CloudTable Service

User Guide

Issue 01

Date 2026-02-03

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

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 Creating a User and Granting Permissions.. 1

2 CloudTable Service Selection..5

3 Using HBase...13

3.1 HBase Usage Process..13

3.2 Creating an HBase Cluster...14

3.3 Connecting to an HBase Cluster..20

3.3.1 Preparing an ECS..20

3.3.2 Connecting to an HBase Normal Cluster Using HBase Shell...24

3.3.3 Connecting to an HBase Security Cluster Using the HBase Shell..................................28

3.3.4 Connecting to the HBase Cluster Through the Thrift Server..31

3.4 Importing Data to an HBase Cluster..32

3.4.1 Using CDM to Migrate Data to CloudTable HBase Clusters...32

3.4.2 Using the Import Tool to Import SequenceFile Data to an HBase Cluster.......................35

3.4.3 Using CopyTable to Import Source Data to an HBase Cluster..36

3.5 HBase Enterprise-class Enhancement...37

3.5.1 Configuring HBase Cold and Hot Data Separation...37

3.5.1.1 Overview...37

3.5.1.2 Configuring HBase Cold and Hot Data Separation Using HBase Shell........................38

3.5.1.3 Configuring HBase Cold and Hot Data Separation Using Java APIs.............................41

3.5.2 Configuring an HBase Global Secondary Index...44

3.5.2.1 About HBase Global Secondary Indexes...45

3.5.2.2 Creating an HBase GSI...47

3.5.2.3 Querying an HBase GSI..48

3.5.2.4 Changing the Status of HBase GSIs...49

3.5.2.5 Deleting GSIs...51

3.6 Managing HBase Clusters...51

3.6.1 Checking the HBase Cluster Status...51

3.6.2 Viewing HBase Cluster Details..54

3.6.3 Restarting an HBase Cluster Node...58

3.6.4 Restarting an HBase Cluster...59

3.6.5 Deleting an HBase Cluster...60

3.7 HBase Cluster O&M..61

3.7.1 Adjusting the Capacity of an HBase Cluster....61
3.7.1.1 Scaling Out HBase Cluster Nodes....61
3.7.1.2 Expanding the Disk Capacity of an HBase Cluster....63
3.7.1.3 Changing the Specifications of an HBase Cluster....64
3.7.2 Modifying HBase Parameters to Optimize Cluster Performance....65
3.7.3 Using Cloud Eye to Monitor HBase Clusters....71
3.7.3.1 HBase Cluster Monitoring Metrics....71
3.7.3.2 Setting Alarm Rules for an HBase Cluster....83
3.7.3.3 Viewing HBase Cluster Monitoring Information....85
3.7.4 Managing HBase Cluster Logs....88
3.7.4.1 Viewing HBase Cluster Logs with LTS....88
3.7.4.2 Viewing HBase Cluster Logs with CTS....89

4 Using Doris....93

4.1 Doris Data Model....93
4.2 Doris Usage Process....110
4.3 Creating a Doris Cluster....112
4.4 Connecting to a Doris Cluster....118
4.4.1 Preparing an ECS....118
4.4.2 Using the MySQL Client to Connect to a Doris Normal Cluster....119
4.4.3 Using the MySQL Client to Connect to a Doris Security Cluster....121
4.5 Configuring Doris User Permissions....124
4.6 Data Import....128
4.6.1 Introduction to Data Import to Doris Clusters....128
4.6.2 Importing Data to a Doris Cluster with Broker Load....129
4.6.3 Importing Data to a Doris Cluster with Stream Load....138
4.7 Doris Enterprise-class Enhancement....146
4.7.1 Configuring a Doris Tenant....146
4.7.1.1 Adding a Doris Tenant....146
4.7.1.2 Commands for Tenant Management....149
4.7.2 Interconnecting Doris with Data Sources....149
4.7.2.1 About Doris Multi-Source Data....149
4.7.2.2 Interconnecting Doris with the Hive Data Source....150
4.8 Managing Doris Clusters....157
4.8.1 Checking the Doris Cluster Status....157
4.8.2 Viewing Doris Cluster Details....160
4.8.3 Restarting a Doris Cluster Node....164
4.8.4 Restarting the Doris Cluster....165
4.8.5 Deleting a Doris Cluster....166
4.9 Doris Cluster O&M....167
4.9.1 Adjusting the Capacity of a Doris Cluster....167
4.9.1.1 Overview....167
4.9.1.2 Doris Node Scale-out....168

4.9.1.3 Expanding the Disk Capacity of a Doris Cluster 169
4.9.1.4 Changing the Specifications of a Doris Cluster 170
4.9.2 Modifying Doris Parameters to Optimize Cluster Performance 172
4.9.3 Using Cloud Eye to Monitor a Doris Cluster 178
4.9.3.1 Doris Cluster Monitoring Metrics 178
4.9.3.2 Setting Doris Cluster Alarm Rules 221
4.9.3.3 Viewing Doris Cluster Monitoring Information 223
4.9.4 Managing Doris Cluster Logs 225
4.9.4.1 Viewing Doris Cluster Logs with LTS 225
4.9.4.2 Viewing Doris Cluster Logs with CTS 226
4.10 Common SQL Commands of Doris 231
4.10.1 Creating a Database 231
4.10.2 Creating a Table 231
4.10.3 Inserting Data 233
4.10.4 Querying Data 234
4.10.5 Modifying a Table Structure 234
4.10.6 Deleting a Table 235
5 Using ClickHouse 236
5.1 ClickHouse Coupled Storage and Compute Table Engine Overview 236
5.2 ClickHouse Usage Process 246
5.3 Creating a ClickHouse Cluster 248
5.4 Connecting to a ClickHouse Cluster 254
5.4.1 Preparing an ECS 254
5.4.2 Using a Client to Connect to a ClickHouse Normal Cluster 254
5.4.3 Using a Client to Connect to a ClickHouse Security Cluster 257
5.4.4 Using HTTPS to Connect to a ClickHouse Secure Cluster 260
5.5 Configuring ClickHouse User Permissions 261
5.6 Data Migration and Synchronization 265
5.6.1 Importing and Exporting data 265
5.6.2 Accessing RDS MySQL Using ClickHouse 268
5.7 ClickHouse Enterprise-class Enhancement 270
5.7.1 Viewing ClickHouse Slow Query Statements 270
5.7.2 Migrating ClickHouse Data 272
5.8 Managing ClickHouse Clusters 277
5.8.1 Checking the ClickHouse Cluster Status 277
5.8.2 Viewing ClickHouse Cluster Details 279
5.8.3 Restarting a ClickHouse Cluster Node 283
5.8.4 Restarting a ClickHouse Cluster 284
5.8.5 Deleting a ClickHouse Cluster 285
5.8.6 Managing ClickHouse Cluster Tags 286
5.9 ClickHouse Cluster O&M 288
5.9.1 Adjusting the Capacity of a ClickHouse Cluster 288

5.9.1.1 ClickHouse Cluster Node Scale-out....288
5.9.1.2 Expanding the Disk Capacity of a ClickHouse Cluster....289
5.9.1.3 Changing the Specifications of a ClickHouse Cluster....290
5.9.2 Using Cloud Eye to Monitor a ClickHouse Cluster....291
5.9.2.1 ClickHouse Cluster Monitoring Metrics....291
5.9.2.2 Setting ClickHouse Cluster Monitoring Rules....303
5.9.2.3 Viewing ClickHouse Cluster Monitoring Information....305
5.9.3 Managing ClickHouse Cluster Logs....307
5.9.3.1 Viewing ClickHouse Cluster Logs with LTS....307
5.9.3.2 Viewing ClickHouse Cluster Logs with CTS....308
5.10 Common SQL Commands for Storage-Compute Coupled ClickHouse....312
5.10.1 Data Types....312
5.10.2 CREATE DATABASE....316
5.10.3 CREATE TABLE....317
5.10.4 DESC|DESCRIBE TABLE....320
5.10.5 CREATE VIEW....321
5.10.6 CREATE MATERIALIZED VIEW....322
5.10.7 INSERT INTO....323
5.10.8 SELECT....324
5.10.9 ALTER TABLE....324
5.10.10 DROP....325
5.10.11 SHOW....326

1 Creating a User and Granting Permissions

Before using CloudTable, register a HUAWEI ID and enable Huawei Cloud services. For details, see **Account Center**. This section describes how to register a Huawei Cloud account. If you already have a Huawei Cloud account, you can directly log in to the management console.

Signing Up with Huawei Cloud and Completing Real-Name Authentication

Step 1 Visit the **website** and click **Sign Up** in the upper right corner of the page.

Step 2 Enter the required information and complete the registration as prompted. After the registration is complete, you are automatically logged in to Huawei Cloud.

Step 3 Click the username in the upper right corner of the page and choose **Basic Information** > **Real-Name Authentication**.

Step 4 Complete the authentication as prompted. For details, see **Real-Name Authentication**.

----End

Using CloudTable Through an IAM Account

This chapter describes **Identity and Access Management (IAM)** fine-grained permissions management for your CloudTable. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing CloudTable resources.
- Grant only the permissions required for users to perform a task.
- Entrust a cloud account or a cloud service to perform professional and efficient O&M on your resources.

If your cloud account does not require individual IAM users, skip this chapter.

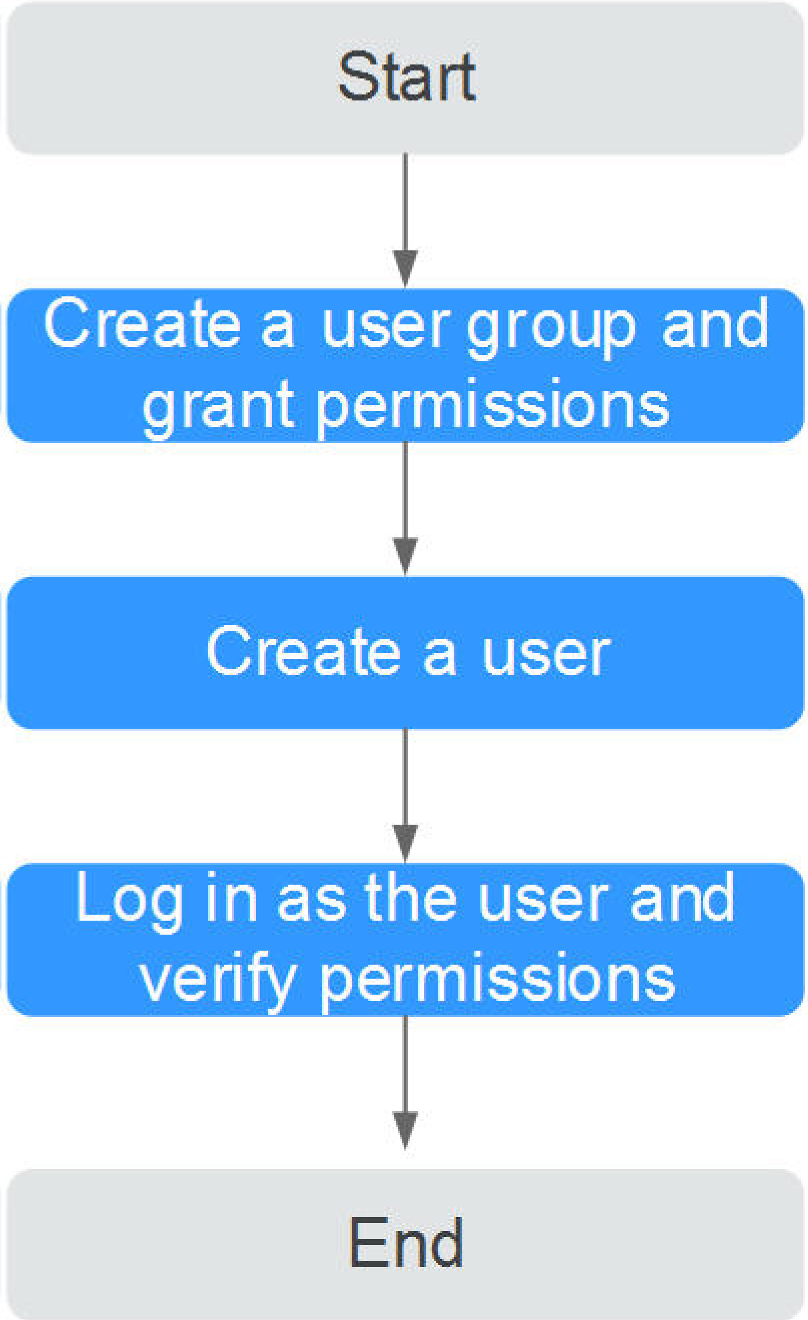
This section describes the procedure for granting permissions (see **Figure 1-1**).

Prerequisites

- CloudTable does not support fine-grained policies. It supports only the system-defined role **cloudtable Administrator**. The default system role **cloudtable Administrator** of CloudTable applies only to accounts that have not enabled the enterprise project function. If the enterprise project function has been enabled, use the IAM custom policy to grant permissions to users.
- Learn about the permissions (see CloudTable Permission Management) supported by CloudTable and choose policies or roles according to your requirements.

Process Flow

Figure 1-1 Process for granting CloudTable permissions



1. **Create a user group and assign permissions** to it.

   Log in to the IAM console, create a user group, and assign the **cloudtable Administrator** permission to it.

2. **Create an IAM user and add it to the user group**.

   Create a user on the IAM console and add the user to the group created in Step **1**.

3. Log in and verify permissions.

   Log in to the console by using the user created, and verify that the user has the granted permissions.

   – Choose **Service List** > **CloudTable Service**. Then click **Buy Cluster** on the CloudTable console. If the cluster can be created (for example, there is only the **cloudtable Administrator** permission), the **cloudtable Administrator** permission has already taken effect

   – Choose any other service in the **Service List** (for example, there is only the **cloudtable Administrator** policy). If a message appears indicating insufficient permissions to access the service, the **cloudtable Administrator** policy has already taken effect.

2 CloudTable Service Selection

Capability Comparison Between CloudTable Components

Table 2-1 Comparison of component capabilities

Comparison Item	HBase	ClickHouse	Doris
Data Storage	• Column-oriented storage which is suitable for large-scale datasets and fast random access • Support for structured, semi-structured, and unstructured data	• Column-oriented data storage which is suitable for real-time analysis and query of mass data • Support only for structured data	• Column-oriented data storage which is suitable for real-time analysis and query of mass data • Support only for structured data
Data Processing	Support for real-time data update and insertion	A high-performance distributed query engine which is strong at complex aggregations but weak at transaction processing	A high-performance distributed query engine which is strong at complex aggregations but weak at transaction processing
Cross-table Query	Not good at	Not good at	Good at foreign and internal table join queries
SQL Support	No support for SQL statements	Support for complex SQL statement operations	Support for complex SQL statement operations

Large Wide Table Support	Support for large tables with millions of columns	Support for querying large wide tables with thousands of columns	Suitable for querying small tables with hundreds of columns
OLTP Capability	No support for transactions, no Atomicity, Consistency, Isolation, and Durability (ACID) capability	No support for transactions, no ACID capability	No support for transactions, no ACID capability
Indexing	Global index, covering index, local index, row key index, and column family index	Sparse index and secondary index	• Point query index: prefix index and sort keys • Skip index: ZoneMap index and BloomFilter index

Application Scenarios of CloudTable Components

Table 2-2 Application scenarios of components

Comparison Item	HBase	ClickHouse	Doris

Application Scenario	• **Real-time data storage and processing**: HBase is a distributed, column-oriented open-source data warehouse. It is designed to handle massive amounts of data, both real-time and non-real-time, and is well-suited for scenarios requiring high-concurrency read and write operations. • **Big data query and analysis**: HBase integrates with big data processing frameworks like Hadoop and Spark to deliver fast data access, making it suitable for big data analysis scenarios. • **Data warehouse**: HBase functions as a component of a data warehousing solution. It stores structured or semi-structured data and supports real-time data query and analysis.	• **Real-time data analysis** This component applies to data analysis scenarios that require low latency and quick response. – Log and monitoring analysis: This component is used for log aggregation and monitoring data analysis systems. – Real-time user behavior analytics: Data of user behaviors (such as clicks, browsing, and purchase behaviors) can be analyzed in real time, which is used for product recommendation, user preference analysis, and ad placement optimization. • **Massive data processing** Data warehouse: TB-level or even PB-level data is supported. It is usually used to build an enterprise data platform to support enterprise decision-making and data insight by aggregating data sources and	• **Real-time data analysis** – Data stream processing: Doris can import and analyze data streams in real time, supporting scenarios like real-time monitoring and real-time user behavior analysis. – Real-time dashboard: Doris is suitable for building real-time visualized dashboards, providing real-time data support for operations and business decision-making. • **Data warehouse** – OLAP: Doris can handle complex OLAP queries on large-scale datasets, delivering quick multidimensional analysis and report generation. – ETL process: Doris can quickly import data from various data sources (such as Kafka,

		analyzing historical data. ● **OLAP** This component delivers excellent performance and can handle complex analytical queries like aggregation, filtering, and sorting. It improves complex query execution efficiency through column-based storage, data compression, and query optimization.	Hadoop, and MySQL) and perform cleanup, aggregation, and analysis. ● **Multi-table federated query and analysis** Join query can be performed between foreign tables in other databases and your internal tables as well as among internal tables in the same database, providing excellent query performance.

Supported Capability	• **Mass data storage** – It can store massive structured, semi-structured, and unstructured data. A single HBase table can accommodate tens of billions of rows and millions of columns, and data can be inserted horizontally and vertically. Therefore, HBase exhibits high elasticity and capacity. – HBase can implement large-scale data storage by deploying inexpensive server clusters with high scalability. • **Fast random query** It is suitable for processing large-scale datasets, supports fast random access, and supports real-time data update and insertion. • **NoSQL query** HBase is suitable for querying and analyzing data	• **Data query and analysis of ultra-large wide tables** It supports data query and analysis of a large wide table with thousands of columns, delivering subsecond-level performance. • **Reads > Writes** Partial columns in a large wide table can be read. Reading one column at a time can meet requirements. Data is integrated in append or update batches. • **Distributed scalability** It adopts a distributed architecture and supports flexible horizontal scaling.	• **Real-time data import, query, and analysis** Data can be imported from various data sources such as Kafka, Hadoop, and MySQL in real time, and can be queried and analyzed in real time. • **High-performance online multi-table join query and analysis** It supports high-concurrency multi-table join query and analysis on PB-level massive data. With subsecond-level performance, it is applicable to real-time analysis and service reports. • **Distributed scalability** It adopts a distributed architecture and supports flexible horizontal scaling. • **Flexible data models** Multiple data models and data types are supported to meet requirements of different service scenarios.

	using non-SQL statements.		

Not Recommended Capability	• **No support for OLTP capability** The ACID capability is not supported. RDS capabilities are unavailable in terms of atomicity, consistency, and real-time performance of point query transactions. • **No support for SQL statement capability** There is no support for using SQL statements to import or query data.	• Weak **OLTP capability** The ACID capability is not supported. RDS capabilities are unavailable in terms of atomicity, consistency, and real-time performance of point query transactions. • **Not applicable to scenarios where a small amount of data is imported and updated frequently** There is no support for full update or deletion operations. Existing data cannot be modified or deleted at a high frequency and low latency. Only batch deletion or modification of data is supported. • **Not good at sorting out unstructured data** It is effective in querying and analyzing structured data based on SQL statements, but not good at processing semi-structured or unstructured data. • **Weak multi-table join operation capability**	• **Weak OLTP capability** The ACID capability is not supported. RDS capabilities are unavailable in terms of atomicity, consistency, and real-time performance of point query transactions. • **Not applicable to scenarios where a small amount of data is imported and updated frequently** There is no support for full update or deletion operations. Existing data cannot be modified or deleted at a high frequency and low latency. Only batch deletion or modification of data is supported. • **Weak query and analysis capabilities for large wide tables** It is suitable for query and analysis of small- and medium-sized data warehouses. The query and analysis performance of ultra-wide tables with more than 1000 columns is weak.

			• **Not good at sorting out unstructured data** It is effective in querying and analyzing structured data based on SQL statements, but not good at processing semi-structured or unstructured data.

3 Using HBase

3.1 HBase Usage Process

The CloudTable HBase cluster mode provides a distributed, scalable, and fully managed NoSQL data storage system based on Apache HBase. It provides strong consistency and single-digit millisecond latency so it is optimal for storage and queries of massive amounts of structured and semi-structured data.

Figure 3-1 HBase usage process

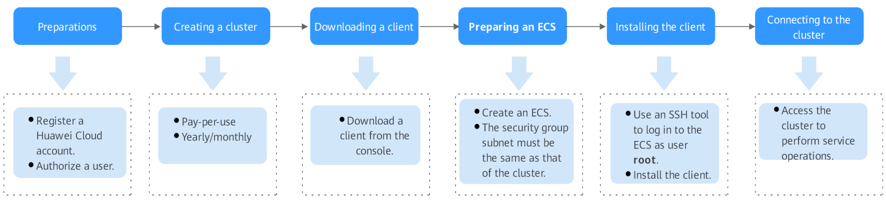


Table 3-1 HBase usage process

Step	Substep	Description	Detailed Instructions
Preparations	Creating a user and granting permissions	• Before using CloudTable HBase, you need to register a Huawei Cloud account, complete real-name authentication, and grant the necessary permissions to your account. • Grant the necessary service permissions to a user group, and then add users to this group to enable their access.	**Creating a User and Granting Permissions**

Step	Substep	Description	Detailed Instructions
Creating a cluster	Creating an HBase cluster	Before using HBase to execute tasks, you need to create an HBase cluster.	**Creating an HBase Cluster**
Downloading the client	Downloading the HBase client	After creating a cluster, download and install the client. After the client is installed, you can use the SSH tool to connect to the cluster.	**Connecting to an HBase Normal Cluster Using HBase Shell**
Preparing an ECS	-	If the client tool runs on Linux, you need to prepare a Linux ECS that is in the VPC as an HBase cluster and the Linux ECS serves as a client host. If the client tool runs on Windows, you need to prepare a Windows ECS that is in the VPC as an HBase cluster and the Windows ECS serves as a client host.	**Preparing an ECS**
Installing the client	-	Place the downloaded client on the ECS, decompress the package, and install the client.	**Manually Installing a Client**
Connecting to the cluster	-	After installing the MySQL client on the ECS, you can run commands to connect to the cluster and perform service operations.	**Connecting to an HBase Cluster**

3.2 Creating an HBase Cluster

You can centrally manage clusters with CloudTable. A cluster is necessary for using CloudTable. This section describes how to create a cluster on the CloudTable console.

HBase clusters support two billing modes: pay-per-use and yearly/monthly. By default, the cluster creation page is set to the pay-per-use mode, which provides flexibility by allowing you to enable or disable clusters as needed and pay only for the actual usage time. Alternatively, you can opt for the yearly/monthly billing mode, which is a prepaid option offering significant discounts compared to the pay-per-use mode. This option is particularly suitable for long-term users. You can also customize a CloudTable HBase cluster with specified computing capabilities and storage space to meet your business needs.

NOTE

Created HBase clusters can be accessed without passing Kerberos authentication. If you have requirements on cluster access security, you are advised to use the HBase component on MRS.

Prerequisites

- The VPC and security group of the cluster to be created must be the same as those of the ECS on the public network. Otherwise, the client cannot access the cluster.
- Before creating a cluster, you must configure inbound security group rules for the host. For details, see **Configuring Security Group Rules**. For details about the security group port, see **HBase Security Group Rules**.
- Before creating a cluster, you must add the ICMP protocol to the security group rules so that you can view the status of each node by pinging the node IP address on the management plane. For details, see **Configuring Security Group Rules**.

Creating a Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Create Cluster**. The **Create Cluster** page is displayed.

Step 4 Configure basic cluster information by referring to the following table.

Table 3-2 Region parameters

Parameter	Description
Region	Region of the cluster.
AZ	AZ associated with the cluster's region. For more information, see **Regions and AZs**.
Billing Mode	Select **Pay-per-use** or **Yearly/Monthly**.
Required Duration	This option is available only when **Billing Mode** is set to **Yearly/Monthly**. Configure this parameter based on your service requirements.
Auto-renew	If you select **Auto-renew** when creating a cluster, the system will automatically renew your subscription before it expires.

Table 3-3 Network configuration parameters

Parameter	Description
Name	Name of a cluster. The cluster name must consist of 4 to 32 characters and must begin with a letter. It may include only letters, digits, and hyphens (-) but must not contain any other special characters. Additionally, the cluster name is case-insensitive.
VPC	A VPC is a secure, isolated, and logical network environment. You can select an existing VPC or click **View VPC** to create a new one. For details about how to create a VPC, see **"User Guide" > "VPC and Subnet" > "Creating a VPC with a Subnet"** in **Virtual Private Cloud**.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks, improving network security. A subnet is automatically created when a VPC is created. If you want to create a subnet, you can refer to **"User Guide" > "VPC and Subnet" > "Creating a VPC with a Subnet"** in .

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Security Group</td><td>A security group is used to control ECS access within a security group or between security groups by defining access rules. You can define different access control rules for a security group. These rules can specify which ECS ports or protocols are accessible and can be used to control inbound and outbound network traffic of VMs. After an ECS is added to the security group, it is protected by these access control rules. ECSs that do not belong to the security group cannot communicate with ECSs in the security group.

The underlying compute unit of a CloudTable cluster is ECS. In terms of security and stable service running, the ECSs must be added to the same security group in the same VPC. VPCs isolate networks and security groups specify which ports and protocols can be opened in the VPC.

You can use an existing security group or click View Security Group to create a new one.

For more information about security groups, see Security Group in the Virtual Private Cloud User Guide.

NOTE
● CloudTable clusters support multiple security groups and security group modification. Security group rules:
– External servers can ping the instances in the security group to verify network connectivity.
– Instances in a security group can communicate with each other via a private network.
– Instances in a security group can be accessed via a private network.
● Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.</td></tr>
<tr><td>Database Engine</td><td>Select the type of cluster to be created.</td></tr>
</table>

Table 3-4 Master node configuration

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** ● Available compute specifications: – 4U16G – 8U16G – 8U32G – 16U32G – 16U64G – 32U64G – 32U128G – 64U128G
Storage	**Ultra-high I/O** is recommended.
Nodes	Use the default value **2**.

Table 3-5 Core node configuration

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** ● Available compute specifications: – 4U16G – 8U16G – 8U32G – 16U32G – 16U64G – 32U64G – 32U128G – 64U128G

Parameter	Description
Storage	Select the disk specifications and capacity of the HBase compute node. **NOTE** ● Available storage specifications: – Common I/O – High I/O – Ultra-high I/O ● The capacity ranges from 400 GB to 30,000 GB per node.
Nodes	Set the number of nodes in a cluster. The value ranges from **2** to **20**.
Advanced Feature	Cold and hot data separation: It classifies data based on access frequency and stores data on different media to save storage costs. This feature is available only in HBase 2.4.14 and later versions. You can toggle it on to enable this feature.
Enable Thrift Server	This is disabled by default. You can enable this to ensure that HBase supports multi-language access.
Tag	A tag is a key-value pair customized by users and used to classify and search for cloud resources. A tag consists of a key and a value.
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** ● You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. ● You can delete a user or multiple users. ● After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.

Step 5 Click **Next**.

Step 6 Confirm the details of the order and click **Submit**. The cluster creation task is submitted successfully.

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

The cluster creation task takes some time. The initial status of the cluster is **Creating**. After the cluster is created, the cluster status changes to **In service**.

Step 8 Submit the creation task of a yearly/monthly cluster.

Click **Pay**. On the displayed purchase page, confirm the information, select a proper payment method, and confirm the payment.

Return to the console and check the cluster status. Cluster creation takes some time. The initial status of the cluster is **Creating**. After the cluster is created, the cluster status changes to **In service**.

----End

3.3 Connecting to an HBase Cluster

3.3.1 Preparing an ECS

If the client tool runs on Linux, you need to prepare a Linux ECS that is in the VPC as a CloudTable cluster and the Linux ECS serves as a client host. If you use the following client tools to access a cluster, you are advised to use the HBase shell, a Linux ECS.

If the client tool runs on Windows, you need to prepare a Windows ECS that is in the VPC as a CloudTable cluster and the Windows ECS serves as a client host.

Preparing an ECS

For details about how to purchase a Linux or Windows ECS, see **Purchasing an ECS in Custom Config Mode** in the *Elastic Cloud Server Getting Started*.

The purchased ECS must meet the following requirements:

- The ECS must have the same region, AZ, VPC, and subnet as the CloudTable cluster.

  For details about how to create a VPC, see **"User Guide" > "VPC and Subnet"** of **Virtual Private Cloud**.
- The ECS must have the same security group as the CloudTable cluster.

  For more information about security groups, see in the *Virtual Private Cloud User Guide*.

  NOTE

  When cross-VPC communication is used to access a CloudTable cluster, the network administrator needs to authorize the access to the VPC, security group, and subnet where the cluster resides.

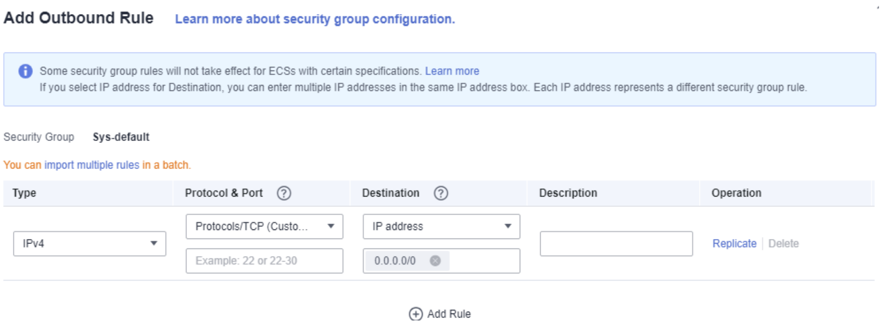
  Check whether the outbound rule of the security group contains the following rule:

  – Protocol & Port: All
  – Type: IPv4
  – Destination: 0.0.0.0/0

  If the preceding rules do not exist, add an outbound rule according to **Figure 3-2**.

  – Protocol: TCP
  – Port: 80
  – Destination: Select **IP address** and set it to **0.0.0.0/0**.

Figure 3-2 Adding an outbound rule



- When purchasing an ECS, you need to set **EIP** to **Automatically assign**. Alternatively, you can bind an EIP to an ECS after the ECS is created.
- To access a Linux ECS, you are advised to use an SSH password.

  For details about how to log in to a Linux ECS, see **Logging In to a Linux ECS** in the *Elastic Cloud Server User Guide*.
- To access a Windows ECS, you are advised to use the MSTSC-based remote desktop connection tool.

  For details, see **Logging In to a Windows ECS** in the *Elastic Cloud Server User Guide*.

Configuring the DNS Address and hosts File for the Linux ECS

You do not need to perform this operation when you install the HBase shell by deploying a client in one click. In other cases, perform the following steps to configure the DNS address and the **hosts** file for the Linux ECS:

Step 1 Configure **/etc/hosts**.

Configure the host IP address in the **/etc/hosts** file to accelerate the access to the CloudTable cluster.

To configure /etc/hosts for a Linux ECS, perform the following steps:

1. Log in to the Linux ECS as user **root**.
2. Run the **hostname** command to query the host name.
```
[root@euler ~]# hostname
euler.novalocal
```
3. Run the **ifconfig** command to query the IP address of the local host.

Figure 3-3 Querying the IP address

```
[root@euler ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.58  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe8d:7acb  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:8d:7a:cb  txqueuelen 1000  (Ethernet)
        RX packets 379443  bytes 522261185 (498.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 148302  bytes 10571485 (10.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 23819  bytes 16629808 (15.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23819  bytes 16629808 (15.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@euler ~]#
```

4. Run the **vi /etc/hosts** command to edit the file and add the host configuration.
```
192.168.0.58 euler.novalocal
127.0.0.1 euler.novalocal
```
5. Press **Esc** and enter **:wq** to save the settings and exit.
6. Run the following command to check whether the IP address is successfully added:
```
cat /etc/hosts
```
7. Run the following command to check whether the host name can be resolved:
```
ping Host name
```

Step 2 Configure the DNS.

The DNS server is used to resolve the domain name in the CloudTable cluster link, for example, the ZooKeeper link. Set the private DNS server address specific to the region where the CloudTable cluster is located. For details about the DNS server address for each region, see **What Are Huawei Cloud Private DNS Server Addresses?** in the *Domain Name Service FAQs*.

To configure DNS for a Linux ECS, perform the following steps:

1. Log in to the Linux ECS as user **root**.
2. Run the **vi /etc/resolv.conf** command to edit the **/etc/resolv.conf** file and add the **nameserver** configuration.

   Above the Internet IP address, add two lines to input the DNS server IP addresses in the following format.
```
nameserver 100.125.1.250
```
3. Press **Esc** and enter **:wq** to save the settings and exit.
4. Run the following command to check whether the IP address is successfully added:
```
cat /etc/resolv.conf
```
5. Run the following command to check whether an IP address can be resolved from the access domain name.
```
ping Access domain name
```
   Access domain name indicates the domain name in the CloudTable cluster link, for example, the ZooKeeper link. On the CloudTable management console, click the cluster name to go to the basic cluster information page and obtain the link of CloudTable. The obtained link contains three private domain names separated by commas (,). You can ping any of the private domain names.

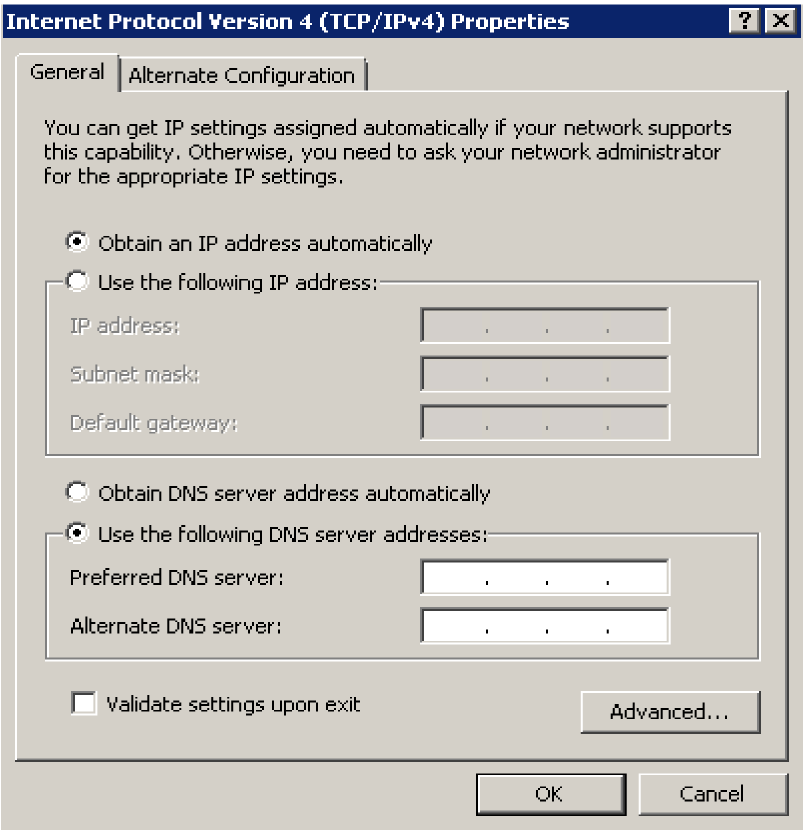
----End

Configuring DNS Address for a Windows ECS

1. Log in to the Windows ECS as user **Administrator**.
2. On the Windows ECS, choose **Start** > **Control Panel** > **Network and Sharing Center**.
3. In the **Network and Sharing Center** window, click **Change adapter settings** on the right.
4. Right-click the network adapter name and choose **Properties** from the shortcut menu. The **Properties** dialog box is displayed.

5. Click **Internet Protocol Version 4 (TCP/IPv4)**. The **Internet Protocol Version 4 (TCP/IPv4) Properties** dialog box is displayed.

Figure 3-4 Internet protocol version 4 (TCP/IPv4) properties



6. In the dialog box shown in **Figure 3-4**, select **Use the following DNS server addresses**, enter the IP address of the DNS server in the **Preferred DNS server** textbox, and click **OK**.

   The DNS server is used to resolve the domain name in the CloudTable cluster link, for example, the ZooKeeper link. Set the private DNS server address specific to the region where the CloudTable cluster is located. For details about the DNS server address for each region, see **What Are Huawei Cloud Private DNS Server Addresses?** in the *Domain Name Service FAQs*.

7. Click **Start**, enter **cmd** in the search box, and press **Enter**. The CLI is displayed.
8. Run the following command on the CLI to check whether the DNS is successfully configured:

   ```
   ping Access domain name
   ```

   Access domain name indicates the domain name in the CloudTable cluster link, for example, the ZooKeeper link. On the CloudTable management console, click the cluster name to go to the basic cluster information page and obtain the link of CloudTable. The obtained links contain three private domain names separated by commas (,). You can ping any of the private domain names.

3.3.2 Connecting to an HBase Normal Cluster Using HBase Shell

You can use the HBase shell to access a cluster by **manually installing a client** or **deploying a client with one click** on an ECS. You are advised to **deploy a client with one click**. If a security channel is enabled for the cluster, connect to the cluster by referring to **Connecting to an HBase Security Cluster**.

Constraints

- The HBase cluster and the ECS must be in the same region, AZ, and VPC.
- The HBase cluster and the ECS must be in the same security group.
- The IP address of the local host has been added to the ECS security group.

Deploying a Client with One Click

Step 1 Prepare a Linux ECS.

To use the one-click client deployment tool, you are advised to use EulerOS, CentOS, Ubuntu, or SUSE Linux ECSs. For details, see **Preparing an ECS**.

Step 2 Download the one-click client deployment tool.

Use an SSH login tool (such as PuTTY) to remotely log in to the Linux ECS through the EIP and run the following command to obtain the one-click client deployment tool:

```
curl -O -k "https://cloudtable-publish.obs.myhuaweicloud.com/quick_start_hbase_shell.sh"
```

NOTE

This command applies to HBase 1.*x*.

```
curl -O -k "https://cloudtable-publish.obs.myhuaweicloud.com/cloudtable-client/quick_start_hbase_shell.sh"
```

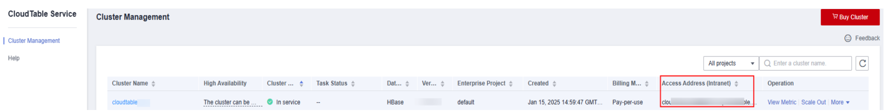
NOTE

- This command applies to HBase 2.*x*.
- The one-click deployment package contains the verification file.

Step 3 Obtain a cluster access address.

Choose **Cluster Management** in the navigation pane. In the cluster list, locate the required cluster and obtain the address in the **Access Address (Intranet)** column. The parameter value is the cluster access address, as shown in **Figure 3-5**.

Figure 3-5 Access address



Step 4 Use the tool to deploy the client.

Replace **$zookeeper_address** in the following command with the address you obtained in **Step 3**. Then, run the command on the CLI of the ECS to deploy the client with one click.

1. Command for one-click client deployment for normal clusters:
```
source quick_start_hbase_shell.sh $zookeeper_address
```
2. Command for one-click client deployment for security clusters:
```
source quick_start_hbase_shell $zookeeper_address enable
```

Step 5 Start the shell to access the cluster.

After you run the **source** command to automatically deploy the client, the HBase shell is automatically started. You can also run the **bin/hbase shell** command to start the HBase shell to access the cluster.

----End

Manually Installing a Client

Step 1 Prepare a Linux ECS.

For details, see **Preparing an ECS**.

Step 2 Download the client and verification file.

In the navigation pane on the left, choose **Help**. In the right pane, click the button for downloading client and **Download the Client Verification File** to download the client installation package and client verification file.

Step 3 Install the client and verify the client.

1. Use a file transfer tool (such as WinSCP) to upload the client installation package to the Linux ECS.
2. Use the SSH login tool (such as PuTTY) to log in to the Linux ECS through the EIP.

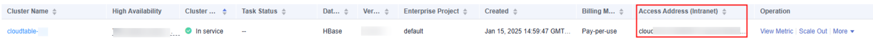
   For details about how to log in to the Linux ECS, see **Logging In to a Linux ECS** > "Login Using an SSH Password" in the *Elastic Cloud Server User Guide*.

   Run the following command to decompress the client installation package:
```
cd <Path of the client installation package>
tar xzvf hbase-1.3.1-bin.tar.gz
cd <Path of the client installation package>
tar xzvf hbase-2.4.14-bin.tar.gz
```
   <Path of the client installation package>: Replace it with the actual path.
3. Decompress the client verification file to the same directory as the client.
   a. Decompress the client verification file.
```
cd <Path for storing the client verification file >
tar xzvf Client_sha256.tar.gz
```
   b. Obtain the client verification code.
```
sha256sum HBase_Client_2.4.14.tar.gz
```
   c. Check the verification code in the client verification file and compare it with the client verification code. If they are the same, the client is not tampered with. If they are different, the client is tampered with.
```
less HBase_Client_2.4.14.tar.gz.sha256
```
4. Configure the ZooKeeper address in a configuration file.

   In the decompression directory of the client installation package, open the **hbase/conf/hbase-site.xml** file and set the following parameters:
   – **hbase.zookeeper.quorum**: The value of this parameter is the cluster's access address (Intranet) obtained in the cluster list.

Choose **Cluster Management** in the navigation pane. In the cluster list, locate the required cluster and obtain the address in the **Access Address (Intranet)** column. See the following figure.

Figure 3-6 Obtaining the access address



– **mapreduce.cluster.local.dir**: Check whether the configuration item exists. If it does not exist, add it.

The following is an example of configuring the client for a cluster with security channel encryption disabled:

```
<configuration>
<property>
<name>hbase.zookeeper.quorum</name>
<value>xxx-zk1.cloudtable.com:2181,xxx-zk2.cloudtable.com:2181,xxx-zk3.cloudtable.com:2181</
value>
</property>
<property>
<name>mapreduce.cluster.local.dir</name>
<value>${hadoop.tmp.dir}/mapred/local</value>
</property>
</configuration>
```

For details about how to configure the client for a cluster with security channel encryption enabled, see **Connecting to an HBase Security Cluster Using the HBase Shell**.

Step 4 Start the shell to access the cluster.

Run the **bin/hbase shell** command to start the shell to access the cluster.

----End

Getting Started with HBase

This section describes common HBase shell commands. For more HBase shell commands, visit **https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/**.

1. Obtain online help.

   After you run the **help** command in the HBase shell, all command information as well as common command instructions and use methods will be returned.

   ```
   hbase(main):001:0> help
   ```

2. Create a table.

   Run the **create** command to create a table. When creating a table, you must specify the table name and column family name.

   ```
   hbase(main):007:0> create 'cloudtable','cf'
   0 row(s) in 1.5530 seconds

   => Hbase::Table - cloudtable
   ```

3. Query a table.

   ```
   hbase(main):009:0> list
   TABLE
   cloudtable
   1 row(s) in 0.0060 seconds

   => ["cloudtable"]
   ```


4. Insert a record to a table.

   Run the **put** command to insert a data record to the table. You need to specify the table name, primary key, custom column, and the value to be inserted.

   ```
   hbase(main):004:0> put 'cloudtable','row1','cf:a','value1'
   0 row(s) in 0.2720 seconds
   ```

   The parameters in the command are as follows:

   – **cloudtable**: table name
   – **row1**: primary key
   – **cf:a**: custom column
   – **value1**: value to be inserted

5. Scan records.

   Run the **scan** command to scan a table. You need to specify the table name to scan the entire table or specify a scan range.

   ```
   hbase(main):001:0> scan 'cloudtable'
   ROW                                                   COLUMN+CELL
   row1                                                  column=cf:a, timestamp=1504866237162,
   value=value1
   1 row(s) in 0.2420 seconds
   ```

   NOTE

   - If the TTL of a cell is set when data is inserted, the TTL attribute cannot be viewed. However, you can check whether the TTL takes effect.
   - If the TTL of a cell is not set when data is inserted, the system automatically inserts the current time as the timestamp.

6. Query a single record.

   Run the **get** command to query a single record. You must specify the name and the primary key of the table.

   ```
   hbase(main):001:0> get 'cloudtable','row1'
   COLUMN                                                CELL
   cf:a                                                  timestamp=1504866237162, value=value1
   1 row(s) in 0.2280 seconds
   ```

7. Disable a table.

   Before modifying or deleting a table, you need to disable the table. Run the **disable** command to disable the table. When you perform operations on a disabled table, **ERROR** is displayed, indicating that the table is disabled.

   ```
   hbase(main):002:0> disable 'cloudtable'
   0 row(s) in 2.3550 seconds
   ```

8. Enable a table.

   If you want to use a table that has been disabled, run the **enable** command to enable it.

   ```
   hbase(main):004:0> enable 'cloudtable'
   0 row(s) in 1.2500 seconds
   ```

9. Delete a table.

   Run the **drop** command to delete a table that is no longer needed. Before deleting a table, disable it first. Otherwise, **ERROR** will be displayed, indicating that the table is enabled. Deleting a table will cause data loss. Exercise caution when deleting a table.

   ```
   hbase(main):007:0> disable 'cloudtable'
   0 row(s) in 2.2380 seconds
   ```

```
hbase(main):008:0> drop 'cloudtable'
0 row(s) in 1.2600 seconds
```

10. Exit the HBase shell.

    Run the **quit** command to exit the HBase shell.

```
hbase(main):009:0> quit
```

3.3.3 Connecting to an HBase Security Cluster Using the HBase Shell

You can enable security channel encryption to encrypt data transmission. This section describes how to enable a secure channel for an HBase cluster.

Constraints

- Enabling a security channel may deteriorate the cluster performance.
- The HBase security channel can be enabled only when a cluster is created.
- The HBase cluster and the ECS must be in the same region, AZ, and VPC.
- The HBase cluster and the ECS must be in the same security group.
- The IP address of the local host has been added to the ECS security group.

Creating a Cluster and Enabling the Security Channel

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Create Cluster** in the upper right corner.

Step 4 Check whether **Security Channel** is toggled on (default).

Step 5 Set the parameters and click **Next**.

Step 6 Confirm the cluster information and click **Submit**. After the cluster is created, go to its details page to view its security channel status.

----End

Connecting to an HBase Security Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Choose **Help** and click **Download the HBase2.x Client** to download the client.

Step 4 Use the SSH login tool (such as PuTTY) to log in to the Linux ECS through the EIP.

For details about how to log in to the ECS, see **Logging In to a Linux ECS** > "Logging In to a Linux ECS Using an SSH Password" in the *Elastic Cloud Server User Guide*.

Step 5 Install the client downloaded in **Step 3**. For details, see **Install the client and verify the client**.

Step 6 Go to the client installation directory and modify the **hbase-env.sh** file.

- Go to the **hbase-client** folder and view the **conf** folder.
```
cd hbase-client
```
- Go to the **conf** folder and edit the **hbase-env.sh** file.
  - Go to the **conf** folder.
  ```
  cd conf/
  ```
  - Edit the **hbase-env.sh** file.
  ```
  vi hbase-env.sh
  ```
  - Add the following content to the end of the **hbase-env.sh** file:
  ```
  CLIENT_JVMFLAGS="-
  Dzookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty -
  Dzookeeper.client.secure=true"
  export HBASE_OPTS="${CLIENT_JVMFLAGS} ${HBASE_OPTS}"
  ```
  - Press **Ecs** to exit the editing page.
  - Save the added content.
  ```
  :wq
  ```

Step 7 Go to the client installation directory to configure the client. Add the required configurations to the **hbase-site.xml** file in the **conf** directory on the client.

- Edit the **hbase-site.xml** file.
```
vi hbase-site.xml
```
- Add the following content to the end of the **hbase-site.xml** file:
```
<property>
   <name>hbase.rpc.protection</name>
   <value>privacy</value>
</property>
```
- Press **Ecs** to exit the editing page.
- Save the added content.
```
:wq
```

Step 8 Connect to the HBase cluster.

- Configure the ZooKeeper address in the configuration file. For details, see **Configure the ZooKeeper address in a configuration file**.
- Start the shell to access the cluster.

  Run the **bin/hbase shell** command to start the shell to access the cluster.

----End

Getting Started with HBase

This section describes common HBase shell commands. For more HBase shell commands, visit **https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/**.

1. Obtain online help.

   After you run the **help** command in the HBase shell, all command information as well as common command instructions and use methods will be returned.
   ```
   hbase(main):001:0> help
   ```
2. Create a table.

   Run the **create** command to create a table. When creating a table, you must specify the table name and column family name.

```
hbase(main):007:0> create 'cloudtable','cf'
0 row(s) in 1.5530 seconds

=> Hbase::Table - cloudtable
```

3. Query a table.

```
hbase(main):009:0> list
TABLE
cloudtable
1 row(s) in 0.0060 seconds

=> ["cloudtable"]
```

4. Insert a record to a table.

   Run the **put** command to insert a data record to the table. You need to specify the table name, primary key, custom column, and the value to be inserted.

```
hbase(main):004:0> put 'cloudtable','row1','cf:a','value1'
0 row(s) in 0.2720 seconds
```

   The parameters in the command are as follows:

   – **cloudtable**: table name
   – **row1**: primary key
   – **cf:a**: custom column
   – **value1**: value to be inserted

5. Scan records.

   Run the **scan** command to scan a table. You need to specify the table name to scan the entire table or specify a scan range.

```
hbase(main):001:0> scan 'cloudtable'
ROW                                        COLUMN+CELL
row1                                      column=cf:a, timestamp=1504866237162,
value=value1
1 row(s) in 0.2420 seconds
```

   NOTE

   - If the TTL of a cell is set when data is inserted, the TTL attribute cannot be viewed. However, you can check whether the TTL takes effect.
   - If the TTL of a cell is not set when data is inserted, the system automatically inserts the current time as the timestamp.

6. Query a single record.

   Run the **get** command to query a single record. You must specify the name and the primary key of the table.

```
hbase(main):001:0> get 'cloudtable','row1'
COLUMN                                     CELL
cf:a                                      timestamp=1504866237162, value=value1
1 row(s) in 0.2280 seconds
```

7. Disable a table.

   Before modifying or deleting a table, you need to disable the table. Run the **disable** command to disable the table. When you perform operations on a disabled table, **ERROR** is displayed, indicating that the table is disabled.

```
hbase(main):002:0> disable 'cloudtable'
0 row(s) in 2.3550 seconds
```

8. Enable a table.

   If you want to use a table that has been disabled, run the **enable** command to enable it.


```
hbase(main):004:0> enable 'cloudtable'
0 row(s) in 1.2500 seconds
```

9. Delete a table.

   Run the **drop** command to delete a table that is no longer needed. Before deleting a table, disable it first. Otherwise, **ERROR** will be displayed, indicating that the table is enabled. Deleting a table will cause data loss. Exercise caution when deleting a table.

   ```
   hbase(main):007:0> disable 'cloudtable'
   0 row(s) in 2.2380 seconds

   hbase(main):008:0> drop 'cloudtable'
   0 row(s) in 1.2600 seconds
   ```

10. Exit the HBase shell.

   Run the **quit** command to exit the HBase shell.

   ```
   hbase(main):009:0> quit
   ```

3.3.4 Connecting to the HBase Cluster Through the Thrift Server

The HBase Thrift Server operates as a service within the HBase cluster, offering a gateway to the HBase database via the Thrift protocol. This server is adept at producing both client and server code, accommodating a multitude of programming languages for diverse application needs.

Constraints

- The Thrift Server supports the following programming languages: Python and C++.
- The Thrift Server does not support features such as cold and hot data separation, automatic hotspot self-healing, or secondary indexing.
- Thrift Server is deployed only on Core nodes.
- Scaling out the Thrift Server is directly dependent on the scaling capabilities of the Core nodes.
- If the Thrift Server encounters issues, services may be disrupted, especially when it is accessed via IP addresses.
- The Thrift Serve feature is supported in only HBase 2.4.14 or later.
- Thrift Server is disabled by default.

Prerequisites

- You have **created an HBase cluster**.
- The cluster status is **In Service**.

Procedure for Enabling Thrift Server

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Buy Cluster** in the upper right corner.

Step 4 Go to the **Create Cluster** page. On the cluster purchase page, select the HBase database engine and enable Thrift Server.

You can also choose **More** > **Enable Thrift Server** in the **Operation** column on the cluster management page to enable Thrift Server.

Step 5 Click the target cluster name. In the **Cluster Information** area on the cluster details page, obtain the Thrift Server IP address.

Step 6 Select the runtime language and configure the IP address. For details, see "HBase Multi-Language Access" in the *Developer Guide*.

----End

Procedure for Disabling Thrift Server

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Select the cluster for which you want to disable Thrift Server and choose **More** > **Disable Thrift Server**.

Step 4 In the displayed window for disabling the Thrift Server service, click **OK**.

----End

3.4 Importing Data to an HBase Cluster

3.4.1 Using CDM to Migrate Data to CloudTable HBase Clusters

CloudTable uses CDM to migrate data from multiple data sources on the cloud and on-premises or third-party cloud to an HBase table of a CloudTable cluster.

The procedure of using CDM to migrate data to CloudTable is as follows:

1. **Creating a CloudTable Cluster**
2. **Using CDM to Migrate Data to CloudTable**
3. **Viewing the Imported Data on CloudTable**

Creating a CloudTable Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Create a **CloudTable HBase** cluster. If there is one available, you can skip this step.

For example, create a CloudTable cluster named **CloudTable-demo**.

----End

Using CDM to Migrate Data to CloudTable

Step 1 Log in to the **CDM console** and create a CDM cluster.

The CDM cluster you create must have the same region, AZ, VPC, subnet, and security group as the CloudTable cluster.

For details, see **Creating a Cluster**.

Step 2 Create a source link in the new CDM cluster.

CDM enables you to migrate data from multiple data sources to CloudTable. Create a link as the source link based on the actual data source. The link is used for CDM to connect to the data source and read data from the data source.

For details about how to create a link, see **Creating a CDM Link**.

Step 3 Create a destination link in the new CDM cluster.

In the CDM cluster, create a link for which **Data Source Type** is set to **CloudTable** as a destination link. The link is used for CDM to connect to the CloudTable cluster, read data from the data source, and import data to the HBase table of the CloudTable cluster.

For details about how to create a link, see **Creating a CDM Link**.

The following is an example of destination link settings.

1. On the **Cluster Management** page, locate the newly created CDM cluster and click **Job Management** in the row of the CDM cluster. The job management page is displayed.
2. Click the **Link Management** tab and then click **Create Link**. On the **Select connector type** page that is displayed, select **CloudTable Service** and click **Next**.
3. Configure link parameters.
   - **Name**: Enter a link name. You can create a link name based on the data source type for easy memorization. For example, enter **cloudtable_connect**.
   - **ZK Link**: Set this parameter to the access address (Intranet) of the CloudTable cluster. Choose **Cluster Management** in the navigation pane. In the cluster list, locate the required cluster and obtain the address in the **Access Address (Intranet)** column.
   - **IAM Authentication**: Set this parameter to **No**.
   - **IAM Authentication**: CloudTable does not use IAM authentication. Therefore, set this parameter to **No**.
   - **Run Mode**: Retain the default value.
4. Click **Save**.

Step 4 In the CDM cluster, create a job on the **Table/File Migration** page.

For details about how to create a **table/file migration** job, see .

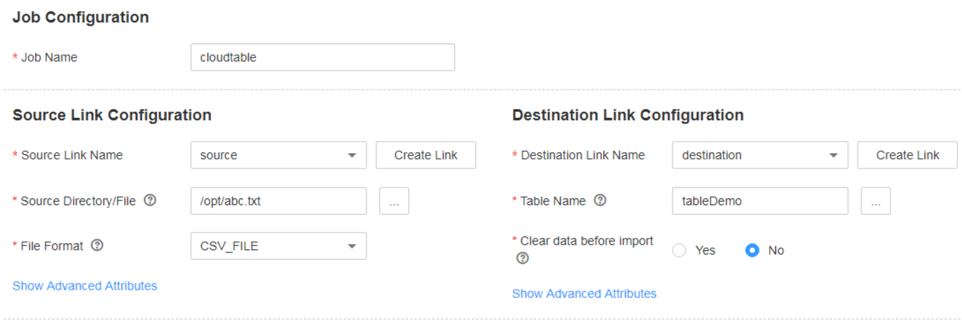
NOTE

When you create a table/file migration job, if the table or file of the source end does not have the corresponding HBase table in the destination CloudTable cluster, set a table name and field mapping in the job for the destination end (the field name can be copied from the source end). Therefore, when a job is running, the corresponding HBase table is automatically created at the destination end.

The following uses the SFTP data source on an ECS as an example to describe how to create a job.

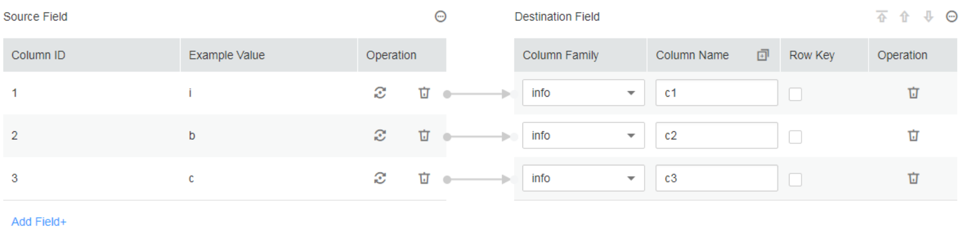
1. On the **Cluster Management** page, locate the newly created CDM cluster and click **Job Management** in the row of the CDM cluster. The job management page is displayed.
2. Choose **Table/File Migration** > **Create Job**.
3. Configure basic information as follows.

Figure 3-7 Configuring basic information



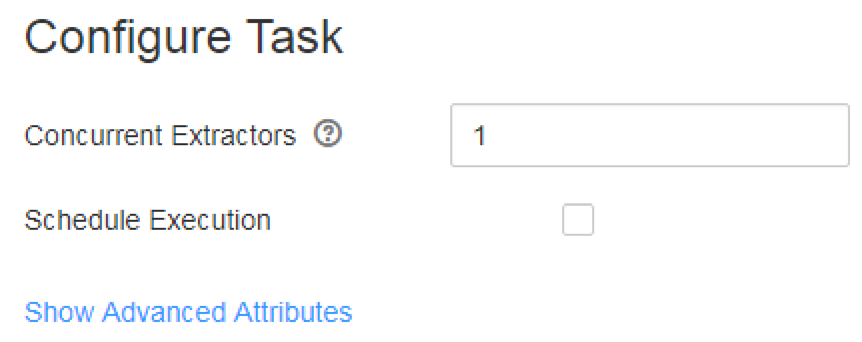
4. Configure field mapping as follows.

Figure 3-8 Field mapping



5. Configure a task as follows.

Figure 3-9 Configuring a task



6. Click **Save and Run**.

----End

Viewing the Imported Data on CloudTable

Step 1 Prepare a Linux ECS.

Assume that the ECS name is **ecs_20170916**. For details, see **Preparing an ECS**.

Step 2 Install the client and start the shell to access the CloudTable cluster.

For details about how to use the HBase shell to access the cluster, see **Connecting to an HBase Normal Cluster Using HBase Shell**.

Step 3 On the CloudTable client, run a command to query the data migrated to CloudTable using CDM.

The following is an example of the command. Replace *table_name* with the name of the table specified when a job is created in the CDM cluster.

```
scan 'table_name'
```

----End

3.4.2 Using the Import Tool to Import SequenceFile Data to an HBase Cluster

Use the Import tool to import the **SequenceFile** data file of HBase to HBase of CloudTable. The Import tool is a tool contained in the installation package of the client tool.

NOTE

The **SequenceFile** file is a data file exported from HBase by the Export tool.

Step 1 Prepare a server where a CloudTable HBase client is installed. For details about how to install a client, see **Connecting to an HBase Normal Cluster Using HBase Shell**.

Step 2 Upload the **SequenceFile** file directory to the server where the HBase client is located. Assume that the directory is **/tmp/sequencefile**.

Step 3 The Import tool must be executed on the shell interface of the operating system of the client host. Go to the HBase directory on the client host and run the Import tool. The command format of the Import tool is as follows:

./bin/hbase org.apache.hadoop.hbase.mapreduce.Import <TableName> <InputPath>

TableName: Name of the table to which data is to be imported

InputPath: Directory of the **SequenceFile** file In the local file system, the directory must start with file://.

Example:

```
./bin/hbase org.apache.hadoop.hbase.mapreduce.Import t1 file:///tmp/sequencefile
```

----End

3.4.3 Using CopyTable to Import Source Data to an HBase Cluster

CopyTable is a utility provided by HBase. It can copy part or of all of a table, either to the same cluster or another cluster. The target table must exist first. The CloudTable client tool includes CopyTable. After deploying the client tool, you can use CopyTable to import data to a CloudTable cluster.

Using CopyTable to Import Data

Step 1 Prepare a Linux ECS as the client host and deploy the CloudTable HBase client tool on it.

For details, see **Connecting to an HBase Normal Cluster Using HBase Shell**.

When deploying the client tool, set the ZK link to the access address (Intranet) of the CloudTable HBase cluster where the source table resides.

Step 2 (Optional) If you want to copy a table to another cluster, obtain the access address (Intranet) of the target CloudTable HBase cluster.

Choose **Cluster Management** in the navigation pane. In the cluster list, locate the required cluster and obtain the address in the **Access Address (Intranet)** column.

Step 3 Before using CopyTable to copy table data, ensure that the target table exists in the target CloudTable HBase cluster. If the target table does not exist, create it first.

For details about how to create a table, see **Getting Started with HBase**.

Step 4 On the client host, open the CLI, access the **hbase** directory in the installation directory of the client tool, and run the CopyTable command to import data to the CloudTable cluster.

The following is an example of the command. In this example, the data in the specified 1 hour in **TestTable** is copied to the target cluster.

```
cd ${Installation directory of the client tool}/hbase
./bin/hbase org.apache.hadoop.hbase.mapreduce.CopyTable --starttime=1265875194289 --
endtime=1265878794289 --peer.adr=${ZK link of the target CloudTable cluster}:/hbase --
families=myOldCf:myNewCf,cf2,cf3 TestTable
```

----End

Overview of the CopyTable Command

The CopyTable command format is as follows:

```
CopyTable [general options] [--starttime=X] [--endtime=Y] [--new.name=NEW] [--peer.adr=ADR]
<tablename>
```

For details about the CopyTable command, see **CopyTable**.

The following provides description about common options:

- **startrow**: the start row
- **stoprow**: the stop row

- **starttime**: beginning of the time range (unixtime in milliseconds). If **endtime** is not specified, it implies that the duration extends from the start time indefinitely.
- **endtime**: end of the time range. If no starttime is specified, ignore it.
- **versions**: number of cell versions to be copied
- **new.name**: name of a new table
- **peer.adr**: Address of the target cluster. The format is **hbase.zookeeper.quorum:hbase.zookeeper.client.port:zookeeper.znode.paren**. For the HBase clusters, the parameter value is **${ZK link of the target CloudTable cluster}:/hbase**.
- **families**: List of column families to be copied. Multiple column families are separated by commas (,).

  If you want to copy from **sourceCfName** to **destCfName**, specify **sourceCfName:destCfName**.

  If the column family name needs to remain unchanged after copying, you only need to specify **cfName**.
- **all.cells**: Deletion markers and the deleted cells are also copied.

The parameter description is as follows:

tablename: name of the table to be copied

3.5 HBase Enterprise-class Enhancement

3.5.1 Configuring HBase Cold and Hot Data Separation

3.5.1.1 Overview

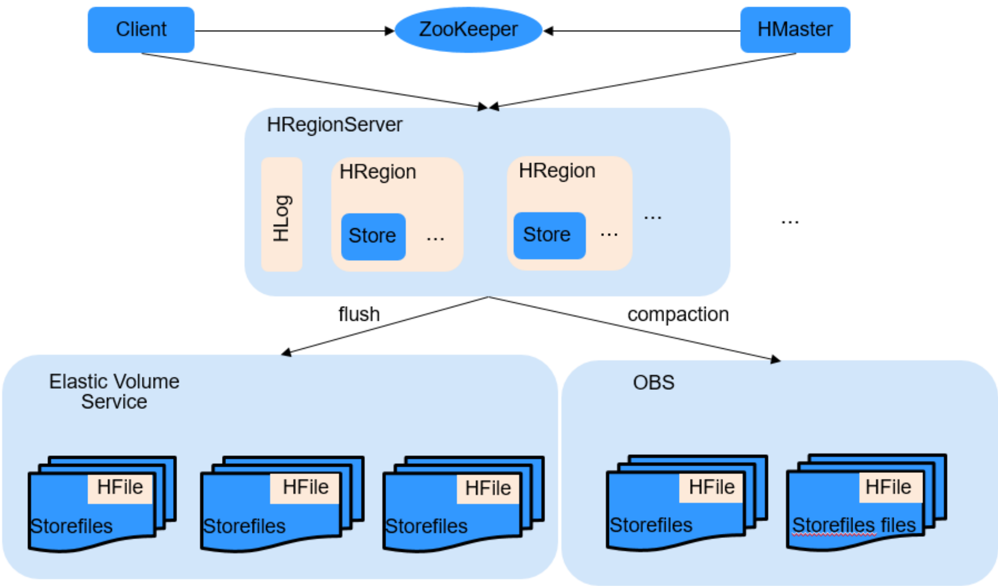
In a big data scenario, business data such as order data or monitoring data grows over time. As your business develops, rarely used data is archived. Enterprises may want to use cost-effective storage to store this type of data to reduce costs. CloudTable HBase supports cold and hot data separation. Data is classified and stored on different media to save costs without sacrificing functions. In data application scenarios, data can be classified into hot data and cold data based on the access frequency. Hot and cold data is classified based on the data access frequency and update frequency.

- Hot data: Data that is frequently accessed and updated and requires fast response.
- Cold data: Data that cannot be updated or is seldom accessed and does not require fast response

Principles

CloudTable HBase determines whether the data that is written to a table is cold data based on the timestamp of the data and the time boundary that is specified by users. The timestamp is in milliseconds. New data is stored in the hot storage and is gradually moved to the cold storage over time. You can change the time boundary for separating cold and hot data based on your business requirements. Data can be moved from the cold storage to the hot storage or vice versa.

Figure 3-10 Working principles



Constraints

- With low read IOPS performance, common I/O is only suitable for low-frequency queries.
- It is not a good choice to use common I/O when there is a large number of concurrent read requests. Otherwise, exceptions may occur.
- The cold and hot data separation feature supports only HBase 2.4.14 and later versions.
- In the cold and hot data separation scenario, hot disks do not support scale-in.
- Common I/O clusters do not support cold and hot data separation.
- HBase cold and hot data separation can only be configured during cluster creation.

3.5.1.2 Configuring HBase Cold and Hot Data Separation Using HBase Shell

HBase supports cold and hot data separation. Cold and hot data can be stored in different media, improving data query efficiency and reducing data storage costs. This section describes how to configure HBase cold and hot data separation using HBase Shell.

Prerequisites

- You have created an HBase cluster by referring to **Creating an HBase Cluster**.
- You have installed an HBase client.

Step 1: Enabling HBase Cold and Hot Data Separation

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Buy Cluster** in the upper right corner.

Step 4 On the **Buy Cluster** page, set **Database Engine** to **HBase** and select **Enable Hot/Cold** in **Advanced Feature**. The cold and hot separation feature is enabled for the created cluster.

Figure 3-11 Enabling cold and hot data separation

----End

Step 2: Setting the Cold and Hot Data Separation Boundary

Step 1 Connect to the HBase cluster. For details, see **Connecting to an HBase Normal Cluster Using HBase Shell**.

Step 2 Set the time boundary for separating hot and cold data. The time boundary must be longer than the major compaction execution period. The default execution period of major compactions is seven days.

- Create a table that separately stores cold and hot data.
```
hbase(main):002:0> create 'hot_cold_table', {NAME=>'f', COLD_BOUNDARY=>'86400'}
```
  Parameter description:
  - **NAME** indicates the column family that requires cold and hot separation.
  - **COLD_BOUNDARY** specifies the time boundary for separating cold and hot data. The time boundary is measured in seconds. For example, if **COLD_BOUNDARY** is set to **86400**, new data is archived as cold data after 86,400 seconds, which is equal to one day.
- Disable cold and hot data separation.
```
hbase(main):004:0> alter 'hot_cold_table', {NAME=>'f', COLD_BOUNDARY=>""}
```
- Enable cold and hot data separation for an existing table or change the time boundary. The time boundary is measured in seconds.
```
hbase(main):005:0> alter 'hot_cold_table', {NAME=>'f', COLD_BOUNDARY=>'86400'}
```

Step 3 Check whether the cold and hot separation is enabled or modified successfully.
```
hbase(main):002:0> desc 'hot_cold_table'
Table hot_cold_table is ENABLED
hot_cold_table
COLUMN FAMILIES DESCRIPTION
{NAME => 'f', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL
=> 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY =>
'false', COMPRE
SSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536', METADATA => {'COLD_BOUNDARY' =>
'86400'}}
1 row(s)
Quota is disabled
Took 0.0339 seconds
```

----End

Step 3: Inserting Data

You can write data to a table that separately stores cold and hot data in a similar manner that you write data to a standard table. When the data is written to a table, new data is stored in the hot storage (EVS disks). If the storage duration of the data exceeds the value specified by the **COLD_BOUNDARY** parameter, the system automatically moves the data to the cold storage (OBS) during the major compaction process.

Run the **put** command to insert a piece of data record to the specified table. You need to specify the table name, primary key, customized column, and inserted value.

```
hbase(main):004:0> put 'hot_cold_table','row1','cf:a','value1'
0 row(s) in 0.2720 seconds
```

The following describes parameters in the command:

- **hot_cold_table**: table name
- **row1**: primary key
- **cf: a**: customized column
- **value1**: inserted value

Step 4: Querying Data

CloudTable HBase allows you to use a table to store cold and hot data. You can query data only from one table. You can configure **TimeRange** to specify the time range of the data that you want to query. The system automatically determines whether the target data is hot or cold based on the time range that you specify and choose the optimal query mode. If the time range is not specified during the query, cold data will be queried. The throughput of reading cold data is lower than the throughput of reading hot data.

The cold storage is used only to archive data that is rarely accessed. If your cluster receives a large number of queries that hit cold data, you can check whether the time boundary (**COLD_BOUNDARY**) is set to an appropriate value. The query performance deteriorates if data that is frequently accessed are stored in the cold storage.

If you update a field in a row that is stored in the cold storage, the field is moved to the hot storage after the update. When this row is hit by a query that carries the **HOT_ONLY** hint or has a time range that is configured to hit hot data, only the updated field in the hot storage is returned. If you want the system to return the entire row, you must delete the **HOT_ONLY** hint from the query statement or make sure that the specified time range covers the time period from when this row is inserted to when this row is last updated. It is recommended that you do not update data that is stored in the cold storage.

- Random queries with Get
  - Do not specify **HOT_ONLY** to query data. In this case, data in cold storage is queried.
    ```
    hbase(main):001:0> get 'hot_cold_table', 'row1'
    ```
  - Specify **HOT_ONLY** to query data. In this case, only data in hot storage is queried.
    ```
    hbase(main):002:0> get 'hot_cold_table', 'row1', {HOT_ONLY=>true}
    ```


  – Query data within a time range that is specified by the **TIMERANGE** parameter. The system determines whether the query hits cold or hot data based on the values of the **TIMERANGE** and **COLD_BOUNDARY** parameters.
```
hbase(main):003:0> get 'hot_cold_table', 'row1', {TIMERANGE => [0, 1568203111265]}
```

    NOTE

    TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

- Range query scan
  – Do not specify **HOT_ONLY** to query data. In this case, data in cold storage is queried.
```
hbase(main):001:0> scan 'hot_cold_table', {STARTROW =>'row1', STOPROW=>'row9'}
```
  – Specify **HOT_ONLY** to query data. In this case, only data in hot storage is queried.
```
hbase(main):002:0> scan 'hot_cold_table', {STARTROW =>'row1', STOPROW=>'row9',
HOT_ONLY=>true}
```
  – Query data within a time range that is specified by the **TIMERANGE** parameter. The system determines whether the query hits cold or hot data based on the values of the **TIMERANGE** and **COLD_BOUNDARY** parameters.
```
hbase(main):003:0> scan 'hot_cold_table', {STARTROW =>'row1', STOPROW=>'row9',
TIMERANGE => [0, 1568203111265]}
```

    NOTE

    TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

- Prioritizing hot data selection

  CloudTable may look up cold and hot data for SCAN queries, for example, queries that are submitted to search all records of a customer. The query results are paginated based on the timestamps of the data in descending order. In most cases, hot data appears before cold data. If the SCAN queries do not carry the **HOT_ONLY** hint, CloudTable must scan cold and hot data. As a result, the query response time increases. When hot data query is prioritized, CloudTable will preferentially retrieve data from hot storage. Cold storage data is only queried if the number of rows in hot storage falls below the specified minimum query threshold. In this way, the frequency of cold data access is minimized and the response time is reduced.
```
hbase(main):001:0> scan 'hot_cold_table', {STARTROW =>'row1',
STOPROW=>'row9',COLD_HOT_MERGE=>true}
```
- Major compaction
  – Merge hot data areas of all partitions in a table.
```
hbase(main):002:0> major_compact 'hot_cold_table', nil, 'NORMAL', 'HOT'
```
  – Merge cold data areas of all partitions in a table.
```
hbase(main):002:0> major_compact 'hot_cold_table', nil, 'NORMAL', 'COLD'
```
  – Merge hot and cold data areas of all partitions in a table.
```
hbase(main):002:0> major_compact 'hot_cold_table', nil, 'NORMAL', 'ALL'
```

3.5.1.3 Configuring HBase Cold and Hot Data Separation Using Java APIs

HBase supports cold and hot data separation. Cold and hot data can be stored in different media, improving data query efficiency and reducing data storage costs.

This section describes how to configure HBase cold and hot data separation using JAVA APIs.

Prerequisites

- You have created an HBase cluster by referring to **Creating an HBase Cluster**.
- You have installed an HBase client.

Step 1: Enabling HBase Cold and Hot Data Separation

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Buy Cluster** in the upper right corner.

Step 4 On the **Buy Cluster** page, set **Database Engine** to **HBase** and select **Enable Hot/Cold** in **Advanced Feature**. The cold and hot separation feature is enabled for the created cluster.

Figure 3-12 Enabling cold and hot data separation

----End

Step 2: Specifying a Time Boundary for a Table

Step 1 Access the HBase cluster through the APIs described in "HBase Application Development Guide" > "Developing HBase Applications" in the *Developer Guide*.

Step 2 Set the time boundary for separating hot and cold data.

- Create a table that separately stores cold and hot data. **COLD_BOUNDARY** specifies the time boundary for separating cold and hot data. The time boundary is measured in seconds. In this example, new data is archived as cold data after one day.

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
HTableDescriptor descriptor = new HTableDescriptor(tableName);
HColumnDescriptor cf = new HColumnDescriptor("f");
cf.setValue(HColumnDescriptor.COLD_BOUNDARY, "86400");
descriptor.addFamily(cf);
admin.createTable(descriptor);
```

- Disable cold and hot data separation.

```
HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
cf.setValue(HColumnDescriptor.COLD_BOUNDARY, null);
admin.modifyTable(tableName, descriptor);
```

- Enable cold and hot data separation for an existing table or change the time boundary. **COLD_BOUNDARY** specifies the time boundary for separating cold and hot data. The time boundary is measured in seconds. In this example, new data is archived as cold data after one day.

```
HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
```

```
cf.setValue(HColumnDescriptor.COLD_BOUNDARY, "86400");
admin.modifyTable(tableName, descriptor);
```

----End

Step 3: Inserting Data

You can write data to a table that separately stores cold and hot data in a similar manner that you write data to a standard table. When the data is written to a table, new data is stored in the hot storage (EVS disks). If the storage duration of the data exceeds the value specified by the **COLD_BOUNDARY** parameter, the system automatically moves the data to the cold storage (OBS) during the major compaction process.

Write data using the the method described in "HBase Application Development Guide > Developing HBase Applications > Sample Code Description > Inserting Data" in the *Developer Guide*.

Step 4: Querying Data

CloudTable HBase allows you to use a table to store cold and hot data. You can query data only from one table. You can configure **TimeRange** to specify the time range of the data that you want to query. The system automatically determines whether the target data is hot or cold based on the time range that you specify and choose the optimal query mode. If the time range is not specified during the query, cold data will be queried. The throughput of reading cold data is lower than the throughput of reading hot data.

The cold storage is used only to archive data that is rarely accessed. If your cluster receives a large number of queries that hit cold data, you can check whether the time boundary (**COLD_BOUNDARY**) is set to an appropriate value. The query performance deteriorates if data that is frequently accessed are stored in the cold storage.

If you update a field in a row that is stored in the cold storage, the field is moved to the hot storage after the update. When this row is hit by a query that carries the **HOT_ONLY** hint or has a time range that is configured to hit hot data, only the updated field in the hot storage is returned. If you want the system to return the entire row, you must delete the **HOT_ONLY** hint from the query statement or make sure that the specified time range covers the time period from when this row is inserted to when this row is last updated. It is recommended that you do not update data that is stored in the cold storage.

- Get
  - The query that does not contain the **HOT_ONLY** hint may hit cold data.
    ```
    Get get = new Get("row1".getBytes());
    ```
  - The query that contains the **HOT_ONLY** hint hits only hot data.
    ```
    Get get = new Get("row1".getBytes());
    get.setAttribute(HBaseConstants.HOT_ONLY, Bytes.toBytes(true));
    ```
  - Query data within a time range that is specified by the **TimeRange** parameter. The system determines whether the query hits cold or hot data based on the values of the **TimeRange** and **COLD_BOUNDARY** parameters.
    ```
    Get get = new Get("row1".getBytes());
    get.setTimeRange(0, 1568203111265)
    ```

TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.

- Range query scan
  - The query that does not contain the **HOT_ONLY** hint may hit cold data.
```
TableName tableName = TableName.valueOf("chsTable");
Table table = connection.getTable(tableName);
Scan scan = new Scan();
ResultScanner scanner = table.getScanner(scan);
```
  - The query that contains the **HOT_ONLY** hint hits only hot data.
```
Scan scan = new Scan();
scan.setAttribute(HBaseConstants.HOT_ONLY, Bytes.toBytes(true));
```
  - Query data within a time range that is specified by the **TimeRange** parameter. The system determines whether the query hits cold or hot data based on the values of the **TimeRange** and **COLD_BOUNDARY** parameters.
```
Scan scan = new Scan();
scan.setTimeRange(0, 1568203111265);
```
    TimeRange specifies the query time range. The time in the range is a UNIX timestamp, which is the number of milliseconds that have elapsed since the Unix epoch.
- Prioritizing hot data selection

  CloudTable may look up to cold and hot data for SCAN queries, for example, queries that are submitted to search all records of a customer. The query results are paginated based on the timestamps of the data in descending order. In most cases, hot data appears before cold data. If the SCAN queries do not carry the **HOT_ONLY** hint, CloudTable must scan cold and hot data. As a result, the query response time increases. If you prioritize hot data selection, CloudTable preferentially queries hot data and cold data is queried only if you want to view more query results. In this way, the frequency of cold data access is minimized and the response time is reduced.
```
TableName tableName = TableName.valueOf("hot_cold_table");
Table table = connection.getTable(tableName);
Scan scan = new Scan();
scan.setAttribute(HBaseConstants.COLD_HOT_MERGE, Bytes.toBytes(true));
scanner = table.getScanner(scan);
```
- Major compaction
  - Merge hot data areas of all partitions in a table.
```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL, CompactionScopeType.HOT);
```
  - Merge cold data areas of all partitions in a table.
```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL, CompactionScopeType.COLD);
```
  - Merge hot and cold data areas of all partitions in a table.
```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("hot_cold_table");
admin. majorCompact (tableName,null, CompactType.NORMAL, CompactionScopeType.ALL);
```

3.5.2 Configuring an HBase Global Secondary Index

3.5.2.1 About HBase Global Secondary Indexes

Scenario

HBase secondary indexes can accelerate conditional queries with filters. There are local secondary indexes (LSIs, also called HIndexes) and global secondary indexes (GSIs). Compared with LSIs, GSIs have better query performance and are suitable for scenarios that require low read latency.

HBase GSIs use independent index tables to store index data. When a given query condition hits an index, a full table query is converted into an exact range query on an index table. This way, query speed is greatly improved. You do not need to modify your application code to enable HBase GSIs.

Key features of HBase GSIs are as follows:

- **Composite index**

  Multiple columns of different column families can be specified as index columns.

- **Covering index**

  Multiple columns or column families can be stored in the index table in redundancy to cover all data needed for a query. With covering indexes, you can quickly query non-index columns in index query.

- **Index TTL**

  Index table TTL takes effect if data table TTL is enabled. To ensure consistency with the data table, the index table TTL is automatically inherited from the index column and the column to overwrite an index of the data table and cannot be specified.

- **Online index change**

  Indexes can be created, deleted, and their status can be modified online without affecting data table read and write.

- **Online index repair**

  If the index data hit by a query is invalid, index data rebuilding is triggered to ensure that the final query result is correct.

- **Index tool**

  The index tool helps you to check consistency, repair, create, and delete indexes, modify index status, and rebuild index data.

Constraints

- Application Scenarios
  - GSIs cannot be used together with HIndexes (LSIs). That is, they cannot be created in the same data table.
  - Index tables do not support DR.
  - DISABLE, DROP, MODIFY, and TRUNCATE cannot be directly performed on index tables.
  - Index definition cannot be modified. You need to delete definitions and create indexes again. Other DDL operations on indexes are allowed, for example, modify index status, and delete and create indexes.

- HBase GSIs cannot be created for a table that contains data.

- Creating Indexes
  - An index name must comply with the regular expression requirements and does not support other characters. The following regular expression is supported: **[a-zA-Z_0-9-.]:**
  - The data table specified for index creation must exist. An index cannot be repeatedly created.
  - The index table cannot have multiple versions.

    Indexes cannot be created on data tables with multiple versions (**VERSION>1**). The **VERSION=1** setting is a must.
  - The number of indexes in a single data table cannot exceed five.

    Do not create too many indexes for a data table. Otherwise, bigger storage is required and write operations become slow. If more than five indexes need to be created, add the **hbase.gsi.max.index.count.per.table** parameter to the custom configuration **hbase.hmaster.config.expandor** of HMaster and set the parameter to a value greater than **5**. Restart HMaster to make the configuration take effect.
  - The index name can contain a maximum of 18 characters.

    Do not use long index names. If you have to, add the **hbase.gsi.max.index.name.length** parameter to the custom configuration **hbase.hmaster.config.expandor** of HMaster, set the parameter to a value greater than **18**, and restart HMaster to make the configuration take effect.
  - Indexes cannot be created for index tables.

    Indexes cannot be nested. Index tables are used only to accelerate queries and do not provide data table functions.
  - Indexes that can be covered by existing indexes cannot be created.

    If indexes you want to create are a subset of the existing indexes, they cannot be created. Duplicate indexes waste storage space. In the following example, index 2 cannot be created:

    Create a table.

    ```
    create 't1','cf1'
    ```

    Create index 1.

    ```
    hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -
    Dtablename.to.index='t1' -Dindexspecs.to.add='idx1=>cf1:[q1],[q2]'
    ```

    Create index 2.

    ```
    hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -
    Dtablename.to.index='t1' -Dindexspecs.to.add='idx2=>cf1:[q1]'
    ```
  - Indexes with the same name cannot be created in the same data table, but can be created in different data tables.
  - The TTL of a column family in an index table is inherited from the original table, and must be the same as that of the original table.

    The TTLs of all column families in an index table are the same and are inherited from a data table. The TTLs of associated column families in the data table must be the same. Otherwise, associated indexes cannot be created.
  - Properties of user-defined index tables are not supported.

3.5.2.2 Creating an HBase GSI

Scenario

For a table that does not have indexes, this tool allows you to add and create indexes.

Usage

Run the following command on the HBase client to add or create indexes to a table (the added or created indexes will be in the ACTIVE state):

```
hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -
Dtablename.to.index='table' -Dindexspecs.to.add='idx1=>cf1:[c1->string],[c2]#idx2=>cf2:[c1->string],
[c2]#idx3=>cf1:[c1];cf2:[c1]' -Dindexspecs.covered.family.to.add='idx2=>cf1' -
Dindexspecs.covered.to.add='idx1=>cf1:[c3],[c4]' -Dindexspecs.coveredallcolumn.to.add='idx3=>true' -
Dindexspecs.splitkeys.to.set='idx1=>[\x010,\x011,\x012]#idx2=>[\x01a,\x01b,\x01c]#idx3=>[\x01d,\x01e,\x0
1f]'
```

The parameters are described as follows:

- **tablename.to.index**: name of the data table for which an index is created
- **indexspecs.to.add**: mapping between the index name and the index column in the data table (definition of index column)
- (Optional) **indexspecs.covered.to.add**: column of the data table that is redundantly stored in an index table (definition of covering index column)
- (Optional) **indexspecs.covered.family.to.add**: column family of the data table that is redundantly stored in an index table (definition of covering index column family)
- (Optional) **indexspecs.coveredallcolumn.to.add**: all data in a data table that is redundantly stored in an index table (definition of all covering index columns)
- (Optional) **indexspecs.splitkeys.to.set**: pre-partition split keys of an index table. **Specify this parameter** in case hotspotting occurs in the region of the index table. You can configure pre-partitioning using the following characters:
  - **'#'** separates indexes.
  - **'[]'** contains **splitkeys**.
  - **','** separates **splitkeys**.

    NOTE

    Each **splitkey** set for per-partitioning must start with **\x01**.
- **indexspecs.to.addandbuild** (optional): Index data will be generated during data table creation. **If the data table is large, do not enable this parameter.** Use an index data generation tool instead.

  The parameters in the preceding command are described as follows:
  - **idx1**, **idx2**, and **idx3** are index names.
  - **cf1** and **cf2** are column family names.
  - **c1**, **c2**, **c3**, and **c4** are column names.
  - **string** indicates a data type. The value can be **STRING**, **INTEGER**, **FLOAT**, **LONG**, **DOUBLE**, **SHORT**, **BYTE**, or **CHAR**.

NOTE

- **'#'** separates indexes, **';'** separates column families, and **';'** separates column qualifiers.
- The column name and its data type must be included in **'[]'**.
- Column names and their data types are separated by **' - >'**.
- If the data type of a column is not specified, the default data type (**string**) will be used.

3.5.2.3 Querying an HBase GSI

Querying an HBase GSI

You can use the GSI tool to view the definition and status of indexes of a data table in batches.

Run the following command on the HBase client to view the definition and status of an index:

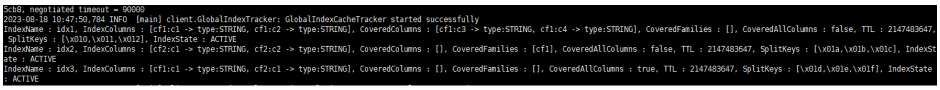
```
hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.show='table'
```

The related parameters are described as follows:

tablename.to.show: indicates the name of the table where the index to be queried is.

Figure 3-13 shows the query result. The index column definition, covering column definition, TTL, pre-partition information, and index status are displayed.

Figure 3-13 Index query result



Querying Data in an HBase Table with Indexes

You can use **SingleColumnValueFilter** to query data in a table with indexes. When the query condition hits an index, the query speed is much faster than that of an ordinary table query.

Typical index query conditions are as follows:

- Query by multiple AND conditions
  - When the columns used for a query contain at least the first indexed column, the query performance is optimized.

    For example, create a composite index for C1, C2, and C3.

    The index takes effect in the following queries:

    ```
    Filter_Condition(IndexCol1) AND Filter_Condition(IndexCol2) AND
    Filter_Condition(IndexCol30)
    Filter_Condition(IndexCol1) AND Filter_Condition(IndexCol2)
    Filter_Condition(IndexCol1) AND Filter_Condition(IndexCol3)
    Filter_Condition(IndexCol1)
    ```

    The index does not take effect in the following queries:


```
Filter_Condition(IndexCol2) AND Filter_Condition(IndexCol3)
Filter_Condition(IndexCol2)
Filter_Condition(IndexCol3)
```

  - When you use "Index Column AND Non-Index Column" as a query condition, the index can improve query performance. If a non-index column hits a covering column, the query performance is optimal. If a non-index column needs to be frequently queried, you are advised to define it as a covering column. Example:

```
Filter_Condition(IndexCol1) AND Filter_Condition(NonIndexCol1)
Filter_Condition(IndexCol1) AND Filter_Condition(IndexCol2) AND
Filter_Condition(NonIndexCol1)
```

  - When multiple columns are used for query, you can specify a value range for only the last column in the composite index and set other columns to specified values.

    Assume that a composite index is created for C1, C2, and C3. In a range query, only the value range of C3 can be set. The search criteria are "C1 = *XXX*, C2 = *XXX*, and C3 = Value range."

- Query by multiple OR conditions

  Assume that a composite index is created for C1, C2, and C3.

  - If only the first field in the index column is searched (range filtering is supported), indexing improves the query performance.

```
Filter_Condition(IndexCol1) OR Filter_Condition(IndexCol1) OR Filter_Condition(IndexCol1)
```

  - When non-index and non-index columns are searched, indexes cannot be hit, and query performance is not improved.

```
Filter_Condition(IndexCol1) OR Filter_Condition(NonIndexCol1)
```

  - During a combined query, if the outermost layer contains the OR condition, the index cannot be hit, and the query performance is not improved.

```
Filter_Condition(IndexCol1) OR Filter_Condition(NonIndexCol1)
(Filter_Condition(IndexCol1) AND Filter_Condition(IndexCol2)) OR
(Filter_Condition(NonIndexCol1))
```

NOTE

Reduce the use of OR conditions, especially an OR condition used together with a range condition. Otherwise, large-scale data is queried in slow speed when indexes are hit.

3.5.2.4 Changing the Status of HBase GSIs

Index Status

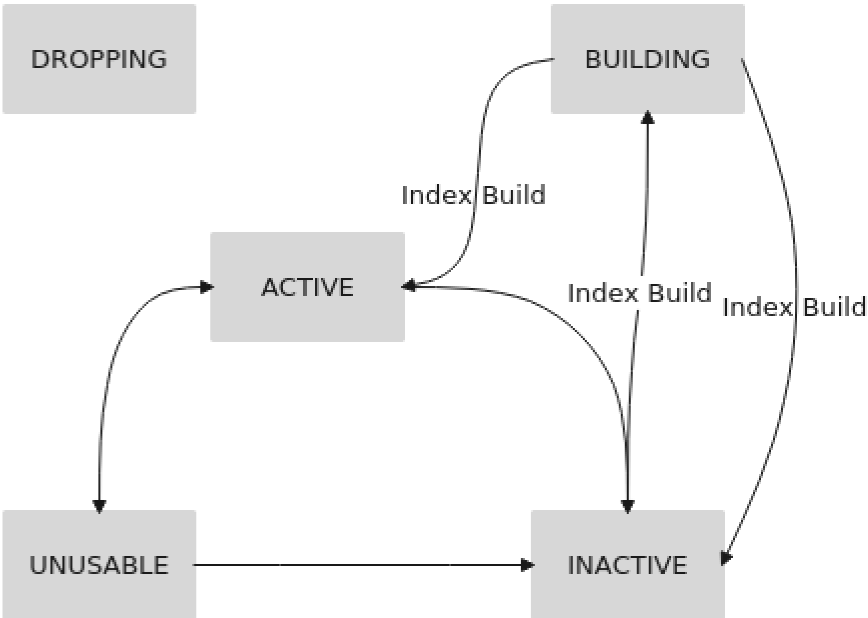
A GSI has the following states:

- **ACTIVE**: The index can be read and written normally.
- **UNUSABLE**: The index is disabled. Index data can be written normally but cannot be used for query.
- **INACTIVE**: The index is abnormal. The index data is inconsistent with that in the data table. The indexed data is skipped and the index cannot be used during data query.
- **BUILDING**: Index data is being generated in batches. After the generation is complete, the index is automatically switched to the **ACTIVE** state. In this state, data can be read and written properly.

- **DROPPING**: The index is being deleted. The indexed data is skipped, and the index cannot be used during data query.

You can change index status with the GSI tool. **Figure 3-14** describes the states and transitions between them.

Figure 3-14 State transitions



Scenario

You can use the GSI tool to disable or enable an index.

Usage

Run the following command on the HBase client to disable or enable an index:

```
hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -
Dtablename.to.index='table' -D[idx_state_opt]='idx1'
```

The related parameters are described as follows:

- **tablename.to.index**: indicates the name of the data table whose index status needs to be changed.
- **idx_state_opt**: indicates the target status of the index to be modified. The options are as follows:
  - **indexnames.to.inactive**: disables a specified index (**INACTIVE**).
  - **indexnames.to.active**: enables a specified index (**ACTIVE**).
  - **indexnames.to.unusable**: switches the specified index to **UNUSABLE**.

The following example changes the state of **idx1** of **table** to **UNUSABLE**:

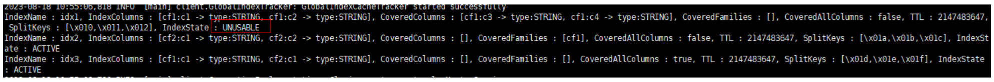
```
hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -
Dtablename.to.index='table' -Dindexnames.to.unusable='idx1'
```

After the command is executed, check the index information.

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.show='table'

As shown in **Figure 3-15**, the status of index **idx1** is changed.

Figure 3-15 idx1 status



3.5.2.5 Deleting GSIs

Scenario

You can use the GSI tool to delete an index.

Usage

Run the following command on the HBase client to delete an index:

hbase org.apache.hadoop.hbase.hindex.global.mapreduce.GlobalTableIndexer -Dtablename.to.index='table' **-Dindexnames.to.drop**='idx1#idx2'

The parameters are described as follows:

- **tablename.to.index**: indicates the name of the table where the index to be deleted is.
- **indexnames.to.drop**: indicates the name of the index to be deleted. You can specify multiple indexes and separate them with number signs (#).

3.6 Managing HBase Clusters

3.6.1 Checking the HBase Cluster Status

You can check the cluster status on the console. In the left navigation, click **Cluster Management**. On the **Cluster Management** page, a cluster list is displayed. If there are a large number of clusters, you can turn pages to view clusters in any state.

Clusters are listed in chronological order by default, with the latest cluster displayed at the top. **Table 3-6** and **Table 3-7** provides parameters and icons in the cluster list.

Table 3-6 Cluster management parameters

Parameter	Description
Cluster Name	Name of a cluster. Set this parameter when creating a cluster.
Cluster Status	Cluster status, including **Creating**, **In service**, **Sub-health**, **Frozen**, and **Creation failed**.
Task Status	Task status of a cluster For details, see **Table 3-9**.

Parameter	Description
Database Engine	HBase
Cluster Version	Cluster kernel version
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** ● You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. ● You can delete a user or multiple users. ● After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.
Created	Time when a cluster is created
Billing Mode	The cluster billing mode can be pay-per-use or yearly/monthly.
ZK Link (Intranet)	ZooKeeper address

Parameter	Description
Operation	● Click **Monitor** to access the CloudTable service monitoring page. ● **Node Scale Out**: Increase compute units in the cluster. For details, see **Scaling Out HBase Cluster Nodes**. ● **More** – **Disk Expansion**: Click **Disk Expansion** to expand the disk capacity. For details, see **Expanding the Disk Capacity of an HBase Cluster**. – **Specification Expansion**: Click **Specification Expansion** to expand the specifications. For details, see **Changing the Specifications of an HBase Cluster**. – Click **More** and then click **Change to Yearly/Monthly** in the **Operation** column to change the billing mode to yearly/monthly. – Click **More** > **Unsubscribe** in the **Operation** column to unsubscribe from the yearly/monthly cluster. – **Restart**: Click **Restart** to restart a cluster. For details, see **Restarting an HBase Cluster**. – **Delete**: You can click **Delete** to delete a cluster. For details, see **Deleting an HBase Cluster**. – **Enable Thrift Server**: Click **Enable Thrift Server** to enable Thrift Server for the cluster. – **Disable Thrift Server**: Click **Disable Thrift Server** to disable Thrift Server for the cluster.

Table 3-7 Icon description

Icon	Description
▼	Click ▼ to view all projects.
	Enter a cluster name in the search box and click  to search for the cluster.
	Click  to manually refresh the cluster list.

Cluster Status

Table 3-8 Cluster status description

Status	Description
Creating	Indicates that a cluster is being created.

Status	Description
In service	If a cluster is successfully created and can provide services, the cluster status is **In service**.
Sub-health	If the cluster status cannot be monitored within the specified time, the cluster status changes to **Sub-health**. Manual intervention is required to recover a cluster that is in **Sub-health** status. For example, you can restart the cluster to recover the cluster.
Creation failed	Indicates that a cluster fails to be created.
Frozen	If the balance is insufficient for renewing a cluster, the cluster status is **Frozen**. If a cluster status is **Frozen**, you need to renew your subscription and ensure that your account balance is not 0 before unfreezing the cluster. **NOTE** A frozen cluster is unavailable and its all ECSs are shut down. After being unfrozen, the cluster recovers to the **In service** state. If you do not renew the cluster before the freeze period ends, the cluster will be deleted.

Task Status

Table 3-9 Task status description

Status	Description
Deleting	Indicates that a cluster is being deleted.
Restarting	Indicates that a cluster is being restarted.
Enabling Thrift Server	Indicates that the Thrift Server is being enabled in the cluster.
Disabling Thrift Server	Indicates that the Thrift Server is being disabled in the cluster.
Scaling out	Indicates that cluster nodes are being scaled out.
Disk expanding	Indicates that the disk capacity of the cluster is being expanded.
Resizing Flavor	Indicates that the cluster specifications are being changed.

3.6.2 Viewing HBase Cluster Details

You can monitor and manage the clusters you have created. On the CloudTable console, click **Cluster Management**. In the cluster list, locate the cluster to be viewed and click the cluster name to access the basic information page. You can view the cluster information and network configurations.

Table 3-10 and **Table 3-11** describe parameters about basic cluster information.

Table 3-10 Cluster information

Parameter	Description
Cluster Name	Name of a cluster. Set this parameter when creating a cluster.
Cluster ID	Cluster ID
Cluster Status	Cluster status information
Used Storage Capacity (GB)	Used storage capacity
Billing Mode	Billing mode of the cluster
Version	Cluster kernel version
Created	Time when a cluster is created
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** • You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. • You can delete a user or multiple users. • After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.
ZK Link (Intranet)	ZooKeeper address You can click to copy the ZooKeeper link to the clipboard.
Used Hot Storage Capacity	Storage capacity of hot data if **Enable Hot/Cold** is selected in **Advanced Feature**.
Used Cold Storage Capacity	Storage capacity of cold data if **Enable Hot/Cold** is selected in **Advanced Feature**.
Enable Security Channel	You can check whether the security channel is enabled on the CloudTable console.
Enable Thrift Server	If **No** is displayed, Thrift Server is disabled. If **Yes** is displayed, Thrift Server is enabled and the IP address of Thrift Server is displayed.

Parameter	Description
Thrift Server IP	The visibility of this field on the cluster details page depends on the Thrift Server's status: it appears when enabled and is hidden when disabled.
Database Engine	The value is **HBase**.
Read-only Threshold	When data is written to the storage device after the cluster's storage usage reaches a specified threshold, the storage space may become fully utilized, potentially causing performance and functionality issues. To prevent this, it is essential to set a threshold for the cluster. Upon reaching this threshold, the cluster will switch to read-only mode, and an alarm will be triggered to prompt timely management of the cluster status. The default threshold is set at 85%, with an adjustable range from 70% to 90%. **NOTE** If the Core node storage specification uses common I/O, the read-only mode is not supported.
Read-only	"Yes" indicates that the read-only threshold is reached. "No" indicates that the read-only threshold is not reached.

Table 3-11 Network configuration

Parameter	Description
Region	Working area of the cluster. Set this parameter when creating a cluster.
AZ	AZ you select during cluster creation
VPC	VPC you select during cluster creation A Virtual Private Cloud (VPC) is a secure, isolated, logical network environment.
Subnet	Subnet you select during cluster creation A subnet provides dedicated network resources that are logically isolated from other networks, improving network security.
Security Group	Security group you select during cluster creation If a security group needs to be added or modified due to service changes, click the pen icon on the right of **Security Group** in the **Network Configuration** area on the cluster details page to add or modify the security group. **NOTE** Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.

Table 3-12 Master node configuration

Parameter	Description

Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node.
	NOTE
	● Available compute specifications:
	  ● 4U16G
	  ● 8U16G
	  ● 8U32G
	  ● 16U32G
	  ● 16U64G
	  ● 32U64G
	  ● 32U128G
	  ● 64U128G

Storage	**Ultra-high I/O** is recommended.
Nodes	Use the default value **2**.

Table 3-13 Core node configuration

Parameter	Description

Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node.
	NOTE
	● Available compute specifications:
	  ● 4U16G
	  ● 8U16G
	  ● 8U32G
	  ● 16U32G
	  ● 16U64G
	  ● 32U64G

Storage	Select the disk specifications and capacity of the HBase compute node.
	NOTE
	● Available storage specifications:
	  ● Common I/O
	  ● High I/O
	  ● Ultra-high I/O
	● The capacity ranges from 400 GB to 30,000 GB per node.

Parameter	Description
Nodes	Set the number of nodes in a cluster. The value ranges from **2** to **20**.

3.6.3 Restarting an HBase Cluster Node

If a CloudTable cluster node is abnormal, you can restart the node to restore the node status.

Precautions

- The node is unavailable during the restart.
- To minimize service disruption, schedule the node restart during off-peak hours.
- Disk scale-out, node scale-out, and specification expansion functionalities will be temporarily disabled during the node restart.
- Please note that a node restart is a process restart, not a full node reboot.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 Click the name of the cluster to be operated. The cluster details page is displayed.

Step 5 On the cluster details page, choose **Operation** > **Restart**. The **Restart Node** dialog box is displayed.

Step 6 Enter **RESTART** or click **Auto Enter**, and click OK to restart the node.

After the node is restarted, the **Restart** button is unavailable.

Step 7 Check the node restart result. If the restart is successful, the **Restart** button is highlighted. If the node fails to be restarted, the task status is **Failed to restart the node**, and the **Restart button** is highlighted, the node can be restarted again.

----End

Node Restart Statuses

Table 3-14 Restart statuses

Status	Description
Restart	The Restart button is highlighted, indicating either no restart operation has been initiated or that the restart has been completed successfully

Status	Description
Restarting	The cluster node is being restarted, and **Task Status** in **Cluster Information** is **Restarting**.
Restart failed	If a cluster node fails to be restarted and **Task Status** is **Failed to restart the node**, you can continue to restart the node.

3.6.4 Restarting an HBase Cluster

If a cluster is in the unbalanced state or cannot work properly, you may need to restart it for restoration. After modifying a cluster's configurations, such as security settings and parameters, manually restart the cluster to make the configurations take effect.

NOTE

- If your cluster is in arrears, this function may be unavailable. Please top up your account in time.
- The function is unavailable when the cluster status is subhealthy. Please contact technical support for assistance with restoring the cluster.

Impact on the System

- A cluster cannot provide services during the restart. Therefore, before the restart, ensure that no task is running and all data is saved.
- If a cluster is processing transactional data, for example, importing data, querying data, files may be damaged or the cluster may fail to be restarted once the cluster is restarted. You are advised to stop all cluster tasks before restarting a cluster.
- If the restart fails, the cluster may be unavailable. Try again later or contact technical support.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the navigation pane, click **Cluster Management**.

Step 4 In the upper right corner of the cluster list, enter the name of a cluster in the search box and click .

Step 5 In the **Operation** column of the cluster, click **Restart**.

Step 6 In the dialog box that is displayed, select the check box and click **OK** to restart the cluster.

----End

3.6.5 Deleting an HBase Cluster

If a cluster is no longer needed, you can delete, unsubscribe from, or release the cluster. Deleting, unsubscribing from, or releasing a CloudTable cluster will clear all resources and data related to the cluster. This operation cannot be undone. Exercise caution when deleting a cluster.

- Pay-per-use clusters can be directly deleted. For details, see **Deleting an HBase Pay-per-Use Cluster**.
- Yearly/monthly clusters cannot be directly deleted. You need to unsubscribe from them (if they have not expired) or release them (if they have expired but have not been renewed). For details, see **Unsubscribing from or Releasing a Yearly/Monthly Cluster**.

NOTE

- A cluster in the **Scaling out** state cannot be deleted. You need to delete it after the scale-out is complete.
- If your cluster is in arrears, this function may be unavailable. Please top up your account in time.

Deleting an HBase Pay-per-Use Cluster

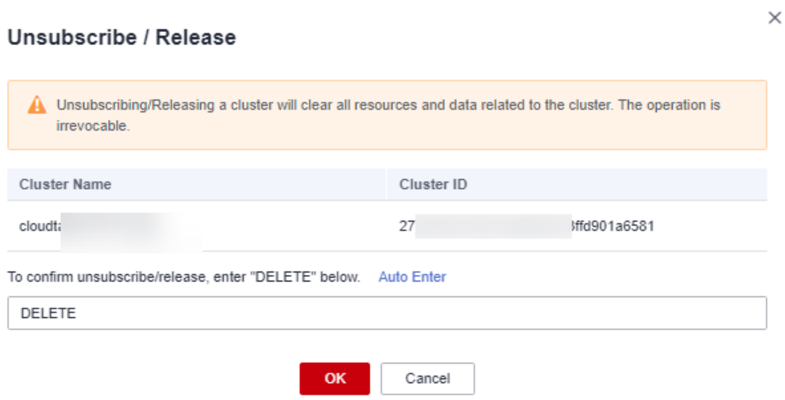
Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the **Operation** column of the cluster, choose **More** > **Delete**.

Step 4 In the displayed dialog box, enter **DELETE** or click **Auto Enter**, and click **OK** to delete the cluster.

Figure 3-16 Confirming the deletion



----End

Unsubscribing from or Releasing a Yearly/Monthly Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the **Operation** column of the cluster, choose **More** > **Unsubscribe/Release**.

Step 4 In the displayed dialog box, enter **DELETE** or click **Auto Enter**, and click **OK** to unsubscribe from or release the cluster.

Step 5 On the **Unsubscribe** page, confirm the cluster information, select reasons for unsubscription, and confirm the unsubscription amount and related fees.

Step 6 Select "I've backed up the data or confirmed that the unsubscribed resources are no longer needed." and "I understand that only resources in the recycle bin can be restored after unsubscription". View the recycle bin description and click **Unsubscribe**.

Step 7 Return to the console and check whether the cluster has been unsubscribed from or released.

----End

3.7 HBase Cluster O&M

3.7.1 Adjusting the Capacity of an HBase Cluster

3.7.1.1 Scaling Out HBase Cluster Nodes

You can perform a node scale-out to expand cluster capacity and improve system computing and storage capabilities. Scale-out will incur fees. You can dynamically add nodes in response to site needs or service conditions. The cluster seamlessly manages load balancing to maintain uninterrupted service and facilitate smooth expansion of capacity.

Precautions

- During the scale-out, the system does not automatically restart the cluster, ensuring service continuity.
- If the scale-out fails, **Scale-out failed** is displayed in the **Task Status** column, and the cluster automatically rolls back to the state before the scale-out. You can try again. If you have any questions, contact technical support.
- Only core nodes can be scaled-out.

Procedure

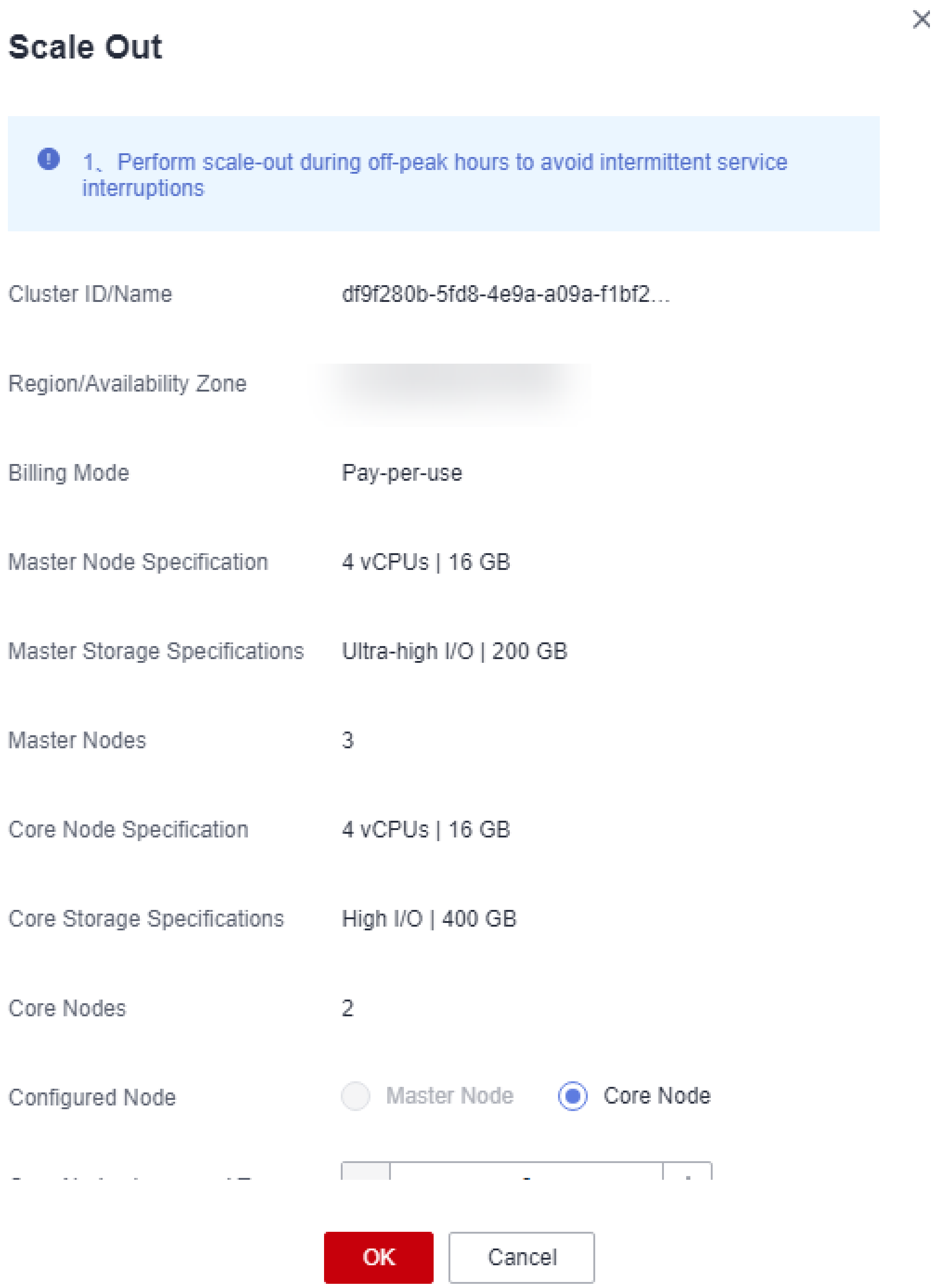
Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the navigation pane, click **Cluster Management**.

Step 4 In the cluster list, locate the row that contains the target cluster, and choose **More** > **Scale Out** in the **Operation** column. The **Scale Out** page is displayed.

Figure 3-17 Scaling out nodes



NOTE

You can also click **Scale Out** on the cluster details page to go to the **Scale Out** page.

Step 5 Set **Configured Node** to **Core Node**.

Step 6 Click **+** to add nodes.

Step 7 Confirm the fee and resource quota, and click **OK**.

----End

3.7.1.2 Expanding the Disk Capacity of an HBase Cluster

The vertical capacity expansion is performed by expanding disk capacity. Data is usually stored on the compute nodes, and disk expansion is required when the disk capacity of the compute nodes is insufficient.

Constraints

- When the disk usage of all core nodes reaches 90%, the cluster is abnormal and read and write operations are unavailable.
- You can perform disk capacity expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as node scale-out and parameter modification).
- During disk capacity expansion, cluster services may jitter.
- If the core node storage specification uses common I/O, disk capacity expansion is not supported.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management**.

Step 3 Select the cluster to be expanded and choose **Operation** > **More** > **Disk Expansion**.

> NOTE
>
> You can also click **Disk Expansion** on the cluster details page to go to the page for expanding disk capacity.

Step 4 Click **+** to increase the disk capacity, confirm the fee and resource quota, and click **OK**.

----End

Disk Expansion Status

Table 3-15 Disk expansion status description

Status	Description
Disk expanding	Indicates that a cluster is being scaled out.
In service (cluster status)	This status is shown in **Cluster Status**, indicating that the scale-out is complete and the cluster can provide services.
Disk capacity expansion failed (task status)	This status is shown in **Task Status**, indicating that the cluster fails to be scaled out.

3.7.1.3 Changing the Specifications of an HBase Cluster

Constraints

- The entire cluster becomes temporarily unavailable during the specification change. Subsequent specification changes are not permitted until the current operation completes.
- You can change the specifications of only one type of nodes at a time. After the change, nodes in other types still maintain their original specifications.
- If the data volume is large, it may take long to modify the node specifications. You are advised to increase the node specifications during off-peak hours.

Prerequisites

You can perform specification expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as node scale-out and disk expansion).

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 On the **Cluster Management** page, locate the target cluster and choose **More** > **Specification Expansion** in its Operation column.

NOTE

You can also click **Specification Expansion** on the cluster details page to switch to the **Specification Expansion** page.

Step 3 Select the target node type.

Step 4 Select the CPU and memory from the node specifications drop-down list.

Step 5 Confirm the fee and resource quota, and click **OK**.

After you click **OK**, the cluster status changes to **Sub-health** and the task status changes to **Resizing Flavor**. After the cluster specifications are changed, the cluster status changes to **In service** and the task status is cleared.

----End

Specification Expansion Status

Table 3-16 Specification expansion status description

Status	Description
Resizing Flavor	Indicates that the specifications of the target cluster are being changed.
In service	Indicates that the specifications of the target cluster are changed and the cluster can provide services.

Status	Description
Resizing flavor failed	Indicates that the specifications of the target cluster failed to be changed.

3.7.2 Modifying HBase Parameters to Optimize Cluster Performance

Scenario

A CloudTable HBase cluster has many HBase parameters related to read/write performance. You can modify the HBase parameters to tune cluster performance under different read/write request loads. You need to restart the cluster to make changes take effect.

Constraints

- Restart the CloudTable HBase cluster after parameter configuration modification. Otherwise, services will be interrupted.
- Do not modify cluster parameters when the CloudTable HBase cluster is being restarted.

Prerequisites

No task is running in the cluster.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the navigation pane, click **Cluster Management**.

Step 4 Click the name of a cluster for which you want to modify HBase parameters to access the cluster details page.

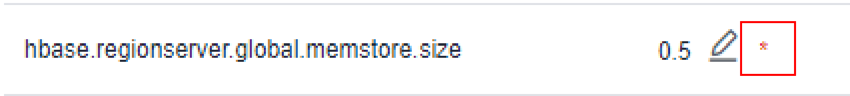
Step 5 In the **Parameter Configuration** area, click the **Parameter Configuration** tab to modify HBase parameters.

For details about the HBase parameters you can modify, see **HBase Parameters**.

1. Select the target parameter and click in the **Value** column.
2. Enter a new value in the text box, and then click . The parameter value has been successfully changed if the system prompts "The parameter changed to *xx* successfully. Save the modified value." The new parameter value is marked with a red asterisk (*).

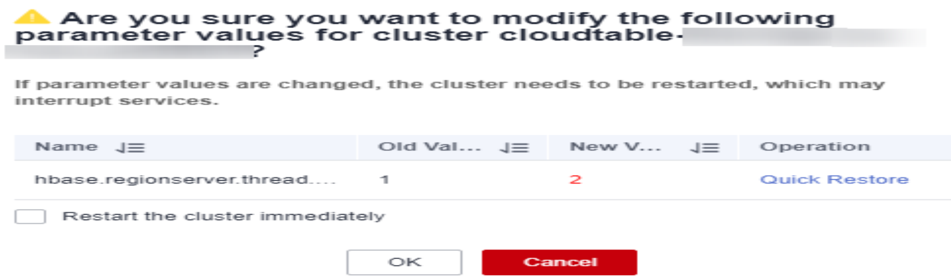
   If you want to cancel it, click .

Figure 3-18 Modifying HBase parameters



3. Click **Save Changes** in the upper left corner above the parameter list. A dialog box is displayed.
4. In the **Save Changes** dialog box, verify whether the parameter settings are correct. Select **Restart the cluster immediately** and click **OK**.

Figure 3-19 Saving the modification



– After you select **Restart the cluster immediately** and click **OK**, the cluster restarts immediately. After the cluster restart is complete, the new parameter configurations take effect. Click  above the parameter list. The system prompts "The new value is applied" in the upper right corner of the parameter list.

– If you do not select **Restart the cluster immediately** and click **OK** in the **Save Changes** dialog box, the system prompts "The new value is not applied" above the parameter list. In this case, you need to restart the cluster for the changes to take effect. For details about how to restart a cluster, see **Restarting an HBase Cluster**.

– If the new parameter value is incorrect, click **Quick Restore** to cancel the modification.

Step 6 After modifying parameters, you can click the **Change History** tab to view the change history.

On the **Change History** tab page, you can view the following information.

- **Name**: Name of the changed parameter.
- **Old Value**: old parameter value
- **New Value**: new parameter value
- **Modified**: time when you modify a parameter value

----End

HBase Parameters

Table 3-17 lists HBase parameters you can modify currently.

NOTE

The sum of the values of **hbase.regionserver.global.memstore.size** and **hfile.block.cache.size** cannot exceed 0.8.

Table 3-17 HBase parameters

Parameter	Value Range	Default Value	Description
hbase.regionserver.thread.compaction.small	[1,20]	1	Indicates the number of HFile compaction threads. You can increase the parameter value in heavy-put-load scenarios.
hbase.regionserver.global.memstore.size	(0,0.8)	0.4	It is recommended that you set this parameter to "hbase.hregion.memstore.flush.size x Number of regions with active writes/ RegionServer GC -Xmx". The default value is **0.4**, indicating that 40% of RegionServer GC -Xmx is used.
hbase.hstore.blockingStoreFiles	[1,2147483647]	60	When the HFile number in the column cluster reaches this threshold, all operations in the region are blocked until the compaction is complete. You can increase the parameter value in heavy-put-load scenarios.
hbase.client.scanner.timeout.period	[1,2147483647]	60000	A client and RegionServer parameter, indicating the scan lease period. It is recommended that you set this parameter to an integral multiple of 60,000 ms, and increase the parameter value in heavy-read-load scenarios. Unit: millisecond
hfile.block.cache.size	(0,0.8)	0.2	Indicates the data cache percentage in the RegionServer GC -Xmx. You can increase the parameter value in heavy-read-load scenarios, in order to improve cache hit ratio and performance. The default value is 0.2, indicating that 20% of RegionServer GC -Xmx is used.
hbase.regionserver.handler.count	[1,300]	100	Indicates the number of RPC server instances on the RegionServer. The recommended value ranges from 100 to 300.

Parameter	Value Range	Default Value	Description
hbase.regionserver.metahandler.count	[1,100]	50	Indicates the number of program instances for processing prioritized requests. The recommended value ranges from 20 to 100.
hbase.hstore.flusher.count	[1,10]	2	Indicates the number of memstore flush threads. You can increase the parameter value in heavy-put-load scenarios.
hbase.ipc.server.callqueue.read.ratio	[0,1]	0.5	When used under different load models, it controls the ratio between the numbers of read and write RPC queues. The value ranges from 0 to 1.0, and the default value is 0.5. ● 0 indicates the total RPC queues of read and write operations. ● If the value is less than 0.5, it indicates that the read load is less than the write load. ● 0.5 indicates that the read load equals to the write load. ● If the value is greater than 0.5, it indicates that the read load is greater than the write load. ● 1.0 indicates that all RPC queues except one are used for read operations.
hbase.regionserver.hotregion.handler.count	[1,65535]	66	Number of RPC listener instances started on RegionServers for hotspot regions.
hbase.ipc.server.hotregion.max.callqueue.length	[1,65535]	330	Maximum length of the queue for RegionServers to process requests of hotspot regions. Upon receiving a new request, the system checks whether the length of the queue exceeds the threshold. If so, the request is discarded.
hbase.metric.controller.analysis.period	[1,2147483647]	60	Hotspot analysis period of MetricController, in seconds.
hbase.metric.controller.analysis.threads.max	[1,100]	10	Maximum number of threads for hotspot analysis in the thread pool.

Parameter	Value Range	Default Value	Description
hbase.metric.controller.collect.threads.max	[1,100]	16	Maximum number of threads for hotspot analysis in the traffic collection thread pool.
hbase.metric.regionserver.hotspot.threshold	[1,2147483647]	20000	Hotspot threshold of a RegionServer. Unit: requests per second.
hbase.metric.region.hotspot.threshold	[1,2147483647]	10000	Hotspot threshold of a single region. Unit: requests per second.
hbase.hotspot.enable	[true,false]	true	Whether to enable hotspot self-healing. The value true means to enable it, and false means to disable it. After this function is enabled, access hotspots will be automatically processed.
hbase.tries.cache.enabled	[true,false]	false	If this parameter is set to true, LoudsTriesLruBlockCache is used to cache index blocks and data blocks.
hbase.write.tries	[true,false]	false	If this parameter is set to true, the succinct tries feature is enabled. In this case, a new data structure is used to improve the utilization of index blocks.
hbase.hfile.hsync	[true,false]	false	Specifies whether to enable the HFile durability to make data persistence on disks. If this parameter is set to true, the performance is affected because each Hfile file is synchronized to the disk.
hbase.wal.hsync	[true,false]	false	Specifies whether to enable WAL file durability to make the WAL data persistence on disks. If this parameter is set to **true**, the performance is affected because each WAL file is synchronized to the disk.
hbase.hot.cold.data.separator.chore.enabled	[true,false]	false	Specifies whether to enable the CDS function. If this parameter is set to **true**, asynchronous dump is enabled for cold data.

Parameter	Value Range	Default Value	Description
hbase.hot.cold.data.separator.chore.interval	[60000, 2147483647]	86400000	Specifies the CDS check period. Unlike common compaction tasks, compaction tasks triggered by CDS will write hot and cold data to separate data files. Unit: millisecond
hbase.regionserver.cold.data.shipper.chore.interval	[60000, 2147483647]	300000	Specifies the period for checking cold data files to be dumped. If there are cold data files that can be dumped, a task is triggered to migrate the cold data files to the cold storage. Unit: millisecond
hbase.regionserver.shipper.cold.file.verification.enabled	[true, false]	true	Specifies whether to check the consistency of data files migrated to the cold storage.
hbase.cold.data.shipper.thread.count	[1, 20]	3	Specifies the number of threads used to copy data files to the cold storage.
hbase.cold.data.separator.offpeak.end.hour	[-1, 23]	-1	Specifies the end time of the off-peak period. The value is an integer hour in 24-hour format. During off-peak hours, CDS uses more resources to speed up the process of changing hot data to cold data. If this parameter is set to **-1**, there is no off-peak hour.
hbase.cold.data.separator.offpeak.start.hour	[-1, 23]	-1	Specifies the start time of the off-peak period. The value is an integer hour in 24-hour format. During off-peak hours, CDS uses more resources to speed up the process of changing hot data to cold data. If this parameter is set to **-1**, there is no off-peak hour.
hbase.compaction.skip.unnecessary.cold.files.enabled	[true, false]	false	Specifies whether to skip cold data files during major compaction.
hbase.regionserver.thread.compaction.cold	[1, 20]	1	Specifies the number of threads for merging cold data files.

Parameter	Value Range	Default Value	Description
hbase.regionserver.compaction.cold.check.period	[60000, 2147483647]	30000	Specifies the period for checking whether cold files need to be merged. Unit: millisecond

3.7.3 Using Cloud Eye to Monitor HBase Clusters

3.7.3.1 HBase Cluster Monitoring Metrics

Description

Monitoring is critical to ensure CloudTable reliability, availability, and performance. You can monitor the running status of CloudTable servers.

This section describes the metrics that can be monitored by Cloud Eye as well as their namespaces and dimensions.

Namespace

SYS.CloudTable

CloudTable HBase HMaster Instance Monitoring Metrics

Table 3-18 CloudTable HBase HMaster instance monitoring metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
cmdForTotalMemory	Total Memory	Total memory size of the monitored object	> 0	Byte	1024 (IEC)	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
cmdProcessCPU	CPU Utilization	CPU utilization of the monitored object	0~100	%	N/A	CloudTable instance node	1 min
cmdProcessMem	Memory Utilization	Memory utilization of the monitored object	0~100	%	N/A	CloudTable instance node	1 min
hm_deadregionservernum	Faulty RegionServers	Number of faulty RegionServers in the cluster	≥ 0	Count	N/A	CloudTable instance node	1 min
hm_regionservernum	Normal RegionServers	Number of normal RegionServers in the cluster	≥ 0	Count	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
hm_ritCount	RIT Count	Number of regions in the Region In Transaction (RIT) state in the cluster where the monitored object is located	≥ 0	Count	N/A	CloudTable instance node	1 min
hm_ritCountOverThreshold	RIT Count Over Threshold	Number of regions in the RIT state and reach the threshold in the cluster where the monitored object is running	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_queuecalltime_max	RPC Queue Call Time (Max)	Maximum RPC queue call time	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_queuecalltime_mean	RPC Queue Call Time (Mean)	Mean RPC queue call time	≥ 0	ms	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
nn_percentallused	Disk Utilization Rate	Disk space usage of the cluster	0~100	%	N/A	CloudTable instance node	1 min
nn_capacityremaining	Disk capacity remaining of cluster	Remaining disk space of the cluster	Depends on the cluster disk capacity.	GB	N/A	CloudTable instance node	1 min
nn_capacityused	Disk capacity used of cluster	Disk space used in the cluster	Depends on the cluster disk capacity.	GB	N/A	CloudTable instance node	1 min
cmdForUsedStorageRate	Ratio of Used Storage Space	Ratio of the used storage space to the total storage space in the cluster	0~100	%	N/A	CloudTable instance node	1 min
network_throughput_inbound_rate	Inbound Throughput	Inbound data volume over network of each node per second	≥ 0	KB/s	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
network_throughput_outgoing_rate	Outbound Throughput	Outbound data volume over network of each node per second	≥ 0	KB/s	N/A	CloudTable instance node	1 min
disk_throughput_read_rate	Disk Read Throughput	Disk read throughput	≥ 0	Byte/s	1024 (IEC)	CloudTable instance node	1 min
disk_throughput_write_rate	Disk Write Throughput	Disk write throughput	≥ 0	Byte/s	1024 (IEC)	CloudTable instance node	1 min

NOTE

hmaster instances include **hmaster-standby** (standby) and **hmaster-active** (active). When **hmaster-active** becomes faulty, **hmaster-standby** becomes active to provide services.

In an HBase cluster, 10% of the disk space is reserved by default. Therefore, the disk alarm value is not equivalent to the raw disk usage percentage.

CloudTable HBase RegionServer Instance Monitoring Metrics

Table 3-19 lists the monitoring metrics supported by CloudTable HBase RegionServer instances.

Table 3-19 Monitored CloudTable metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
cmdProcessCPU	CPU Utilization	CPU utilization of the monitored object	0~100	%	N/A	CloudTable instance node	1 min
cmdForTotalMemory	Total Memory	Total memory size of the monitored object	> 0	Byte	1024 (IEC)	CloudTable instance node	1 min
cmdProcessMem	Memory Utilization	Memory utilization of the monitored object	0~100	%	N/A	CloudTable instance node	1 min
disk_throughput_write_rate	Disks Write Rate	Volume of data written to the monitored object per second	≥ 0	Byte/s	1024 (IEC)	CloudTable instance node	1 min
disk_throughput_read_rate	Disks Read Rate	Volume of data read from the monitored object per second	≥ 0	Byte/s	1024 (IEC)	CloudTable instance node	1 min
hm_regionservernum	Normal RegionServers	Number of normal RegionServers	≥ 0	Count	N/A	CloudTable instance node	1 min
hm_deadregionservernum	Faulty RegionServers	Number of faulty RegionServers	≥ 0	Count	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
hm_ritCountOverThreshold	RIT Count Over Threshold	Region in transaction count over threshold	≥ 0	Count	N/A	CloudTable instance node	1 min
hm_ritCount	RIT Count	Region in transaction count	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_requests	Requests Per Second	Number of requests of a RegionServer per second	≥ 0	requests/s	N/A	CloudTable instance node	1 min
rs_regions	Regions	Number of regions of a RegionServer	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_writerequestscount	Write Requests	Number of write requests of a RegionServer	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_readrequestscount	Read Requests	Number of read requests of a RegionServer	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_blockcachehitcachingratio	Hit Cache Block Caching Ratio	Block cache hit caching ratio	0~100	%	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
rs_blockCacheCountHitPercent	Hit Cache Block Ratio	Block cache hit ratio	0~100	%	N/A	CloudTable instance node	1 min
rs_getavgtime	Get Delay (Avg)	Average Get operation delay of the RegionServer per unit time	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_putavgtime	Put Delay (Avg)	Average Put operation delay of the RegionServer per unit time	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_deleteavgtime	Delete Delay (Avg)	Average Delete operation delay of the RegionServer per unit time	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_getnumops	Get Operations	Number of Get operations of the RegionServer per unit time	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_putnumops	Put Operations	Number of Put operations of the RegionServer per unit time	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_deletenumops	Delete Operations	Number of Delete operations of the RegionServer per unit time	≥ 0	Count	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
rs_queuecalltime_max	RPC Queue Call Time (Max)	Maximum RPC queue call time	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_queuecalltime_mean	RPC Queue Call Time (Mean)	Mean RPC queue call time	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_flushtime_mean	Flush Time(Mean)	Mean time of flush	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_compactionqueuesize	Compaction Queue Size	Point in time length of the compaction queue. The number of Stores for compaction in the RegionServer.	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_flushqueuesize	Flush Queue Size	Flush queue size	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_compactionscompletedcount	Compaction Count	Count of compaction	≥ 0	Count	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
rs_flushtimeops_num	Flush Operation Count	Count of flush operation **NOTE** ● Flush Operation Count belongs to a counter type. When its value reaches the upper limit, the counter wraps around and starts counting from zero. ● After a cluster is restarted, Flush Operation Count will also be cleared and recalculated.	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_blockcacheevictedcount	Discarded Cache Blocks	Block cache evict count	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_syncTime_max	Sync WAL Time(Max)	Maximum time it took to sync the WAL	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_syncTime_mean	Sync WAL Time(Mean)	Mean time it took to sync the WAL	≥ 0	ms	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
dn_byteswritten_speed	Bytes written per second	Bytes written per second of the node	≥ 0	Byte	1024 (IEC)	CloudTable instance node	1 min
dn_bytesread_speed	Bytes read per second	Bytes read per second of the node	≥ 0	Byte	1024 (IEC)	CloudTable instance node	1 min
rs_numActiveHandler	Number of RegionServer Active Handlers	Number of active RegionServer handlers (total number of handlers for processing user table requests, meta table requests, and replication requests)	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_numActiveGeneralHandler	Number of RegionServer Active Handlers for Processing User Table Requests	Number of active RegionServer handlers for processing user table requests	≥ 0	Count	N/A	CloudTable instance node	1 min
rs_scanTime_p999	99.9th Percentile of the Scan Operation Delay	99.9th percentile of the RegionServer Scan operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
rs_syncTime_p999	99.9th Percentile of the WAL Sync Operation Delay	99.9th percentile of the RegionServer WAL Sync operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_Get_99th_percentile	99th Percentile of the Get Operation Delay	99th percentile of the RegionServer Get operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_Put_99th_percentile	99th Percentile of the Put Operation Delay	99th percentile of the RegionServer Put operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_Delete_99th_percentile	99th Percentile of the Delete Operation Delay	99th percentile of the RegionServer Delete operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_Get_999th_percentile	99.9th Percentile of the Get Operation Delay	99.9th percentile of the RegionServer Get operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min
rs_Put_999th_percentile	99.9th Percentile of the Put Operation Delay	99.9th percentile of the RegionServer Put operation delay	≥ 0	ms	N/A	CloudTable instance node	1 min

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)
rs_Delete_999th_percentile	99.9th Percentile of the Delete Operation Delay	99.9th percentile of the RegionServer Delete operation delay	≥ 0	ms	N/A	Cloud Table instance node	1 min

Dimension

Key	Value
cluster_id	CloudTable cluster ID. To obtain the value, go to the cluster management page, click the cluster name, access its details page, and obtain the cluster ID in the cluster information area.
instance_name	Name of a CloudTable cluster node. To obtain the value, go to the cluster management page, click the cluster name, access its details page, and obtain the value of **instance_name**.

3.7.3.2 Setting Alarm Rules for an HBase Cluster

To monitor the usage of cloud service resources or key operations on them, you can create an alarm rule. After an alarm rule is created, if a metric reaches the specified threshold or there is a specified event, Cloud Eye immediately informs you of the exception through Simple Message Notification (SMN).

You can set CloudTable HBase alarm rules to customize the monitored objects and notification policies. Then, you can learn about the running status of the CloudTable HBase Cluster in a timely manner.

CloudTable HBase alarm rules consist of name, instance, metric, threshold, monitoring period, and notification settings. This section describes how to set CloudTable alarm rules.

Setting Alarm Rules for an HBase Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 In the navigation pane, choose **Alarm Management** > **Alarm Rules**.

Step 3 In the **Resource Type** column, select CloudTable Service to filter alarm rules.

If there are no alarm rules that meet your requirements, create a CloudTable alarm rule by referring to **Creating an Alarm Rule and Notifications**. You need to set the following parameters and retain the default values of other parameters.

Table 3-20 Alarm rule configuration parameters

Parameter	Description
Alarm Type	Select **Metric**.
Cloud Product	Select CloudTable Service.
Resource Level	Select a level as required. ● **Cloud product**: Alarm rules are specified by product. Metrics across dimensions can be configured in the same alarm rule. ● **Specific dimension** > **Cluster IDs - Compute Units**: Alarm rules are specified by cluster node.

Figure 3-20 Configuring cluster alarm rules

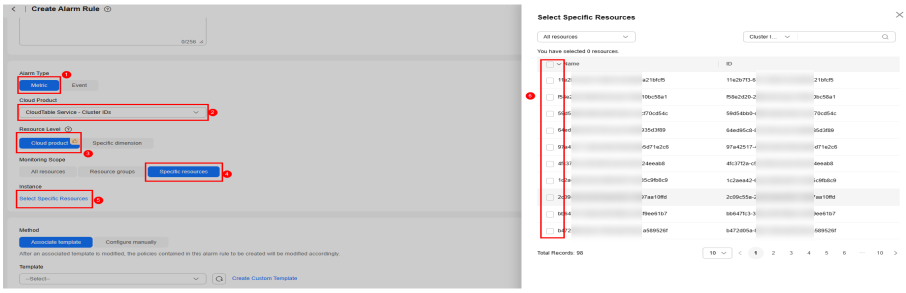
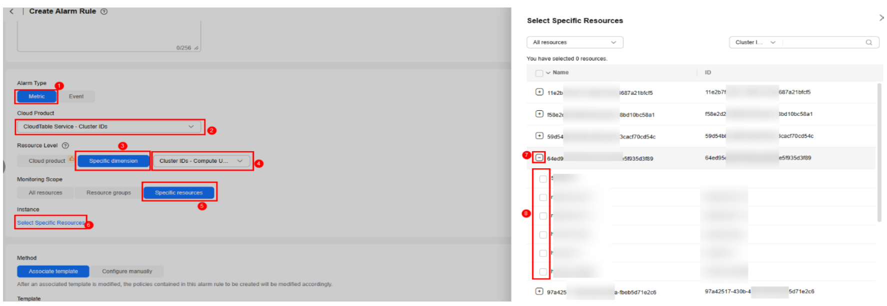


Figure 3-21 Configuring alarm rules for cluster nodes



Step 4 After the configuration is complete, click **Next**.

After the alarm rule is created, if the metric data reaches the specified threshold, Cloud Eye will immediately inform you that an exception has occurred.

----End

Viewing an Alarm Rule

Step 1 Log in to **the CloudTable console**.

Step 2 On the **Cluster Management** page, locate the target cluster and click **View Metric** in the **Operation** column to go to the **Details** page.

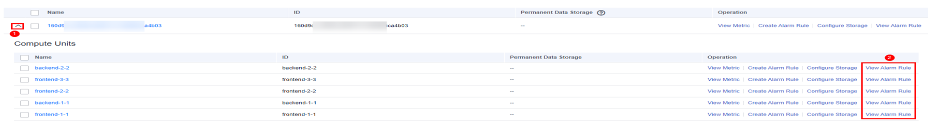
Step 3 Viewing alarm rules for clusters or cluster nodes.

1. View cluster alarm rules. Click **View Alarm Rule** in the **Operation** of the target cluster. In the displayed dialog box, you can view all alarm rules of the cluster.

Figure 3-22 Viewing cluster alarm rules

2. View alarm rules for cluster nodes. Click to expand cluster nodes. Click **View Alarm Rule** in the **Operation** of the target node. In the displayed dialog box, you can view all alarm rules of the cluster node.

Figure 3-23 Viewing alarm rules for cluster nodes



----End

3.7.3.3 Viewing HBase Cluster Monitoring Information

Scenario

Cloud Eye monitors the operational status of the HBase cluster. You can view cluster monitoring metrics on the management console. According to the monitoring information, you can quickly learn about cluster health status and key system information.

Background Information

- Monitoring metrics of an unavailable cluster node cannot be displayed on the Cloud Eye page. You can view the monitoring information only after the CloudTable cluster node is restarted or recovered.
- Cloud Eye will delete a cluster node that becomes faulty for 1 hour from the monitoring list and will not monitor it anymore. However, you need to manually clear its alarm rules.

Viewing HBase Cluster Monitoring Information on the Cloud Eye Console

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the navigation tree on the left, click **Cluster Management**.

Step 4 In the cluster list, locate the row where the target cluster resides, click **View Metric** in the **Operation** column. The Cloud Eye console is displayed.

The status of the cluster to be viewed must be **In service**.

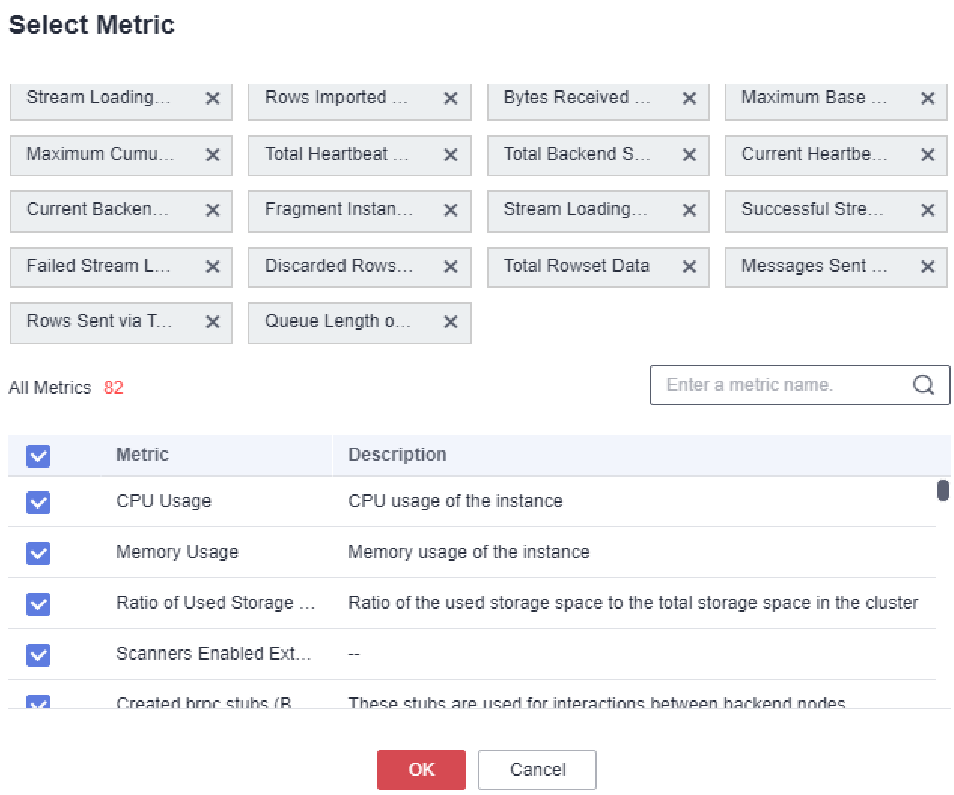
Step 5 On the **Cloud Service Monitoring** page, click on the left of the cluster ID to expand the compute unit list, and select the corresponding node to view the monitoring information.

- **ID**: ID of the monitored instance, that is, the CloudTable cluster ID
- **Viewing monitoring metrics**: Locate the compute unit you want to view and click **View Metric**.

Step 6 Set the metrics to be viewed if there are too many metrics on the monitoring page.

1. If there are too many metrics, delete them on the **Select Metric** page.
2. If the metrics displayed on the page do not contain the desired metrics, add the metrics on the **Select Metric** page.
3. Select at least one metric. You can drag a selected metric and drop it to a desired location to sort the metrics.

Figure 3-24 Selecting metrics



----End

Viewing HBase Cluster Monitoring Information on the Cluster Details Page

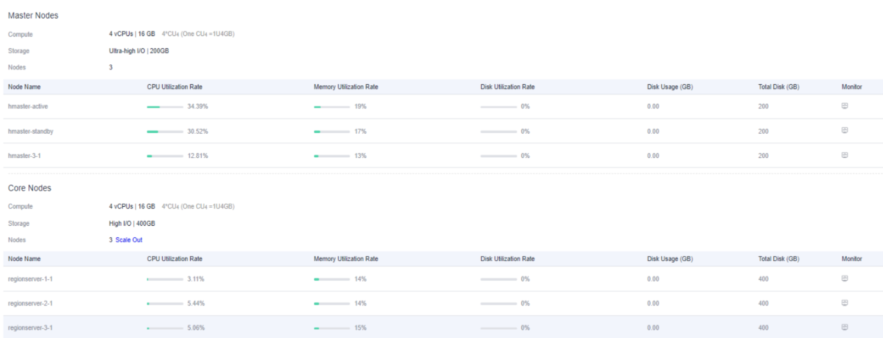
The cluster details page displays the CPU, memory, disk utilization rate, as well as the disk usage and total disk capacity of each HBase node.

Step 1 Log in to **the CloudTable console**.

Step 2 Go to the **Cluster Management** page, select the target HBase cluster, and click the cluster name to go to the cluster details page.

Step 3 View the monitoring metrics on the cluster details page.

Figure 3-25 HBase monitoring page



Step 4 To view all node metrics, navigate to the Cloud Eye monitoring page from the details page by clicking the monitoring icon . This page provides detailed monitoring metrics for individual HBase cluster nodes.

----End

3.7.4 Managing HBase Cluster Logs

3.7.4.1 Viewing HBase Cluster Logs with LTS

Cluster logs are collected and sent to Log Tank Service (LTS). You can check or dump the collected cluster logs on LTS.

NOTE

Currently, the following log types are supported:

Path for storing HBase cluster logs: /var/chroot/cloudtable/hbase/logs/

- hbase-Ruby-master-$hostname.log
- hbase-Ruby-regionserver-$hostname.log
- hbase-audit.log

Enabling LTS

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Click the toggle button in the upper left corner of the page to enable LTS.

NOTE

- If this function is enabled for the first time, the **Create Agency** dialog box is displayed. Click **OK** to authorize the agency.
- If LTS has been enabled and authorized to create an agency, no authorization is required when LTS is enabled again.

----End

Checking Cluster Logs

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Select **View Logs** in the **Operation** column. The LTS console is displayed.

----End

Disable Logging

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Toggle off the LTS switch.

Step 6 Click **OK** in the dialog box.

----End

3.7.4.2 Viewing HBase Cluster Logs with CTS

CloudTable uses CTS to record operations associated with CloudTable for later query, audit, and backtrack operations.

The following key operation traces of CloudTable are recorded in audit logs. For details, see **Table 3-21**.

Table 3-21 CloudTable HBase cluster operation traces supported by CTS

Operation	Trace Name	Resource Type
Creating a cluster	createCloudTableClusterV3	cluster
Scaling out a node	growCloudTableCluster	cluster
Restarting a cluster	rebootCloudTableCluster	cluster
Setting the storage quota	storageClusterAction	cluster
Modifying a feature	modifyClusterFeatures	cluster
Configuring parameters	modifyClusterSetting	cluster
Creating a data migration task	copierCreateTask	cluster
Enumerating database information	copierListDatabaseInfo	cluster
Enumerating cluster node information	copierListNodeInfo	cluster
Deleting a cluster	deleteCloudTableClusterV2	cluster

Operation	Trace Name	Resource Type
Disabling cluster logs	disableLTSAccess	cluster
Enabling cluster logs	enableLTSAccess	cluster
Obtaining cluster information	getClusterInfo	cluster
Obtaining database information	getDatabases	cluster
Obtaining role information	getRoles	cluster
Obtaining table information	getTables	cluster
Accessing the disk expansion page	growCloudTableDisk	cluster
Expanding specifications	growCloudTableFlavor	cluster
Modifying HBase configuration parameters	modifyClusterSetting	cluster
Restarting a node	restartInstance	cluster
Restart	REBOOTING	cluster
Capacity expansion	GROWING	cluster
Deletion	DELETING	cluster
Enabling or disabling the HBase Thrift Server	switchThriftServer	cluster
Changing the specifications of a yearly/monthly-billed cluster	changeCloudTableCluster	cluster
Enabling cold and hot data separation	switchHotColdFeature	cluster
Scanning and killing SQL statements	killQueryBySqlId	cluster

Enabling CTS

A tracker will be automatically created after CTS is enabled. All traces recorded by CTS are associated with a tracker. Currently, only one tracker can be created for each account.

Step 1 Log in to the **CTS console**.

Step 2 Enabling CTS

If you are a first-time CTS user and do not have any created tackers in the tracker list, refer to **Overview**.

If you have enabled CTS, the system has automatically created a management tracker. Only one management tracker can be created and it cannot be deleted. You can also manually create a data tracker. For details, see **Creating a Tracker** in the *Cloud Trace Service User Guide*.

----End

Disabling the Audit Log Function

If you want to disable the audit log function, disable the tracker in CTS.

Step 1 Log in to the **CTS console**.

Step 2 Disable the audit log function by disabling the tracker. To enable the audit log function again, you only need to enable the tracker.

For details about how to enable or disable a tracker, see **Disabling or Enabling a Tracker** in the *Cloud Trace Service Getting Started*.

----End

Viewing CTS Logs of CloudTable

Step 1 Log in to the **CTS console**.

Step 2 In the navigation pane on the left, choose **Trace List**.

Step 3 In the upper right corner of the trace list, click **Filter** to set the search criteria.

The following four filter criteria are available:

- **Trace Source**, **Resource Type**, and **Search By**
  - **Trace Source**: Select **CloudTable**.
  - **Resource Type**: Select **All resource types** or specify a resource type.
  - **Search By**: Select **All** or any of the following options:
    - **Trace name**: If you select this option, you also need to select a specific trace name.
    - **Resource ID**: If you select this option, you also need to select or enter a specific resource ID.
    - **Resource name**: If you select this option, you also need to select or enter a specific resource name.

- **Operator**: Select a specific operator (at user level rather than tenant level).
- **Trace Status**: Available options include **All trace statuses**, **normal**, **warning**, and **incident**. You can only select one of them.
- **Start Date** and **End Date**: You can specify the time period to query traces.

Step 4 Click **Query**.

Step 5 Click  on the left of the trace to be queried to extend its details.

Figure 3-26 Trace

Step 6 Locate the row containing the target trace and click **View Trace** in the **Operation** column.

Figure 3-27 Viewing a trace

For details about key fields in the CTS trace structure, see the **Trace Structure** in the *Cloud Trace Service User Guide*.

----End

4 Using Doris

4.1 Doris Data Model

In Doris, data is logically described in the form of tables. A table is a collection of homogeneous data with the same schema. A table consists of rows and columns. Row indicates a row of user data. Column describes different fields in a row of data. Different data types (such as integers, strings, and Boolean values) can be used as required.

In OLAP scenarios, columns can be divided into two categories: Key and Value. Key and Value can correspond to the dimension column and indicator column respectively.

Doris data models are classified into the following types:

- **Aggregate Key Model**
- **Unique Key Model**
- **Duplicate Key Model**

Aggregate Key Model

This section illustrates what an Aggregate model is and how to use it correctly with practical examples.

- Example 1: Importing data aggregation

  Assume that the business has the following data table schema:

Table 4-1 Data description

ColumnName	Type	AggregationType	Comment
user_id	LARGEINT	-	User ID
date	DATE	-	Data import date

ColumnName	Type	AggregationType	Comment
city	VARCHAR(20)	-	City where a user is located
age	SMALLINT	-	User age
sex	TINYINT	-	User gender
last_visit_date	DATETIME	REPLACE	Last visit date
cost	BIGINT	SUM	Total consumption
max_dwell_time	INT	MAX	Maximum residence time
min_dwell_time	INT	MIN	Minimum residence time

The corresponding **CREATE TABLE** statement would be as follows:

```
CREATE TABLE IF NOT EXISTS demo.example_tbl
(
    `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `date` DATE NOT NULL COMMENT "Data import date and time",
    `city` VARCHAR(20) COMMENT "City where the user is located",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "Last visit date of
the user",
    `cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",
    `max_dwell_time` INT MAX DEFAULT "0" COMMENT "Maximum residence time",
    `min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum residence time",
)
AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `sex`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
    "replication_allocation" = "tag.location.default: 3"
);
```

This is a typical fact table of user information and visit behaviors. In star models, user information and visit behaviors are usually stored in dimension tables and fact tables, respectively. Here, for easier explanation of Doris data models, the two types of information are stored in one single table.

The columns in the table are divided into Key (dimension) columns and Value (indicator) columns based on whether they are set with an **AggregationType.** Key columns are not set with an **AggregationType**, such as **user_id**, **date**, **age**, and **sex**, while Value columns are.

When data is imported, rows with the same contents in the Key columns will be aggregated into one row, and their values in the Value columns will be aggregated as their **AggregationType** specify. AggregationType has the following modes:

- SUM: Sum up the values in multiple rows.
- REPLACE: Replace the previous value with the newly imported value.

– MAX: Keep the maximum value.
– MIN: Keep the minimum value.

Table 4-2 User information table

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	A	20	0	2017-10-01 06:00:00	20	10	10
10000	2017-10-01	A	20	0	2017-10-01 07:00:00	15	2	2
10001	2017-10-01	A	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	B	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	C	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	D	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	D	35	0	2017-10-03 10:20:22	11	6	6

Assume that this is a table recording user behaviors when they are visiting a certain commodity page. The first row of data, for example, is explained as follows:

Table 4-3 Parameters

Parameter	Value	Description
user_id	10000	User ID, which uniquely identifies a user.

Parameter	Value	Description
date	2017-10-01	Time when data is imported to the database. The value is accurate to date.
city	A	City where a user is located
age	20	User age
sex	0	Gender: male (**1** indicates female)
last_visit_date	2017-10-01 06:00:00	Time when a user visits the page. The value is accurate to second.
cost	20	Consumption generated by the current visit
max_dwell_time	10	Maximum residence time of a user's access
min_dwell_time	10	Minimum residence time of a user's access

After this batch of data is imported into Doris correctly, it will be stored in Doris as follows:

Table 4-4 Inserting data

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	A	20	0	2017-10-01 07:00:00	35	10	2
10001	2017-10-01	A	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	B	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	C	32	0	2017-10-02 11:20:00	30	11	11

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10004	2017-10-01	D	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	D	35	0	2017-10-03 10:20:22	11	6	6

As shown in the figure, the data of user **10000** has been aggregated to one row, while those of other users remain the same. The explanation for the aggregated data of User **10000** is as follows.

The first 5 columns remain unchanged, so it starts with Column 6 **last_visit_date**.

- **2017-10-01 07:00:00**: The **last_visit_date** column is aggregated by REPLACE, so **2017-10-01 07:00** has replaced **2017-10-01 06:00:00**.

  NOTE

  When using REPLACE to aggregate data from the same import batch, the order of replacement is uncertain. That means, in this case, the data eventually saved in Doris could be **2017-10-01 06:00:00**. However, for different import batches, data from the new batch will replace those from the old batch.

- **35**: The **cost** column is aggregated by SUM, so the update value **35** is the result of **20** plus **15**.
- **10**: The **max_dwell_time** column is aggregated by MAX, so **10** is saved as it is the maximum between **10** and **2**.
- **2**: The **min_dwell_time** column is aggregated by MIN, so **2** is saved as it is the minimum between **10** and **2**.

  After aggregation, Doris only stores the aggregated data. The detailed raw data is not retained.

- Example 2: Keeping detailed data

  Here is a modified version of the table schema in Example 1:

Table 4-5 Data description

ColumnName	Type	AggregationType	Comment
user_id	LARGEINT	-	User ID
date	DATE	-	Data import date

ColumnName	Type	AggregationType	Comment
timestamp	DATETIME	-	Date and time when the data is imported. The value is accurate to second.
city	VARCHAR(20)	-	City where the user is located
age	SMALLINT	-	User age
sex	TINYINT	-	User gender
last_visit_date	DATETIME	REPLACE	Last visit date
cost	BIGINT	SUM	Total consumption
max_dwell_time	INT	MAX	Maximum residence time
min_dwell_time	INT	MIN	Minimum residence time

A new column **timestamp** (accurate to second) has been added to record the date and time when the data is imported.

In addition, **AGGREGATE KEY** is set to **AGGREGATE KEY(user_id, date, timestamp, city, age, sex)**.

Suppose that the imported data is as follows:

Table 4-6 User information table

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	2017-10-01 08:00:05	A	20	0	2017-10-01 06:00:00	20	10	10
10000	2017-10-01	2017-10-01 09:00:05	A	20	0	2017-10-01 07:00:00	15	2	2

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10001	2017-10-01	2017-10-01 18:12:10	A	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	2017-10-02 13:10:00	B	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	2017-10-02 13:15:00	C	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	2017-10-01 12:12:48	D	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	2017-10-03 12:38:20	D	35	0	2017-10-03 10:20:22	11	6	6

After this batch of data is imported into Doris correctly, it will be stored in Doris as follows:

Table 4-7 Stored data

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	2017-10-01 08:00:05	A	20	0	2017-10-01 06:00:00	20	10	10

user_id	date	timestamp	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	2017-10-01 09:00:05	A	20	0	2017-10-01 07:00:00	15	2	2
10001	2017-10-01	2017-10-01 18:12:10	A	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	2017-10-02 13:10:00	B	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	2017-10-02 13:15:00	C	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	2017-10-01 12:12:48	D	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	2017-10-03 12:38:20	D	35	0	2017-10-03 10:20:22	11	6	6

- Example 3: Aggregate the imported data with the existing data

  Based on the **table** in Example 1, suppose that you have the following data stored in Doris:

Table 4-8 User information table

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	A	20	0	2017-10-01 07:00:00	35	10	2
10001	2017-10-01	A	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	B	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	C	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	D	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	D	35	0	2017-10-03 10:20:22	11	6	6

Now you need to import a new batch of data:

Table 4-9 New data

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10004	2017-10-03	D	35	0	2017-10-03 11:22:00	44	19	19
10005	2017-10-03	E	29	1	2017-10-03 18:11:02	3	1	1

After this batch of data is imported into Doris correctly, the data stored in Doris will be updated as follows:

Table 4-10

user_id	date	city	age	sex	last_visit_date	cost	max_dwell_time	min_dwell_time
10000	2017-10-01	A	20	0	2017-10-01 07:00:00	35	10	2
10001	2017-10-01	A	30	1	2017-10-01 17:05:45	2	22	22
10002	2017-10-02	B	20	1	2017-10-02 12:59:12	200	5	5
10003	2017-10-02	C	32	0	2017-10-02 11:20:00	30	11	11
10004	2017-10-01	D	35	0	2017-10-01 10:00:15	100	3	3
10004	2017-10-03	D	35	0	2017-10-03 11:22:00	55	19	6
10005	2017-10-03	E	29	1	2017-10-03 18:11:02	3	1	1

In this table, the existing data and the newly imported data of User **10004** have been aggregated. Meanwhile, the new data of User **10005** has been added.

In Doris, data aggregation happens in the following three stages:

- The ETL stage of each batch of import data. At this stage, the batch of import data will be aggregated internally.
- The data compaction stage of the underlying BE nodes. At this stage, BE nodes will aggregate data from different imported batches.

– The data query stage. The data involved in the query will be aggregated accordingly.

At different stages, data will be aggregated to varying degrees. For example, when a batch of data is just imported, it may not be aggregated with the existing data. But users can only query aggregated data and they should not assume that what they have seen are not or partly aggregated.

Unique Key Model

In some multi-dimensional analysis scenarios, users are highly concerned about how to create uniqueness constraints for the Primary Key. The Unique model is introduced to solve this problem. This model is a special case of the Aggregate model and a simplified representation of table schema. Assume that the business has the following data table schema:

NOTE

Unique model table. You are not advised to enable the **merge-on-write** attribute. The **merge-on-read** attribute is used by default.

Table 4-11 Data description

ColumnName	Type	IsKey	Comment
user_id	BIGINT	Yes	User ID
username	VARCHAR(50)	Yes	Username
city	VARCHAR(20)	No	City where the user is located
age	SMALLINT	No	User age
sex	TINYINT	No	User gender
phone	LARGEINT	No	Phone number of a user
address	VARCHAR(500)	No	User address
register_time	DATETIME	No	User registration time

This is a table that contains the basic information of a user. There is no aggregation requirement for such data. The only concern is to ensure the uniqueness of the primary key (**user_id+username**). The **CREATE TABLE** statement would be as follows:

```
CREATE TABLE IF NOT EXISTS example_db.expamle_tbl
(
    `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `username` VARCHAR(50) NOT NULL COMMENT "Username",
    `city` VARCHAR(20) COMMENT "City where the user is located",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `phone` LARGEINT COMMENT "User phone number",
```

```
    `address` VARCHAR(500) COMMENT "User address",
    `register_time` DATETIME COMMENT "User registration time"
)
UNIQUE KEY(`user_id`, `username`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3"
);
```

This table schema is the same as the following table schema using the Aggregate model:

Table 4-12 Data description

ColumnName	Type	AggregationType	Comment
user_id	BIGINT	-	User ID
username	VARCHAR(50)	-	Username
city	VARCHAR(20)	REPLACE	City where the user is located
age	SMALLINT	REPLACE	User age
sex	TINYINT	REPLACE	User gender
phone	LARGEINT	REPLACE	Phone number of a user
address	VARCHAR(500)	REPLACE	User address
register_time	DATETIME	REPLACE	User registration time

The **CREATE TABLE** statement would be as follows:

```
CREATE TABLE IF NOT EXISTS example_db.expamle_tbl
(
    `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `username` VARCHAR(50) NOT NULL COMMENT "Username",
    `city` VARCHAR(20) REPLACE COMMENT "City where the user is located",
    `age` SMALLINT REPLACE COMMENT "User age",
    `sex` TINYINT REPLACE COMMENT "User gender",
    `phone` LARGEINT REPLACE COMMENT "User phone number",
    `address` VARCHAR(500) REPLACE COMMENT "User address",
    `register_time` DATETIME REPLACE COMMENT "User registration time"
)
AGGREGATE KEY(`user_id`, `username`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3"
);
```

The Unique Model is equivalent to the REPLACE aggregation function in the Aggregate model. The internal implementation and data storage are exactly the same.

Duplicate Key Model

In some multi-dimensional analysis scenarios, there is no need for primary keys or data aggregation. Duplicate models can be introduced to meet such requirements.

Table 4-13 Data

ColumnName	Type	SortKey	Comment
timestamp	DATETIME	Yes	Log time
type	INT	Yes	Log type
error_code	INT	Yes	Error code
error_msg	VARCHAR(1024)	No	Error details
op_id	BIGINT	No	Operator ID
op_time	DATETIME	No	Operation time

The **CREATE TABLE** statement would be as follows:

```
CREATE TABLE IF NOT EXISTS example_db.expamle_tbl
(
    `timestamp` DATETIME NOT NULL COMMENT "Log time",
    `type` INT NOT NULL COMMENT "Log type",
    `error_code` INT COMMENT "Error code",
    `error_msg` VARCHAR(1024) COMMENT "Error details",
    `op_id` BIGINT COMMENT "Operator ID",
    `op_time` DATETIME COMMENT "Operation time"
)
DUPLICATE KEY(`timestamp`, `type`)
DISTRIBUTED BY HASH(`type`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3"
);
```

Different from the Aggregate and Unique models, the Duplicate model stores the data as it is and executes no aggregation. Even if there are two identical rows of data, they will both be retained. The "DUPLICATE KEY" in the **CREATE TABLE** statement is only used to specify based on which columns the data are sorted. A more appropriate name than "DUPLICATE KEY" would be "Sorted Column", but it is named as such to specify the data model used. The DUPLICATE KEY is suitable for the first 2 to 4 columns.

The Duplicate model is suitable for storing raw data without aggregation requirements or primary key uniqueness constraints. For more usage scenarios, see the part "Limitations of the Aggregate Model".

Limitations of the Aggregate Model

- Aggregate model and Unique models

  The following describes the limitations of the Aggregate model. (The Unique model will also be discussed.)

  The Aggregate model only presents the aggregated data. That is, for any data that has not been aggregated (for example, data in two different import batches), the consistency of the data displayed externally must be ensured.

Suppose that you have the following table schema:

Table 4-14 Data

ColumnName	Type	AggregationType	Comment
user_id	LARGEINT	-	User ID
date	DATE	-	Data import date
cost	BIGINT	SUM	Total consumption

Assume that there are two batches of data that have been imported into the storage engine as follows:

batch1

Table 4-15 Data

user_id	date	cost
10001	2017-11-20	50
10002	2017-11-21	39

batch2

Table 4-16 Data

user_id	date	cost
10001	2017-11-20	1
10001	2017-11-21	5
10003	2017-11-22	22

Data about user **10001** in these two import batches has not yet been aggregated. However, users can only query the aggregated data shown in the following table.

Table 4-17 Data

user_id	date	cost
10001	2017-11-20	51
10001	2017-11-21	5

user_id	date	cost
10002	2017-11-21	39
10003	2017-11-22	22

Create a data table.

```
CREATE TABLE IF NOT EXISTS example_db.expamle_tb2
(
    `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `date` DATE NOT NULL COMMENT "Data import date and time",
    `cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",
)
AGGREGATE KEY(`user_id`, `date`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
"replication_allocation" = "tag.location.default: 3"
);
```

Insert the data of **Table 15** and **Table 16**.

```
INSERT INTO example_db.expamle_tb2 (user_id,date,cost) VALUES('10001','2017-11-20','50'),
('10002','2017-11-21','39'),('10001','2017-11-20','1'),('10001','2017-11-21','5'),
('10003','2017-11-22','22');
```

The aggregation operator is added to the query engine to ensure data consistency.

In addition, on the aggregated columns (Value columns), when executing aggregation class queries that are inconsistent with the aggregation function, pay attention to the semantics. For example, in the example above, if you execute the following query:

```
mysql> SELECT MIN(cost) FROM example_db.expamle_tb2;
+-------------+
| min(`cost`) |
+-------------+
|           5 |
+-------------+
1 row in set (0.02 sec)
```

The result is 5, not 1.

In addition, this consistency guarantee could considerably reduce efficiency in some queries.

Take the basic **count (*)** query as an example:

```
SELECT COUNT(*) FROM table;
```

In other databases, such queries return results quickly. Because in actual implementation, the models can get the query result by counting rows and saving the statistics upon import, or by scanning only one certain column of data to get count value upon query, with very little overhead. But in Doris' Aggregate model, the overhead of such queries is large. Take the following two batches of data as example:

batch1

Table 4-18 Data

user_id	date	cost
10001	2017-11-20	50
10002	2017-11-21	39

batch2

Table 4-19 Data

user_id	date	cost
10001	2017-11-20	1
10001	2017-11-21	5
10003	2017-11-22	22

The final aggregation result is as follows:

Table 4-20 Data

user_id	date	cost
10001	2017-11-20	51
10001	2017-11-21	5
10002	2017-11-21	39
10003	2017-11-22	22

Query the result.

```
mysql> SELECT COUNT(date) FROM example_db.expamle_tb2;
+---------------+
| count(`date`) |
+---------------+
|             4 |
+---------------+
1 row in set (0.01 sec)
```

The correct result of **select count (*) from table;** should be **4**. But if the model only scans the **user_id** column and operates aggregation upon query, the final result will be **3 (10001, 10002, 10003)**. And if it does not operate aggregation, the final result will be **5** (a total of five rows in two batches). Apparently, both results are wrong.

In order to get the correct result, we must read both the **user_id** and **date** columns, and perform aggregation when querying. In the **count (*)** query, Doris must scan all AGGREGATE KEY columns (the **user_id** and **date** columns) and aggregate them to get the semantically correct results. If there are many aggregated columns, **count (*)** queries could involve scanning large amounts of data.

Therefore, if you need to perform frequent **count (*)** queries, simulate **count (*)** by adding a column of value **1** and the SUM aggregation function. The table schema in the previous example will be modified as follows:

Table 4-21 Data description

ColumnName	Type	AggregateType	Comment
user_id	BIGINT	-	User ID
date	DATE	-	Data import date
cost	BIGINT	SUM	Total consumption
count	BIGINT	SUM	Used for **count** queries

Add a **count** column, the value of which will always be **1**. The result of **select count(*) from table;** is equivalent to that of **select sum(count) from table;**. The latter is much more efficient than the former. However, this method has its shortcomings. It requires that users will not import rows with the same values in the AGGREGATE KEY columns. Otherwise, **select sum (count) from table;** can only express the number of rows of the originally imported data, instead of the semantics of **select count (*) from table;**. The value of the former increases incorrectly.

Another method is to add a **count** column of value **1** but use the REPLACE aggregation function. Then **select sum (count) from table;** and **select count (*) from table;** could produce the same results. This method does not have restrictions on duplicate row import.

- Duplicate

  The Duplicate model does not impose the same limitations as the Aggregate model because the Duplicate model does not involve aggregation semantics. For any columns, the Duplicate model can return the semantically correct results in **count (*)** queries.

Data Model Selection

The Doris data model is classified into three types: AGGREGATE KEY, UNIQUE KEY, and DUPLICATE KEY. Data in all three models is sorted by KEY.

- AGGREGATE KEY models

  The AGGREGATE KEY model aggregates data in advance, greatly reducing data scanning and calculation workload. Therefore, it is suitable for reporting query business, which has fixed schema. But it is not suitable for **count(*)** queries. In addition, because the aggregation function in Value columns is fixed, semantic correctness needs to be considered when aggregation queries using other functions are performed.

When **AGGREGATE KEY** is the same, old and new records are aggregated. The aggregation functions currently supported are SUM, MIN, MAX, REPLACE.

```
CREATE TABLE site_visit
(
    siteid      INT,
    city        SMALLINT,
    username    VARCHAR(32),
    pv BIGINT   SUM DEFAULT '0'
)
AGGREGATE KEY(siteid, city, username)
DISTRIBUTED BY HASH(siteid) BUCKETS 10;
```

- UNIQUE KEY models

  The UNIQUE KEY model applies to scenarios where a unique primary key constraint is required. If UNIQUE KEY is the same, the new record overwrites the old record to ensure that the primary key constraint is unique. This model is suitable for analytical business with updated requirements. The UNIQUE KEY model implements the same REPLACE aggregation function as the AGGREGATE KEY model, and they are essentially the same. However, the high-performance query with pre-aggregation using functions such as ROLLUP cannot be used (because the REPLACE function is used instead of SUM).

```
CREATE TABLE sales_order
(
    orderid     BIGINT,
    status      TINYINT,
    username    VARCHAR(32),
    amount      BIGINT DEFAULT '0'
)
UNIQUE KEY(orderid)
DISTRIBUTED BY HASH(orderid) BUCKETS 10;
```

- DUPLICATE KEY models

  The DUPLICATE KEY model does not merge same rows and is suitable for ad-hoc queries in any dimension. Although the pre-aggregation feature cannot be used, the DUPLICATE KEY models do not impose the same limitations as the AGGREGATE KEY models. It supports column tailoring and vectorized execution.

```
CREATE TABLE session_data
(
    visitorid   SMALLINT,
    sessionid   BIGINT,
    visittime   DATETIME,
    city        CHAR(20),
    province    CHAR(20),
    ip          varchar(32),
    brower      CHAR(20),
    url         VARCHAR(1024)
)
DUPLICATE KEY(visitorid, sessionid)
DISTRIBUTED BY HASH(sessionid, visitorid) BUCKETS 10;
```

4.2 Doris Usage Process

The CloudTable cluster mode provides a distributed, scalable, and fully managed Doris-based real-time data warehouse. It returns mass data query results within subseconds and supports high-concurrency point queries and high-throughput complex analysis.

In Doris cluster management, you need to create a dedicated cluster and can use it on demand. Dedicated clusters are physically isolated and are not affected by other services, facilitating user management.

After a Doris cluster is created, you can use a client to access the cluster. For details, see **Connecting to a Doris Cluster**.

Figure 4-1 Doris Usage Process

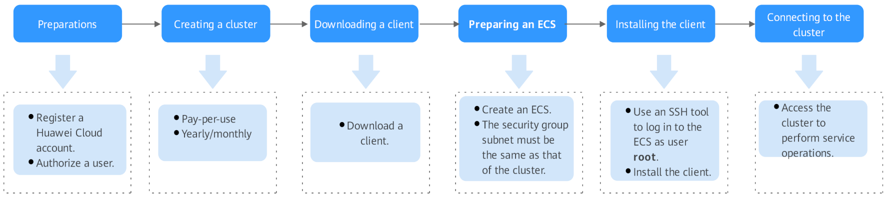


Table 4-22 Doris Usage Process

Step	Substep	Description	Detailed Instructions
Preparations	Creating a user and granting permissions	• Before using CloudTable Doris, you need to register a Huawei Cloud account, complete real-name authentication, and grant the necessary permissions to your account. • Grant the necessary service permissions to a user group, and then add users to this group to enable their access.	**Creating a User and Granting Permissions**
Creating a cluster	Creating a Doris cluster	Before using Doris to execute tasks, you need to create a Doris cluster.	**Creating a Doris Cluster**
Downloading the client	Downloading the Doris client	After creating a cluster, download and install the client. After the client is installed, you can use the SSH tool to connect to the cluster.	-

Step	Substep	Description	Detailed Instructions
Preparing an ECS	-	If the client tool runs on Linux, you need to prepare a Linux ECS that is in the VPC as a Doris cluster and the Linux ECS serves as a client host. If the client tool runs on Windows, you need to prepare a Windows ECS that is in the VPC as a Doris cluster and the Windows ECS serves as a client host.	**Preparing an ECS**
Installing the client	-	Place the downloaded client on the ECS, decompress the package, and install the client.	**Using the MySQL Client to Connect to a Doris Normal Cluster**
Connecting to the cluster	-	After installing the MySQL client on the ECS, you can run commands to connect to the cluster and perform service operations.	**Using the MySQL Client to Connect to a Doris Normal Cluster**

4.3 Creating a Doris Cluster

You can centrally manage clusters with CloudTable. A cluster is necessary for using CloudTable. This section describes how to create a cluster on the CloudTable console.

Doris clusters support two billing modes: pay-per-use and yearly/monthly. The pay-per-use billing mode is used by default when you create a cluster. In pay-per-use mode, compute units are charged by the duration you use them, with a billing cycle of one hour. With this mode, you can start or stop a cluster at any time and pay what you use. Alternatively, you can opt for the yearly/monthly billing mode, which is a prepaid option offering significant discounts compared to the pay-per-use mode. The yearly/monthly billing mode is particularly suitable for long-term users. You can also customize a CloudTable Doris cluster with specified computing capabilities and storage space to meet your business needs.

Prerequisites

- The VPC and security group of the cluster to be created must be the same as those of the ECS on the public network. Otherwise, the client cannot access the cluster.
- Before creating a cluster, you must configure inbound security group rules. For details, see **Configuring Security Group Rules**.

- Before creating a cluster, you must add the ICMP protocol to the security group rules so that you can view the status of each node by pinging the node IP address on the management plane. For details, see **Configuring Security Group Rules**.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Create Cluster** in the upper right corner.

Step 4 Configure basic cluster information by referring to the following table.

Table 4-23 Region

Parameter	Description
Region	Region of the cluster. ● Select the region for the cluster nodes to run.
AZ	Select the AZ associated with the cluster's region. For more information, see **Regions and AZs**.
Billing Mode	Select **Pay-per-use** or **Yearly/Monthly**.
Required Duration	This option is available only when **Billing Mode** is set to **Yearly/Monthly**. Configure this parameter based on your service requirements.
Auto-renew	If you select **Auto-renew** when creating a cluster, the system will automatically renew your subscription before it expires.

Table 4-24 Cluster and network configuration

Parameter	Description
Name	Name of a cluster. The cluster name must consist of 4 to 32 characters and must begin with a letter. It may include only letters, digits, and hyphens (-) but must not contain any other special characters. Additionally, the cluster name is case-insensitive.
VPC	A Virtual Private Cloud (VPC) is a secure, isolated, logical network environment. Retain the default settings. If there is no available VPC, click **View VPC** to access the VPC console and create a VPC.

Parameter	Description
Subnet	Specify a VPC subnet. A subnet provides dedicated network resources that are isolated from other networks, improving network security.
Security Group	A security group is used to control ECS access within a security group or between security groups by defining access rules. You can define different access control rules for a security group. These rules can specify which ECS ports or protocols are accessible and can be used to control inbound and outbound network traffic of VMs. After an ECS is added to the security group, it is protected by these access control rules. ECSs that do not belong to the security group cannot communicate with ECSs in the security group. You can use an existing security group or click **View Security Group** to create a new one. For more information about security groups, see **Security Group** in the *Virtual Private Cloud User Guide*. **NOTE** ● CloudTable clusters support multiple security groups and security group modification. Security group rules: – External servers can ping the instances in the security group to verify network connectivity. – Instances in a security group can communicate with each other via a private network. – Instances in a security group can be accessed via a private network. ● Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.
Database Engine	Select the type of cluster to be created.
Cluster Storage Mode	The cluster storage mode is coupled storage and compute.
Doris Kernel	Engine version of the component.

NOTE

- A CloudTable Doris cluster should contain at least three FE nodes are for high availability.
- When updating the kernel version, choose appropriate new specifications to prevent issues with cluster creation.
- Small specifications support yearly/monthly and pay-per-use billing modes.

Figure 4-2 FE node configuration

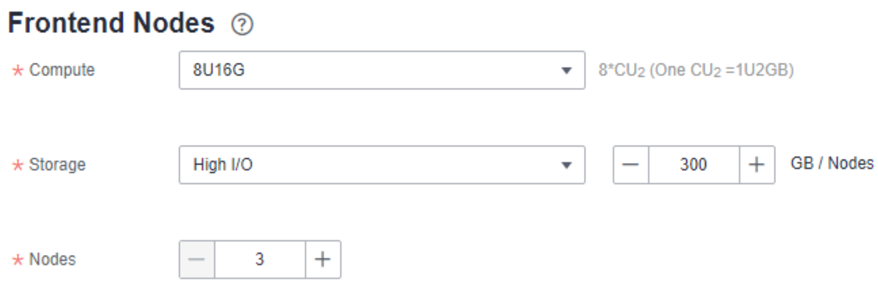


Table 4-25 FE node configuration parameters

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: • 8U16G • 16U32G • 32U64G • 64U128G • 4U16G • 8U32G • 16U64G • 32U128G • 64U256G

Storage	Select the disk specifications and capacity of the Doris compute node.
	NOTE
	• Available storage specifications:
	  – High I/O
	  – General-purpose SSD
	  – Ultra-high I/O
	  – Extreme SSD
	• The capacity ranges from 200 GB to 2,000 GB per node.

Nodes	Specify the number of nodes in the cluster. You can add 3 or 5 nodes.

Figure 4-3 BE node configuration

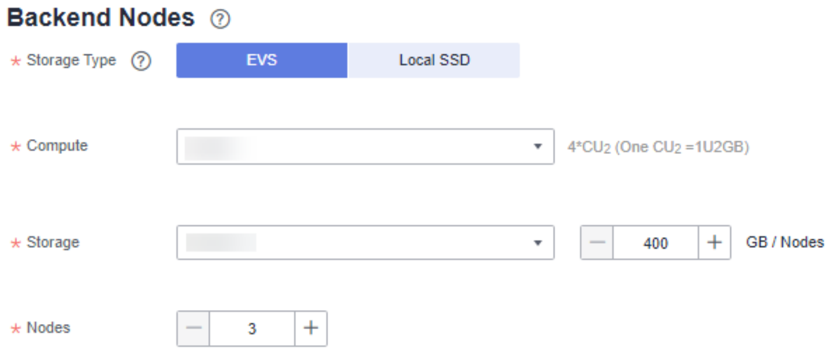


Table 4-26 BE node configuration (EVS)

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: • 4 vCPUs and 16 GB memory (It is advised to utilize this specification solely for functional testing purposes. For deployment in a production environment, a configuration with 8 vCPUs and 32 GB memory or higher is recommended.) • 8U16G • 16U32G • 32U64G • 64U128G • 8U32G • 16U64G • 32U128G • 64U256G
Storage	Select the disk specifications and capacity of the Doris compute node. **NOTE** • Available storage specifications: – High I/O – General-purpose SSD – Ultra-high I/O – Extreme SSD • The capacity ranges from 400 GB to 10,000 GB per node.
Nodes	Specify the number of nodes in the cluster. You can add 3 to 100 BE nodes.

Table 4-27 Password setting parameters

Parameter	Description
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** ● You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. ● You can delete a user or multiple users. ● After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.
Username	Doris management user, which is used to connect to the Doris cluster. The default value is **admin** and cannot be changed.
Password	Set the password of the **admin** user. The password complexity requirements are as follows: ● The password must contain 8 to 12 characters. ● The password must contain at least four types of the following characters: uppercase letters, lowercase letters, digits, and special characters ($@!%*?&._{}()+=^<>) ● The password cannot be the same as the username or the username spelled backwards. ● The strong and weak password check should be performed. **NOTE** Change the password regularly and keep it secure.
Confirm Password	Enter the password of the **admin** user again for confirmation.
Broker Process	Whether to enable the Broker process, which is used to import data for customers. After the cluster is created, this process cannot be enabled.
Enable Https	Whether to enable HTTPS access channel encryption. The HTTPS port number is **8050**. **NOTE** ● When HTTPS is enabled, only the MySQL 8.0 or later client can be used to connect to the cluster. The connection port is **9030**. For details, see **Connecting to a Doris Cluster**. ● The HTTPS option is enabled during cluster creation and cannot be disabled later. ● If the HTTPS option is not enabled during cluster creation, it cannot be enabled later.

Parameter	Description
Enable Interface Authentication	After this function is enabled, account and password authentication needs to be added for REST APIs within the cluster. This may marginally impact performance. **NOTE** ● This parameter is displayed when **Enable Https** is toggled on. ● After this option is enabled, you need to add account and password authentication when using REST APIs in the Doris cluster, such as APIs for modifying parameters and obtaining cluster monitoring data.
Tag	A tag is a key-value pair customized by users and used to classify and search for cloud resources. A tag consists of a key and a value.

Step 5 Set the parameters and click **Buy Now**.

Step 6 On the displayed page, confirm the cluster specification order information and click **Submit**. The cluster creation task is submitted.

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

The cluster creation task takes some time. The initial status of the cluster is **Creating**. After the cluster is created, the cluster status changes to **In service**.

Step 8 Submit the creation task of a yearly/monthly cluster.

Click **Pay**. On the displayed purchase page, confirm the information, select a proper payment method, and confirm the payment.

Return to the console and check the cluster status. Cluster creation takes some time. The initial status of the cluster is **Creating**. After the cluster is created, the cluster status changes to **In service**.

----End

4.4 Connecting to a Doris Cluster

4.4.1 Preparing an ECS

Before using a cluster, you need to use a client to connect to the database. After the client is installed, you can access the cluster via the private network address of the cluster.

Preparing an ECS

For details about how to purchase an ECS, see **Purchasing an ECS**.

To purchase an ECS, the following requirements must be met:

- The ECS must have the same region, AZ, VPC, and subnet as the CloudTable cluster.

For details about how to create a VPC, see **"User Guide" > "VPC and Subnet"** of **Virtual Private Cloud**.

- The ECS must have the same security group as the CloudTable cluster.

  For more information about security groups, see **Security Group** in the *Virtual Private Cloud User Guide*.

  NOTE

  When cross-VPC communication is used to access a CloudTable cluster, the network administrator needs to authorize the access to the VPC, security group, and subnet where the cluster resides.

- When purchasing an ECS, you need to set **EIP** to **Automatically assign**. Alternatively, you can bind an EIP to an ECS after the ECS is created.
- To access a Linux ECS, you are advised to use an SSH password.

  For details about how to log in to a Linux ECS, see **Logging In to a Linux ECS** in the *Elastic Cloud Server User Guide*.

4.4.2 Using the MySQL Client to Connect to a Doris Normal Cluster

Doris supports the MySQL protocol and is compatible with MySQL command line tools, JDBC/ODBC, and various visualization tools. This section uses the MySQL 5.*x* client to connect to a Doris cluster.

Constraints

- The Doris cluster and the ECS must be in the same region, AZ, and VPC.
- The Doris cluster and the ECS must be in the same security group.
- The IP address of the local host has been added to the ECS security group.

Installing the Client

Step 1 Download the **MySQL client** and select a version as required.

Step 2 Prepare a Linux ECS. For details, see **Preparing an ECS**.

Step 3 Install the client and connect to the cluster.

1. Use the SSH login tool to remotely log in to the Linux ECS through the EIP.

   For details, see **Logging In to a Linux ECS Using an SSH Password** in the *Elastic Cloud Server User Guide*.

2. Go to the root directory of the SSH tool.
   ```
   cd /
   ```
3. Create a folder in the root directory and upload the installation package downloaded in **Step 1** to the folder.
   ```
   mkdir Folder name
   ```
4. Decompress the installation package.
   ```
   cd <Path of the client installation package>
   tar -xzvf Name of the client package
   ```
   NOTE

   Replace *<Path of the client installation package>* mentioned in **Step 3.4** with the actual path.

5. Go to the **bin** directory.
```
cd mysql-5.7.22-linux-glibc2.12-x86_64/bin/
```
6. Connect to the Doris cluster.
```
./mysql -uadmin -pPassword -hInternal IP address of the cluster -P9030
```

NOTE

– *Internal IP address of the cluster*: Enter the cluster access address on the cluster details page. Replace it with the access address of the cluster you purchased. (All access addresses of the FE node can be used to access the cluster.)
– *Password* is the password set when you purchase the cluster. If there are special characters, use backslashes (\) to escape them. If the password is enclosed in single quotation marks ('), the special characters do not need to be escaped.
– *Port*: MySQL server port on the FE node. For details, see **Table 4-28**.

Table 4-28 Custom security group rules

Direction	Action	Port/Range	Type	Destination/Source Address	Usage
Outbound	Allow	All	IPv4/IPv6	0.0.0.0/0	Permit in the outbound direction
Inbound	Allow	9030		Security group of the CloudTable Doris cluster	MySQL server port on the FE node
	Allow	8030			HTTP server port on the FE node
	Allow	8040			HTTP server port on the BE node
	Allow	8050			HTTPS server port on the FE node

----End

Getting Started with Doris

1. Create a database.
```
CREATE DATABASE demo;
```
2. Create a data table.
   – Use the database.
```
USE demo;
```
   – Create a table.
```
CREATE TABLE IF NOT EXISTS demo.example_tbl
(
    `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `date` DATE NOT NULL COMMENT "Data import date and time",
    `city` VARCHAR(20) COMMENT "City where the user locates",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "Last visit date of the user",
    `cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",
    `max_dwell_time` INT MAX DEFAULT "0" COMMENT "Maximum residence time",
```

```
    `min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum residence time",
)
AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `sex`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
    "replication_allocation" = "tag.location.default:3"
);
```

3. Insert data.
```
INSERT INTO demo.example_tbl
(user_id,date,city,age,sex,last_visit_date,cost,max_dwell_time,min_dwell_time)
VALUES('10000','2017-10-01','A','20','0','2017-10-01 07:00:00','35','10','2'),
('10001','2017-10-01','A','30','1','2017-10-01 17:05:45','2','22','22'),
('10002','2017-10-02','B','20','1','2017-10-02 12:59:12','200','5','5'),
('10003','2017-10-02','C','32','0','2017-10-02 11:20:12','30','11','11'),
('10004','2017-10-01','D','35','0','2017-10-01 10:00:15','100','3','3'),
('10004','2017-10-03','D','35','0','2017-10-03 10:20:22','11','6','6');
```

4. Query the data.
   – Use Doris to perform quick data query and analysis.
```
mysql> SELECT * FROM
demo.example_tbl;
                                      +---------+------------+------+------+------+---------------------
+------+----------------+----------------+
| user_id | date       | city | age  | sex  | last_visit_date     | cost | max_dwell_time | min_dwell_time
|
+---------+------------+------+------+------+--------------------+------+----------------+----------------+
| 10000   | 2017-10-01 | A    |   20 |    0 | 2017-10-01 07:00:00 |   35 |             10 |              2 |
| 10001   | 2017-10-01 | A    |   30 |    1 | 2017-10-01 17:05:45 |    2 |             22 |             22 |
| 10002   | 2017-10-02 | B    |   20 |    1 | 2017-10-02 12:59:12 |  200 |              5 |              5 |
| 10003   | 2017-10-02 | C    |   32 |    0 | 2017-10-02 11:20:12 |   30 |             11 |             11 |
| 10004   | 2017-10-01 | D    |   35 |    0 | 2017-10-01 10:00:15 |  100 |              3 |              3 |
| 10004   | 2017-10-03 | D    |   35 |    0 | 2017-10-03 10:20:22 |   11 |              6 |              6 |
+---------+------------+------+------+------+--------------------+------+----------------+----------------+
6 rows in set (0.02 sec)
```
   – View information about a specified city.
```
mysql> SELECT * FROM demo.example_tbl where city='B';
+---------+------------+------+------+------+--------------------+------+----------------+----------------+
| user_id | date       | city | age  | sex  | last_visit_date     | cost | max_dwell_time | min_dwell_time
|
+---------+------------+------+------+------+--------------------+------+----------------+----------------+
| 10002   | 2017-10-02 | B    |   20 |    1 | 2017-10-02 12:59:12 |  200 |              5 |              5 |
+---------+------------+------+------+------+--------------------+------+----------------+----------------+
1 row in set (0.10 sec)
```

5. Delete data.
   a. Delete a specified row of data.
```
mysql> DELETE FROM demo.example_tbl WHERE user_id = 10003;
Query OK, 0 rows affected (0.04 sec)
{'label':'delete_77ed273a-a052-4d64-bac0-23916b698003', 'status':'VISIBLE', 'txnId':'39'}
```
   b. Delete the table.
```
mysql> DROP TABLE demo.example_tbl;
Query OK, 0 rows affected (0.01 sec)
```

4.4.3 Using the MySQL Client to Connect to a Doris Security Cluster

You can enable HTTPS to encrypt data transmission. This section describes how to enable HTTPS for a Doris cluster.

Constraints

- Enabling security channel may cause cluster performance deterioration.

- The Doris cluster and the ECS must be in the same region, AZ, and VPC.
- The Doris cluster and the ECS must be in the same security group.
- The IP address of the local host has been added to the security group the ECS belongs to.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Create Cluster** in the upper right corner.

Step 4 Check whether **Enable Https** (which is toggled on by default) is toggled on after completing other configurations.

Step 5 Set the parameters and click **Next**.

Step 6 Confirm the cluster information and click **Submit**. After the cluster is created, go to its details page to view its security channel status.

----End

Connecting to a Doris Cluster

Step 1 Click the name of the target security cluster to download the certificates on its details page.

Step 2 Specify the path for storing the certificates.

Step 3 Connect to the cluster. For details about how to download and install the client, see **Installing the Client**.

```
./mysql -uadmin -h Internal IP address of the cluster -P Port --ssl-ca={path}/certificate.crt --ssl-mode=VERIFY_CA -p Password
```

- **Cluster private IP address**: private IP address of the cluster to be connected
- **path**: path for storing certificates.
- **Port**: MySQL server port 9030 on the FE node
- **Password**: password set during cluster creation
- **{path}/certificate.crt**: path for storing the downloaded certificate

NOTE

Use a **MySQL 8.0** or later client after HTTPS is enabled.

----End

Getting Started with Doris

1. Create a database.
```
CREATE DATABASE demo;
```
2. Create a data table.
   - Use the database.
   ```
   USE demo;
   ```
   - Create a table.

```
CREATE TABLE IF NOT EXISTS demo.example_tbl
(
    `user_id` LARGEINT NOT NULL COMMENT "User ID",
    `date` DATE NOT NULL COMMENT "Data import date and time",
    `city` VARCHAR(20) COMMENT "City where the user locates",
    `age` SMALLINT COMMENT "User age",
    `sex` TINYINT COMMENT "User gender",
    `last_visit_date` DATETIME REPLACE DEFAULT "1970-01-01 00:00:00" COMMENT "Last visit date of the user",
    `cost` BIGINT SUM DEFAULT "0" COMMENT "Total consumption",
    `max_dwell_time` INT MAX DEFAULT "0" COMMENT "Maximum residence time",
    `min_dwell_time` INT MIN DEFAULT "99999" COMMENT "Minimum residence time",
)
AGGREGATE KEY(`user_id`, `date`, `city`, `age`, `sex`)
DISTRIBUTED BY HASH(`user_id`) BUCKETS 1
PROPERTIES (
    "replication_allocation" = "tag.location.default:3"
);
```

3. Insert data.

```
INSERT INTO demo.example_tbl
(user_id,date,city,age,sex,last_visit_date,cost,max_dwell_time,min_dwell_time)
VALUES('10000','2017-10-01','A','20','0','2017-10-01 07:00:00','35','10','2'),
('10001','2017-10-01','A','30','1','2017-10-01 17:05:45','2','22','22'),
('10002','2017-10-02','B','20','1','2017-10-02 12:59:12','200','5','5'),
('10003','2017-10-02','C','32','0','2017-10-02 11:20:12','30','11','11'),
('10004','2017-10-01','D','35','0','2017-10-01 10:00:15','100','3','3'),
('10004','2017-10-03','D','35','0','2017-10-03 10:20:22','11','6','6');
```

4. Query the data.
   – Use Doris to perform quick data query and analysis.

```
mysql> SELECT * FROM
demo.example_tbl;
+---------+------------+------+------+------+---------------------
+------+----------------+----------------+
| user_id | date       | city | age  | sex  | last_visit_date     | cost | max_dwell_time | min_dwell_time
|
+---------+------------+------+------+------+---------------------+------+----------------+----------------+
| 10000   | 2017-10-01 | A    |   20 |    0 | 2017-10-01 07:00:00 |   35 |             10 |              2 |
| 10001   | 2017-10-01 | A    |   30 |    1 | 2017-10-01 17:05:45 |    2 |             22 |             22 |
| 10002   | 2017-10-02 | B    |   20 |    1 | 2017-10-02 12:59:12 |  200 |              5 |              5 |
| 10003   | 2017-10-02 | C    |   32 |    0 | 2017-10-02 11:20:12 |   30 |             11 |             11 |
| 10004   | 2017-10-01 | D    |   35 |    0 | 2017-10-01 10:00:15 |  100 |              3 |              3 |
| 10004   | 2017-10-03 | D    |   35 |    0 | 2017-10-03 10:20:22 |   11 |              6 |              6 |
+---------+------------+------+------+------+---------------------+------+----------------+----------------+
6 rows in set (0.02 sec)
```

   – View information about a specified city.

```
mysql> SELECT * FROM demo.example_tbl where city='B';
+---------+------------+------+------+------+---------------------+------+----------------+----------------+
| user_id | date       | city | age  | sex  | last_visit_date     | cost | max_dwell_time | min_dwell_time
|
+---------+------------+------+------+------+---------------------+------+----------------+----------------+
| 10002   | 2017-10-02 | B    |   20 |    1 | 2017-10-02 12:59:12 |  200 |              5 |              5 |
+---------+------------+------+------+------+---------------------+------+----------------+----------------+
1 row in set (0.10 sec)
```

5. Delete data.
   a. Delete a specified row of data.

```
mysql> DELETE FROM demo.example_tbl WHERE user_id = 10003;
Query OK, 0 rows affected (0.04 sec)
{'label':'delete_77ed273a-a052-4d64-bac0-23916b698003', 'status':'VISIBLE', 'txnId':'39'}
```

   b. Delete the table.

```
mysql> DROP TABLE demo.example_tbl;
Query OK, 0 rows affected (0.01 sec)
```


4.5 Configuring Doris User Permissions

In data processing scenarios, when multiple users access a Doris cluster for data analysis at the same time, data leakage or misoperations may occur due to the lack of an effective permission management mechanism, hindering service security and efficiency. Doris user permission management enables unified management of users, roles, and permissions on each node in the cluster. You can create roles, create users, and bind users to roles on the console to control user permissions. Operations of different users do not affect each other, improving service efficiency. In a multi-user environment, Doris user permission management uses fine-grained permission control to ensure that each user can access only authorized data and operations. This effectively prevents data leakage and misoperations, improving the security and efficiency of overall services.

Constraints

- The deletion operation is irreversible. Even if a role with the same name is added immediately after the deletion, the permission may be different from that of the deleted one. Ensure that the role is not in use before deleting it.
- Before deleting a user, make sure that it is not needed anymore. The deletion operation is irreversible. Even if an account with the same name is added immediately after the deletion, the permission may be different from that of the deleted one.
- The password of an existing account cannot be viewed. If you forget the password, you can reset the password.
- The operation permissions of an account on the database can be viewed.
- The username and role name are case sensitive.

Prerequisites

- A Doris cluster has been created and is running properly.
- The MySQL client has been installed.

Step 1: Create a Doris Role

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management**. On the displayed cluster list page, select the target cluster and choose **Cluster Name** > **Role Management**. The role management page is displayed.

Step 3 Click **Create Role**. On the displayed page, configure parameters below.

Table 4-29 Role permissions

Parameter	Description
Rolename	The role name must start with a letter and contain 1 to 64 characters.

Parameter	Description
Global Permission	Global permissions refer to the permissions on all databases and tables, including the **SELECT**, **LOAD**, **ALTER**, **CREATE**, and **DROP** permissions.
Each Level Permission	**Database And Table** refers to the databases and tables created in the cluster. ● For databases, high-risk permissions and common permissions can be configured. For tables, only common permissions can be configured. ● Permission types – High-risk permissions: **CREATE TABLE** and **DROP TABLE** – Common permissions: **SELECT**, **LOAD**, and **ALTER**

Step 4 Click **OK**. Return to the role management page and check whether the new role is displayed in the role list.

----End

Step 2: Create a Doris User

Step 1 After a role is created, choose **Account Management**.

Step 2 Click **Create Account**. On the **Create Account** page, set the username and password.

Table 4-30 Account parameters

Parameter	Description
Username	The username must start with a letter and contain 1 to 64 characters.
IP Address	IP address used by a user for login
Tenant Name	When creating a user, you can assign a tenant to the user. Assigning a tenant to a username will change the tenants for all users sharing that username.
Password	Enter a password for the user. **NOTE** The password must meet the following requirements: ● Contain 8 to 16 characters. ● Contain at least four types of the following characters: uppercase letters, lowercase letters, digits, and special characters ($@!%*?&._{}()+=^<>) ● Cannot be the same as the username or the username spelled backwards.

Parameter	Description
Confirm Password	Enter the password again.

Step 3 Click **OK**. Return to the user management page and check whether the new user is displayed in the user list.

----End

Step 3: Bind a Role to a Doris User and Access the Cluster

For example, role A has the permission to query, insert, modify, create, and delete data. After being bound to role A, the user has the permissions of role A.

Step 1 After a role and user are created, choose **Account Management**, locate the target account, click **More**, and select **Assign Role** in the **Operation** column.

Step 2 In the **Assign Role** dialog box, select a role. Click **OK**.

Step 3 Click **Permission** in the **Operation** column and check that the user has the role permissions.

Step 4 Connect to the cluster as the created user.

For details about how to connect to a non-security cluster, see **Using the MySQL Client to Connect to a Doris Normal Cluster**.

```
./mysql -uadmin -pPassword -hInternal IP address of the cluster -P9030
```

For details about how to connect to a security cluster, see **Using the MySQL Client to Connect to a Doris Security Cluster**.

```
./mysql -uadmin -hInternal IP address of the cluster -P9030 --ssl-ca={path}/certificate.crt --ssl-mode=VERIFY_CA -pPassword
```

Step 5 Execute the query, insert, change, create, and delete commands in the CLI.

- If these commands can be executed, the role is bound successfully.
- If these commands cannot be executed, check whether the role has been configured with permissions and whether it has been bound to the user. If the fault persists, contact technical support.

----End

Managing User Permissions

- Managing roles
  - Deleting a role: Click **Delete** in the **Operation** column. On the displayed page, enter **DELETE** in the text box or click **Auto Enter**, and click **OK**.
  - Modifying role permissions: Click **Edit** in the **Operation** column. On the displayed page, select permissions as needed and click **OK**.
  - Viewing role permissions: Click **Permission** in the **Operation** column. On the displayed page, view the role's permissions on databases and tables.
- Managing users
  - Assigning a role: Choose **More** > **Assign Role** in the **Operation** column. In the displayed dialog box, select a role and click **OK**.

- Assigning a tenant: Choose **More** > **Assign Tenant** in the **Operation** column. In the displayed dialog box, select a tenant and click **OK**.
- Viewing user permissions: Click **Permission** in the **Operation** column. On the displayed page, view the user's permissions on databases and tables.
- Deleting a user: Click **Delete** in the **Operation** column. In the displayed dialog box, click **Yes**.
- Changing the user password: Click **More** and select **Update Password** in the **Operation** column. On the displayed page, change the password and click **OK**.

Common Commands for User Permissions

1. Creating a role
```
CREATE Role role_name;
```
role_name: name of the role to be created
2. Assigning permissions to a role
   - Grant the permissions on a specified database table.
```
GRANT LOAD_PRIV ON ctl1.db1.* TO ROLE 'my_role';
```
   - Grant the permission to use specified resources.
```
GRANT USAGE_PRIV ON RESOURCE 'spark_resource' TO ROLE 'my_role';
```
3. Deleting a role
```
DROP ROLE role1;
```
4. Creating a user
   - Use the **CREATE USER** command to create a Doris user (without a role).
```
CREATE USER 'Jack' IDENTIFIED BY 'password';
```

Table 4-31 Parameters

Parameter	Description
Jack	Username
password	Password of the created user

NOTE

You can run the command to specify an IP address.
```
CREATE USER 'jack02' @'192.168.%'  identified by '123456'  DEFAULT ROLE 'default_role';
```
   - Use the **CREATE USER** command to create a Doris user (with the default role assigned).
```
CREATE USER 'jeo' IDENTIFIED BY 'password' DEFAULT ROLE 'default_role';
```
5. Changing a user password

Use the **ALTER USER** command to change the password of a user.
```
ALTER USER 'Jack' IDENTIFIED BY "password";
```
6. Assigning a role to the user
   - Assign a role to the user.
```
GRANT 'role1','role2' TO 'jack'@'%';
```


Table 4-32 Parameters

Parameter	Description
role	Created role
jack	Created username

– Revoke the role assigned to the user.
```
REVOKE 'role1' FROM 'candy';
```

7. Deleting a user
```
DROP USER'Jack';
```

8. Querying the permissions and roles of a user
```
SHOW GRANTS;
```

4.6 Data Import

4.6.1 Introduction to Data Import to Doris Clusters

The load function is used to import the raw data to Doris. After the import is complete, you can query data on the MySQL client. Doris provides a variety of data import methods.

Supported Data Sources

You can select different data import methods for different data sources.

- **Importing Data to a Doris Cluster with Broker Load**
- **Importing Data to a Doris Cluster with Stream Load**

Supported Data Formats

Different import methods support different data formats.

Table 4-33 Import methods

Import Method	Supported Format
Broker Load	parquet, orc, and obs
Stream Load	csv, json, parquet, and orc

Import Instructions

The data import implementation of Doris has the following common features, which are introduced here to help you better use the data import function.

Import Atomicity Guarantee

Each import job of Doris, whether it is batch import using Broker Load or single import using the INSERT statement, is a complete transaction operation. The import transaction can ensure that the data in a batch takes effect atomically, and there will be no partial data write.

At the same time, an import job will have a label. This label is used to uniquely identify an import job in a database. Labels can be specified users, and some import functions are automatically generated by the system.

A label is used to ensure the success import of the corresponding import job and can be used only once. A successfully imported label, when used again, will be rejected with the error **Label already used**. Through this mechanism, **At-Most-Once** semantics can be implemented in Doris. In combination with the **At-Least-Once** semantics of the upstream system, the **Exactly-Once** semantics of imported data can be achieved.

Synchronous and Asynchronous Imports

Import methods are divided into synchronous and asynchronous imports. If an external program accesses the import function of Doris, you need to determine the import method and then determine the access logic.

- Synchronous import

  Doris executes the import synchronously when you create an import job and returns the import result after the import is complete. You can check whether the import is successful based on the output of the import creation command.

- Asynchronous import

  Doris directly returns a creation success message after you create an import job. But successful creation does not mean that the data has been imported. The import job is executed asynchronously. After an import job is created, you need to send a query command in polling mode to view the status of the import job. If the creation fails, you can determine whether to create the job again based on the failure information.

  NOTE

  Both two methods should not endlessly retry after Doris returns an import failure or an import job creation failure. After the external system retries for a limited number of times and fails, the failure information is retained. Most of the failures are caused by incorrect usage methods or data skew.

4.6.2 Importing Data to a Doris Cluster with Broker Load

Broker Load is an asynchronous import method, and the supported data sources depend on the data sources supported by the Broker process. This section describes the basic principles, basic operations, system configurations, and application examples of Broker Load import.

Application Scenarios

- The source data is in a storage system that the Broker can access, such as HDFS and OBS.
- The amount of data is at the level of tens to hundreds of GB.

Basic Principles

After an import job is submitted, the FE will generate a plan and distribute the plan to multiple BEs for execution based on the current number of BEs and file size, and each BE executes a part of the imported data.

A BE pulls data from the Broker during execution, and imports the data into the system after transforming the data. After all BEs complete the import, the FE determines whether the import is successful.

```
+
             | 1. user create broker load
             v
        +----+----+
        |         |
        |   FE    |
        |         |
        +----+----+
             |
             | 2. BE etl and load the data
    +--------------------------+
    |            |             |
+---v---+     +--v----+    +---v---+
|       |     |       |    |       |
|  BE   |     |  BE   |    |  BE   |
|       |     |       |    |       |
+---+-^-+     +---+-^-+    +--+-^--+
    | |           | |         | |
    | |           | |         | | 3. pull data from broker
+---v-+-+     +---v-+-+    +--v-+--+
|       |     |       |    |       |
|Broker |     |Broker |    |Broker |
|       |     |       |    |       |
+---+-^-+     +---+-^-+    +---+-^-+
    | |           | |          | |
+---v-+-----------v-+----------v-+-+
|       HDFS/BOS/AFS cluster       |
|                                  |
+----------------------------------+
```

Starting the Import

The following are some examples of using Broker Load import.

Data sample:

'100','101','102','103','104','105',100.00,100.01,100.02,'100',200,100.08,2022-04-01

'101','102','103','104','105','105',100.00,100.01,100.02,'100',200,100.08,2022-04-02

'102','103','104','105','106','105',100.00,100.01,100.02,'100',200,100.08,2022-04-03

Preparations:

Create a sample data file **source_text.txt** on the local host and upload it to the **/tmp/** directory of HDFS.

Create the **ods_source** table in Hive.

```
CREATE TABLE `ods_source`(
  `id` string,
  `store_id` string,
  `company_id` string,
  `tower_id` string,
  `commodity_id` string,
  `commodity_name` string,
```

```
  `commodity_price` double,
  `member_price` double,
  `cost_price` double,
  `unit` string,
  `quantity` string,
  `actual_price` double,
  `day ` string
)
row format delimited fields terminated by ','
lines terminated by '\n'
stored as textfile;
```

Import the TXT file created in HDFS to the **ods_source** table.

```
load data inpath '/tmp/source_text.txt' into table ods_source;
```

- Example 1: Import the data in textfile format.
  - Create a partitioned table in Hive and write data to it.
    - Create the **ods_demo_detail** table.

      ```
      CREATE TABLE `ods_demo_detail`(
        `id` string,
        `store_id` string,
        `company_id` string,
        `tower_id` string,
        `commodity_id` string,
        `commodity_name` string,
        `commodity_price` double,
        `member_price` double,
        `cost_price` double,
        `unit` string,
        `quantity` string,
        `actual_price` double
      )
      PARTITIONED BY (day string)
      row format delimited fields terminated by ','
      lines terminated by '\n'
      stored as textfile;
      ```

    - Import data in the **ods_source** table to the **ods_demo_detail** table.

      ```
      set hive.exec.dynamic.partition.mode=nonstrict;
      set hive.exec.dynamic.partition=true;
      insert overwrite table ods_demo_detail partition(day) select * from ods_source;
      ```

  - Check whether Hive table **ods_demo_detail** contains data.

    ```
    select * from ods_demo_detail;
    ```

  - Create a database in Doris.

    ```
    create database doris_demo_db;
    ```

  - Create Doris table **doris_ods_test_detail**.

    NOTE

    If the ultra-high I/O specification is not selected for the cluster during cluster creation, delete **'storage_medium'='SSD'**.

    ```
    use doris_demo_db;
    CREATE TABLE `doris_ods_test_detail` (
      `rq` date NULL,
      `id` varchar(32) NOT NULL,
      `store_id` varchar(32) NULL,
      `company_id` varchar(32) NULL,
      `tower_id` varchar(32) NULL,
      `commodity_id` varchar(32) NULL,
      `commodity_name` varchar(500) NULL,
      `commodity_price` decimal(10, 2) NULL,
      `member_price` decimal(10, 2) NULL,
      `cost_price` decimal(10, 2) NULL,
      `unit` varchar(50) NULL,
    ```

```
  `quantity` int(11) NULL,
  `actual_price` decimal(10, 2) NULL
) ENGINE=OLAP
UNIQUE KEY(`rq`, `id`, `store_id`)
PARTITION BY RANGE(`rq`)
(
PARTITION P_202204 VALUES [('2022-04-01'),('2022-08-30')))
DISTRIBUTED BY HASH(`store_id`) BUCKETS 1
PROPERTIES (
'replication_allocation' = 'tag.location.default: 3',
'dynamic_partition.enable' = 'true',
'dynamic_partition.time_unit' = 'MONTH',
'dynamic_partition.start' = '-2147483648',
'dynamic_partition.end' = '2',
'dynamic_partition.prefix' = 'P_',
'dynamic_partition.buckets' = '1',
'in_memory' = 'false',
'storage_format' = 'V2',
'storage_medium' = 'SSD'
);
```

– Import data.

```
LOAD LABEL broker_name_test01
(
    DATA INFILE('hdfs://{HDFS remote IP address}:{HDFS remote port number}/user/hive/
warehouse/ods_demo_detail/*/*')
    INTO TABLE doris_ods_test_detail
    COLUMNS TERMINATED BY ','
(id,store_id,company_id,tower_id,commodity_id,commodity_name,commodity_price,member_pric
e,cost_price,unit,quantity,actual_price)
    COLUMNS FROM PATH AS (`day`)
   SET
   (rq = str_to_date(`day`,'%Y-%m-
%d'),id=id,store_id=store_id,company_id=company_id,tower_id=tower_id,commodity_id=commodi
ty_id,commodity_name=commodity_name,commodity_price=commodity_price,member_price=m
ember_price,cost_price=cost_price,unit=unit,quantity=quantity,actual_price=actual_price)
    )
WITH BROKER 'broker1'
    (
      'username' = 'hdfs',
      'password' = ''
    )
PROPERTIES
(
    'timeout'='1200',
    'max_filter_ratio'='0.1'
);
```

– Check the import status.

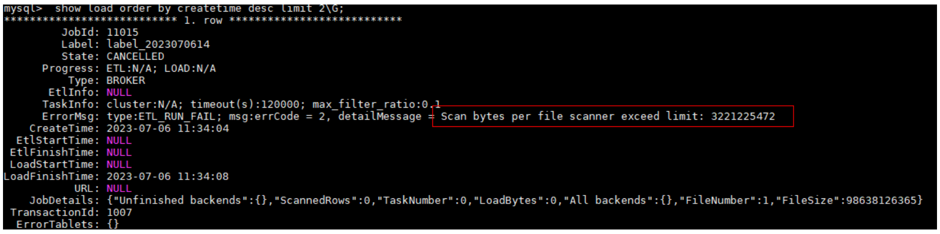
You can run the following command to view the status of the import job:

```
show load order by createtime desc limit 1\G;
```

NOTE

If the status information contains the "Scan bytes per file scanner exceed limit: 3221225472", the import fails. In this case, you need to modify the **max_bytes_per_broker_scanner** parameter. For details, see FE node parameters in **Modifying Doris Parameters to Optimize Cluster Performance**.

Figure 4-4 Checking the import status



- Example 2: Import an ORC table.
  - Create a Hive partitioned table in ORC format.

```
CREATE TABLE `ods_demo_orc_detail`(
  `id` string,
  `store_id` string,
  `company_id` string,
  `tower_id` string,
  `commodity_id` string,
  `commodity_name` string,
  `commodity_price` double,
  `member_price` double,
  `cost_price` double,
  `unit` string,
  `quantity` double,
  `actual_price` double
)
PARTITIONED BY (day string)
row format delimited fields terminated by ','
lines terminated by '\n'
STORED AS ORC;
```

  - Query the source table and write it to the partitioned table.

```
insert overwrite table ods_demo_orc_detail partition(day) select * from ods_source;
```

  - Create a Doris table.

```
CREATE TABLE `doris_ods_orc_detail` (
  `rq` date NULL,
  `id` varchar(32) NOT NULL,
  `store_id` varchar(32) NULL,
  `company_id` varchar(32) NULL,
  `tower_id` varchar(32) NULL,
  `commodity_id` varchar(32) NULL,
  `commodity_name` varchar(500) NULL,
  `commodity_price` decimal(10, 2) NULL,
  `member_price` decimal(10, 2) NULL,
  `cost_price` decimal(10, 2) NULL,
  `unit` varchar(50) NULL,
  `quantity` int(11) NULL,
  `actual_price` decimal(10, 2) NULL
) ENGINE=OLAP
UNIQUE KEY(`rq`, `id`, `store_id`)
PARTITION BY RANGE(`rq`)
(
PARTITION P_202204 VALUES [('2022-04-01'), ('2022-08-30')))
DISTRIBUTED BY HASH(`store_id`) BUCKETS 1
PROPERTIES (
'replication_allocation' = 'tag.location.default: 3',
'dynamic_partition.enable' = 'true',
'dynamic_partition.time_unit' = 'MONTH',
'dynamic_partition.start' = "-2147483648",
```



```
'dynamic_partition.end' = '2',
'dynamic_partition.prefix' = 'P_',
'dynamic_partition.buckets' = '1',
'in_memory' = 'false',
'storage_format' = 'V2');
```

– Import data.

```
LOAD LABEL orc_2022_02_17
(
    DATA INFILE("hdfs://{HDFS remote IP address}:{HDFS remote port number}/user/hive/
warehouse/ods_demo_orc_detail/*/*")
    INTO TABLE doris_ods_orc_detail
    COLUMNS TERMINATED BY ","
    FORMAT AS 'orc'
(id,store_id,company_id,tower_id,commodity_id,commodity_name,commodity_price,member_pric
e,cost_price,unit,quantity,actual_price)
    COLUMNS FROM PATH AS (`day`)
   SET
   (rq = str_to_date(`day`,'%Y-%m-
%d'),id=id,store_id=store_id,company_id=company_id,tower_id=tower_id,commodity_id=commodi
ty_id,commodity_name=commodity_name,commodity_price=commodity_price,member_price=m
ember_price,cost_price=cost_price,unit=unit,quantity=quantity,actual_price=actual_price)
    )
WITH BROKER 'broker1'
    (
      'username' = 'hdfs',
      'password'= ""
    )
PROPERTIES
(
    'timeout'="1200",
    'max_filter_ratio'="0.1"
);
```

– Query the imported data.

```
show load order by createtime desc limit 1\G;
```

- Example 3: Import data in OBS format.

– Create a Doris table.

```
CREATE TABLE `obs_detail_test` (
  `id` varchar(32) NOT NULL,
  `store_id` varchar(32) NULL,
  `company_id` varchar(32) NULL,
  `tower_id` varchar(32) NULL,
  `commodity_id` varchar(32) NULL,
  `commodity_name` varchar(500) NULL,
  `commodity_price` decimal(10, 2) NULL,
  `member_price` decimal(10, 2) NULL,
  `cost_price` decimal(10, 2) NULL,
  `unit` varchar(50) NULL,
  `quantity` int(11) NULL,
  `actual_price` decimal(10, 2) NULL
) ENGINE=OLAP
UNIQUE KEY(`id`, `store_id`)
DISTRIBUTED BY HASH(`store_id`) BUCKETS 1
PROPERTIES (
'replication_allocation' = 'tag.location.default: 3',
'in_memory' = 'false',
'storage_format' = 'V2'
);
```

– Import OBS data to the Doris table.

Construct 100 pieces of text data, which corresponds to the fields in the Doris table. Upload the data to OBS buckets.

```
LOAD LABEL label_2023021801
(
DATA INFILE("obs://xxx/source_text2.txt")
INTO TABLE `obs_detail_test`
COLUMNS TERMINATED BY ","
```

```
(id,store_id,company_id,tower_id,commodity_id,commodity_name,commodity_price,member_pric
e,cost_price,unit,quantity,actual_price)
)
WITH BROKER 'broker1' (
'fs.obs.access.key' = 'xxx',
'fs.obs.secret.key' = 'xxxxxx',
'fs.obs.endpoint' = 'xxxxxx'
)
PROPERTIES
(
'timeout'="1200",
'max_filter_ratio'='0.1'
);
```

Methods for obtaining **fs.obs.access.key**, **fs.obs.secret.key**, and **fs.obs.endpoint**:

- For details about how to obtain **fs.obs.access.key** and **fs.obs.secret.key**, see **Access Keys** in the OBS documentation.
- For details about how to obtain **fs.obs.endpoint**, see **Endpoints and Domain Names** in the OBS documentation.

– Query data.

```
show load order by createtime desc limit 1\G;
```

- Example 4: Import HDFS data to a Doris table through With HDFS.

– Create a Doris table.

```
CREATE TABLE `ods_dish_detail_test` (
  `id` varchar(32) NOT NULL,
  `store_id` varchar(32) NULL,
  `company_id` varchar(32) NULL,
  `tower_id` varchar(32) NULL,
  `commodity_id` varchar(32) NULL,
  `commodity_name` varchar(500) NULL,
  `commodity_price` decimal(10, 2) NULL,
  `member_price` decimal(10, 2) NULL,
  `cost_price` decimal(10, 2) NULL,
  `unit` varchar(50) NULL,
  `quantity` int(11) NULL,
  `actual_price` decimal(10, 2) NULL
) ENGINE=OLAP
UNIQUE KEY(`id`, `store_id`)
DISTRIBUTED BY HASH(`store_id`) BUCKETS 1
PROPERTIES (
'replication_allocation' = 'tag.location.default: 3',
'in_memory' = 'false',
'storage_format' = 'V2'
);
```

– Import data.

Construct 100 text data records, which correspond to fields in the Doris table.

```
LOAD LABEL label_2023021703
        (
            DATA INFILE("hdfs://{HDFS remote IP address}:{HDFS remote port number}/tmp/
source_text2.txt")
            INTO TABLE `ods_dish_detail_test`
            COLUMNS TERMINATED BY ","
(id,store_id,company_id,tower_id,commodity_id,commodity_name,commodity_price,member_pric
e,cost_price,unit,quantity,actual_price)
        )
        with HDFS (
            'fs.defaultFS'="hdfs://{HDFS remote IP address}:{HDFS remote port}",
            'hadoop.username'="hdfs",
            'password'=""
        )
```



```
PROPERTIES
(
    'timeout'="1200",
    'max_filter_ratio'='0.1'
);
```

– Query data.

```
show load order by createtime desc limit 1\G;
```

Canceling Data Import

If a Broker Load job is not cancelled or finished, you can manually cancel it. When canceling an import job, you need to specify the label of the job.

System Configurations

Configure the FE. The following configurations belong to the system-level configuration of Broker Load, that is, the configurations that apply to all Broker Load import jobs. You can change the configuration values by modifying the **FE configuration items**.

max_bytes_per_broker_scanner/max_broker_concurrency

max_bytes_per_broker_scanner limits the maximum amount of data processed by a single BE node. **max_broker_concurrency** limits the maximum number of concurrent imports in a job. The minimum amount of data processed (64 MB by default), the maximum number of concurrent imports, source file size, and the number of BE nodes in the current cluster together determine the number of concurrent imports in a job.

```
Number of concurrent imports = Math.min(Source file size/Minimum amount of data processed (64 MB by default), Maximum concurrent imports, Current number of BE nodes)
Amount of data processed by a single BE node = Source file size/Number of concurrent imports
```

Usually, the maximum amount of data supported by an import job is the product of the **max_bytes_per_broker_scanner** value and the number of BE nodes. If you need to import a larger amount of data, you need to adjust the value of the **max_bytes_per_broker_scanner** parameter appropriately.

Default configuration:

- **max_broker_concurrency**: The default value is **10**.
- **max_bytes_per_broker_scanner**: The default value is 3 GB, which should be converted to the unit of bytes.

Application Examples

- Application scenarios

  If raw data is stored in a file system (such as HDFS, BOS, or AFS), Broker Load is the most suitable solution. Broker Load is the only way of asynchronous import in a single import. Therefore, if you need to use asynchronous access when importing large files, you can also use Broker Load.
- Data volume

  Only the case of a single BE node is discussed here. If you have multiple BE nodes in your cluster, the amount of data below should be multiplied by the number of BE nodes. For example, if you have three BE nodes, the value below 3 GB should be multiplied by 3, that is, 9 GB.

- Below 3 GB (included): You can directly submit a request to create a Broker Load import.
- Above 3 GB: Since the maximum amount of data processed by a single BE node is 3 GB in a single import job, you can import the files larger than 3 GB only by adjusting the import parameters of Broker Load.
  - Change the maximum number of scans and the maximum number of concurrent imports of a single BE node based on the number of BE nodes and the source file size.

```
Modify FE configuration items.
max_broker_concurrency = Number of BE nodes
Maximum amount of data processed by a single BE node in the current import job =
Source file size/max_broker_concurrency
max_bytes_per_broker_scanner ≥ Maximum amount of data processed by a single BE
node in the current import job
For example, for a 100 GB file, the number of BE nodes in the cluster is 10.
max_broker_concurrency = 10
max_bytes_per_broker_scanner ≥10 GB = 100 GB/10
```

    After the modification, all BE nodes concurrently process the import job, and each BE node processes a part of the source file.

    NOTE

    The configurations of two FE nodes above are all system-level configurations and their modifications apply to all Broker Load jobs.

  - When creating an import job, customize its timeout period.

    You are not advised to directly increase the maximum timeout period. If the time for a single import exceeds the default maximum import timeout period (4 hours), you are advised to split the file in a single import and then import the file fragments multiple times. If a single import exceeds 4 hours, it also takes a long time to retry upon an import failure.

    You can use the following formula to calculate the expected maximum data volume of files to be imported in a Doris cluster:

```
Expected maximum data volume of files to be imported = 14,400s × 10 MB/s × Number of
BE nodes
Assume that the number of BE nodes in a cluster is 10.
Expected maximum data volume of files to be imported = 14,400s × 10 MB/s × 10 =
1,440,000 MB ≈ 1,440 GB
```

    NOTE

    Generally, a user's environment may not reach the speed of 10 MB/s, so it is recommended that files over 500 GB be split before being imported.

Job Scheduling

The system limits the number of running Broker Load jobs in a cluster to prevent too many Load jobs from running at the same time.

First, the configuration parameter **desired_max_waiting_jobs** of the FE node will limit the number of Broker Load jobs that have not started or are running (jobs in the **PENDING** or **LOADING** state) in a cluster. The default value is **100**. If the threshold is exceeded, newly submitted jobs will be rejected.

A Broker Load job has the **pending task** and **loading task** phases. The **pending task** phase is responsible for obtaining the information of the file to be imported, and the **loading task** phase will send the specific import job to the BE node to execute.

The FE configuration parameter **async_pending_load_task_pool_size** is used to limit the number of pending tasks running at the same time. In this way, the number of running import jobs is controlled. The default value is **10**. Assume that 100 Load jobs are submitted. At the same time only 10 jobs will enter the **LOADING** state and start execution, while other jobs stay in the **PENDING** state.

The FE configuration parameter **async_loading_load_task_pool_size** is used to limit the number of loading tasks running at the same time. A Broker Load job will have one pending task and multiple loading tasks (equal to the number of **DATA INFILE** clauses in the **LOAD** statement). Therefore, the value of **async_loading_load_task_pool_size** must be greater than or equal to that of **async_pending_load_task_pool_size**.

Performance Analysis

You can run **set enable_profile=true** to enable session variables before submitting a Load job. Then, submit the import job. After the import job is complete, you can view the profile of the import job on the **Queries** page of the FE web UI.

4.6.3 Importing Data to a Doris Cluster with Stream Load

Stream Load is a synchronous import method. You can import local files or data streams into Doris by sending HTTP requests. Stream Load synchronously executes the import and returns the import result. You can determine whether the import is successful based on the response body.

Stream Load is used to import local files or import data in data streams through programs.

Basic Principles

The following shows the main flow of Stream Load, omitting some import details.

```
 ^      +
                         |      |
                         |      | 1A. User submit load to FE
                         |      |
                         |   +--v-----------+
                         |   | FE           |
5. Return result to user |   +--+-----------+
                         |      |
                         |      | 2. Redirect to BE
                         |      |
                         |   +--v-----------+
                         +---+Coordinator BE| 1B. User submit load to BE
                             +-+-----+----+-+
                               |     |    |
                         +-----+     |    +-----+
                         |           |          | 3. Distribute data
                         |           |          |
                       +-v-+       +-v-+      +-v-+
                       |BE |       |BE |      |BE |
                       +---+       +---+      +---+
```

In Stream Load, Doris selects a node as the Coordinator node. This node receives data and distributes data to other data nodes. You can submit the import command through HTTP. If the command is submitted to the FE node, the FE node forwards the request to a BE node through the **HTTP redirect** command. You can also directly submit the import command to a specified BE node. The Coordinator BE node returns the final import result.

Basic Operations

NOTE

Before using Stream Load to import data, ensure that the security group ports of the Doris cluster (ports 8030 and 8040) are enabled. Otherwise, the connection to Stream Load will time out.

- Creating an import job

  Stream Load submits and transfers data through HTTP. Here, the **curl** command shows how to submit an import.

  You can also perform operations through other HTTP clients.

```
curl --location-trusted -u user:passwd [-H ""...] -T data.file -XPUT http://fe_host:http_port/api/{db}/{table}/_stream_load
```

NOTE

- The properties supported in the header are described in the import parameters below.
- The format is **-H "***Key 1***:***Value 1***"**.
- **port** specifies the HTTP port number.

You can run **HELP STREAM LOAD** to view the detailed syntax for creating an import job. All parameters related to Stream Load import jobs are set in the header. The following table describes related parameters.

Table 4-34 Parameters

Parameter		Description
Signature parameter	user/passwd	Stream Load uses the HTTP protocol to create the import job and signs it through Basic Access authentication. Doris verifies user identity and import permissions based on the signature.
Import parameters	label	A label identifies an import job. Each import job has a unique label in a single database. A label is a user-defined name in the import command. With this label, you can view the execution status of the import job. Another function of labels is to prevent repeated data import. It is strongly recommended that you use the same label for the same batch of data. In this way, repeated requests for the same batch of data will only be accepted once, guaranteeing At-Most-Once. When the corresponding import operation state of a label is **CANCELLED**, the label can be used again.

Parameter		Description
	column_separator	This parameter specifies the column separator in the imported file. **\t** is used by default. If it is an invisible character, you need to prefix **\x** and use its hexadecimal notation to represent the separator. For example, the separator **\x01** of the Hive file needs to be specified as **-H "column_separator:\x01"**. You can use a combination of multiple characters as the column separator.
	line_delimiter	This parameter specifies the line delimiter in the imported file. **\n** is used by default. You can use a combination of multiple characters as the line delimiter.
	max_filter_ratio	This parameter specifies the maximum tolerance rate of the import job. The default value is **0**, and the range is **0** to **1**. When the import error rate exceeds this value, the import fails. If you want to ignore error rows, set this parameter to a value greater than **0** to ensure a successful import. The calculation formula is as follows: (dpp.abnorm.ALL / (dpp.abnorm.ALL + dpp.norm.ALL)) > max_filter_ratio **dpp.abnorm.ALL** indicates the number of rows with unqualified data quality. For example, the type, number of columns, or length does not match. **dpp.norm.ALL** indicates the number of correct data records during the import. You can run the **SHOW LOAD** command to query the correct data volume of the import job. Number of rows in the source file = **dpp.abnorm.ALL** + **dpp.norm.ALL**
	where	This parameter specifies the filter criteria for an import job. Stream Load supports filtering of raw data by specifying the **where** clause. The filtered data will not be imported or calculated into the filter ratio, but will be counted into **num_rows_unselected**.
	Partitions	This parameter specifies the partition information of the table to be imported. If the data to be imported does not belong to the specified partition, the information will not be imported and will be included in **dpp.abnorm.ALL**.

Parameter		Description
	columns	The function transformation configuration of data to be imported includes the sequence change of columns and the expression transformation. The expression transformation method is the same as that of the query statement.
	exec_mem_limit	This parameter limits the import memory. The default value is 2 GB, in bytes.
	strict_mode	The strict mode can be enabled for Stream Load import. To enable the strict mode, specify **strict_mode** to **true** in the header. By default, the strict mode is disabled. In strict mode, column type conversion during data import is strictly filtered. The strict filtering policy is as follows: For column type conversion, if the strict mode is enabled, incorrect data will be filtered out. Incorrect data here refers to the data that is originally non-null but is converted into null. If a column to be imported is converted by a function, the strict mode does not affect the column. If the type of a column to be imported contains a range restriction and the raw data can be converted but cannot pass the range restriction, the strict mode does not affect the column. For example, if the type is decimal(1,0) and the raw data is 10, the data can be converted but is not within the range specified by the column. The strict mode does not affect the data.
	merge_type	Data merge supports three types, APPEND, DELETE, and MERGE. APPEND is the default type, which means that all this batch of data needs to be appended to the existing data. DELETE means to delete all rows with the same key in this batch of data. MERGE semantics need to be used in conjunction with the DELETE condition, which means that the data that meets the DELETE condition is processed according to DELETE semantics and the rest is processed according to APPEND semantics.

<table>
<tr><th colspan="2">Parameter</th><th>Description</th></tr>
<tr><td></td><td>two_phase_commit</td><td>
Stream Load import supports the two-phase transaction commit mode. In the Stream Load process, data is written and a message is returned. At this time, the data is invisible and the transaction status is PRECOMMITTED. After you manually trigger the commit operation, the data becomes visible.
Example:

Initiate a Stream Load pre-commit operation.
<pre><code>curl --location-trusted -u user:passwd -H "two_phase_commit:true" -T test.txt http://fe_host:http_port/api/{db}/{table}/_stream_load
{
    "TxnId": 18036,
    "Label": "55c8ffc9-1c40-4d51-b75e-f2265b3602ef",
    "TwoPhaseCommit": "true",
    "Status": "Success",
    "Message": "OK",
    "NumberTotalRows": 100,
    "NumberLoadedRows": 100,
    "NumberFilteredRows": 0,
    "NumberUnselectedRows": 0,
    "LoadBytes": 1031,
    "LoadTimeMs": 77,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 1,
    "ReadDataTimeMs": 0,
    "WriteDataTimeMs": 58,
    "CommitAndPublishTimeMs": 0
}</code></pre>
Commit a transaction.

Note 1: The request can be sent to either the FE or BE node.
Note 2: {table} in the URL can be omitted during commit.
<pre><code>curl -X PUT --location-trusted -u user:passwd -H "txn_id:18036" -H "txn_operation:commit" http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
{
    "status": "Success",
    "msg": "transaction [18036] commit successfully."
}</code></pre>

Abort a transaction.

Note 1: The request can be sent to either the FE or BE node.
Note 2: {table} in the URL can be omitted during abort.
<pre><code>curl -X PUT --location-trusted -u user:passwd -H "txn_id:18037" -H "txn_operation:abort" http://fe_host:http_port/api/{db}/{table}/_stream_load_2pc
{
    "status": "Success",
    "msg": "transaction [18037] abort successfully."
}</code></pre>

</td></tr>
</table>

- Example 1: Import data in CSV format.
  - Create a Doris table.

```
CREATE TABLE cloudtable0327.doris_streameload_test01
(
user_id bigint,
date date,
group_id bigint,
modify_date date,
keyword VARCHAR(128)
)
UNIQUE KEY(user_id, date, group_id)
DISTRIBUTED BY HASH (user_id) BUCKETS 32
PROPERTIES(
'function_column.sequence_col' = 'modify_date',
'replication_num' = '3',
'in_memory' = 'false'
);
```

  - Prepare the data table **sequencedata01.csv**.

Table 4-35 sequencedata01.csv

1	2020-02-22	1	2020-02-21	a
1	2020-02-22	1	2020-02-22	b
1	2020-02-22	1	2020-03-05	c
1	2020-02-22	1	2020-02-26	d
1	2020-02-22	1	2020-02-23	e
1	2020-02-22	1	2020-02-24	b

  - Run the **curl** command to load data.

```
curl -k --location-trusted -u admin:passwd -T sequencedata01.csv -H 'column_separator:,'
https://{fe_host}:{http_port}/api/cloudtable0327/doris_streameload_test01/_stream_load
```

  - View the returned result.

    Since Stream Load imports data synchronously, the result of the import is directly returned to the user through the return value of the import.

```
{
    "TxnId": 1003,
    "Label": "b6f3bc78-0d2c-45d9-9e4c-faa0a0149bee",
    "Status": "Success",
    "ExistingJobStatus": "FINISHED", // optional
    "Message": "OK",
    "NumberTotalRows": 1000000,
    "NumberLoadedRows": 1000000,
    "NumberFilteredRows": 1,
    "NumberUnselectedRows": 0,
    "LoadBytes": 40888898,
    "LoadTimeMs": 2144,
    "BeginTxnTimeMs": 1,
    "StreamLoadPutTimeMs": 2,
    "ReadDataTimeMs": 325,
    "WriteDataTimeMs": 1933,
    "CommitAndPublishTimeMs": 106,
    "ErrorURL": "http://fe_host:http_port/api/_load_error_log?file=__shard_0/
error_log_insert_stmt_db18266d4d9b4ee5-
abb00ddd64bdf005_db18266d4d9b4ee5_abb00ddd64bdf005"
}
```


- The following table describes the parameters of the Stream Load import result.

Table 4-36 Parameters

Parameter	Description
TxnId	Import transaction ID, which is invisible to users.
Label	Import label, which is user-defined or automatically generated by the system.
Status	Import completion status. ● **Success**: indicates that the import is successful. ● **Publish Timeout**: indicates that the import has been completed but data visibility may be delayed. You do not need to retry. ● **Label Already Exists**: indicates that the label is duplicate and needs to be replaced. ● **Fail**: indicates that the import fails.
ExistingJobStatus	Status of the import job corresponding to the existing label. This field is displayed only when the status is **Label Already Exists**. You can view the status of the import job corresponding to the existing label. **RUNNING** indicates that the job is still being executed, and **FINISHED** indicates that the job is successful.
Message	Import error information.
NumberTotalRows	Total number of rows processed during the import.
NumberLoadedRows	Number of rows successfully imported.
NumberFilteredRows	Number of rows whose data quality is unqualified.
NumberUnselectedRows	Number of rows filtered by the **where** condition.
LoadBytes	Number of imported bytes.
LoadTimeMs	Time when the import is complete, in milliseconds.
BeginTxnTimeMs	Time taken to request the FE node to start a transaction, in milliseconds.

Parameter	Description
StreamLoadPutTimeMs	Time taken to request the FE node to obtain the execution plan for importing data, in milliseconds.
ReadDataTimeMs	Time taken to read data, in milliseconds.
WriteDataTimeMs	Time taken to write data, in milliseconds.
CommitAndPublishTimeMs	Time taken to submit and publish a transaction to the FE node, in milliseconds.
ErrorURL	URL to view a specific error line if there are data quality problems.

NOTE

Since Stream Load is a synchronous import mode, import information will not be recorded in Doris. You cannot see Stream Load asynchronously by checking import commands. You need to view the return value of the import request to obtain the import result.

- Example 2: Import data in JSON format.

  Prepare data in JSON format and save the data in the **testjson.json** file. Upload the JSON data to the Doris client.

  {"id": 100, "city": "B", "code" : 1}

  - Create a Doris table.
```
CREATE TABLE `doris_testjson01` (
  `id` varchar(32) NOT NULL,
  `city` ARRAY<int(11)>,
  `code` int(11)
) ENGINE=OLAP
DUPLICATE KEY(`id`)
COMMENT "OLAP"
DISTRIBUTED BY HASH(`id`) BUCKETS 1
PROPERTIES (
'replication_allocation' = 'tag.location.default: 3',
'in_memory' = 'false',
'storage_format' = 'V2'
);
```
  - Run the **curl** command to load data.
```
curl --location-trusted -u admin:{Doris cluster password} -H 'format: json' -T testjson.json
https://fe_host:http_port/api/{Doris database}/doris_testjson01/_stream_load -k
```
  - Query data.
```
select * from doris_testjson01;
```

Canceling Data Import

You cannot manually cancel Stream Load jobs. It will be automatically canceled by the system upon a timeout or import error.

Viewing Stream Load Jobs

You can run the **show stream load** command to view completed Stream Load jobs.

4.7 Doris Enterprise-class Enhancement

4.7.1 Configuring a Doris Tenant

4.7.1.1 Adding a Doris Tenant

Doris' multi-tenant management feature is designed to minimize cross-tenant interference while multiple users conduct data operations within the same Doris cluster. It ensures optimal distribution of cluster resources among users. It encompasses tenant lifecycle management, resource configuration for tenants, and monitoring of tenant resource consumption. It offers a sophisticated multi-tenant management framework that enables enterprises to execute centralized management of tenants and services effectively.

Constraints

- By default, a new user is bound to the default tenant **normal**.
- A user can be bound to only one tenant.
- A tenant can be bound to multiple users.
- The tenant name cannot be changed.

Creating a Doris Tenant

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Cluster Management** on the left. The current cluster list is displayed.

Step 4 Locate the cluster to be operated and click *Cluster Name* > **Tenant Management**. The **Tenant Management** page is displayed.

Step 5 Click **Create Tenant** in the upper right corner of the page. On the displayed page, configure tenant attributes by referring to **Table 1**.

Table 4-37 Parameters for creating a tenant

Parameter	Value Range	Description
Tenant Name	-	Name of the current tenant. The value must start with a letter and contain 4 to 64 characters. Only uppercase letters, lowercase letters, digits, and underscores (_) are allowed.

Parameter	Value Range	Description
CPU Quota Weight	1~9223372036854 77580	Weight of CPU resources that can be used by a tenant For example, if only tenant A is executing queries, the CPU usage of tenant A is 100%. If two tenants A and B are executing queries, and the CPU quota usage for tenant A is 10 and that for tenant B is 20, the CPU resources can be used by the query tasks of tenant A is one-third of total resources, that is, 10/(10 + 20). If tenant C starts query tasks and its CPU quota usage is 30, a CPU quota of tenant A is one-sixth of total resources, that is, 10/(10 + 20 + 30).
Memory Quota Percentage (%)	1%~100%	Maximum proportion of memory that can be used by a tenant
Concurrent Requests	1~2147483647	Maximum number of concurrent query tasks that a tenant can run. This parameter specifies the maximum number of tasks on each FE node. For example, if the number of concurrent requests is set to 1 and the Doris has three FE nodes, the maximum number of SQL statements that can be executed in a cluster is 3.
Queue Length	1~2147483647	Maximum number of waiting query tasks. Excessive SQL statements are queued. When the queue is full, newly submitted queries are rejected.
Queue Waiting Duration (ms)	1~2147483647	Maximum waiting duration of a tenant query task. If the query waiting duration exceeds the value of this parameter, the query is rejected. The unit is millisecond.

Parameter	Value Range	Description
Enabling Memory Soft Isolation	Enable/Disable	Whether a tenant can use more memory resources than the limit ● If this function is disabled, the system immediately cancels the tasks that occupy the most memory in the tenant groups when detecting that the memory usage of the tenant exceeds the upper limit. ● If this function is enabled and the cluster has idle memory resources, the tenant can use the system memory more than the limit. The tasks that occupy the most memory in the tenant groups are canceled only when cluster resources are insufficient.

Step 6 Click **OK**. The tenant is created.

----End

Binding a User to a Tenant

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Cluster Management** on the left. The current cluster list is displayed.

Step 4 Locate the cluster to be operated and click *Cluster Name* > **User Management**. The **User Management** page is displayed.

Step 5 Select the user to be bound and choose **More** > **Assign Tenant** in the **Operation** column on the user management page. The **Assign Tenant** dialog box is displayed.

Step 6 After selecting a tenant, click **OK**. The tenant is bound to the user.

----End

Managing Tenants

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Cluster Management** on the left. The current cluster list is displayed.

Step 4 Locate the cluster to be operated and click *Cluster Name* > **Tenant Management**. The **Tenant Management** page is displayed.

Step 5 Select the tenant to be edited and click **Modify** in the **Operation** column. In the displayed dialog box, modify parameters and click **OK**.

Step 6 If the tenant is unavailable, you can also delete the tenant by clicking **Delete** in the **Operation** column. In the displayed confirmation dialog box, click **OK**.

Step 7 To view the monitoring information of a tenant under multi-tenancy, click **View Monitoring Information** in the **Operation** column of the target tenant.

----End

4.7.1.2 Commands for Tenant Management

This section describes the basic syntax and usage of SQL statements for tenant management.

1. Add a tenant.
```
create workload group if not exists test_group properties (
    "cpu_share"="10",
    "memory_limit"="30%",
    "enable_memory_overcommit"="true",
    "max_concurrency" = "10",
    "max_queue_size" = "20",
    "queue_timeout" = "3000"
);
```
2. Modify a tenant.
```
alter workload group test_group properties
('cpu_share'='20','memory_limit'='4%','enable_memory_overcommit'='true','max_concurrency'='15','max
_queue_size'='25','queue_timeout'='4000');
```
3. Bind a user to a tenant: Bind a user to a workload group by setting **user property**. The default value is **normal**.
```
set property for doris_user 'default_workload_group' = 'test_group';
```
4. Query the tenant to which the user belongs.
```
show property for user_name like 'default_workload_group';
```
5. Grant permissions to the user bound to the tenant.
```
GRANT USAGE_PRIV ON WORKLOAD GROUP 'test_group' TO 'jack';
```
6. Revoke the permissions of the old tenant.
```
revoke USAGE_PRIV ON WORKLOAD GROUP 'test_group' from 'jack';
```
7. View tenants.
```
select * from information_schema.workload_groups;
```
8. Delete a tenant.
```
mysql> DROP WORKLOAD GROUP if exists 'test_group';
Query OK, 0 rows affected (0.01 sec)
```

4.7.2 Interconnecting Doris with Data Sources

4.7.2.1 About Doris Multi-Source Data

Hive foreign tables of Doris support CREATE CATALOG. By connecting to Hive Metastore, or a metadata service compatible with Hive Metastore, Doris can automatically obtain Hive database table information and perform data queries. This avoids complex manual mapping and data migration when there is a large number of conventional external data directories.

Multi-Catalog is designed to make it easier to connect to external data catalogs to enhance Doris' data lake analysis and federated data query capabilities. It is a function launched in Doris 1.2.0.

In older versions of Doris, user data is in a two-tiered structure: database and table. Therefore, connections to external catalogs could only be done at the database or table level. For example, you could create a mapping to a table in an external catalog via **create external table**, or to a database via **create external database**. If there were large amounts of databases or tables in the external catalog, you would need to create mappings to them one by one, which could be a heavy workload.

With the advent of Multi-Catalog, Doris now has a new three-tiered metadata hierarchy (including catalog, database, and table), A catalog may directly correspond to an external data catalog.

Basic Concepts

1. Internal Catalog

   Existing databases and tables in Doris are all under the Internal Catalog, which is the default catalog in Doris and cannot be modified or deleted.

2. External Catalog

   You can run the **CREATE CATALOG** command to create an External Catalog, and view the existing catalogs using the **SHOW CATALOGS** command.

3. Switch Catalog

   After login, you will enter the Internal Catalog by default. Then, you can view or switch to your target database via **SHOW DATABASES** and **USE DB**.

   You can run the SWITCH command to switch the catalog. Example:

   ```
   SWITCH internal;
   SWITCH hive_catalog;
   ```

   After switching the catalog, you can view or switch to your target database in that catalog via **SHOW DATABASES** and **USE DB**. Doris automatically passes through databases and tables in Catalog. You can view and access data in External Catalogs the same way as doing that in Internal Catalogs.

   Doris only supports read-only access to data in External Catalogs currently.

4. Delete Catalog

   Databases and tables in external catalogs are read-only. External Catalogs are deletable via the **DROP CATALOG** command. (The Internal catalog cannot be deleted.) You can run the **DROP CATALOG** command to delete an External Catalog.

   This operation only deletes the mapping information of the Catalog in Doris, but does not modify or change the content of any external data catalog.

5. Resource

   Resource is a set of configurations. You can run the **CREATE RESOURCE** command to create a Resource. Then, you can use the Resource when creating a catalog.

   A resource can be used by multiple catalogs to reuse the configuration of the resource.

4.7.2.2 Interconnecting Doris with the Hive Data Source

By connecting to Hive Metastore or a metadata service compatible with Hive Metastore, Doris can automatically obtain Hive database table information and query table data. This avoids complex operations such as manual mapping and data migration when there is a large number of conventional external data catalogs.

Currently, CloudTable does not support the S3 protocol. You are advised to use the OBS protocol for Hive Catalog creation.

Prerequisites

- You have **created an MRS cluster**.
- **Create a Doris cluster.**
- To ensure network connectivity, the MRS cluster must have the same VPC and subnet as the Doris cluster, and all node IP addresses of the Doris cluster must be added to the security group of the MRS cluster.

Step 1: Creating an MRS Hive Connection

Step 1 Log in to **the CloudTable console**.

Step 2 On the **Cluster Management** page, click the target cluster to go to the cluster details page.

Step 3 Choose **Connection Management** > **Create**. The **Create** page is displayed.

Step 4 Set the connection type:

- Set **Authentication Type** to **SIMPLE** and disable **OBS Storage**.
- Set **Authentication Type** to **KERBEROS** and disable **OBS Storage**.
- Set **Authentication Type** to **SIMPLE** and enable **OBS Storage**.
- Set **Authentication Type** to **KERBEROS** and enable **OBS Storage**.

Table 4-38 Parameter description

Parameter	Description
Connection Name	Connection name. You can enter a name indicating the type of the data source to be connected. **NOTE** A connection name must start with a letter and contain 4 to 64 letters, digits, and hyphens (-). It cannot contain other special characters.
Connector	External data directory component supported by Doris. Currently, only Hive is available.
Hadoop Type	Hadoop component supported by Doris. Currently, only MRS is available.
Cluster Name	Name of an MRS cluster. **NOTE** Doris can connect to only one user in a Kerberos cluster.
Manager IP	Floating IP address of MRS Manager. After you select the cluster to be connected, the IP address is automatically filled in.

Parameter	Description
Authentication Type	MRS cluster authentication type. Values of this parameter include **KERBEROS** (for security clusters) and **SIMPLE** for (non-security clusters). After you select the cluster to be connected, this parameter is automatically filled in. **NOTE** Only one KERBEROS connection can be created.
Connector Version	Version of the data source component of the MRS cluster. Hive supports version 3.*x*. The value can be **3.X**.
Username	Username of the MRS cluster. The user must have adequate permissions to access the underlying data of Hive and HDFS. For details about how to create a user, see **Creating an MRS Cluster User**.
Password	Password of the MRS user **mrs_user_name**.
OBS Storage	When this toggle is on, the data table associated with the created catalog connection is stored in OBS. When toggled off, the data table associated with the catalog connection is stored in HDFS.

Step 5 After setting the parameters, click **Test**. If the test fails, check whether the username and password are correct.

Step 6 After the test is complete, click **OK**. The created connection record will be displayed on the **Connection Management** page.

----End

Step 2: Running SQL Commands to Create a Hive Catalog Table

Step 1 Use the SSH login tool to remotely log in to the Linux ECS through the EIP.

For details, see **Logging In to an ECS Using an SSH Password** in the *Elastic Cloud Server User Guide*.

Step 2 Access the Doris cluster. For details, see **Using the MySQL Client to Connect to a Doris Normal Cluster**.

Step 3 Create a Hive catalog table.

- Create a catalog table with the authentication type set to **SIMPLE** to access Hive data stored in HDFS.

```
CREATE CATALOG hive_catalog_simple PROPERTIES (
    'type'='hms',
    'hive.metastore.uris'='thrift://192.X.X.X:port,thrift://192.x.x.x:port',
    'hive.metastore.sasl.enabled' = 'false',
    'dfs.nameservices'='hacluster',
    'dfs.ha.namenodes.hacluster'='3,4',
    'dfs.namenode.rpc-address.hacluster.3'='192.x.x.x:port',
    'dfs.namenode.rpc-address.hacluster.4'='192.x.x.x:port',
    'dfs.client.failover.proxy.provider.hacluster'='***',
```



```
    'hive.version' = '3.1.0'
);
```

- Create a catalog table with the authentication type set to **KERBEROS** to access Hive data stored in HDFS.

```
CREATE CATALOG hive_catalog PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://192.x.x.x:port,thrift://192.x.x.x:port',
    'hive.metastore.sasl.enabled' = 'true',
    'hive.server2.thrift.sasl.qop'='auth-conf',
    'hive.server2.authentication' = 'KERBEROS',
    'hive.server2.authentication.kerberos.principal' = '***',
    'hive.metastore.kerberos.principal' = '***',
    'dfs.nameservices'='hacluster',
    'dfs.ha.namenodes.hacluster'='3,4',
    'dfs.namenode.rpc-address.hacluster.3'='192.x.x.x:port',
    'dfs.namenode.rpc-address.hacluster.4'='192.x.x.x:port',
    'dfs.client.failover.proxy.provider.hacluster'='***',
    'hadoop.security.authentication'='kerberos',
    'hadoop.kerberos.principal' = '{kinit_result}', -- Result of kinit {USER_NAME}
    'hive.version' = '3.1.0',
    'fs.defaultFS'='hdfs://hacluster',
    'hadoop.rpc.protection'='privacy'
);
```

- Create a catalog table with the authentication type set to **SIMPLE** and access Hive data stored in OBS.

```
CREATE CATALOG hive_catalog_simple_obs PROPERTIES (
    'type'='hms',
    'hive.metastore.uris'='thrift://192.x.x.x:port,thrift://192.x.x.x:port',
    'obs.access_key' = '***',
    'obs.secret_key' = '***',
    'obs.endpoint' = '***',
    'obs.region' = '***',
    'hive.metastore.sasl.enabled' = 'true',
    'hive.version' = '3.1.0'
);
```

- Create a catalog table with the authentication type set to **KERBEROS** and access Hive data stored in OBS.

```
CREATE CATALOG hive_catalog_OBS PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://192.x.x.x:port,thrift://192.x.x.x:port',
    'hive.metastore.sasl.enabled' = 'true',
    'hive.server2.thrift.sasl.qop'='auth-conf',
    'hive.server2.authentication' = 'KERBEROS',
    'hive.server2.authentication.kerberos.principal' = '***',
    'hive.metastore.kerberos.principal' = '***',
    'hadoop.security.authentication'='kerberos',
    'hadoop.kerberos.principal' = 'USER_NAME',
    'hive.version' = '3.1.0',
    'fs.defaultFS'='hdfs://hacluster',
    'hadoop.rpc.protection'='privacy',
    'obs.access_key' = '***',
    'obs.secret_key' = '***',
    'obs.endpoint' = '***',
    'obs.region' = '***'
);
```

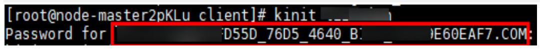
The following table describes related parameters and how to obtain parameter values.

Table 4-39 Parameter description

Parameter	Description
type	Type of the external data to be connected.

Parameter	Description
hive.metastore.uris	Hive metadata URI, which can be viewed in the **hive-site.xml** configuration file.
hive.metastore.sasl.enabled	Obtain the value in the **hive-site.xml** configuration file.
dfs.nameservices	Obtain the value in the **hdfs-site.xml** configuration file.
dfs.ha.namenodes.hacluster	Obtain the value in the **hdfs-site.xml** configuration file.
dfs.namenode.rpc-address.hacluster.3	Obtain the value in the **hdfs-site.xml** configuration file. **NOTE** Obtain the IP address of this parameter on FusionInsight Manager. Log in to the home page of the FusionInsight Manager, choose the target component and click the corresponding instance.
dfs.namenode.rpc-address.hacluster.4	Obtain the value in the **hdfs-site.xml** configuration file. **NOTE** Obtain the IP address of this parameter on FusionInsight Manager. Log in to the home page of the FusionInsight Manager, choose the target component and click the corresponding instance.
dfs.client.failover.proxy.provider.hacluster	Obtain the value in the **hdfs-site.xml** configuration file.
hive.version	Hive version.
hive.server2.thrift.sasl.qop	Obtain the value in the **hive-site.xml** configuration file.
hive.server2.authentication	Authentication type, which can be viewed on the **Connection Management** page.
hive.server2.authentication.kerberos.principal	Obtain the value in the **hive-site.xml** configuration file.
hive.metastore.kerberos.principal	Obtain the value in the **hive-site.xml** configuration file.

Parameter	Description

hadoop.kerberos.principal	Obtain the value of **hadoop.kerberos.principal** as follows:
	– Method 1:
	1. Log in to the MRS Manager page of the cluster.
	2. Choose **System** > **Permission** > **User** to access the user page and obtain the user.
	3. Click **Domain and Mutual Trust** to access the **Domain and Mutual Trust** page and obtain the local domain.
	4. The parameter value is in the format of User name+Local domain.
	– Method 2:
	1. Log in to the node where the MRS client is located as user **root**.
	2. Go to the client installation path.
	`cd /opt/Bigdata/client/`
	3. Run the following command to load the environment variables:
	`source bigdata_env`
	4. Run the **kinit** command to authenticate the user and obtain the parameter value.
	`kinit` *Username*
	

hive.version	Hive version.
fs.defaultFS	Obtain the value in the **core-site.xml** configuration file.
hadoop.rpc.protection	Obtain the value in the **core-site.xml** configuration file.
obs.access_key	Access key. For details about how to obtain the access key, see **Access Keys**. **NOTE** **obs.access_key** can be used to access underlying data files.
obs.secret_key	Secret key. For details about how to obtain the access key, see Access Keys.
obs.endpoint	OBS address. For details about how to obtain the address, see **Endpoints and Domain Names**.
obs.region	OBS region, which can be viewed on the OBS console.

a. Log in to the node (**Master1**) where the MRS client is deployed as user **root**.

b. Go to the **client** directory. This directory contains the folders of all MRS components. Currently, only the folders of HDFS and Hive are displayed.

```
cd /opt/Bigdata/client/
```

> **NOTE**
>
> The HDFS folder contains the **core-site.xml** and **hdfs-site.xml** configuration files and the Hive folder contains the **hive-site.xml** configuration file.

c. Obtain the parameter values in these configuration files.

i. Check the HDFS component.

```
cd ./HDFS/hadoop/etc/
```

Go to the **hadoop** directory.

```
cd hadoop
```

Check the **hdfs-site.xml** configuration file and find the related parameters.

```
cat hdfs-site.xml
```

ii. View the **hive-site.xml** configuration file of the Hive component and search for related parameters.

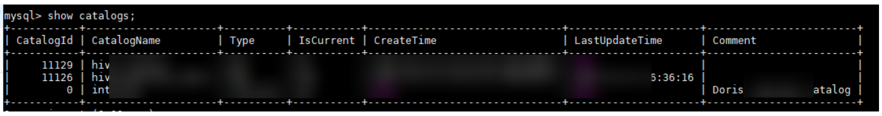
```
cat /opt/Bigdata/client/Hive/config/hive-site.xml
```

----End

Step 3: Querying the Data Mapping Table

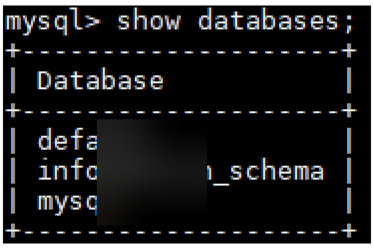
Step 1 View catalogs.

```
show catalogs;
```



Step 2 Run the following command to query the databases in a catalog:

show databases from *catalog name*;



Step 3 Switch to the specified catalog.

switch *catalog name*;

Step 4 View the specified table in a database.

show tables from '*catalog name*'.'*database*';

Specify the table.

select * from '*catalog name*'.'*database name*'.'*table name*';

NOTE

The **use database;** command is intended solely for testing environments with a limited number of data tables. Use this command with extreme caution in production clusters. In environments with a large number of data tables, executing this command will load metadata for all tables, potentially causing a significant surge in pressure on the Hive Metastore. This increased load can negatively impact the stability and performance of the MRS production environment.

----End

FAQs

- If the following error message is displayed, perform an active/standby switchover for HDFS nodes on FusionInsight Manager.

```
mysql> select * from hive_hdfs_test2;
ERROR 1105 (HY000): errCode = 2, detailMessage = get file split failed for table: hive_hdfs_test2, err: 
org.apache.hadoop.ipc.RemoteException: Operation category READ is not supported in state standby. 
Visit https://s.apache.org/sbnn-error
    at 
org.apache.hadoop.hdfs.server.namenode.ha.StandbyState.checkOperation(StandbyState.java:108)
    at org.apache.hadoop.hdfs.server.namenode.NameNode
$NameNodeHAContext.checkOperation(NameNode.java:2200)
    at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkOperation(FSNamesystem.java:
```

- When creating a Catalog, you can use the parameter **file.meta.cache.ttl-second** to set the automatic expiration time of the Hive partition file cache, or set this value to **0** (unit: second) to disable the partition file cache. The following provides an example.

```
CREATE CATALOG hive_catalog PROPERTIES (
    'type'='hms',
    'hive.metastore.uris' = 'thrift://127.x.x.x:port',
    'obs.access_key' = '***',
    'obs.secret_key' = '***',
    'obs.endpoint' = '***',
    'obs.region' = '***'
    'file.meta.cache.ttl-second' = '60'
);
```

4.8 Managing Doris Clusters

4.8.1 Checking the Doris Cluster Status

The cluster list contains all CloudTable clusters. You can view clusters in various states. If a large number of clusters are involved, navigate through multiple pages to view all of the clusters.

Clusters are listed in chronological order by default, with the in-service clusters displayed at the top.

Table 4-40 Cluster management parameters

Parameter	Description
Cluster Name	Name of a cluster. Set this parameter when creating a cluster.

Parameter	Description
HA	When you create a Doris cluster with node specifications less than 8 vCPUs and 32 GB memory, HA is enabled by default, and "The cluster can be used only for testing services." is displayed. If HA is not enabled, "--" is displayed.
Cluster Status	Cluster status, including **Creating**, **In service**, **Sub-health**, **Frozen**, and **Creation failed**.
Task Status	Task status of a cluster.
Database Engine	Doris
Cluster Version	Cluster version.
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** ● You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. ● You can delete a user or multiple users. ● After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.
Created	Time when a cluster is created
Billing Mode	Cluster billing mode.
Access Address (Intranet)	Private network link address.

Parameter	Description
Operation	• Choose **More** > **Delete** to delete a pay-per-use cluster. • Choose **More** > **Restart** to restart a cluster. • Click **Monitor** to access the CloudTable service monitoring page. • Capacity expansion: Expand the compute specifications (specification expansion), storage specifications (disk expansion), and number of nodes (node scale-out) in the cluster. For details, see **Adjusting the Capacity of a Doris Cluster**. • Click **More** and then click **Change to Yearly/Monthly** in the **Operation** column to change the billing mode to yearly/ monthly. • Click **More** > **Unsubscribe** in the **Operation** column to unsubscribe from the yearly/monthly cluster. • Choose **More** > **Renew** to renew a yearly/monthly cluster. • Choose **More** > **View Order Details** to view the order details of a cluster. • Choose **More** > **Enable Auto-Renewal** to enable auto-renewal for a yearly/monthly cluster.

Table 4-41 Icon description

Icon	Description
	Click to view all projects.
	Click to refresh the cluster list.
	Enter a cluster name in the search box and click to search for it.

Cluster Status

Table 4-42 Cluster status description

Status	Description
Creating	Indicates that a cluster is being created.
In service	If a cluster is successfully created and can provide services, the cluster status is **In service**.

Status	Description
Sub-health	If the cluster status cannot be monitored within the specified time, the cluster status changes to **Sub-health**. Manual intervention is required to recover a cluster that is in **Sub-health** status. For example, you can restart the cluster to recover the cluster.
Creation failed	Indicates that a cluster fails to be created.
Frozen	If the balance is insufficient for renewing a cluster, the cluster status is **Frozen**. If a cluster status is **Frozen**, you need to renew your subscription and ensure that your account balance is not 0 before unfreezing the cluster. **NOTE** A frozen cluster is unavailable and its all ECSs are shut down. After being unfrozen, the cluster recovers to the **In service** state. If you do not renew the cluster before the freeze period ends, the cluster will be deleted.

Task Status

Table 4-43 Task status description

Status	Description
Deleting	Indicates that a cluster is being deleted.
Restarting	Indicates that a cluster is being restarted.
Modifying setting	Indicates that the cluster parameters are being modified.
Scaling out	Indicates that cluster nodes are being scaled out.
Expanding disk capacity	Indicates that the disk capacity of the cluster is being expanded.
Resizing flavor	Indicates that the cluster specifications are being changed.

4.8.2 Viewing Doris Cluster Details

You can monitor and manage the clusters you create. In the cluster list, locate the cluster to be viewed and click the cluster name to access the basic information page. You can view the cluster information and network configurations.

Table 4-44 Cluster information

Parameter	Description
Cluster Name	Name of a cluster. Set this parameter when creating a cluster.
Cluster ID	Cluster ID
Cluster Status	Cluster status information
Cluster Version	Kernel version of the cluster.
Access Address	Address for accessing the cluster. You can click to copy the access link to the clipboard.
Billing Mode	Billing mode of the cluster
Created	Time when a cluster is created
Database Engine	A core service used to store, process, and protect data.
Broker Process	Whether to enable the Broker process, which is used to import data for customers.
Enable Https	If **Enable Https** is toggled on, you can download the certificates in a secure environment.
Cluster Storage Mode	The cluster storage mode is coupled storage and compute.
Enable Interface Authentication	After this function is enabled, account and password authentication is required for REST interfaces in the cluster. **NOTE** This parameter is displayed when **Enable Https** is toggled on.
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** • You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. • You can delete a user or multiple users. • After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.

Table 4-45 Network configuration

Parameter	Description
Region	Working area of the cluster. Set this parameter when creating a cluster.

Parameter	Description
AZ	AZ you select during cluster creation.
VPC	VPC you select during cluster creation. A VPC is a secure, isolated, logical network environment.
Subnet	Subnet you select during cluster creation. A subnet provides dedicated network resources that are logically isolated from other networks, improving network security.
Security Group	Security group you select during cluster creation. If a security group needs to be added or modified due to service changes, click the pen icon on the right of **Security Group** in the **Network Configuration** area on the cluster details page to add or modify the security group. **NOTE** Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.

Table 4-46 FE node configuration

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: • 8U16G • 16U32G • 32U64G • 64U128G • 4U16G • 8U32G • 16U64G • 32U128G • 64U256G

Parameter	Description

Storage	Select the disk specifications and capacity of the Doris compute node.
	NOTE
	• Select the disk specifications and capacity of the Doris compute node.
	  • High I/O
	  • General-purpose SSD
	  • Ultra-high I/O
	  • Extreme SSD
	• The capacity ranges from 200 GB to 2,000 GB per node.

Nodes	Specify the number of nodes in the cluster. You can add 3 or 5 FE nodes.

Table 4-47 BE node configuration

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: • 8U16G • 16U32G • 32U64G • 64U128G • 4U16G • 8U32G • 16U64G • 32U128G • 64U256G

Storage	Select the disk specifications and capacity of the Doris compute node.
	NOTE
	• Available storage specifications:
	  • High I/O
	  • General-purpose SSD
	  • Ultra-high I/O
	  • Extreme SSD
	• The capacity ranges from 400 GB to 10,000 GB per node.

Parameter	Description
Nodes	Specify the number of nodes in the cluster. You can add 3 to 100 BE nodes.

4.8.3 Restarting a Doris Cluster Node

If a CloudTable cluster node is abnormal, you can restart the node to restore the node status.

Precautions

- The node is unavailable during the restart.
- To minimize service disruption, schedule the node restart during off-peak hours.
- Disk scale-out, node scale-out, and specification expansion functionalities will be temporarily disabled during the node restart.
- Please note that a node restart is a process restart, not a full node reboot.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 Click the name of the cluster to be operated. The cluster details page is displayed.

Step 5 On the cluster details page, choose **Operation** > **Restart**. The **Restart Node** dialog box is displayed.

Step 6 Enter **RESTART** or click **Auto Enter**, and click OK to restart the node.

After the node is restarted, the **Restart** button is unavailable.

Step 7 Check the node restart result. If the restart is successful, the **Restart** button is highlighted. If the node fails to be restarted, the task status is **Failed to restart the node**, and the **Restart button** is highlighted, the node can be restarted again.

----End

Node Restart Statuses

Table 4-48 Restart statuses

Status	Description
Restart	The Restart button is highlighted, indicating either no restart operation has been initiated or that the restart has been completed successfully

Status	Description
Restarting	The cluster node is being restarted, and **Task Status** in **Cluster Information** is **Restarting**.
Restart failed	If a cluster node fails to be restarted and **Task Status** is **Failed to restart the node**, you can continue to restart the node.

4.8.4 Restarting the Doris Cluster

If a cluster is in the unbalanced state or cannot work properly, you may need to restart it for restoration. After modifying a cluster's configurations, such as security settings and parameters, manually restart the cluster to make the configurations take effect.

NOTE

- If your cluster is in arrears, this function may be unavailable. Please top up your account in time.
- The function is unavailable when the cluster status is subhealthy. Please contact technical support for assistance with restoring the cluster.

Impact on the System

- A cluster cannot provide services during the restart. Therefore, before the restart, ensure that no task is running and all data is saved.
- If a cluster is processing transactional data, for example, importing data, querying data, files may be damaged or the cluster may fail to be restarted once the cluster is restarted. You are advised to stop all cluster tasks before restarting a cluster.
- If the restart fails, the cluster may be unavailable. Try again later or contact technical support.

Procedure

Step 1 Log in to **the CloudTable console**.

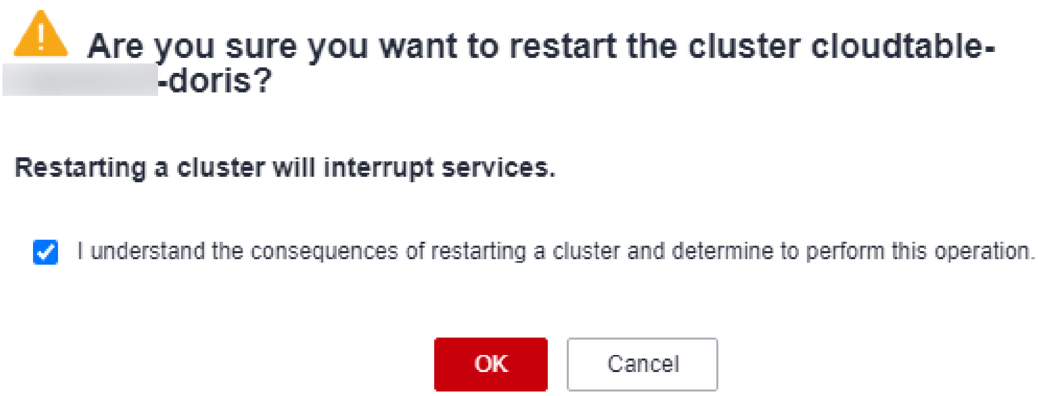
Step 2 Select a region in the upper left corner.

Step 3 In the upper right corner of the cluster list, enter the name of a cluster in the search box and click .

Step 4 In the **Operation** column of the cluster, click **More** > **Restart**.

Step 5 In the dialog box that is displayed, select the check box and click **OK** to restart the cluster.

Figure 4-5 Confirming the restart



----End

4.8.5 Deleting a Doris Cluster

If a cluster is no longer needed, you can delete, unsubscribe from, or release the cluster. Deleting, unsubscribing from, or releasing a CloudTable cluster will clear all resources and data related to the cluster. This operation cannot be undone. Exercise caution when deleting a cluster.

- Pay-per-use clusters can be directly deleted. For details, see **Deleting a Pay-per-Use Doris Cluster**.
- Yearly/monthly clusters cannot be directly deleted. You need to unsubscribe from them (if they have not expired) or release them (if they have expired but have not been renewed). For details, see **Unsubscribing from or Releasing a Yearly/Monthly Cluster**.

Deleting a Pay-per-Use Doris Cluster

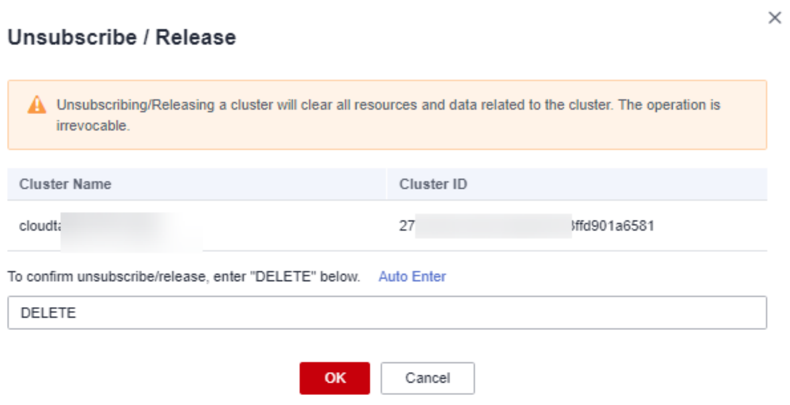
Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the **Operation** column of the cluster, click **More** > **Delete**.

Step 4 In the displayed dialog box, enter **DELETE** or click **Auto Enter**, and click **OK** to delete the cluster.

Figure 4-6 Confirming the deletion



----End

Unsubscribing from or Releasing a Yearly/Monthly Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the **Operation** column of the cluster, choose **More** > **Unsubscribe/Release**.

Step 4 In the displayed dialog box, enter **DELETE** or click **Auto Enter**, and click **OK** to unsubscribe from or release the cluster.

Step 5 On the **Unsubscribe** page, confirm the cluster information, select reasons for unsubscription, and confirm the unsubscription amount and related fees.

Step 6 Select "I've backed up the data or confirmed that the unsubscribed resources are no longer needed." and "I understand that only resources in the recycle bin can be restored after unsubscription". View the recycle bin description and click **Unsubscribe**.

Step 7 Return to the console and check whether the cluster has been unsubscribed from or released.

----End

4.9 Doris Cluster O&M

4.9.1 Adjusting the Capacity of a Doris Cluster

4.9.1.1 Overview

You can perform capacity expansion on the console if you need more compute or storage resources. There are two methods for cluster capacity expansion, that is, **adding nodes** (node scale-out) and **expanding disk capacity** (vertical expansion). Doris allows you to add FE and BE nodes with easy operations. Generally, you can expand the number of FE nodes to more than three to achieve high availability of these nodes. The scale-out of BE nodes does not affect system running, the system performance, and the ongoing tasks. Data is automatically balanced and the cluster will be restored to the load balancing state within several hours to one day. Currently, Doris clusters do not support vertical scaling. Nonetheless, Doris inherently offers a variety of strategies to achieve vertical scaling. Integration of vertical scaling services into Doris clusters is necessary to leverage these capabilities.

NOTE

New nodes are charged based on the billing mode of the current cluster.

Precautions

- You can perform capacity expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as specification change, node scale-out, and disk capacity expansion).
- The number of compute units to be added must be no bigger than the remaining quotas. Otherwise, the system will prompt a message indicating that capacity expansion is not allowed.
- During node scale-out, the system does not automatically restart the cluster, ensuring service continuity and smooth capacity expansion. During vertical capacity expansion, the cluster stops running. Each node in the cluster is restarted only after all parameters are set.
- If the capacity expansion fails, **Unit capacity expansion failed** is displayed in the **Task Status** column, and the cluster automatically rolls back to the state before capacity expansion. You can try again.

4.9.1.2 Doris Node Scale-out

The node scale-out is performed by adding FE or BE nodes in the cluster and this operation does not affect system running.

Constraints

- The number of FE nodes after the scale-out must be an odd number, such as **3** or **5**.
- The number of BE nodes after the scale-out ranges from 4 to 100.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management** to view the clusters.

Step 3 In the cluster list, locate the row that contains the target cluster and choose **More** > **Scale Out** in the **Operation** column.

NOTE

You can also click **Scale Out** on the cluster details page to go to the **Scale Out** page.

Step 4 Set **Configured Node** to **Frontend Nodes** or **Backend Nodes** as required.

Step 5 Click **+** to add nodes.

Step 6 Confirm the fee and resource quota, and click **OK**.

----End

Scale-out Statuses

Table 4-49 Node scale-out statuses

Task Status	Description
Scaling out	Indicates that a cluster is being scaled out.
Scale-out failed	Indicates that the cluster fails to be scaled out.

Viewing Scaling Details

- After you click **OK**, the cluster task status changes to **Scaling out**. After the cluster scale-out is complete, the cluster status changes to **In service**.
- After you click **OK**, the cluster task status changes to **Scaling out**. If the scale-out fails, the cluster status is **In service** and the task status is **Scale-out failed**.

4.9.1.3 Expanding the Disk Capacity of a Doris Cluster

The vertical capacity expansion is performed by expanding disk capacity. Data is usually stored on the BE nodes, and disk expansion is required when the disk capacity of the BE nodes is insufficient.

NOTE

During disk capacity expansion, cluster services may jitter.

Prerequisites

You can perform disk capacity expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as scale-out, parameter modification, and specification change). Note that:

- The storage capacity of the instance to be expanded is measured in GB.
- The maximum disk capacity of a FE node after expansion is 2,000 GB.
- The maximum disk capacity of a BE node after expansion is 10,000 GB.
- The disk capacity after expansion is a multiple of 100 GB.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management** to view the clusters.

Step 3 In the cluster list, locate the row that contains the target cluster and click **Disk Expansion** in the **Operation** column. If you fail to find **Disk Expansion**, click **More** and choose **Disk Expansion** in the **Operation** column.

NOTE

You can also click **Disk Expansion** on the cluster details page to go to the page for expanding disk capacity.

Step 4 Set **Configured Node** to **Frontend Nodes** or **Backend Nodes** as required.

Step 5 Press **+** to increase nodes.

Step 6 Confirm the fee and resource quota, and click **OK**.

----End

Disk Expansion Status

Table 4-50 Disk expansion status description

Status	Description
Disk expanding	This status is shown in **Task Status**, indicating that a cluster is being scaled out.
In service (cluster status)	This status is shown in **Cluster Status**, indicating that the scale-out is complete and the cluster can provide services.
Disk capacity expansion failed (task status)	This status is shown in **Task Status**, indicating that the cluster fails to be scaled out.

Viewing Disk Expansion Information

After you click **OK**, the cluster task status changes to **Disk expanding**. After the expansion is complete, the cluster status changes to **In service**.

4.9.1.4 Changing the Specifications of a Doris Cluster

Constraints

- You can only increase cluster specifications. To decrease cluster specifications, create a new cluster with the desired specifications and migrate data from the existing cluster.
- You can change the specifications of only one node type (the BE or FE node) at a time. After the change, nodes in other types still maintain their original specifications.
- Modifying node specifications, especially with large data volumes, can be time-consuming. We recommend performing such operations during periods of low system utilization.
- The entire cluster becomes temporarily unavailable during the specification change. Subsequent specification changes are not permitted until the current operation completes.

- To maintain system stability, repeated specification changes within a short timeframe are not supported. Operational frequency of such changes must be limited.
- The system is unavailable during the specification change.
- In pay-per-use billing mode, the change fee refers to the hourly unit price after the scale-out.

Prerequisites

You can perform specification expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as node scale-out, disk expansion, and parameter modification).

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management** to view the clusters.

Step 3 In the cluster list, locate the row that contains the target cluster and choose **More** > **Specification Expansion** in the **Operation** column.

NOTE

You can also click **Specification Expansion** on the cluster details page to switch to the **Specification Expansion** page.

Step 4 Set **Configured Node** to **Frontend Nodes** or **Backend Nodes** as required.

Step 5 Select the CPU and memory from the node specifications drop-down list.

Step 6 Confirm the fee and resource quota, and click **OK**.

After you click **OK**, the cluster status changes to **Sub-health** and the task status changes to **Resizing Flavor**. After the cluster specifications are changed, the cluster status changes to **In service** and the task status is cleared.

----End

Specification Expansion Status

Table 4-51 Specification expansion status

Status	Description
Resizing Flavor	Indicates that the specifications of the target cluster are being changed.
In service	Indicates that the specifications of the target cluster are changed and the cluster can provide services.
Resizing flavor failed	Indicates that the specifications of the target cluster failed to be changed.

4.9.2 Modifying Doris Parameters to Optimize Cluster Performance

In data analysis scenarios, you need to adjust the configuration parameters of Doris clusters as required to optimize performance. However, manually adjusting the configuration parameters is time-consuming and error-prone, which leads to fact that the cluster performance cannot reach the optimal state. You can adjust the configuration parameters of Doris clusters on the console for multiple times to optimize cluster performance, thereby improving data processing efficiency and system response speed.

Constraints

- Modifying the static parameters of a cluster requires a cluster restart, which may interrupt services. To avoid impacting services, perform this operation during off-peak hours.
- You can modify parameters only when no task is running in the cluster.
- Do not modify cluster parameters when the cluster is being restarted.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the navigation pane on the left, choose **Cluster Management**.

Step 4 In the cluster list, locate the target cluster, click the cluster name, and go to the **Parameter Configuration** tab page.

Step 5 Select the target FE or BE node, select the parameter to be modified, and click in the **Value** column.

Step 6 Enter a new value in the text box and click . The system displays a message "The parameter changed to *xx* successfully. Save the modified value." The new value is followed by a red asterisk (*).

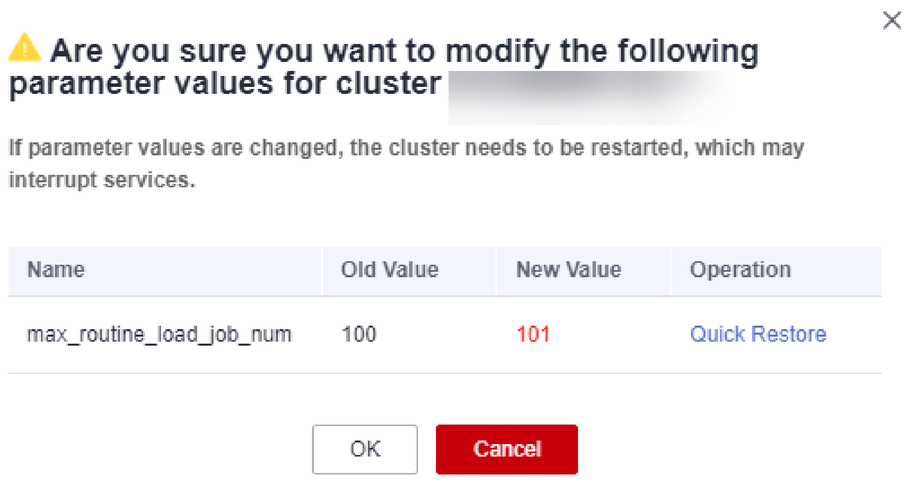
To cancel the modification, click .

Figure 4-7 Modifying a parameter

Step 7 Click **Save Changes** in the upper left corner above the parameter list. A dialog box is displayed.

Step 8 Check the parameter settings and click **OK**.

Figure 4-8 Saving the modification



- Click  in the upper right corner. If **Status** of the parameter is **Applied**, the cluster does not need to be restarted. If **Status** is **Unapplied**, you need to manually restart the cluster for the modification to take effect.
- Click **Restart** in the upper left corner of the **Parameter Configuration** page. Alternatively, return to the cluster list to restart the cluster.

  Manually restart the cluster after the static parameters are modified. Dynamic parameters take effect after being modified. You do not need to restart the cluster.
- If the new parameter value is incorrect, click **Quick Restore** to cancel the modification.

Step 9 After modifying parameters, you can click the **Change History** tab to view the change history. The following table lists the parameters displayed on the **Change History** page.

Table 4-52 Parameters

Parameter	Description
Name	Name of the modified parameter
Old Value	Parameter value before modification
New Value	Parameter value after modification
Modified	Time when the parameter was modified

----End

Doris Parameter Modification Status

Table 4-53 Parameter statuses

Status	Description
Unmodified	The parameter has not been modified.
Unapplied	The parameter has been modified but has not been applied.
Applying	This is a static parameter and you need to restart the cluster after saving the modification.
Applied	The modification of the parameter takes effect. ● After the modification of a static parameter is saved, the cluster needs to be restarted for the modification to take effect. ● The modification of a dynamic parameter takes effect after being saved.

FE Node Parameters

Table 4-54 FE node parameters

Parameter	Static	Description
max_routine_load_job_num	No	● Description: The total number of routine import jobs, including the jobs in the **NEED_SCHEDULED**, **RUNNING**, and **PAUSE** states. After this parameter value is exceeded, no new jobs can be submitted. ● Type: int ● Value range: (0,300) ● Default value: **100**
use_new_tablet_scheduler	Yes	● Description: Whether to enable the new replica scheduling mode. ● Constraints: This parameter is only available for storage-compute coupled clusters. ● Type: boolean ● Value range: [true,false] ● Default value: **true**

Parameter	Static	Description
audit_plugin_max_sql_length	No	● Description: Maximum length of an executed SQL statement recorded in Doris audit logs. The excess part will be truncated. If the value is too large, too much memory may be occupied, affecting cluster availability. Exercise caution when adjusting the value. ● Type: long ● Value range: (0,9223372036854775807) ● Default value: **4096**
max_broker_concurrency	No	● Description: Maximal concurrency of broker scanners. ● Type: int ● Value range: (5,30) ● Default value: **10**
tablet_repair_delay_factor_second	No	● Description: For different scheduling priorities, we will delay different time to start repairing in order to prevent a large number of unnecessary replica repair tasks from occurring in the process of routine restart and upgrade. This parameter is a reference coefficient. For HIGH priority, the delay is the reference coefficient multiplied by 1; for NORMAL priority, the delay is the reference coefficient multiplied by 2; for LOW priority, the delay is the reference coefficient multiplied by 3. That is, the lower the priority, the longer the delay waiting time. If the user wants to repair the replica as soon as possible, this parameter can be reduced appropriately. ● Type: long ● Value range: (0,9223372036854775807) ● Default value: **60**

Parameter	Static	Description
balance_load_score_threshold	No	● Description: Threshold of the cluster balance percentage. The default value is 0.1, that is, 10%. When the load core of a BE node is not higher than or less than 10% of the average load core, we think that the node is balanced. If you want to make the cluster load more even, you can adjust this parameter appropriately. ● Type: double ● Value range: (0,1) ● Default value: **0.1**
dynamic_partition_check_interval_seconds	No	● Description: The execution frequency of dynamic partition threads. The default value is **600** (10 minutes), that is, scheduling is performed every 10 minutes. Unit: second ● Type: long ● Value range: (0,9223372036854775807) ● Default value: **600**
max_routine_load_task_concurrent_num	No	● Description: Limit on the maximum number of concurrent subtasks for a routine import job. It is advised to keep the default value. Setting it too large may result in too many concurrent tasks and consume cluster resources. ● Type: int ● Value range: (0,10) ● Default value: **5**
dynamic_partition_enable	No	● Description: Whether to enable Doris' dynamic partition feature. The default value is **false**, indicating that the dynamic partition feature is disabled. This parameter only affects the partitioning operation of dynamic partition tables, not normal tables. ● Type: boolean ● Value range: [true,false] ● Default value: **true**

Parameter	Static	Description
max_bytes_per_broker_scanner	No	• Description: Maximum bytes a broker scanner can process in one broker load job. Generally, each BE has a broker scanner. Maximum amount of data processed by a single BE. The default value is 500 GB. If the amount of data imported by a single BE is greater than the **max_bytes_per_broker_scanner** value, an error is reported. In this case, you need to increase the value of **max_bytes per_broker_scanner** or adjust the number of concurrent import tasks. The calculation logic is as follows: Number of concurrent import tasks = Math.min (Source file size/Minimum processing volume **min_bytes_per_broker scanner**, Maximum number of concurrent tasks **max broker concurrency**, Number of current BE nodes). Processing volume of a single BE = Source file size/Number of concurrent import tasks. The default value of **min_bytes_per_broker scanner** is 64 MB, and the default value of **max broker concurrency** is 10. Unit: byte • Type: long • Value range: (0,9223372036854775807) • Default value: **3221225472**

BE Node Parameters

Table 4-55 BE node parameters

Parameter	Static	Description
max_consumer_num_per_grou	Yes	• Description: Maximum number of consumers in a data consumer group, used for routine load. • Type: int • Value range: (0,10) • Default value: **3**

Parameter	Static	Description
clone_worker_count	Yes	• Description: Number of threads used to perform cloning tasks. The value affects the speed of replica equalization. In the case of low disk pressure, you can speed up replica balancing by adjusting this parameter. • Type: int • Value range: (0,10) • Default value: **3**

4.9.3 Using Cloud Eye to Monitor a Doris Cluster

4.9.3.1 Doris Cluster Monitoring Metrics

Description

Monitoring is critical to ensure CloudTable reliability, availability, and performance. You can monitor the running status of CloudTable servers. This section describes the metrics that can be monitored by Cloud Eye as well as their namespaces and dimensions.

Namespace

SYS.CloudTable

FE node monitoring metrics

Table 4-56 lists the FE node monitoring metrics.

Table 4-56 FE node monitoring metrics

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
cmdProcessMem	Memory utilization	Memory utilization rate of the monitored object	0~100	%	N/A	CloudTable instance node	60s	SYS.CloudTable
cmdProcessCPU	CPU utilization	CPU utilization of the monitored object	0~100	%	N/A	CloudTable instance node	60s	SYS.CloudTable
cmdForUsedStorageRate	Used storage rate	Ratio of the used storage space to the total storage space in the cluster	0~100	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_image_clean_failed	Failed historical metadata image clearing	The operation should not fail. If it fails, contact O&M engineers.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_image_clean_success	Successful Historical Metadata Image Clearing	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_image_push_success	Successful metadata image pushes to FE nodes	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_image_write_failed	Failed metadata image generation	The operation should not fail. If it fails, contact O&M engineers.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_image_write_success	Successful metadata image generation	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_max_journal_id	Maximum metadata log ID of the FE node	For the master frontend node, this value indicates the maximum ID of the metadata log that is being written. For other frontend nodes, the value indicates the maximum ID of the metadata log that is being replayed. If the ID difference between these frontend nodes is too much, metadata synchronization is abnormal.	≥ 0	N/A	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_max_tablet_compaction_score	Maximum compaction score of all BE nodes	The maximum compaction score of the current cluster. If the score is too high, there may be query or write latency.	≥ 0	N/A	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_qps	FE node QPS (query requests only)	QPS	≥ 0	Count/s	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_query_err	Total failed queries	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_query_err_rate	Number of error queries per second	-	≥ 0	Count/s	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_query_latency_ms_99	99th percentile query latency	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_query_latency_ms_999	999th percentile query latency	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_query_olap_table	Query requests to an internal table (OLAPTable)	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_query_total	Number of all query requests	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_report_queue_size	Queue length of BE periodic report tasks on the FE node	This value indicates the number of report tasks waiting on the master FE node. A large value indicates a poorer FE processing capability.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_request_total	Operation requests (query and other statements) received through the MySQL port	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_routine_load_error_rows	Total error rows of all Routine Load jobs in the cluster	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_routine_load_receive_bytes	Size of data received by all Routine Load jobs in the cluster	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
doris_fe_routine_load_rows	Number of data rows received by all Routine Load jobs in the cluster	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_rps	FE Node QPS (Query and Other Statements)	This metric is used together with QPS to monitor the number of requests processed by the cluster.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_scheduled_tablet_num	Tablets being scheduled by the master FE node	Copies being repaired and balanced are included. This value reflects the number of tablets that are being migrated in the current cluster. If there always are such tablets, the cluster is unstable.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_tablet_status_count_added	Number of tablets scheduled on the master FE node	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_tablet_status_count_in_sched	Number of tablets that are repeatedly scheduled on the master FE node	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_tablet_status_count_not_ready	Number of tablets that do not meet the scheduling triggering conditions on the master FE node	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_tablet_status_count_total	Number of tablets that have been checked on the master FE node	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_tablet_status_count_unhealthy	Unhealthy tablets on the master FE node	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_txn_counter_begin	Number of submitted transactions	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_txn_counter_failed	Number of failed transactions	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_txn_counter_reject	Number of rejected transactions	If the number of running transactions exceeds the threshold, new transactions will be rejected.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_txn_counter_success	Number of successful transactions	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_txn_exec_latency_ms_99	99th percentile transaction execution duration	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_txn_exec_latency_ms_999	999th percentile transaction execution duration	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_txn_publish_latency_ms_99	99th percentile publish operation duration	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_txn_publish_latency_ms_999	999th percentile publish operation duration	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_heap_size_bytes_max	Maximum heap memory	The metric is used for observing the JVM memory usage.	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
jvm_heap_size_bytes_committed	Allocated heap memory	The metric is used for observing the JVM memory usage.	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_heap_size_bytes_used	Used heap memory	The metric is used for observing the JVM memory usage.	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_non_heap_size_bytes_committed	Allocated off-heap memory	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_non_heap_size_bytes_used	Used off-heap memory	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
jvm_old_gc_count	Old generation GC	This metric is used to check whether Full GC persists for a long time.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_old_gc_time	Old generation GC duration	This metric is used to check whether Full GC persists for a long time.	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_old_size_bytes_used	Old generation memory usage	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_old_size_bytes_peak_used	Peak memory usage of the old generation	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
jvm_old_size_bytes_max	Maximum old generation memory	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_thread_new_count	Peak thread count	This metric is used to check whether there are too many JVM threads.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_thread_new_count	Number of threads in the new state	This metric is used to check whether there are too many JVM threads.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_thread_runnable_count	Number of threads in the runnable state	This metric is used to check whether there are too many JVM threads.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
jvm_thread_blocked_count	Number of threads in the blocked state	This metric is used to check whether there are too many JVM threads.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_thread_waiting_count	Number of threads in the waiting state	This metric is used to check whether there are too many JVM threads.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_thread_terminated_count	Number of threads in the terminated state	This metric is used to check whether there are too many JVM threads.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_young_gc_count	Young generation GC count	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
jvm_young_gc_time	Young generation GC duration	Cumulative value	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
jvm_young_size_bytes_used	Young generation memory usage	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_young_size_bytes_peak_used	Peak memory usage of the young generation	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
jvm_young_size_bytes_max	Maximum young generation memory	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_cache_added_partition	New partition caches	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_cache_added_sql	New SQL caches	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_cache_hit_partition	Hit partition caches	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_cache_hit_sql	Hit SQL caches	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_connection_total	FE MySQL port connections	This metric is used to monitor the number of query connections. If the number of connections exceeds the upper limit, new connections cannot be established.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_counter_hit_sql_block_rule	Queries intercepted by SQL block rule	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_edit_log_clean_failed	Failed historical metadata log clearing	The operation should not fail. If it fails, contact O&M engineers.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_edit_log_clean_success	Successful historical metadata log clearing	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_edit_log_read	Metadata log reads	The slope indicates the metadata read frequency.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_edit_log_write	Metadata log writes	The slope indicates the metadata read frequency.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_fe_image_push_failed	Failed metadata image pushes to FE nodes	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Table 4-57 Custom monitoring metrics of FE nodes

Metric Name	Metric	Description	Value Range	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_thrift_rpc_total_{method_name}	doris_fe_thrift_rpc_total_{method_name}	RPC requests received by each method of the FE thrift interface.	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thrift_rpc_latency_ms_{method_name}	doris_fe_thrift_rpc_latency_ms_{method_name}	Duration of RPC requests received by each method of the FE thrift interface.	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_thrift_server_pool_active_thread_num	doris_fe_thread_pool_thrift_server_pool_active_thread_num	Number of tasks that are being executed in the thread pool **thrift-server-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_thrift_server_pool_active_thread_pct	doris_fe_thread_pool_thrift_server_pool_active_thread_pct	Percentage of the number of tasks that are being executed in the thread pool **thrift-server-pool** to the maximum number of threads	[0, 100]	CloudTable instance node	60s	Service.CloudTable

Metric Name	Metric	Description	Value Range	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_thread_pool_thrift_server_pool_task_in_queue	doris_fe_thread_pool_thrift_server_pool_task_in_queue	Number of tasks that are queuing in the thread pool **thrift-server-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_thrift_server_pool_task_rejected	doris_fe_thread_pool_thrift_server_pool_task_rejected	Number of tasks rejected by the thread pool **thrift-server-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_mysql_nio_pool_active_thread_num	doris_fe_thread_pool_mysql_nio_pool_active_thread_num	Number of tasks that are being executed in the thread pool **mysql-nio-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_mysql_nio_pool_active_thread_pct	doris_fe_thread_pool_mysql_nio_pool_active_thread_pct	Percentage of the number of tasks that are being executed in the **mysql-nio-pool** thread pool to the maximum number of threads.	[0, 10]	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_mysql_nio_pool_task_in_queue	doris_fe_thread_pool_mysql_nio_pool_task_in_queue	Number of tasks that are queuing in the thread pool **mysql-nio-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable

Metric Name	Metric	Description	Value Range	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_thread_pool_mysql_nio_pool_task_rejected	doris_fe_thread_pool_mysql_nio_pool_task_rejected	Number of tasks rejected by the thread pool **mysql-nio-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_connect_scheduler_pool_active_thread_num	doris_fe_thread_pool_connect_scheduler_pool_active_thread_num	Number of tasks that are being executed in the thread pool **connect-scheduler-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_connect_scheduler_pool_active_thread_pct	doris_fe_thread_pool_connect_scheduler_pool_active_thread_pct	Percentage of the number of tasks that are being executed in the thread pool **connect-scheduler-pool** to the maximum number of threads.	[0, 100]	CloudTable instance node	60s	Service.CloudTable
doris_fe_thread_pool_connect_scheduler_pool_task_in_queue	doris_fe_thread_pool_connect_scheduler_pool_task_in_queue	Number of tasks that are queuing in the thread pool **connect-scheduler-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable

Metric Name	Metric	Description	Value Range	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_fe_thread_pool_connect_scheduler_pool_task_rejected	doris_fe_thread_pool_connect_scheduler_pool_task_rejected	Number of tasks rejected by the thread pool **connect-scheduler-pool**	≥ 0	CloudTable instance node	60s	Service.CloudTable

BE node monitoring metrics

Table 4-58 lists the BE node monitoring metrics.

Table 4-58 BE node monitoring metrics

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_active_scan_context_count	Scanners enabled externally	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_add_batch_task_queue_size	Queue size of the thread pool for receiving batches	-	≥ 0	N/A	N/A	CloudTable instance node	60s	SYS.CloudTable
cmdForUsedStorageRate	Used storage rate	Ratio of the used storage space to the total storage space in the cluster	0~100	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_brpc_endpoint_stub_count	Created brpc stubs (BE nodes)	These stubs are used for interactions between BE nodes.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_brpc_function_endpoint_stub_count	Created brpc stubs (remote RPCs)	These stubs are used for interactions between remote RPCs.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_cache_usage_LastestSuccessChannelCache	LRU ChannelCache usage	LRU DataPage Cache usage	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_cache_usage_ratio_DataPageCache	LRU DataPageCache utilization	-	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_cache_usage_ratio_IndexPageCache	LRU IndexPage Cache utilization	-	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_cache_usage_ratio_SegmentCache	LRU SegmentCache Utilization	-	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_cache_hit_ratio_DataPageCache	LRU DataPageCache hit ratio	The data cache directly affects the query efficiency.	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_cache_hit_ratio_IndexPageCache	LRU IndexPage Cache hit ratio	The index cache directly affects the query efficiency.	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_cache_hit_ratio_LastestSuccessChannelCache	LRU ChannelCache hit ratio	-	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_cache_hit_ratio_SegmentCache	LRU SegmentCache hit ratio	-	[0,100]	%	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_chunk_pool_local_core_alloc_count	Memory allocation by ChunkAllocator from the bound core	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_chunk_pool_other_core_alloc_count	Memory allocation by ChunkAllocator from other cores	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_chunk_pool_reserved_bytes	Reserved memory size for ChunkAllocator	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_chunk_pool_system_alloc_cost_ns	SystemAllocator memory application duration	Accumulated value. The slope indicates the memory allocation duration.	≥ 0	ns	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_chunk_pool_system_alloc_count	SystemAllocator memory application count	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_chunk_pool_system_free_cost_ns	SystemAllocator memory release duration	The slope indicates the memory release duration.	≥ 0	ns	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_chunk_pool_system_free_count	SystemAllocator memory release count	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_compaction_bytes_total_base	Base Compaction data volume	Cumulative value	≥ 0	Byte	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_compaction_bytes_total_cumulative	Cumulative Compaction data volume	Cumulative value	≥ 0	Byte	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_compaction_deltas_total_base	Rowsets processed by Base Compaction	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_compaction_deltas_total_cumulative	Rowsets processed by cumulative compaction	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_compaction_waitting_permits	Tokens waiting for compaction	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_data_stream_receiver_count	Number of receivers	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_fragment_endpoint_count	Number of receivers	Same as **doris_be_data_stream_receiver_count**	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_fragment_request_duration_us	Execution duration of all fragment instances	Accumulated value. The slope indicates the execution duration.	≥ 0	us	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_fragment_requests_total	Executed fragment instances	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_load_channel_count	Load channels	A larger value indicates that more import tasks are being executed.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_mem_consumption_tablet_meta	Total memory overhead of the **tablet_meta** module	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_mem_consumption_load	Total memory overhead of the **load** module	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
doris_be_memory_allocated_bytes	Virtual memory occupied by TcMalloc	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
doris_be_memory_pool_bytes_total	Memory occupied by all MemPools.	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
doris_be_memtable_flush_duration_us	Time required for writing memtables to disks	Accumulated value. The slope indicates the write latency.	≥ 0	us	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_memtable_flush_total	Number of memtables written to disks	Accumulated value. The slope can be used to calculate the file writing frequency.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_meta_request_duration_read	Time required for reading RocksDB meta	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_meta_request_duration_write	Time required for writing RocksDB meta	-	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_meta_request_total_read	RocksDB meta reads	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_meta_request_total_write	RocksDB meta writes	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_plan_fragment_count	Received fragment instances	This metric is used to check whether instances are stacked.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_process_fd_num_limit_hard	Hard limit of backend process file handles	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_process_fd_num_limit_soft	Soft limit of backend process file handles	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_process_fd_num_used	File handles used by the backend process	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_process_thread_num	Threads used by the backend process	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_query_cache_memory_total_byte	Bytes occupied by query cache	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
doris_be_query_cache_partition_total_count	Partition caches	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_query_cache_sql_total_count	SQL caches	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_query_scan_bytes	Read data volume	Only the amount of data read from the OLAP table is counted.	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_query_scan_bytes_per_second	Read rate	-	≥ 0	Byte/s	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable
doris_be_query_scan_rows	Read rows	Accumulated value. Only data read from the OLAP table is counted. The slope indicates the query rate.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_result_block_queue_count	Fragment instances in the query result cache	This queue is used only when it is directly read by external systems.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_result_buffer_block_count	Queries in the current query result cache	This metric indicates the number of backend query results waiting for frontend consumption.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_routine_load_task_count	Routine load tasks being executed	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_rowset_count_generated_and_in_use	New rowset IDs being used since startup	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_scanner_thread_pool_queue_size	Queue length of OLAPScanner thread pool	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_segment_read_segment_read_total	Read segments	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_segment_read_segment_row_total	Read segment rows	Accumulated value, which also contains the rows searched by index	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_send_batch_thread_pool_queue_size	Queue length of the thread pool for importing data	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_send_batch_thread_pool_thread_num	Threads of the thread pool for importing data	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_small_file_cache_count	Small files cached on BE nodes	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_streaming_load_current_processing	Running stream loading tasks	Only tasks sent by the **curl** command are included.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_streaming_load_duration_ms	Execution duration of all stream loading tasks	Cumulative value	≥ 0	ms	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_streaming_load_requests_total	Stream loading tasks	Accumulated value. The slope indicates the task submission frequency.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_stream_load_pipe_count	Stream loading pipelines	Stream loading and routine loading tasks are included.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_stream_load_load_rows	Rows imported by stream loading	Stream loading and routine loading tasks are included.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_stream_load_receive_bytes	Bytes received by stream loading	Data received by stream loading from HTTP and data read by routine loading from Kafka are included.	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_tablet_base_max_compaction_score	Maximum base compaction score	The value changes in real time, and peak data may be lost. A larger value indicates a heavier compaction accumulation.	≥ 0	N/A	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_tablet_cumulative_max_compaction_score	Maximum cumulative compaction score	-	≥ 0	N/A	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_thrift_connections_total_heartbeat	Total heartbeat service connections	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_thrift_connections_total_backend	Connections of the backend service	Cumulative value	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_thrift_current_connections_heartbeat	Current connections of the heartbeat service	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_thrift_current_connections_backend	Current connections of the backend service	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_timeout_canceled_fragment_count	Fragment instances canceled due to timeout	This value may be recorded repeatedly.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_stream_load_txn_request_begin	Stream loading start transactions	Stream loading and routine loading tasks are included.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_stream_load_txn_request_commit	Successful stream loading transactions	Stream loading and routine loading tasks are included.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_stream_load_txn_request_rollback	Failed stream loading transactions	Stream loading and routine loading tasks are included.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_unused_rowsets_count	Discarded rowsets	These rowsets are deleted periodically.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_load_bytes	Messages sent via tablet sink	Accumulated value, which can be used to calculate the amount of imported data.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_load_rows	Rows sent via tablet sink	Accumulated value, which can be used to calculate the amount of imported data.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable

Metric Name	Metric	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
doris_be_fragment_thread_pool_queue_size	Queue length of query execution thread pool	-	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_compaction_used_permits	Tokens used by the compaction tasks	This metric indicates the resource consumption of Compaction.	≥ 0	Count	N/A	CloudTable instance node	60s	SYS.CloudTable
doris_be_upload_total_byte	Total rowset data	-	≥ 0	Byte	1024(IEC)	CloudTable instance node	60s	SYS.CloudTable

Table 4-59 Custom monitoring metrics of BE nodes

Metric Name	Metric	Description	Value Range	Monitored Object (Dimension)	Monitoring Interval (Raw Data)	Namespace
light_work_active_threads	light_work_active_threads	Number of active threads in the thread pool **brpc light**.	≥ 0	CloudTable instance node	60s	Service.CloudTable
light_work_pool_queue_size	light_work_pool_queue_size	Maximum queue length of the thread pool **brpc light**. If the length exceeds the maximum, work submission is blocked.	≥ 0	CloudTable instance node	60s	Service.CloudTable
fragment_thread_pool_queue_size	fragment_thread_pool_queue_size	Queue length of query execution thread pool	≥ 0	CloudTable instance node	60s	Service.CloudTable
process_thread_num	process_thread_num	Threads used by the BE process	≥ 0	CloudTable instance node	60s	Service.CloudTable

Dimension

Key	Value
cluster_id	CloudTable cluster ID. To obtain the value, go to the cluster management page, click the cluster name, access its details page, and obtain the cluster ID in the cluster information area.
instance_name	Name of a CloudTable cluster node. To obtain the value, go to the cluster management page, click the cluster name, access its details page, and obtain the value of **instance_name**.

4.9.3.2 Setting Doris Cluster Alarm Rules

To monitor the usage of cloud service resources or key operations on them, you can create an alarm rule. After an alarm rule is created, if a metric reaches the specified threshold or there is a specified event, Cloud Eye immediately informs you of the exception through Simple Message Notification (SMN).

You can set CloudTable Doris alarm rules to customize the monitored objects and notification policies. Then, you can learn about Doris running status in a timely manner. The Doris alarm rules include alarm rule name, instance, metric, threshold, monitoring interval and whether to send notification. This section describes how to set alarm rules.

Setting a Doris Cluster Alarm Rule

Step 1 Log in to **the CloudTable console**.

Step 2 In the navigation pane, choose **Alarm Management** > **Alarm Rules**.

Step 3 In the **Resource Type** column, select CloudTable Service to filter alarm rules.

If there are no alarm rules that meet your requirements, create a CloudTable alarm rule by referring to **Creating an Alarm Rule and Notifications**. You need to set the following parameters and retain the default values of other parameters.

Table 4-60 Alarm rule configuration parameters

Parameter	Description
Alarm Type	Select **Metric**.
Cloud Product	Select CloudTable Service.

Parameter	Description
Resource Level	Select a level as required. ● **Cloud product**: Alarm rules are specified by product. Metrics across dimensions can be configured in the same alarm rule. ● **Specific dimension** > **Cluster IDs - Compute Units**: Alarm rules are specified by cluster node.

Figure 4-9 Configuring cluster alarm rules

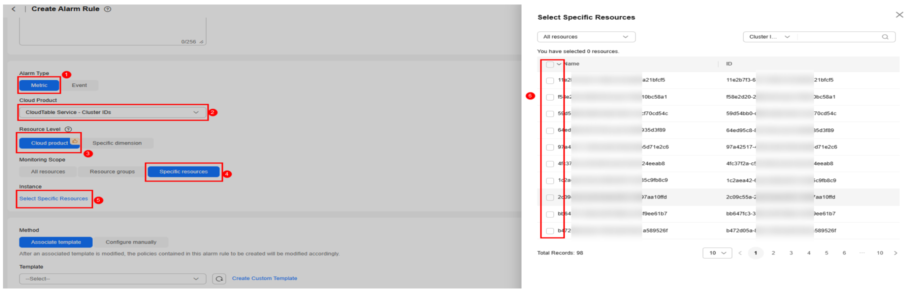
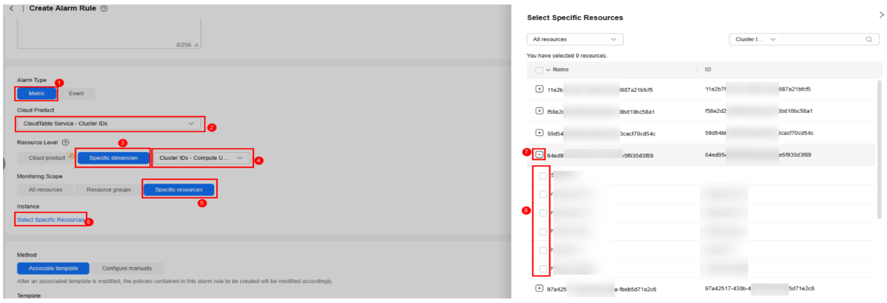


Figure 4-10 Configuring alarm rules for cluster nodes



Step 4 After the configuration is complete, click **Next**.

After the alarm rule is created, if the metric data reaches the specified threshold, Cloud Eye will immediately inform you that an exception has occurred.

----End

Viewing an Alarm Rule

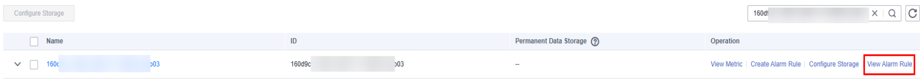
Step 1 Log in to **the CloudTable console**.

Step 2 On the **Cluster Management** page, locate the target cluster and click **View Metric** in the **Operation** column to go to the **Details** page.

Step 3 Viewing alarm rules for clusters or cluster nodes.

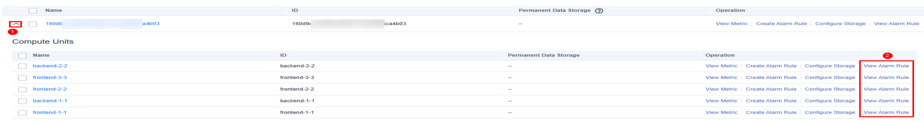
1. View cluster alarm rules. Click **View Alarm Rule** in the **Operation** of the target cluster. In the displayed dialog box, you can view all alarm rules of the cluster.

Figure 4-11 Viewing cluster alarm rules



2. View alarm rules for cluster nodes. Click to expand cluster nodes. Click **View Alarm Rule** in the **Operation** of the target node. In the displayed dialog box, you can view all alarm rules of the cluster node.

Figure 4-12 Viewing alarm rules for cluster nodes



----End

Setting an Alarm Rule with Custom Monitoring Metrics for a Doris Cluster

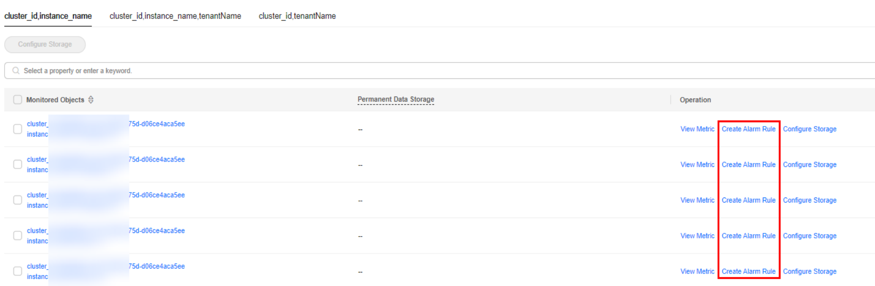
Step 1 Log in to **the CloudTable console**.

Step 2 Choose **Service List** > **Cloud Eye** > **Custom Monitoring** > **Service.CloudTable**.

Step 3 Query the cluster based on the cluster ID, instance name, or tenant name.

Step 4 Click **Create Alarm Rule** in the **Operation** column. On the displayed page, set parameters and click **Create Now**.

Figure 4-13 Custom monitoring



----End

4.9.3.3 Viewing Doris Cluster Monitoring Information

Scenario

Cloud Eye monitors the running status of Doris clusters. You can view the monitoring metrics of Doris on the management console. According to the monitoring information, you can quickly learn about cluster health status and key system information.

Monitoring Function

The cluster monitoring function consists of FE and BE metric monitoring. You can adjust the time range of monitoring data in each module to view historical data at different time. You can also adjust the time granularity of monitoring data to view data in different dimensions. If you want to view monitoring data in real time, enable automatic refresh. The platform can automatically refresh monitoring graphs at intervals of 1 hour, 3 hours, 12 hours, 24 hours, and 7 days.

If you are interested in a specific metric graph, you can click the zoom-in button to view the graph or export the graph.

Viewing Doris Cluster Monitoring Information on the Cloud Eye Console

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the navigation tree on the left, click **Cluster Management**.

Step 4 In the cluster list, locate the row where the target cluster resides, click **View Metric** in the **Operation** column. The Cloud Eye console is displayed.

The status of the cluster to be viewed must be **In service**.

Step 5 On the **Cloud Service Monitoring** page, click on the left of the cluster ID to expand the compute unit list, and select the corresponding node to view the monitoring information.

- **ID**: ID of the monitored instance, that is, the CloudTable cluster ID
- **Viewing monitoring metrics**: Locate the compute unit you want to view and click **View Metric**.

Step 6 Set the metrics to be viewed if there are too many metrics on the monitoring page.

1. If there are too many metrics, delete them on the **Select Metric** page.
2. If the metrics displayed on the page do not contain the desired metrics, add the metrics on the **Select Metric** page.
3. Select at least one metric. You can drag a selected metric and drop it to a desired location to sort the metrics.

NOTE

- If you want to view monitoring data in real time, enable automatic refresh. The platform can automatically refresh monitoring graphs at intervals of 1 hour, 3 hours, 12 hours, 24 hours, and 7 days.
- If you want to zoom in on a single metric view, click in the upper right corner of the metric view to view its details.

----End

Viewing Doris Cluster Monitoring Information on the Cluster Details Page

The cluster details page displays the CPU usage, memory usage, and disk usage of each Doris node.

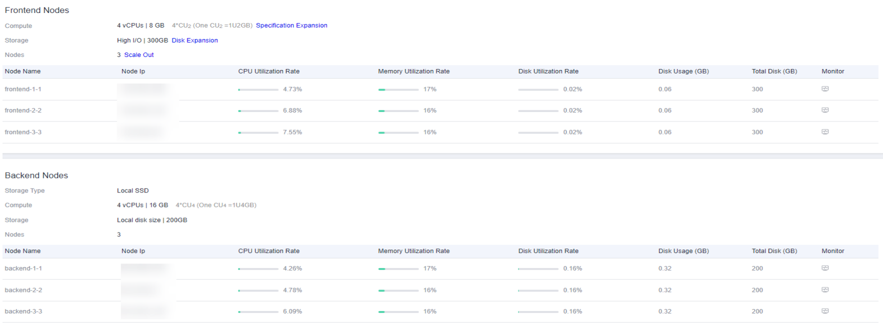
Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Go to the **Cluster Management** page, select the target Doris cluster, and click the cluster name to go to the cluster details page.

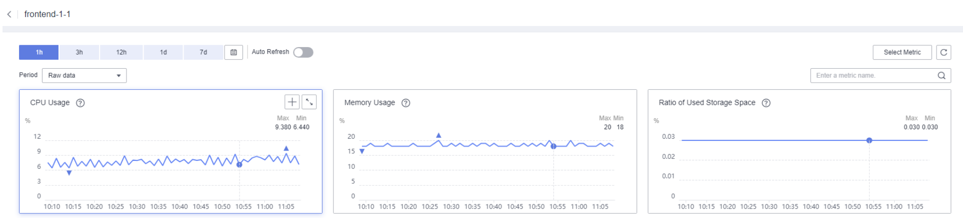
Step 4 View the monitoring metrics on the cluster details page.

Figure 4-14 Viewing the monitoring metrics



Step 5 To view all node metrics, navigate to the Cloud Eye monitoring page from the details page by clicking the monitoring icon . This page provides detailed monitoring metrics for individual Doris cluster nodes.

Figure 4-15 Monitoring metrics



----End

4.9.4 Managing Doris Cluster Logs

4.9.4.1 Viewing Doris Cluster Logs with LTS

Cluster logs are collected and sent to Log Tank Service (LTS). You can check or dump the collected cluster logs on LTS.

NOTE

Path for storing Doris cluster logs: /var/log/doris/

Doris log types:

- fe.log
- fe.warn.log
- fe.audit.log
- be.INFO
- be.WARNING

Enabling LTS

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Click the toggle button in the upper left corner of the page to enable LTS.

NOTE

- If this function is enabled for the first time, the **Create Agency** dialog box is displayed. Click **OK** to authorize the agency.
- If LTS has been enabled and authorized to create an agency, no authorization is required when LTS is enabled again.

----End

Checking Cluster Logs

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Select **View Logs** in the **Operation** column. The LTS console is displayed.

----End

Disable Logging

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Toggle off the LTS switch.

Step 6 Click **OK** in the dialog box.

----End

4.9.4.2 Viewing Doris Cluster Logs with CTS

CloudTable uses CTS to record operations associated with CloudTable for later query, audit, and backtrack operations.

The following key operation traces of CloudTable are recorded in audit logs. For details, see **Table 4-61**.

Table 4-61 CloudTable Doris cluster operation traces supported by CTS

Operation	Trace Name	Resource Type
Creating a cluster	createCloudTableClusterV3	cluster
Scaling out a node	growCloudTableCluster	cluster
Restarting a cluster	rebootCloudTableCluster	cluster
Setting the storage quota	storageClusterAction	cluster
Modifying a feature	modifyClusterFeatures	cluster
Configuring parameters	modifyClusterSetting	cluster
Binding a role to a user in Doris	addAccountRole	cluster
Binding a user to a tenant in Doris	bindAccountWithTenant	cluster
Enumerating database information	copierListDatabaseInfo	cluster
Enumerating cluster node information	copierListNodeInfo	cluster
Creating a catalog in Doris	createCatalogV3	cluster
Creating a Doris user	createCloudTableAccount	cluster
Creating a role	createRole	cluster
Creating a tenant	createTenante	cluster
Testing the function of creating a catalog in Doris	createTestCatalogV3	cluster
Deleting a catalog connection in Doris	deleteCatalogV3	cluster
Deleting a cluster	deleteCloudTableClusterV2	cluster
Deleting a role	deleteRole	cluster
Disabling cluster logs	disableLTSAccess	cluster

Operation	Trace Name	Resource Type
Enabling cluster logs	enableLTSAccess	cluster
Obtaining catalogs in Doris	getCatalogs	cluster
Obtaining cluster information	getClusterInfo	cluster
Obtaining database information	getDatabases	cluster
Obtaining role information	getRoles	cluster
Obtaining table information	getTables	cluster
Accessing the disk expansion page	growCloudTableDisk	cluster
Expanding specifications	growCloudTableFlavor	cluster
Modifying Doris configuration parameters	modifyClusterSettingV3	cluster
Modifying role permissions in Doris	modifyRolePermission	cluster
Restarting a node	restartInstance	cluster
Modifying a tenant in Doris	updateTenant	cluster
Restart	REBOOTING	cluster
Capacity expansion	GROWING	cluster
Deletion	DELETING	cluster
Changing the specifications of a yearly/monthly-billed cluster	changeCloudTableCluster	cluster
Creating a tenant	createTenant	cluster
Deleting a tenant in Doris	deleteTenant	cluster
Scanning and killing SQL statements	killQueryBySqlId	cluster

Operation	Trace Name	Resource Type
Deleting a Doris user	deleteCloudTableAccount	cluster
Updating a Doris user	updateCloudTableAccount	cluster

Enabling CTS

A tracker will be automatically created after CTS is enabled. All traces recorded by CTS are associated with a tracker. Currently, only one tracker can be created for each account.

Step 1 Log in to the **CTS console**.

Step 2 Enabling CTS

If you are a first-time CTS user and do not have any created tackers in the tracker list, refer to **Overview**.

If you have enabled CTS, the system has automatically created a management tracker. Only one management tracker can be created and it cannot be deleted. You can also manually create a data tracker. For details, see **Creating a Tracker** in the *Cloud Trace Service User Guide*.

----End

Disabling the Audit Log Function

If you want to disable the audit log function, disable the tracker in CTS.

Step 1 Log in to the **CTS console**.

Step 2 Disable the audit log function by disabling the tracker. To enable the audit log function again, you only need to enable the tracker.

For details about how to enable or disable a tracker, see **Disabling or Enabling a Tracker** in the *Cloud Trace Service Getting Started*.

----End

Viewing CTS Logs of CloudTable

Step 1 Log in to the **CTS console**.

Step 2 In the navigation pane on the left, choose **Trace List**.

Step 3 In the upper right corner of the trace list, click **Filter** to set the search criteria.

The following four filter criteria are available:

- **Trace Source**, **Resource Type**, and **Search By**
  - **Trace Source**: Select **CloudTable**.
  - **Resource Type**: Select **All resource types** or specify a resource type.

- **Search By**: Select **All** or any of the following options:
  - **Trace name**: If you select this option, you also need to select a specific trace name.
  - **Resource ID**: If you select this option, you also need to select or enter a specific resource ID.
  - **Resource name**: If you select this option, you also need to select or enter a specific resource name.
- **Operator**: Select a specific operator (at user level rather than tenant level).
- **Trace Status**: Available options include **All trace statuses**, **normal**, **warning**, and **incident**. You can only select one of them.
- **Start Date** and **End Date**: You can specify the time period to query traces.

Step 4 Click **Query**.

Step 5 Click ˅ on the left of the trace to be queried to extend its details.

Figure 4-16 Trace

Step 6 Locate the row containing the target trace and click **View Trace** in the **Operation** column.

Figure 4-17 Viewing a trace

For details about key fields in the CTS trace structure, see the **Trace Structure** in the *Cloud Trace Service User Guide*.

----End

4.10 Common SQL Commands of Doris

4.10.1 Creating a Database

This section describes the basic syntax and usage of the SQL statements for creating a database in a Doris cluster.

Basic Syntax

```
CREATE DATABASE [IF NOT EXISTS] db_name
[PROPERTIES ("key"="value", ...)];
```

Usage Example

Step 1 Connect to Doris through the MySQL client as a user with Doris administrator permissions.

Step 2 Run the following command to create the **example_db** database:

```
create database if not exists example_db;
```

Step 3 Run the following statement to view the database information:

```
SHOW DATABASES;
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| example_db         |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

Step 4 Run the following command to switch to **example_db**:

```
use example_db;
```

----End

4.10.2 Creating a Table

This section describes the basic syntax and usage of the SQL statements for creating a table in a Doris cluster.

Basic Syntax

```
CREATE TABLE [IF NOT EXISTS] [database.]table
(
column_definition_list,
[index_definition_list]
)
[engine_type]
[keys_type]
[table_comment]
[partition_info]
distribution_desc
```



```
[rollup_list]
[properties]
[extra_properties]
```

Usage Example

- Create a common table named **table1**.
  ```
  CREATE TABLE example_db.table1
  (
  k1 TINYINT,
  k2 DECIMAL(10, 2) DEFAULT "10.5",
  k3 CHAR(10) COMMENT "string column",
  k4 INT NOT NULL DEFAULT "1" COMMENT "int column"
  )
  COMMENT "table comment"
  DISTRIBUTED BY HASH(k1) BUCKETS 32;
  ```
- Create a partitioned table named **table2**. Use the **event_day** column to partition the table into **p201706**, **p201707**, and **p201708**. The values are as follows:
  - **p201706**: The value range is [Minimum value, 2017-07-01).
  - **p201707**: The value range is [2017-07-01, 2017-08-01).
  - **p201708**: The value range is [2017-08-01, 2017-09-01).

  Each partition is hashed into 10 buckets based on **siteid**. The command for creating a table is as follows:
  ```
  CREATE TABLE table2
  (
  event_day DATE,
  siteid INT DEFAULT '10',
  citycode SMALLINT,
  username VARCHAR(32) DEFAULT '',
  pv BIGINT SUM DEFAULT '0'
  )
  AGGREGATE KEY(event_day, siteid, citycode, username)
  PARTITION BY RANGE(event_day)
  (
  PARTITION p201706 VALUES LESS THAN ('2017-07-01'),
  PARTITION p201707 VALUES LESS THAN ('2017-08-01'),
  PARTITION p201708 VALUES LESS THAN ('2017-09-01')
  )
  DISTRIBUTED BY HASH(siteid) BUCKETS 10
  PROPERTIES("replication_num" = "3");
  ```

  NOTE

  - You must specify at least three replicas when creating a table to ensure high availability.
  - For a non-node cluster, you do not need to specify replicas.
  - You can add a rollup to a table to improve query performance.
  - By default, the **Null** property of a column in a table is **true**, which affects the query performance.
  - The bucket column must be specified for a Doris table.
- View the table content.
  - View tables in the current database.
    ```
    SHOW TABLES;
    +----------------------+
    | Tables_in_example_db |
    +----------------------+
    | table1               |
    | table2               |
    ```

```
+----------------------+
2 rows in set (0.01 sec)
```

– View the structure of **table1**.

```
DESC table1;
+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | Yes  | true  | 10      |       |
| citycode | smallint(6) | Yes  | true  | N/A     |       |
| username | varchar(32) | Yes  | true  |         |       |
| pv       | bigint(20)  | Yes  | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
4 rows in set (0.00 sec)
```

– View the structure of **table2**.

```
DESC table2;
+-----------+-------------+------+-------+---------+-------+
| Field     | Type        | Null | Key   | Default | Extra |
+-----------+-------------+------+-------+---------+-------+
| event_day | date        | Yes  | true  | N/A     |       |
| siteid    | int(11)     | Yes  | true  | 10      |       |
| citycode  | smallint(6) | Yes  | true  | N/A     |       |
| username  | varchar(32) | Yes  | true  |         |       |
| pv        | bigint(20)  | Yes  | false | 0       | SUM   |
+-----------+-------------+------+-------+---------+-------+
5 rows in set (0.00 sec)
```

4.10.3 Inserting Data

This section describes the basic syntax and usage of the SQL statements for inserting table data in a Doris cluster.

Basic Syntax

```
INSERT INTO table_name
[ PARTITION (p1, ...) ]
[ WITH LABEL label]
[ (column [, ...]) ]
[ [ hint [, ...] ] ]
{ VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | query }
```

Usage Example

- Create table **test**.

```
CREATE TABLE test
(
c1 TINYINT,
c2 DECIMAL(10, 2) DEFAULT "10.5",
)
COMMENT "table comment"
DISTRIBUTED BY HASH(k1) BUCKETS 32;
```

- Create table **test2**.

```
CREATE TABLE test2 like test;
```

- Insert multiple rows of data into the **test** table at a time.

```
INSERT INTO test VALUES (1, 2), (3, 4);
```

- Check whether data is inserted into **test2**.

```
SELECT * from test2;
```

- Import the result of a query statement to the **test** table.

```
INSERT INTO test (c1, c2) SELECT * from test2;
```

4.10.4 Querying Data

This section describes the basic syntax and usage of the SQL statements for querying table data in a Doris cluster.

Basic Syntax

```
SELECT
[hint_statement, ...]
[ALL | DISTINCT | DISTINCTROW | ALL EXCEPT ( col_name1 [, col_name2, col_name3, ...] )]
select_expr [, select_expr ...]
[FROM table_references
[PARTITION partition_list]
[TABLET tabletid_list]
[TABLESAMPLE sample_value [ROWS | PERCENT]
[REPEATABLE pos_seek]]
[WHERE where_condition]
[GROUP BY [GROUPING SETS | ROLLUP | CUBE] {col_name | expr | position}]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[INTO OUTFILE 'file_name']
```

Usage Example

- Query the names of students aged 18, 20, and 25 in the **student** table.
  ```
  SELECT Name from student where age in (18,20,25);
  ```
- Query the names of the three youngest students in the **student** table. Use **order by** to sort the **age** column and use **limit** to display three records. **asc** indicates the ascending order (default), and **desc** indicates the descending order.
  ```
  SELECT Name from student order by age asc limit 3;
  ```

4.10.5 Modifying a Table Structure

There are different methods for modifying table structures in an aggregate model and a non-aggregate model. The methods for modifying the key and value columns are also different. Specifically:

- If **AGGREGATE KEY** is specified during table creation, the table uses an aggregate model. In other scenarios, a non-aggregate model is used.
- In the table creation statement, the columns following keyword **'unique key'**, **'aggregate key'**, or **'duplicate key'** are the key columns, and the remaining columns are value columns.

Example for an Aggregate Model

The aggregate type of the aggregate columns cannot be changed.

- Add the **new_col** column (key column) after the **col1** column.
  ```
  ALTER TABLE example_db.my_table  ADD COLUMN new_col INT DEFAULT "0" AFTER col1;
  ```
- Add the **new_col** column (value column with the SUM aggregate type) after the **col1** column.
  ```
  ALTER TABLE example_db.my_table ADD COLUMN new_col INT SUM DEFAULT "0" AFTER col1;
  ```
- Change the type of the **col1** column (key column) to BIGINT.
  ```
  ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT DEFAULT "1";
  ```


- Change the type of the **col1** column (value column) to BIGINT.
```
ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT MAX DEFAULT "1";
```
- Delete the **col1** column.
```
ALTER TABLE example_db.my_table DROP COLUMN col1;
```

Example for a Non-Aggregate Model

- Add the **new_col** column (key column) after the **col1** column.
```
ALTER TABLE example_db.my_table ADD COLUMN new_col INT KEY DEFAULT "0" AFTER col1;
```
- Add the **new_col** column (value column) after the **col1** column.
```
ALTER TABLE example_db.my_table ADD COLUMN new_col INT DEFAULT "0" AFTER col1;
```
- Change the type of the **col1** column (key column) to BIGINT.
```
ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT KEY DEFAULT "1";
```
- Change the type of the **col1** column (value column) to BIGINT.
```
ALTER TABLE example_db.my_table MODIFY COLUMN col1 BIGINT DEFAULT "1";
```
- Delete the **col1** column.
```
ALTER TABLE example_db.my_table DROP COLUMN col1;
```

4.10.6 Deleting a Table

This section describes the basic syntax and usage of the SQL statements for deleting a table in a Doris cluster.

Basic Syntax

```
DROP TABLE [IF EXISTS] [db_name.]table_name [FORCE];
```

Usage Example

Delete the **my_table** table.

```
DROP TABLE IF EXISTS example_db.my_table;
```


5 Using ClickHouse

5.1 ClickHouse Coupled Storage and Compute Table Engine Overview

Background

The table engine determines:

- Where to write and read data.
- Which queries are supported.
- Whether concurrent data access is supported.
- Whether indexes are supported.
- Whether multi-thread requests can be executed.
- Parameters used for data replication.

This section describes MergeTree and Distributed engine families, which are the most important and frequently used.

Overview

A table engine is a table type. ClickHouse table engine determines how to store and read data, whether indexes are supported, and whether active/standby replication is supported. The following table lists ClickHouse table engines to help you get started with ClickHouse.

NOTE

The MergeTree engine maintains data in a single replica, which cannot ensure high availability and data reliability. You are advised to use the MergeTree engine only in the test environment. For production environments, use the Replicated*MergeTree engine.

Table 5-1 Table engines

Engine Family	Description	Engine	Description
MergeTree	• MergeTree engines are the most universal and functional and are mainly used for heavy-load tasks. They support quick write of a large amount of data and subsequent data processing. • MergeTree engines support data replication, partitioning, and data sampling.	MergeTree	• Data is stored by partition and block based on partitioning keys. • Data index is sorted based on primary keys and the **ORDER BY** sorting keys. • Data replication is supported by table engines prefixed with Replicated. • Data sampling is supported. When data is written, a table with this type of engine divides data into different folders based on the partitioning key. Each column of data in the folder is an independent file. A file that records serialized index sorting is created. This structure reduces the volume of data to be retrieved during data reading, greatly improving query efficiency.
		RelacingMergeTree	This table engine removes duplicates that have the same primary key value. The MergeTree table engine does not support this feature.
		CollapsingMergeTree	CollapsingMergeTree defines a **Sign** field to record status of data rows. If **Sign** is **1**, the data in this row is valid. If **Sign** is **-1**, the data in this row needs to be deleted.
		VersionedCollapsingMergeTree	This table engine allows you to add the **Version** column to the **CREATE TABLE** statement. This helps resolve the issue that the CollapsingMergeTree table engine cannot collapse or delete rows as expected if the rows are inserted in an incorrect order.

Engine Family	Description	Engine	Description
		SummigMergeTree	This table engine pre-aggregates primary key columns and combines all rows that have the same primary key into one row. This helps reduce storage usage and improves aggregation performance.
		AggregatingMergeTree	This table engine is a pre-aggregation engine and is used to improve aggregation performance. When merging partitions, the AggregatingMergeTree engine aggregates data based on predefined conditions, calculates data based on predefined aggregate functions, and saves the data in binary format to tables.
		GraphiteMergeTree	This table engine is used to store and roll up Graphite data. This helps reduce storage space and makes Graphite data queries more efficient.
Replicated*MergeTree	All engines of the MergeTree family in ClickHouse prefixed with Replicated become MergeTree engines that support replicas.	Replicated*MergeTree series	Replicated series engines use ZooKeeper to synchronize data. When a replicated table is created, all replicas of the same shard are synchronized based on the information registered with ZooKeeper.
Distributed	-	Distributed	This table engine does not store data and performs distributed queries on multiple servers.

MergeTree

- Creating a table.

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER ClickHouse cluster name]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
    ...
    INDEX index_name1 expr1 TYPE type1(...) GRANULARITY value1,
    INDEX index_name2 expr2 TYPE type2(...) GRANULARITY value2
```



```
) ENGINE = MergeTree()
ORDER BY expr
[PARTITION BY expr]
[PRIMARY KEY expr]
[SAMPLE BY expr]
[TTL expr [DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'], ...]
[SETTINGS name=value, ...]
```

- The following is an example.

```
CREATE TABLE default.test (name1 DateTime,name2 String,name3 String,name4 String,name5 Date)
ENGINE = MergeTree() PARTITION BY toYYYYMM(name5) ORDER BY (name1, name2) SETTINGS
index_granularity = 8192;
```

Parameters in the example are described as follows:

Table 5-2 Parameter description

Parameter	Description
ENGINE = MergeTree()	MergeTree table engine.
PARTITION BY toYYYYMM(name5)	Partition. The sample data is partitioned by month, and a folder is created for each month.
ORDER BY	Sorting field. A multi-field index can be sorted. If the first field is the same, the second field is used for sorting, and so on.
index_granularity = 8192	Granularity of a sorting index. One index value is recorded for every 8,192 data records.

NOTE

If the data to be queried exists in a partition or sorting field, the data query efficiency is greatly improved.

ReplacingMergeTree

ClickHouse provides the ReplacingMergeTree table engine to remove duplicates that have the same primary key value. The MergeTree table engine does not support this feature.

- Create a table.

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER ClickHouse cluster name]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = ReplacingMergeTree([ver])
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

CollapsingMergeTree

The CollapsingMergeTree table engine removes the limits of the ReplacingMergeTree table engine. This table engine allows you to add the **Sign** column to the **CREATE TABLE** statement. Rows are classified into two types. If **Sign** is **1**, the row is a "state" row and is used to add states. If **Sign** is **–1**, the row is a "cancel" row and is used to delete states.

- Statements for creating a table:
```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER ClickHouse cluster name]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = CollapsingMergeTree(sign)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```
- Example
  - Sample data

    Assume that you need to calculate how many pages users visited on a website and how long they were there. At a specific time point, write the following row with the state of the user's activity.

Table 5-3 Sample data

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

  - **Sign**: Name of the column with the type of row. **1** is a "state" row and **–1** is a "cancel" row.
  - Create the **Test** table.
```
CREATE TABLE Test(UserID UInt64,PageViews UInt8,Duration UInt8,Sign Int8)ENGINE =
CollapsingMergeTree(Sign) ORDER BY UserID;
```
  - Insert data.
    - Insert data for the first time.
```
INSERT INTO Test VALUES (4324182021466249494, 5, 146, 1);
```
    - Insert data for the second time.
```
INSERT INTO Test VALUES (4324182021466249494, 5, 146, -1),(4324182021466249494, 6,
185, 1);
```
  - View data.
```
SELECT * FROM Test;
```
    The following query result is returned:
```
┌──────────────UserID─┬─PageViews─┬─Duration─┬─Sign─┐
│ 4324182021466249494 │         5 │      146 │   -1 │
│ 4324182021466249494 │         6 │      185 │    1 │
└─────────────────────┴───────────┴──────────┴──────┘
┌──────────────UserID─┬─PageViews─┬─Duration─┬─Sign─┐
```

```
│ 4324182021466249494 │         5 │      146 │    1 │
└─────────────────────┴───────────┴──────────┴──────┘
```

– Aggregate data in a specified column.

```
SELECT UserID,sum(PageViews * Sign) AS PageViews,sum(Duration * Sign) AS Duration FROM Test GROUP BY UserID HAVING sum(Sign) > 0;
```

The command output is as follows:

```
┌──────────────UserID─┬─PageViews─┬─Duration─┐
│ 4324182021466249494 │         6 │      185 │
└─────────────────────┴───────────┴──────────┘
```

– Perform force collapsing on data.

```
SELECT * FROM Test FINAL;
```

The command output is as follows:

```
┌──────────────UserID─┬─PageViews─┬─Duration─┬─Sign─┐
│ 4324182021466249494 │         6 │      185 │    1 │
└─────────────────────┴───────────┴──────────┴──────┘
```

VersionedCollapsingMergeTree

ClickHouse provides the VersionedCollapsingMergeTree table engine to resolve the issue that the CollapsingMergeTree table engine cannot collapse or delete rows as expected if the rows are inserted in an incorrect order. The VersionedCollapsingMergeTree table engine allows you to add the **Version** column to the **CREATE TABLE** statement to record the mapping between the "state" rows and "cancel" rows. During background compaction, rows with the same primary key, **Version**, and **Sign** are collapsed (deleted).

- Create a table.

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER ClickHouse cluster name]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = VersionedCollapsingMergeTree(sign, version)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

- Example
  – Sample data

    Assume that you need to calculate how many pages users visited on a website and how long they were there. At a specific time point, write the following row with the state of the user's activity.

Table 5-4 Sample data

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

- **Sign**: Name of the column with the type of row. **1** is a "state" row and **–1** is a "cancel" row.
- **Version**: Name of the column with the version of the object state.

– Create the **T** table.
```
CREATE TABLE T(UserID UInt64,PageViews UInt8,Duration UInt8,Sign Int8,Version
UInt8)ENGINE = VersionedCollapsingMergeTree(Sign, Version)ORDER BY UserID;
```
– Insert two different parts of data.
```
INSERT INTO T VALUES (4324182021466249494, 5, 146, 1, 1);
INSERT INTO T VALUES (4324182021466249494, 5, 146, -1, 1),(4324182021466249494, 6, 185,
1, 2);
```
– View data.
```
SELECT * FROM T;
```
– Aggregate data in a specified column.
```
SELECT UserID, sum(PageViews * Sign) AS PageViews,sum(Duration * Sign) AS Duration,Version
FROM T GROUP BY UserID, Version HAVING sum(Sign) > 0;
```
The query result is as follows:
```
┌──────────────UserID─┬─PageViews─┬─Duration─┬─Version─┐
│ 4324182021466249494 │         6 │      185 │       2 │
└─────────────────────┴───────────┴──────────┴─────────┘
```
– Perform force collapsing on data.
```
SELECT * FROM T FINAL;
```
The query result is as follows:
```
┌──────────────UserID─┬─PageViews─┬─Duration─┬─Sign─┬─Version─┐
│ 4324182021466249494 │         6 │      185 │    1 │       2 │
└─────────────────────┴───────────┴──────────┴──────┴─────────┘
```

SummingMergeTree

The SummingMergeTree table engine pre-aggregates primary key columns and combines all rows that have the same primary key into one row. This helps reduce storage usage and improves aggregation performance.

- Create a table.
```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER ClickHouse cluster name]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = SummingMergeTree([columns])
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```
- Example

– Create a SummingMergeTree table named **testTable**.
```
CREATE TABLE testTable(id UInt32,value UInt32)ENGINE = SummingMergeTree() ORDER BY id;
```
– Inserts data to the **testTable** table.
```
INSERT INTO testTable Values(5,9),(5,3),(4,6),(1,2),(2,5),(1,4),(3,8);
INSERT INTO testTable Values(88,5),(5,5),(3,7),(3,5),(1,6),(2,6),(4,7),(4,6),(43,5),(5,9),(3,6);
```
– Query all data in unmerged parts.
```
SELECT * FROM testTable;
```
The following query result is returned:
```
┌─id─┬─value─┐
│  1 │     6 │
```

```
│  2 │     5 │
│  3 │     8 │
│  4 │     6 │
│  5 │    12 │
└────┴───────┘
┌─id─┬─value─┐
│  1 │     6 │
│  2 │     6 │
│  3 │    18 │
│  4 │    13 │
│  5 │    14 │
│ 43 │     5 │
│ 88 │     5 │
└────┴───────┘
```

– If ClickHouse has not summed up all rows and you need to aggregate data by ID, use the **sum** function and **GROUP BY** statement.

```
SELECT id, sum(value) FROM testTable GROUP BY id;
```

The following query result is returned:

```
┌─id─┬─sum(value)─┐
│  4 │         19 │
│  3 │         26 │
│ 88 │          5 │
│  2 │         11 │
│  5 │         26 │
│  1 │         12 │
│ 43 │          5 │
└────┴────────────┘
```

– Merge rows manually.

```
OPTIMIZE TABLE testTable;
```

Query data in the table.

```
SELECT * FROM testTable;
```

The following query result is returned:

```
┌─id─┬─value─┐
│  1 │    12 │
│  2 │    11 │
│  3 │    26 │
│  4 │    19 │
│  5 │    26 │
│ 43 │     5 │
│ 88 │     5 │
└────┴───────┘
```

NOTE

- SummingMergeTree uses the **ORDER BY** sorting keys as the condition keys to aggregate data. If sorting keys are the same, data records are merged into one and the specified merged fields are aggregated.
- Data is pre-aggregated only when merging is executed in the background, and the merging execution time cannot be predicted. Therefore, it is possible that some data has been pre-aggregated and some data has not been aggregated. Therefore, the **GROUP BY** statement must be used during aggregation.

AggregatingMergeTree

The AggregatingMergeTree table engine is also used for pre-aggregation and can improve the aggregation performance.

- Create a table.

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER ClickHouse cluster name]
(
```

```
  name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
  name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
  ...
) ENGINE = AggregatingMergeTree()
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[TTL expr]
[SETTINGS name=value, ...]
```

- Example

  You do not need to set the AggregatingMergeTree parameter separately. When partitions are merged, data in each partition is aggregated based on the **ORDER BY** sorting key. You can set the aggregate functions to be used and column fields to be calculated by defining the AggregateFunction type.

  - Create a table.

    ```
    create table test_table (name1 String,name2 String,name3
    AggregateFunction(uniq,String),name4 AggregateFunction(sum,Int),name5 DateTime) ENGINE
    = AggregatingMergeTree() PARTITION BY toYYYYMM(name5) ORDER BY (name1,name2)
    PRIMARY KEY name1;
    ```

  When data of the AggregateFunction type is written or queried, the ***state** and ***merge** functions need to be called. The asterisk (*) indicates the aggregate functions used for defining the field type. In the table creation example, the **uniq** and **sum** functions are specified for the **name3** and **name4** fields defined in the **test_table**, respectively. Therefore, you need to call the **uniqState** and **sumState** functions and run the **INSERT** and **SELECT** statements when writing data into the table.

  - Insert data.

    ```
    insert into test_table select '8','test1',uniqState('name1'),sumState(toInt32(100)),'2021-04-30
    17:18:00';
    insert into test_table select '8','test1',uniqState('name1'),sumState(toInt32(200)),'2021-04-30
    17:18:00';
    ```

  - Query the data.

    ```
    select name1,name2,uniqMerge(name3),sumMerge(name4) from test_table group by
    name1,name2;
    ```

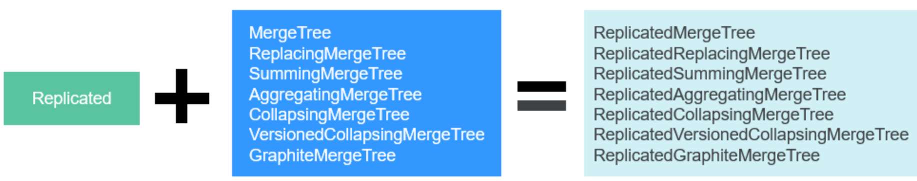
    The following query result is returned:

    ```
    ┌─name1─┬─name2─┬─uniqMerge(name3)─┬─sumMerge(name4)─┐
    │ 8     │ test1 │                1 │             300 │
    └───────┴───────┴──────────────────┴─────────────────┘
    ```

Replicated*MergeTree Engines

All engines of the MergeTree family in ClickHouse prefixed with Replicated become MergeTree engines that support replicas.

Figure 5-1 MergeTree table engines



- Template for creating a Replicated engine:
```
ENGINE = Replicated*MergeTree('ZooKeeper storage path','Replica name', ...)
```

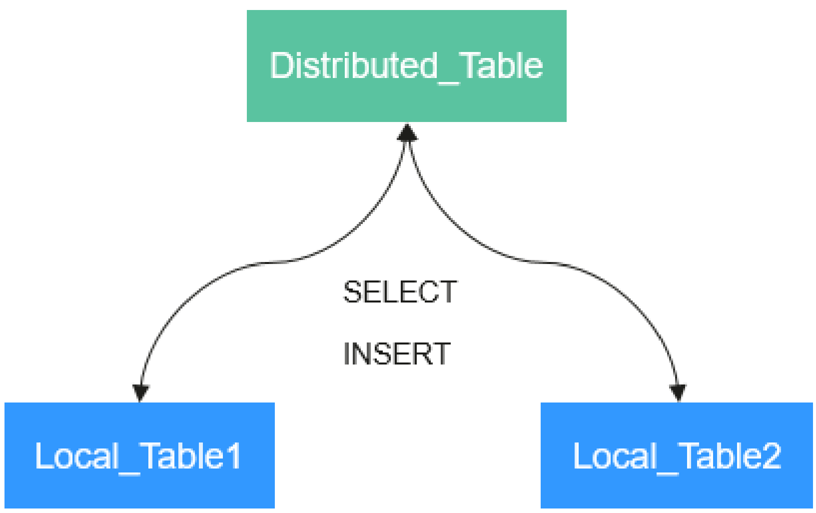
Table 5-5 Parameters

Parameter	Description
ZooKeeper storage path	Path for storing table data in ZooKeeper. The path format is **/clickhouse/tables/{shard}/** *Database name*/ *Table name*.
Replica name	**{replica}** is typically used to represent the replica name.

Distributed Table Engines

Tables with Distributed engine do not store any data of their own, but serve as a transparent proxy for data shards and can automatically transmit data to each node in the cluster. Distributed tables need to work with other local data tables. Distributed tables distribute received read and write tasks to each local table where data is stored.

Figure 5-2 Distributed



- Template for creating a distributed engine:
```
ENGINE = Distributed(cluster_name, database_name, table_name, [sharding_key])
```


Table 5-6 Distributed parameters

Parameter	Description
cluster_name	Cluster name. When a distributed table is read or written, the cluster configuration information is used to search for the corresponding ClickHouse instance node.
database_name	Database name.
table_name	Name of a local table in the database. It is used to map a distributed table to a local table.
sharding_key	Sharding key, based on which a distributed table distributes data to each local table.

- Example
  - Create a local ReplicatedMergeTree table named **demo**.
    ```
    CREATE TABLE default.demo ON CLUSTER default_cluster( `EventDate` DateTime, `id`
    UInt64)ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/demo', '{replica}')
    PARTITION BY toYYYYMM(EventDate) ORDER BY id;
    ```
  - Create a Distributed table named **demo_all** based on the local table **demo**.
    ```
    CREATE TABLE default.demo_all ON CLUSTER default_cluster( `EventDate` DateTime, `id`
    UInt64)ENGINE = Distributed(default_cluster, default, demo, rand());
    ```
- Rules for creating a distributed table:
  - When creating a distributed table, add **ON CLUSTER** *cluster_name* to the table creation statement so that the statement can be executed once on a ClickHouse instance and then distributed to all instances in the cluster for execution.
  - Generally, a distributed table is named in the following format: *Local table name*_all. It forms a one-to-many mapping with local tables. Then, multiple local tables can be operated using the distributed table proxy.
  - Ensure that the structure of a distributed table is the same as that of local tables. If they are inconsistent, no error is reported during table creation, but an exception may be reported during data query or insertion.

5.2 ClickHouse Usage Process

ClickHouse is a columnar database oriented to online analysis and processing. It is independent of the Hadoop big data system and features compression rate and fast query performance. In addition, ClickHouse supports SQL query and provides good query performance, especially the aggregation analysis and query performance based on large and wide tables. The query speed is one order of magnitude faster than that of other analytical databases. CloudTable ClickHouse cluster management also provides the following functions:

In ClickHouse cluster management, you need to create a dedicated cluster and can use it on demand. Dedicated clusters are physically isolated and are not affected by other services, facilitating user management.

After the ClickHouse cluster is created, use the client to access the cluster. For details, see the following sections.

- **Using a Client to Access a ClickHouse Cluster**.
- **Getting Started with ClickHouse**.

Figure 5-3 ClickHouse Usage Process

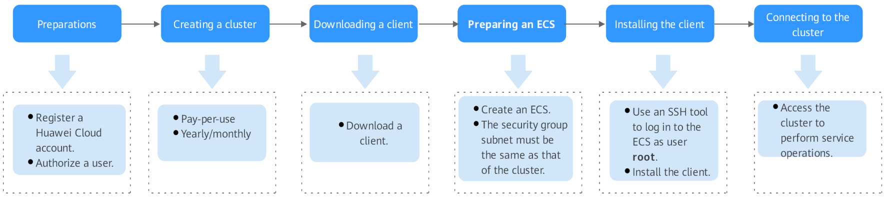


Table 5-7 ClickHouse usage process

Step	Substep	Description	Detailed Instructions
Preparations	Creating a user and granting permissions to use CloudTable	• Before using CloudTable ClickHouse, you need to register a Huawei Cloud account, complete real-name authentication, and grant the necessary permissions to your account. • Grant the necessary service permissions to a user group, and then add users to this group to enable their access.	**Creating a User and Granting Permissions**
Creating a cluster	Creating a ClickHouse cluster	Before using ClickHouse to execute tasks, you need to create a ClickHouse cluster.	**Creating a ClickHouse Cluster**
Downloading the client	Downloading the ClickHouse client	After creating a cluster, download and install the client. After the client is installed, you can use the SSH tool to connect to the cluster.	**Installing the ClickHouse Client**

Step	Substep	Description	Detailed Instructions
Preparing an ECS	-	If the client tool runs on Linux, you need to prepare a Linux ECS that is in the VPC as a ClickHouse cluster and the Linux ECS serves as a client host. If the client tool runs on Windows, you need to prepare a Windows ECS that is in the VPC as a ClickHouse cluster and the Windows ECS serves as a client host.	**Preparing an ECS**
Installing the client	-	Place the downloaded client on the ECS, decompress the package, and install the client.	• **Using a Client to Connect to a ClickHouse Normal Cluster** • **Using a Client to Connect to a ClickHouse Security Cluster**
Connecting to the cluster	-	After installing the MySQL client on the ECS, you can run commands to connect to the cluster and perform service operations.	**Installing the ClickHouse Client**

5.3 Creating a ClickHouse Cluster

You can centrally manage clusters with CloudTable. A cluster is necessary for using CloudTable. This section describes how to create a cluster on the CloudTable console.

ClickHouse clusters support two billing modes: pay-per-use and yearly/monthly. The pay-per-use billing mode is used by default when you create a cluster. In pay-per-use mode, compute units are charged by the duration you use them, with a billing cycle of one hour. With this mode, you can start or stop a cluster at any time and pay what you use. Alternatively, you can opt for the yearly/monthly billing mode, which is a prepaid option offering significant discounts compared to the pay-per-use mode. The yearly/monthly billing mode is particularly suitable for long-term users. You can also customize a ClickHouse cluster with specified computing capabilities and storage space to meet your business needs.

Prerequisites

- The VPC and security group of the cluster to be created must be the same as those of the ECS on the public network. Otherwise, the client cannot access the cluster.
- Before creating a cluster, you must configure inbound security group rules. For details, see **Configuring Security Group Rules**.
- Before creating a cluster, you must add the ICMP protocol to the security group rules so that you can view the status of each node by pinging the node IP address on the management plane. For details, see **Configuring Security Group Rules**.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Create Cluster** in the upper right corner.

Step 4 Configure basic cluster information by referring to the following table.

Table 5-8 Parameter description

Parameter	Description
Region	Current working region of the cluster. ● Select the region for the cluster nodes to run.
AZ	Select the AZ associated with the cluster's region. For more information, see **Regions and AZs**.
Billing Mode	Select **Pay-per-use** or **Yearly/Monthly**.
Required Duration	This option is available only when **Billing Mode** is set to **Yearly/Monthly**. Configure this parameter based on your service requirements.
Auto-renew	If you select **Auto-renew** when creating a cluster, the system will automatically renew your subscription before it expires.

Table 5-9 Cluster and network configuration

Parameter	Description
Name	Name of a cluster. The cluster name must consist of 4 to 32 characters and must begin with a letter. It may include only letters, digits, and hyphens (-) but must not contain any other special characters. Additionally, the cluster name is case-insensitive.

Parameter	Description
VPC	A VPC is a secure, isolated, and logical network environment. Retain the default settings. If there is no available VPC, click **View VPC** to access the VPC console and create a VPC.
Subnet	Specify a VPC subnet. A subnet provides dedicated network resources that are isolated from other networks, improving network security.
Security Group	A security group is used to control ECS access within a security group or between security groups by defining access rules. You can define different access control rules for a security group. These rules can specify which ECS ports or protocols are accessible and can be used to control inbound and outbound network traffic of VMs. After an ECS is added to the security group, it is protected by these access control rules. ECSs that do not belong to the security group cannot communicate with ECSs in the security group. You can use an existing security group or click **View Security Group** to create a new one. For more information about security groups, see **Security Group**. **NOTE** ● CloudTable clusters support multiple security groups and security group modification. Security group rules: – External servers can ping the instances in the security group to verify network connectivity. – Instances in a security group can communicate with each other via a private network. – Instances in a security group can be accessed via a private network. ● Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.
Database Engine	Select the type of cluster to be created.
Cluster Storage Mode	The cluster storage mode is coupled storage and compute.
ClickHouse Kernel	Retain the default settings.

Parameter	Description
Cluster HA	Enable: Two replicas are maintained for each shard, with three ZooKeeper nodes as the default. Disable: Only one single ClickHouse node with a single shard and replica is provided for testing purposes. This configuration does not support node scale-out and cannot be used in any production environments. **NOTE** ● A single-node cluster does not support scale-out or data migration. ● A single-node cluster can only be used in the non-production environment. ● Users of a single-node cluster must establish a non-replicated table. Example of creating a table: `` CREATE TABLE test_p (`EventDate` DateTime,`CounterID` UInt32,`UserID` UInt32,`ver` UInt16)ENGINE = ReplacingMergeTree()PARTITION BY sipHash64(CounterID) ORDER BY (CounterID, EventDate, intHash32(UserID))SAMPLE BY intHash32(UserID)SETTINGS index_granularity = 8192; ``

Table 5-10 Compute node parameters

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: ● 8U32G ● 8U64G ● 16U64G ● 16U128G ● 32U128G ● 32U256G ● 64U256G ● 64U512G ● 128U512G

Parameter	Description
Storage	Select the disk specifications and capacity of the ClickHouse compute node. ● Available storage specifications: – High I/O – General-purpose SSD – Ultra-high I/O – Extreme SSD ● The capacity ranges from 100 GB to 10,000 GB per node.
Nodes	Set the number of nodes in a cluster. The value ranges from **2** to **40**.

Table 5-11 ZooKeeper node parameters

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: ● 4U8G (This specification is not available for production environments.) ● 4U16G ● 8U32G ● 16U64G ● 32U128G The ZooKeeper small specifications include only 4U8G (4 vCPUs, 8 GB memory) and 4U16G (4 vCPUs, 16 GB memory). In the production environment, you are advised to select 8 vCPUs, 32 GB memory, or higher specifications.
Storage	**Ultra-high I/O** is recommended. Set the disk capacity to 100 GB per node.
Nodes	Number of nodes in a cluster. The default value is **3**.

Table 5-12 Password setting parameters

Parameter	Description
Username	Management user, which is used to connect to the cluster. The default value is **admin** and cannot be changed.

Parameter	Description
Password	Set the password of the **admin** user. The password complexity requirements are as follows: ● The password must contain 8 to 16 characters. ● The password must contain at least four types of the following characters: uppercase letters, lowercase letters, digits, and special characters ($@!%*?&._{}()+=^<>) ● The password cannot be the same as the username or the username spelled backwards. ● The strong and weak password check should be performed. **NOTE** Change the password regularly and keep it secure.
Confirm Password	Set the password of the **admin** user.
Enable Channel Encryption	This function uses HTTPS ports and secure TCP ports to access ClickHouse, ensuring that customer communication data is encrypted. ● If the secure channel encryption option is enabled during cluster creation, it cannot be disabled later. ● If the secure channel encryption option is not enabled for the ZooKeeper node selected during cluster creation, you can enable it on the cluster details page later.
Tags	A tag is a key-value pair customized by users and used to classify and search for cloud resources. A tag consists of a key and a value.
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** ● You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. ● You can delete a user or multiple users. ● After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.

Step 5 Set the parameters and click **Buy Now**.

Step 6 On the displayed page, confirm the cluster specification order information and click **Submit**. The cluster creation task is submitted.

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

The cluster creation task takes some time. The initial status of the cluster is **Creating**. After the cluster is created, the cluster status changes to **In service**.

Step 8 Submit the creation task of a yearly/monthly cluster.

Click **Pay**. On the displayed purchase page, confirm the information, select a proper payment method, and confirm the payment.

Return to the console and check the cluster status. Cluster creation takes some time. The initial status of the cluster is **Creating**. After the cluster is created, the cluster status changes to **In service**.

----End

5.4 Connecting to a ClickHouse Cluster

5.4.1 Preparing an ECS

Preparing an ECS

For details about how to purchase an ECS, see **Purchasing an ECS**.

To purchase an ECS, the following requirements must be met:

- The ECS must have the same region, AZ, VPC, and subnet as the CloudTable cluster.

  For details about how to create a VPC, see **"User Guide" > "VPC and Subnet"** of **Virtual Private Cloud**.
- The ECS must have the same security group as the CloudTable cluster.

  For more information about security groups, see **Security Group** in the *Virtual Private Cloud User Guide*.

  NOTE

  When cross-VPC communication is used to access a CloudTable cluster, the network administrator needs to authorize the access to the VPC, security group, and subnet where the cluster resides.
- When purchasing an ECS, you need to set **EIP** to **Automatically assign**. Alternatively, you can bind an EIP to an ECS after the ECS is created.
- To access a Linux ECS, you are advised to use an SSH password.

  For details about how to log in to a Linux ECS, see **Logging In to a Linux ECS** in the *Elastic Cloud Server User Guide*.

5.4.2 Using a Client to Connect to a ClickHouse Normal Cluster

You can use SQL to access a cluster on an ECS. For details about how to install the client, see **Installing the ClickHouse Client**.

NOTE

The VPC and security group of the cluster to be created must be the same as those of the ECS on the public network. Otherwise, the client cannot access the cluster.

Installing the ClickHouse Client

Step 1 Prepare a Linux ECS. For details, see **Preparing an ECS**.

Step 2 Download a client. Log in to the CloudTable console. In the navigation pane on the left, choose **Help**. In the right pane, click **Download Client** and **Client Verification File** to download the client installation package and client verification file.

Step 3 Install the client.

1. Use the SSH login tool to remotely log in to the Linux ECS through the EIP.

   For details, see **Logging In to a Linux ECS Using an SSH Password** in the *Elastic Cloud Server User Guide*.

2. Go to the root directory of the SSH login tool.
   ```
   cd /
   ```
3. Create a folder in the root directory.
   ```
   mkdir Folder name
   ```
4. Go to the directory of the created folder.
   ```
   cd /Folder name/
   ```
5. Place the client in the directory.
6. Decompress the client package.
   ```
   tar  -zxf   Client package name
   ```
7. Decompress the client verification file to the same directory as the client.
   a. Decompress the client verification file.
      ```
      cd <Path for storing the client verification file>
      tar -xzvf Client_sha256.tar.gz
      ```
   b. Obtain the client verification code.
      ```
      sha256sum ClickHouse_Client_23.3.tar.gz
      ```
   c. Check the verification code in the client verification file and compare it with the client verification code. If they are the same, the client is not tampered with. If they are different, the client is tampered with.
      ```
      less ClickHouse_Client_23.3.tar.gz.sha256
      ```
8. Go to the **clickhouse** folder and load the **.so** file.
   ```
   sh install.sh
   ```
9. Go to the **bin** directory.
   ```
   cd bin/
   ```
   Grant the 700 permission to the directory.
   ```
   chmod 700 clickhouse
   ```

Step 4 After the client is installed, run the following command to connect to the ClickHouse cluster. For details about the port, see **Table 5-13**.

```
./clickhouse client --host Internal IP address of the cluster   --port 9000 --user admin --password Password
```

- *Internal IP address of the cluster*: cluster access address on the cluster details page. Replace it with the access address of the cluster you purchased.
- *Password*: the password set when you purchase the cluster. If there are special characters, use backslashes (\) to escape them. If the password is enclosed in single quotation marks ('), the special characters do not need to be escaped.

Table 5-13 Custom security group rules

Direction	Action	Port/Range	Type	Destination/Source Address	Usage
Outbound	Allow	All	IPv4/IPv6	0.0.0.0/0	Permit in the outbound direction
Inbound	Allow	8123		Security group of the CloudTable ClickHouse cluster	ClickHouse HTTP port number
	Allow	9000			ClickHouse TCP port number
	Allow	8443			ClickHouse HTTPS port number
	Allow	9440			Secure TCP security port of ClickHouse

----End

Getting Started with ClickHouse

1. Create a database.
```
create database demo;
```
2. Use the database.
```
use demo;
```
3. Check the database in use.
```
select currentDatabase();
```
4. Create a table.
```
create table demo_t(uid Int32,name String,age UInt32,gender String)engine = TinyLog;
```
5. View the table structure.
```
desc demo_t;
```
6. Insert data.
```
insert into demo_t values(1,'Candy','23','M'),(2,'cici','33','F');
```
7. View the table.
```
select * from demo_t;
```
8. View the database and table.
   - View the database.
   ```
   show databases;
   ```
   - View the table.
   ```
   show tables;
   ```
9. Delete the database and table.
   - Delete the table.
   ```
   drop table demo_t;
   ```

NOTE

- Before deleting a table, check whether the table is in use.
- After a table is deleted, it can be restored within 24 hours. The restoration command is as follows:

```
set allow_experimental_undrop_table_query = 1;
UNDROP TABLE Table name;
```

– Delete the database.

```
drop database demo;
```

5.4.3 Using a Client to Connect to a ClickHouse Security Cluster

You can enable channel encryption to secure data transmission. This section describes how to enable a secure channel for a ClickHouse cluster.

Constraints

- Disabling HTTPS will pose risks to enterprise services.
- The HTTPS option is enabled during cluster creation and cannot be disabled later.
- If the HTTPS option is not enabled during cluster creation, it cannot be enabled later.
- The cluster restarts after the secure channels are enabled on the cluster details page.
- The secure and non-secure channels cannot be disabled after being enabled concurrently.

Enabling the Secure Channel

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 On the **Cluster Management** page, click **Buy Cluster** in the upper right corner. The **Buy Cluster** page is displayed.

Step 4 Check whether **Security Channel** is toggled on (default).

Additionally, you can toggle **Enable Secure and Non-secure Channels** on the cluster details page post-creation. This enables both secure and non-secure channels.

Figure 5-4 Secure channel

Step 5 Set the parameters and click **Next**.

Step 6 Confirm the cluster information and click **Submit**. After the cluster is created, go to its details page to view its security channel status.

----End

Downloading the Certificate and Configuring the config.xml File

Step 1 After the cluster is created, go to the cluster details page and click **Download certificates** on the right of **Channel Status** in the cluster information area.

Step 2 Use the SSH login tool to remotely log in to the Linux ECS through the EIP.

For details, see **Logging In to a Linux ECS Using an SSH Password** in the *Elastic Cloud Server User Guide*.

Step 3 Create and configure the config.xml file.

1. Save the certificate downloaded in **Step 1** to a custom path.
2. Create a config.xml file based on the following configuration file:
```
<config>
  <secure>true</secure>
  <openSSL>
   <client>
    <caConfig>/etc/ssl/certificate.crt</caConfig>
   </client>
  </openSSL>
</config>
```
   – **<caConfig>/etc/ssl/certificate.crt</caConfig>** indicates the path where certificates are stored.
   – **root** indicates the path for storing the configuration file.
   – The certificate can be downloaded only once per minute.
3. Save the configuration file to the root path.

----End

Using the ClickHouse Client to Connect to a Cluster

Step 1 After the certificate is configured, download the client. Log in to the CloudTable console. In the navigation pane on the left, choose **Help**. In the right pane, click **Download Client** and **Client Verification File** to download the client installation package and client verification file.

Step 2 Install the client.

1. Use the SSH login tool to remotely log in to the Linux ECS through the EIP.

   For details, see **Logging In to a Linux ECS Using an SSH Password** in the *Elastic Cloud Server User Guide*.
2. Go to the root directory of the SSH login tool.
```
cd /
```
3. Create a folder in the root directory.
```
mkdir Folder name
```
4. Go to the directory of the created folder.
```
cd /Folder name/
```
5. Place the client in the directory.
6. Decompress the client package.
```
tar  -zxf   Client package name
```
7. Decompress the client verification file to the same directory as the client.
   a. Decompress the client verification file.
```
cd <Path for storing the client verification file >
tar -xzvf Client_sha256.tar.gz
```


b. Obtain the client verification code.
```
sha256sum ClickHouse_Client_23.3.tar.gz
```
c. Check the verification code in the client verification file and compare it with the client verification code. If they are the same, the client is not tampered with. If they are different, the client is tampered with.
```
less ClickHouse_Client_23.3.tar.gz.sha256
```

8. Go to the **clickhouse** folder and load the **.so** file.
```
sh install.sh
```
9. Go to the **bin** directory.
```
cd bin/
```
Grant the 700 permission to the directory.
```
chmod 700 clickhouse
```

Step 3 Connect to the cluster.
```
./clickhouse client --host Internal IP address of the cluster --port 9440 --user admin --password Password --secure --config-file /root/config.xml
```

----End

Getting Started with ClickHouse

1. Create a database.
```
create database demo;
```
2. Use the database.
```
use demo;
```
3. Check the database in use.
```
select currentDatabase();
```
4. Create a table.
```
create table demo_t(uid Int32,name String,age UInt32,gender String)engine = TinyLog;
```
5. View the table structure.
```
desc demo_t;
```
6. Insert data.
```
insert into demo_t values(1,'Candy','23','M'),(2,'cici','33','F');
```
7. View the table.
```
select * from demo_t;
```
8. View the database and table.
   – View the database.
   ```
   show databases;
   ```
   – View the table.
   ```
   show tables;
   ```
9. Delete the database and table.
   – Delete the table.
   ```
   drop table demo_t;
   ```
   NOTE
   - Before deleting a table, check whether the table is in use.
   - After a table is deleted, it can be restored within 24 hours. The restoration command is as follows:
   ```
   set allow_experimental_undrop_table_query = 1;
   UNDROP TABLE Table name;
   ```
   – Delete the database.
   ```
   drop database demo;
   ```

5.4.4 Using HTTPS to Connect to a ClickHouse Secure Cluster

HTTPS is a secure version of HTTP. It protects data transmission by adding an SSL/TLS encryption layer between HTTP and TCP. You can connect to a ClickHouse cluster using HTTPS.

Constraints

- Disabling HTTPS will pose risks to enterprise services.
- The HTTPS option is enabled during cluster creation and cannot be disabled later.
- If the HTTPS option is not enabled during cluster creation, it cannot be enabled later.
- The cluster restarts after the secure channels are enabled on the cluster details page.
- The secure and non-secure channels cannot be disabled after being enabled concurrently.

Enabling the Secure Channel

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 On the **Cluster Management** page, click **Buy Cluster** in the upper right corner. The **Buy Cluster** page is displayed.

Step 4 Check whether **Security Channel** is toggled on (default).

Additionally, you can toggle **Enable Secure and Non-secure Channels** on the cluster details page post-creation. This enables both secure and non-secure channels.

Figure 5-5 Secure channel

Step 5 Set the parameters and click **Next**.

Step 6 Confirm the cluster information and click **Submit**. After the cluster is created, go to its details page to view its security channel status.

----End

Downloading the Security Certificate and Connecting to a ClickHouse Secure Cluster

Step 1 After a ClickHouse secure cluster is created, click the cluster name and click **Details**.

Step 2 On the cluster details page and click **Download certificates** on the right of **Channel Status** in the cluster information area.

Step 3 Use the SSH login tool to remotely log in to the Linux ECS through the EIP.

For details, see **Logging In to a Linux ECS Using an SSH Password** in the *Elastic Cloud Server User Guide*.

Step 4 Save the certificate downloaded in **Step 2** to a custom path. Create a config.xml file based on the following configuration file and save the file to the root path.

- **<caConfig>/etc/ssl/certificate.crt</caConfig>** indicates the path where certificates are stored.
- The certificate can be downloaded only once per minute.

```
<config>
   <secure>true</secure>
   <openSSL>
     <client>
       <caConfig>/etc/ssl/certificate.crt</caConfig>
     </client>
   </openSSL>
</config>
```

Step 5 Connect to the ClickHouse secure cluster.

Run the curl command to connect to the cluster.

```
echo 'select 1' | curl -H 'X-ClickHouse-User: user' -H 'X-ClickHouse-Key: password' --cacert /clickhouse/client/client/bin/certificate.crt 'https://host:port/?' --data-binary @-
```

Table 5-14 Parameter description

Parameter	Description
select 1	Executed SQL statement
user	Username for connecting to the cluster
password	Password created during cluster creation
/clickhouse/client/client/bin/certificate.crt	Path for storing the certificate
host, port	**host** indicates the private IP address, and **port** indicates the HTTPS port.

----End

5.5 Configuring ClickHouse User Permissions

You can centrally manage users, roles, and permissions on each ClickHouse node in a cluster. You can control user permissions by creating roles, creating users, and binding roles to users on the console.

Notes

- The deletion operation is irreversible. Even if a role with the same name is added immediately after the deletion, its permission may be different from that of the deleted one. Ensure that the role is not in use before deleting it.

- Before deleting a user, make sure that it is not needed anymore. The deletion operation is irreversible. Even if an account with the same name is added immediately after the deletion, the permission may be different from that of the deleted one.
- The **admin**, **root**, and **default** users are default users in a ClickHouse cluster and cannot be deleted.
- Only the passwords of users in the user list can be changed. The passwords of the **root** user and default users cannot be changed.
- Users and roles cannot be created for a single node in a cluster.
- If the permissions on a table are granted to a role, the table permissions are not removed when the table is deleted. If a table with the same name is created, the role inherits the original permissions on the table. When necessary, you can manually remove the table permissions that have been granted to the role.
- If user A has the permissions to operate table B, user A must manually revoke this permission before deleting table B. If not, user A will retain the same operation permission on any new table B created by user C.
- The password of an existing account cannot be viewed. If you forget the password, you can reset it.

Prerequisites

- A ClickHouse cluster has been created and is running properly.
- You have installed the ClickHouse client.

Step 1: Create a ClickHouse Role

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management**. On the displayed cluster list page, select the target cluster and choose **Cluster Name** > **Role Management**. The role management page is displayed.

Step 3 Click **Create Role**. On the displayed page, configure parameters below.

Table 5-15 Role permissions

Parameter	Description
Rolename	The role name must start with a letter and can contain 1 to 64 characters.
Global Permission	Global permissions refer to the permissions on all databases and tables, including the **SELECT**, **LOAD**, **ALTER**, **CREATE**, and **DROP** permissions.

Parameter	Description
Each Level Permission	**Database And Table** refers to the databases and tables created in the cluster. ● A role has high-risk permissions for databases but only common permissions for tables. ● Permission Type – High-risk permissions: **CREATE TABLE**, **DROP TABLE**, **CREATE VIEW**, and **DROP VIEW** – Common permissions: **SELECT**, **LOAD**, and **ALTER**

Step 4 Click **OK**.

----End

Step 2 Create a ClickHouse User

Step 1 After a role is created, choose **Account Management**.

Step 2 Click **Create Account**. On the **Create Account** page, set the username and password.

Table 5-16 User parameters

Parameter	Description
Username	The new username must start with a letter and contain 1 to 64 characters.
Password	Enter a password for the user. **NOTE** The password must meet the following requirements: ● Contain 8 to 16 characters. ● Contain at least four types of the following characters: uppercase letters, lowercase letters, digits, and special characters ($@!%*?&._{}()+=^<>) ● Cannot be the same as the username or the username spelled backwards.
Confirm Password	Enter the password again.

Step 3 Click **OK**.

----End

Step 3: Bind a Role to the ClickHouse User

For example, role A has the permission to query, insert, modify, create, and delete data. After being bound to role A, the user has the permissions of role A.

Step 1 After a role and user are created, choose **Account Management**, locate the target account, click **More**, and select **Assign Role** in the **Operation** column.

Step 2 In the **Assign Role** dialog box, select a role. Then, click **OK**.

Step 3 Click **Permission** in the **Operation** column and check that the user has the role permissions.

Step 4 Connect to the cluster as the created user.

For details about how to connect to a non-security cluster, see **Using a Client to Connect to a ClickHouse Normal Cluster**.

```
./clickhouse client --host Internal IP address of the cluster  --port 9000 --user admin --password Password
```

For details about how to connect to a security cluster, see **Using the ClickHouse Client to Connect to a Cluster**.

```
./clickhouse client --host Internal IP address of the cluster  --port 9440 --user admin --password Password --secure --config-file /root/config.xml
```

Step 5 Execute the query, insert, change, create, and delete commands in the CLI.

- If these commands can be executed, the role is bound successfully.
- If these commands cannot be executed, check whether the role has been configured with permissions and whether it has been bound to the user. If the fault persists, contact technical support.

----End

Managing User Permissions

- Managing roles
  - Deleting a role: Click **Delete** in the **Operation** column. On the displayed page, enter **DELETE** in the text box or click **Auto Enter**, and click **OK**.
  - Modifying role permissions: Click **Edit** in the **Operation** column. On the displayed page, select permissions as needed and click **OK**.
  - Viewing role permissions: Click **Permission** in the **Operation** column. On the displayed page, view the role's permissions on databases and tables.
- Managing users
  - Viewing user permissions: Click **Permission** in the **Operation** column. On the displayed page, view the user's permissions on databases and tables.
  - Deleting a user: Click **Delete** in the **Operation** column. In the displayed dialog box, click **Yes**.
  - Changing the user password: Click **More** and select **Update Password** in the **Operation** column. On the displayed page, change the password and click **OK**.

Common Commands for User Permissions

- Commands used to grant permissions to storage-compute coupled users
  a. Creating a role (**role_name** indicates the name of the role to be created)
  ```
  CREATE role IF NOT EXISTS 'role_name' ON CLUSTER default_cluster;
  ```
  b. Assigning permissions to a role
     - Granting all creation (database and table) permissions

```
GRANT CREATE ON *.* TO role1 ON CLUSTER default_cluster;
```

- Granting the permission to create tables in the **test_db** database
```
GRANT CREATE TABLE ON test_db.* TO role1;
```

- Granting the permission to delete all databases and tables
```
GRANT DROP ON test_db.* TO role1;
```

c. Deleting a role
```
DROP ROLE 'role_name' ON CLUSTER default_cluster;
```

d. Creating a user
```
CREATE USER  IF NOT EXISTS name ON CLUSTER default_cluster IDENTIFIED WITH
sha256_password BY 'password';
```

e. Specifying an IP address
```
CREATE USER name HOST IP '127.0.0.x' IDENTIFIED WITH sha256_password BY 'password';
```

f. Changing a user password
```
ALTER USER  IF EXISTS name ON CLUSTER  default_cluster IDENTIFIED WITH sha256_password
BY 'password';
```

g. Assigning a role to the user
```
GRANT role1, role2 TO new_user ON CLUSTER default_cluster;
```

h. Canceling the role assignment
```
REVOKE role1 FROM user ON CLUSTER default_cluster;
```

i. Deleting a user
```
DROP USER IF EXISTS 'name1' ON CLUSTER default_cluster;
```

j. Querying the permissions and roles of a user

- Querying permissions of a user
```
show grants for all;
```

- Querying permissions of a role
```
show grants for role1;
```

5.6 Data Migration and Synchronization

5.6.1 Importing and Exporting data

This section describes the basic syntax and usage of the SQL statements for importing and exporting file data using the ClickHouse client.

Importing and Exporting Data in CSV Format

- Import data in CSV format.
  - Non-security cluster
```
cat csv_ssl | ./clickhouse client --host 192.168.x.x --port port --user admin --password password
--database test010 --query="INSERT INTO test145 FORMAT CSV"
```
  - Security cluster
```
cat csv_no_ssl | ./clickhouse client --host 192.168.x.x --port port --user admin --password
password --config-file ./config.xml --database test010 --query="INSERT INTO test146 FORMAT
CSV"
```

1. **host**: indicates the host name or ClickHouse instance IP address.
2. **port**: indicates the port number (available on the cluster details page).
3. **user**: indicates the username created during cluster creation.
4. **database**: indicates the database name.

5. **password**: indicates the password specified during cluster creation.
6. **INSERT INTO**: indicates the target data table.
7. **cat** *File path*: indicates the user-defined path for storing the file.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

- Export data in CSV format.
  - Non-security cluster
    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password Password --database test010 -m --query="select * from test139 FORMAT CSV" > ./csv_no_ssl
    ```
  - Security cluster
    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 -m --query="select * from test139 FORMAT CSV" > ./csv_no_ssl
    ```

1. **host**: indicates the host name or ClickHouse instance IP address.
2. **port**: indicates the port number (available on the cluster details page).
3. **user**: indicates the username created during cluster creation.
4. **database**: indicates the database name.
5. **password**: indicates the password specified during cluster creation.
6. **SELECT * FROM**: indicates the target data table.
7. **./csv_no_ssl**: indicates the user-defined path for storing the file.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

Importing and Exporting Data in Parquet Format

- Import data in Parquet format.
  - Non-security cluster
    ```
    cat parquet_no_ssl.parquet | ./clickhouse client --host 192.168.x.x --port port --user admin --password password --database test010 --query="INSERT INTO test145 FORMAT Parquet"
    ```
  - Security cluster
    ```
    cat parquet_no_ssl.parquet | ./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 --query="INSERT INTO test146 FORMAT Parquet"
    ```

1. **parquet_no_ssl.parquet**: indicates the user-defined path for storing the file.
2. **host**: indicates the host name or ClickHouse instance IP address.
3. **port**: indicates the port number (available on the cluster details page).
4. **user**: indicates the username created during cluster creation.
5. **database**: indicates the database name.
6. **password**: indicates the password specified during cluster creation.
7. **INSERT INTO**: indicates the target data table.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

- Export data in Parquet format.
  - Non-security cluster
    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password password --database test010 -m --query="select * from test139 FORMAT Parquet" > ./parquet_no_ssl.parquet
    ```
  - Security cluster

```
./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 -m --query="select * from test139 FORMAT Parquet" > ./parquet_ssl.parquet
```

1. **host**: indicates the host name or ClickHouse instance IP address.
2. **port**: indicates the port number (available on the cluster details page).
3. **user**: indicates the username created during cluster creation.
4. **database**: indicates the database name.
5. **password**: indicates the password specified during cluster creation.
6. **select * from**: indicates the target data table.
7. **./parquet_no_ssl.parquet**: indicates the user-defined path for storing the file.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

Importing and Exporting Data in ORC Format

- Import data in ORC format.
  - Non-security cluster

    ```
    cat orc_no_ssl.orc | ./clickhouse client --host 192.168.x.x --port port --user admin --password password --database test010 --query="INSERT INTO test143 FORMAT ORC"
    ```

  - Security cluster

    ```
    cat orc_no_ssl.orc | ./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 --query="INSERT INTO test144 FORMAT ORC"
    ```

1. **cat orc_no_ssl.orc**: indicates the user-defined path for storing the file.
2. **host**: indicates the host name or ClickHouse instance IP address.
3. **port**: indicates the port number (available on the cluster details page).
4. **user**: indicates the username created during cluster creation.
5. **database**: indicates the database name.
6. **password**: indicates the password specified during cluster creation.
7. **INSERT INTO**: indicates the target data table.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

- Export data in ORC format.
  - Security cluster

    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 -m --query="select * from test139 FORMAT ORC" > ./orc_ssl.orc
    ```

  - Non-security cluster

    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password password --database test010 -m --query="select * from test139 FORMAT ORC" > ./orc_no_ssl.orc
    ```

1. **host**: indicates the host name or ClickHouse instance IP address.
2. **port**: indicates the port number (available on the cluster details page).
3. **user**: indicates the username created during cluster creation.
4. **database**: indicates the database name.
5. **password**: indicates the password specified during cluster creation.
6. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

7. **select * from**: indicates the target data table.
8. **/opt/student.orc**: indicates the user-defined path for storing the file.

Importing and Exporting Data in JSON Format

- Import data in JSON format.
  - Non-security cluster
    ```
    cat ./jsonnossl.json | ./clickhouse client --host 192.168.x.x --port port --user admin --password password --database test010 --query="INSERT INTO test141 FORMAT JSON"
    ```
  - Security cluster
    ```
    cat ./jsonssl.json | ./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 --query="INSERT INTO test142 FORMAT JSON"
    ```

1. **cat** *File path*: indicates the user-defined path for storing the file.
2. **host**: indicates the host name or ClickHouse instance IP address.
3. **port**: indicates the port number (available on the cluster details page).
4. **user**: indicates the username created during cluster creation.
5. **database**: indicates the database name.
6. **password**: indicates the password specified during cluster creation.
7. **INSERT INTO**: indicates the target data table.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

- Export data in JSON format.
  - Security cluster
    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password password --database test010 -m --query="select * from test139 FORMAT JSON" > ./jsonnossl.json
    ```
  - Non-security cluster
    ```
    ./clickhouse client --host 192.168.x.x --port port --user admin --password password --config-file ./config.xml --database test010 -m --query="select * from test139 FORMAT JSON" > ./jsonssl.json
    ```

1. **host**: indicates the host name or ClickHouse instance IP address.
2. **port**: indicates the port number (available on the cluster details page).
3. **user**: indicates the username created during cluster creation.
4. **database**: indicates the database name.
5. **password**: indicates the password specified during cluster creation.
6. **SELECT * FROM**: indicates the target data table.
7. **./jsonssl.json**: indicates the user-defined path for storing the file.
8. **config-file ./config.xml**: indicates the configuration file. For details, see **Using a Client to Connect to a ClickHouse Security Cluster**.

5.6.2 Accessing RDS MySQL Using ClickHouse

ClickHouse provides efficient data analysis in OLAP scenarios. It can map a table on the remote database server to a ClickHouse cluster through a database engine such as MySQL, so data can be analyzed in the ClickHouse cluster. This section describes how to interconnect a ClickHouse cluster with a MySQL database instance of RDS.

Prerequisites

- You have prepared the RDS database instance and the username and password of the database. For details, see **Connecting to a DB Instance from a Linux ECS**.
- A ClickHouse cluster has been created and is running properly.

Constraints

- The RDS database instance and ClickHouse cluster are in the same VPC and subnet.
- Before synchronizing data, you need to evaluate the impact on the performance of the source and destination databases. You are advised to synchronize data during off-peak hours.
- Currently, ClickHouse can interconnect with MySQL and PostgreSQL instances of RDS, but cannot interconnect with SQL Server instances.

Interconnecting ClickHouse with RDS Using the MySQL Engine

The MySQL engine is used to map tables on the remote MySQL server to ClickHouse and allows you to run INSERT and SELECT statements on tables to facilitate data exchange between ClickHouse and MySQL.

- Syntax for using the MySQL engine:

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster]
ENGINE = MySQL('host:port', ['database' | database], 'user', 'password')
```

Table 5-17 Parameters of the MySQL database

Parameter	Description
hostport	IP address and port number of the RDS MySQL database instance.
database	Name of the RDS MySQL database.
user	Username of the RDS MySQL database.
password	Password of the RDS MySQL database user.

Example of using the MySQL engine:

a. Connect to the MySQL database of RDS. For details, see **Connecting to a DB Instance from a Linux ECS**.

b. Create a table in the MySQL database and insert data into the table.

c. Run a client command to connect to ClickHouse.

In a non-security cluster:

```
./clickhouse client --host Internal IP address of the cluster  --port 9000 --user admin --password Password
```

For details about how to connect to a security cluster, see **Using a Client to Connect to a ClickHouse Security Cluster**.

```
./clickhouse client --host Internal IP address of the cluster  --port 9440 --user admin --password Password --secure --config-file /root/config.xml
```

NOTE

Internal IP address of the cluster: cluster access address on the cluster details page. Replace it with the access address of the cluster you purchased.

d. Create a MySQL database in ClickHouse. After the database is created, it automatically exchanges data with the MySQL server.

```
CREATE DATABASE mysql_db ENGINE = MySQL(IP address of the RDS MySQL database instance:Port number of the MySQL database instance, MySQL database name, MySQL database username, Password of the MySQL database user);
```

e. Switch to the created database **mysql_db**.

```
USE mysql_db;
```

Query the table data in the MySQL database in ClickHouse.

```
SELECT * FROM mysql_table;
┌─int_id─┬─float─┐
│      1 │     2 │
└────────┴───────┘
```

Data can be properly queried after being inserted.

```
INSERT INTO mysql_table VALUES (3,4);
SELECT * FROM mysql_table;
┌─int_id─┬─float─┐
│      1 │     2 │
│      3 │     4 │
└────────┴───────┘
```

5.7 ClickHouse Enterprise-class Enhancement

5.7.1 Viewing ClickHouse Slow Query Statements

There are two tabs on the **Slow Query Management** page: **Ongoing Slow Queries** and **Completed Queries**. By default, SQL statements whose latency exceeds 500 milliseconds in the cluster are displayed. You can manually determine whether to terminate an ongoing SQL statement.

Querying Ongoing Slow SQL Statements

Step 1 Log in to **the CloudTable console**.

Step 2 Click the name of the target cluster to access its details page. In the navigation pane, choose **Slow Query Management** > **Ongoing Slow Queries**.

Step 3 Use filter criteria such as the latency, time range, user, remote IP, and SQL statement to quickly find out or kill the desired slow queries.

Table 5-18 Ongoing slow query parameters

Parameter	Description
Query ID	ID generated internally
Start Time	Time when the execution of a slow query SQL statement starts
End Time	Time when the execution of a slow query SQL statement ends

Parameter	Description
Database	Database where the SQL statement is executed
SQL Statement	Ongoing SQL statement
Latency (ms)	Execution time of a slow query SQL statement. Unit: ms
User	User who executes the query
Remote IP	IP address of the client that submits the query
Status	Status of the slow query SQL statement: **RUNNING**
Operation	You can click **Kill Query** as required.

Table 5-19 Slow query filter conditions

Condition	Description
Slow query duration	Enter a filter value as required. The value cannot be less than 10 ms. You can also select 15 minutes, 30 minutes, or 1 hour on the console.
Slow query running date	Enter a date and time range for filtering as required.
Query by SQL	Filters slow queries based on SQL statements.
Query by user	Filters the slow queries based on users who execute the SQL statements. Fuzzy search based on part of a username is supported.
Query by remote IP address	Filters the slow queries based on the IP address of the client that submits slow query SQL statements.

----End

Querying Completed Slow SQL Statements

Step 1 Log in to **the CloudTable console**.

Step 2 Click the name of the target cluster to access its details page. In the navigation pane, choose **Slow Query Management** > **Completed Queries**.

Step 3 Use filter criteria such as the latency, time range, status, user, remote IP, and SQL statement to quickly find out the desired slow queries.

Table 5-20 Completed slow query parameters

Parameter	Description
Query ID	ID of each SQL command
Start Time	Time when the SQL statement starts to be executed
End Time	Time when the SQL statement execution ends
Database	Queried database
SQL Statement	Completed SQL statement
Latency (ms)	SQL execution duration
Status	SQL status, including **FINISHED** and **CANCELLED**.
User	User who submitted the SQL statement
Remote IP	Address of the client used by the user who submitted the SQL statement

----End

Slow SQL Query Statuses

The slow SQL query statuses include **FINISHED**, **RUNNING**, or **CANCELLED**. When the status is **RUNNING**, **Kill Query** is displayed in the **Operation** column.

Table 5-21 Status description

Status	Description
FINISHED	The SQL statement has been executed.
RUNNING	The SQL statement is running. You can determine whether to stop it.
CANCELLED	The execution of the SQL statement is canceled.

5.7.2 Migrating ClickHouse Data

This section describes the data migration capability.

Application Scenarios

Once you have performed a node scale-out in ClickHouse, a data migration is necessary.

Precautions

- Tables with partitions exceeding 50 GB do not support migration on the GUI and require manual migration.

- The table intended for migration must contain data, and a unique partitioning key must be defined. Empty tables or tables with multiple partitioning keys cannot be selected when creating a migration task.
- Each data table is limited to a single task. A cluster can only execute one task at a time.
- The local table's database must be either atomic (the default) or ordinary, and the table itself should be from the Mergetree family of engine tables, which includes both replicated and non-replicated types. Materialized view charts are not compatible.
- The backup relationship for local tables mirrors that of clusters. In the context of shards, distributed tables are utilized.
- By default, the original table becomes read-only during data migration.
- Initially, data is moved to a temporary table. Subsequently, the original table is swapped with the table containing the migrated data. This transition, which may result in reading incorrect data, is completed within seconds.
- Should cluster issues arise, the data migration process can be paused. After addressing the reported error in the cluster, the migration task can resume.
- The source and redistribution nodes involved in data migration must share identical tables.
- Data migration is not supported on a single node system.

Creating a Data Migration Task for Automatic Data Migration

Prerequisite: The data volume of the partitioned table is less than 50 GB.

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Go to the **Cluster Management** page and click a cluster name to go to the cluster details page.

Step 4 In the navigation pane, choose **Data Migration**.

Table 5-22 Data migration parameters

Parameter	Description
Task ID/Name	ID or name of the new migration task.
Logical Cluster	Name of the selected logical cluster.
Source Nodes	Node where data is stored.
Distribution Nodes	Node where data is distributed.
Status/Progress	Status/Progress of data distribution. The task can be in the initializing, running, or completed state.
Created	Task creation time.
Start Time	Task start time.

Parameter	Description
Update Time	Task modification time.
Operation	• **Start**: Start the task. • **Edit**: Edit task information. • **Cancel**: Cancel the task. • **Details**: View task details. • **Delete**: Delete the task.

Step 5 Click **New Task** in the upper left corner.

1. Enter a task name (starting with a letter).
2. Select a logical cluster.
3. Select the migration percentage.
4. Select the source node.
5. Select a redistribution node.
6. Select the data table to be migrated.

Step 6 Click **OK** to create the task.

Step 7 Click **Start** in the **Operation** column to start the created task.

----End

Manually Migrating Data

Prerequisites: The data volume of the partitioned table is greater than 50 GB.

- Step 1: Create a distributed table and a local table.
  a. Use the SSH tool to access the target cluster. For details, see **Installing the ClickHouse Client**.
  b. Go to the command window and create a database.
  ```
  CREATE DATABASE IF NOT EXISTS test_fetch on cluster default_cluster;
  ```
  c. Use the database.
  ```
  USE test_fetch;
  ```
  d. Create a local table.
  ```
  CREATE TABLE IF NOT EXISTS test_fetch.example_table on cluster default_cluster (d DateTime,
  a Int) ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_fetch/example_table',
  '{replica}') PARTITION BY toYYYYMM(d) ORDER BY d;
  ```
  e. Create a distributed table.
  ```
  CREATE TABLE IF NOT EXISTS test_fetch.example_table_dist ON CLUSTER default_cluster (d
  DateTime, a Int) ENGINE = Distributed('default_cluster', 'test_fetch', 'example_table', rand());
  ```
  f. Insert data into the distributed table and local table on shard 1. Consequently, all shards now contain data of 202306, 202307, and 202308, while only shard 1 contains data of 202309 and 2023010.
  ```
  INSERT INTO test_fetch.example_table_dist select '2023-06-20',rand64() from numbers(1000);
  INSERT INTO test_fetch.example_table_dist select '2023-07-20',rand64() from numbers(1000);
  INSERT INTO test_fetch.example_table_dist select '2023-08-20',rand64() from numbers(1000);
  INSERT INTO test_fetch.example_table select '2023-09-20',rand64() from numbers(1000);
  INSERT INTO test_fetch.example_table select '2023-10-20',rand64() from numbers(1000);
  ```

- Step 2: Migrate data between partitions using the **fetch partition** command. (Migrate data from partition 1 to partition 2.)
  a. Select the shard and partition of the data to be migrated.

```
SELECT
partition AS `Partition name`,
formatReadableSize(sum(data_uncompressed_bytes)) AS `Uncompressed size`,
formatReadableSize(sum(data_compressed_bytes)) AS `Compressed size`,
sum(rows) AS `Total number of rows`
FROM system.parts
WHERE (active = 1) AND (database = 'test_fetch') AND (table = 'example_table')
GROUP BY partition
ORDER BY partition DESC;
```

  The execution result is displayed.

```
┌─Partition Name─┬─Uncompressed Size─┬─Compressed Size─┬─Rows─┐
 202308    │ 3.81 KiB │ 1.98 KiB │    488 │
 202307    │ 4.12 KiB │ 2.14 KiB │    527 │
 202306    │ 3.88 KiB │ 2.02 KiB │    496 │
└──────────┴──────────┴──────────┴────────┘
```

  b. Execute the **fetch partition** command on any node within the target shard to retrieve the partition data. Subsequently, execute the **attach partition** command to incorporate the partition 202309 from the source shard and synchronize the data with the corresponding partition on the current shard.

```
ALTER TABLE test_fetch.example_table FETCH PARTITION '202309' FROM '/clickhouse/tables/1/
test_fetch/example_table/';
```

  1 in **'/clickhouse/tables/1/test_fetch/example_table/** indicates shard 1. You can change the value as required.

  Load the data obtained from the source shard to the current shard.

```
ALTER TABLE test_fetch.example_table ATTACH PARTITION '202309';
```

  NOTE

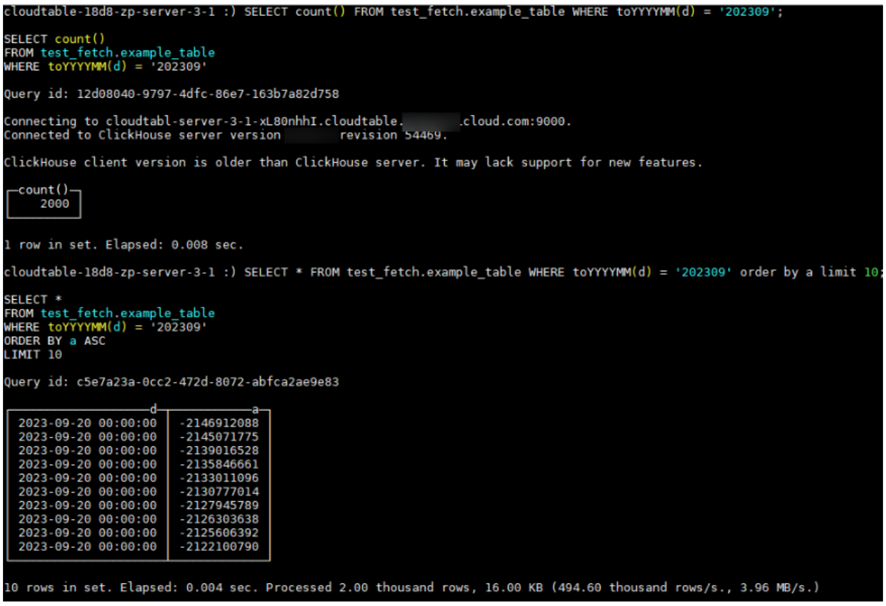
  During the migration process, if the target partition node displays a message indicating the presence of an existing detached partition, you can query the **detached_parts** system table to verify the partition's existence. If the partition already exists and contains redundant or inconsistent data, proceed to delete it. If the partition is required for other purposes, skip the migration of this specific partition and continue with the remaining partitions.

```
SELECT * FROM system.detached_parts WHERE table = 'example_table' AND partition_id =
'202309';
ALTER TABLE test_fetch.example_table DROP DETACHED PARTITION '202309' SETTINGS
allow_drop_detached = 1;
```

- Step 3: Verify the result.
  a. Check the number of data rows and sample data content.

```
SELECT count() FROM test_fetch.example_table WHERE toYYYYMM(d) = '202309';
SELECT * FROM test_fetch.example_table WHERE toYYYYMM(d) = '202309' order by a limit 10;
```

  Results display



b. Delete the corresponding partition on the source shard to deduplicate data.

   i. Method 1: Connect to the source shard to delete the data partition.

   ```
   ALTER TABLE test_fetch.example_table DROP PARTITION '202309';
   ```

   ii. Method 2: Detach the data.

   ```
   ALTER TABLE test_fetch.example_table DETACH PARTITION '202309';
   ```

   NOTE

   If a message appears indicating that the partition data cannot be deleted due to its size exceeding 50 GB, configure the following parameters and attempt to delete the partition again.

   ```
   set max_table_size_to_drop=0;
   set max_partition_size_to_drop=0;
   ```

   i. Following the deletion of the partition on the source shard, query the distributed table within the source shard and the local table within the target shard.

   ```
   SELECT count() FROM test_fetch.example_table_dist WHERE toYYYYMM(d) = '202309';
   SELECT count() FROM test_fetch.example_table WHERE toYYYYMM(d) = '202309';
   ```

Modifying a Data Migration Task

Step 1 Log in to **the CloudTable console**.

Step 2 Go to the **Cluster Management** page and click a cluster name to go to the cluster details page.

Step 3 Choose **Data Migration**.

Step 4 Click **Edit** in the **Operation** column.

Step 5 After modifying the parameters, click **OK**.

----End

Viewing Migration Task Details

Step 1 Log in to **the CloudTable console**.

Step 2 Go to the **Cluster Management** page and click a cluster name to go to the cluster details page.

Step 3 Choose **Data Migration**.

Step 4 Click **Details** in the **Operation** column to access the task details page.

Step 5 View task information.

----End

Deleting a Migration Task

Step 1 Log in to **the CloudTable console**.

Step 2 Go to the **Cluster Management** page and click a cluster name to go to the cluster details page.

Step 3 Choose **Data Migration**.

Step 4 Click **Delete** in the **Operation** column. In the displayed dialog box, click **OK** to delete the task.

----End

5.8 Managing ClickHouse Clusters

5.8.1 Checking the ClickHouse Cluster Status

You can check the cluster status on the console. In the navigation pane, click **Cluster Management**. On the **Cluster Management** page, a cluster list is displayed. If there are a large number of clusters, you can turn pages to view clusters in any state.

Table 5-23 Cluster management list parameters

Parameter	Description
Cluster Name	Cluster name specified when a cluster is created.
High Availability	If cluster HA is enabled when a ClickHouse cluster is created, "The cluster can be used only for testing services." is displayed. If it is not enabled, "--" is displayed.
Cluster Status	Cluster status, including **Creating**, **In service**, **Sub-health**, **Frozen**, and **Creation failed**.
Task Status	Task status of a cluster.
Database Engine	ClickHouse

Parameter	Description
Version	Cluster version.
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** • You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. • You can delete a user or multiple users. • After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.
Created	Time when a cluster is created.
Billing Mode	Billing mode of a cluster, which can be pay-per-use or yearly/monthly.
Operation	• Click **Monitor** to access the CloudTable service monitoring page. • Click **Specification Expansion**, **Disk Expansion**, or **Scale Out** to expand the cluster capacity. For details, see **Adjusting the Capacity of a ClickHouse Cluster**. • Click **Delete** to delete a cluster. • Click **Restart** to restart the cluster. • Choose **More** > **Change to Yearly/Monthly** to change the billing mode to yearly/monthly. • Click **More** > **Unsubscribe/Release** in the **Operation** column to unsubscribe from the yearly/monthly cluster.

Table 5-24 Icon description

Icon	Description
	Click to view all projects.
	Click  to refresh the cluster list.
	Enter a cluster name in the search box and click  to search for it.

Cluster Status

Table 5-25 Cluster status description

Status	Description
Creating	Indicates that a cluster is being created.
In service	If a cluster is successfully created and can provide services, the cluster status is **In service**.
Sub-health	If the cluster status cannot be monitored within the specified time, the cluster status changes to **Sub-health**. Manual intervention is required to recover a cluster that is in **Sub-health** status. For example, you can restart the cluster to recover the cluster.
Creation failed	Indicates that a cluster fails to be created.
Frozen	If the balance is insufficient for renewing a cluster, the cluster status is **Frozen**. If a cluster status is **Frozen**, you need to renew your subscription and ensure that your account balance is not 0 before unfreezing the cluster. **NOTE** A frozen cluster is unavailable and its all ECSs are shut down. After being unfrozen, the cluster recovers to the **In service** state. If you do not renew the cluster before the freeze period ends, the cluster will be deleted.

Task Status

Table 5-26 Task status description

Status	Description
Deleting	Indicates that a cluster is being deleted.
Restarting	Indicates that a cluster is being restarted.
Scaling out	Indicates that a cluster is being scaled out.

5.8.2 Viewing ClickHouse Cluster Details

You can monitor and manage the clusters you create. In the cluster list, locate the cluster to be viewed and click the cluster name to access the basic information page. You can view the cluster information and network configurations.

Table 5-27 Cluster information

Parameter	Description
Cluster Name	Cluster name specified when a cluster is created.
Cluster ID	Cluster ID.
Cluster Status	Cluster status information.
Version	Kernel version of the cluster.
Access Address	Address for accessing the cluster. You can click to copy the access link to the clipboard.
Access Address Port	Port for accessing the cluster. ● HTTP port: 8123 ● TCP port: 9000 ● HTTPS port: 8443 ● Secure TCP port: 9440
Billing Mode	Billing mode of the cluster. ● Pay-per-use ● Yearly/Monthly
Created	Time when a cluster is created.
Enterprise Project	You can group related cloud resources (for example, resources used for the same purpose) and manage them by enterprise project. **NOTE** ● You can delete a user that was authorized to manage an enterprise project if the enterprise's business has changed and the user should not have the permissions to manage the enterprise project. ● You can delete a user or multiple users. ● After a user that was authorized to manage an enterprise project is deleted, the user cannot manage the enterprise project. If the user needs to manage the enterprise project, authorize the user to manage the enterprise project again.
Database Engine	A core service used to store, process, and protect data.
Channel Status	Whether secure channels are enabled.
Enable Secure and Non-secure Channels	This parameter is displayed if secure channels are not enabled.
Client	Currently, the ClickHouse component has two types of clients: old and new. For a new client, the client identifier displays 'New.' Conversely, for existing clients, the identifier displays 'Old.'

Parameter	Description
Cluster Storage Mode	Coupled storage and compute

Table 5-28 Network configuration

Parameter	Description
Region	Working area of the cluster. Set this parameter when creating a cluster.
AZ	AZ you select during cluster creation.
VPC	VPC you select during cluster creation. A VPC is a secure, isolated, logical network environment.
Subnet	Subnet you select during cluster creation. A subnet provides dedicated network resources that are logically isolated from other networks, improving network security.
Security Group	Security group you select during cluster creation. If a security group needs to be added or modified due to service changes, click the pen icon on the right of **Security Group** in the **Network Configuration** area on the cluster details page to add or modify the security group. **NOTE** Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.

Table 5-29 Compute node parameters

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: ● 8U32G ● 8U64G ● 16U64G ● 16U128G ● 32U128G ● 32U256G ● 64U256G ● 64U512G ● 128U512G
Storage	Select the disk specifications and capacity of the ClickHouse compute node. ● Available storage specifications: – High I/O – General-purpose SSD – Ultra-high I/O – Extreme SSD ● The capacity ranges from 100 GB to 10,000 GB per node.
Nodes	Set the number of nodes in a cluster. The value ranges from **2** to **40**.

Table 5-30 ZooKeeper node parameters

Parameter	Description
Compute	Select proper compute specifications based on service requirements. Each specification displays the CPU, memory, and recommended application scenarios of a single node. **NOTE** Available compute specifications: ● 4U8G (This specification is not available for production environments.) ● 4U16G ● 8U32G ● 16U64G ● 32U128G The ZooKeeper small specifications include only 4U8G (4 vCPUs, 8 GB memory) and 4U16G (4 vCPUs, 16 GB memory). In the production environment, you are advised to select 8 vCPUs, 32 GB memory, or higher specifications.
Storage	**Ultra-high I/O** is recommended. Set the disk capacity to 100 GB per node.
Nodes	Number of nodes in a cluster. The default value is **3**.

5.8.3 Restarting a ClickHouse Cluster Node

If a CloudTable cluster node is abnormal, you can restart the node to restore the node status.

Precautions

- The node is unavailable during the restart.
- To minimize service disruption, schedule the node restart during off-peak hours.
- Disk scale-out, node scale-out, and specification expansion functionalities will be temporarily disabled during the node restart.
- Please note that a node restart is a process restart, not a full node reboot.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 Click the name of the cluster to be operated. The cluster details page is displayed.

Step 5 On the cluster details page, choose **Operation** > **Restart**. The **Restart Node** dialog box is displayed.

Step 6 Enter **RESTART** or click **Auto Enter**, and click OK to restart the node.

After the node is restarted, the **Restart** button is unavailable.

Step 7 Check the node restart result. If the restart is successful, the **Restart** button is highlighted. If the node fails to be restarted, the task status is **Failed to restart the node**, and the **Restart button** is highlighted, the node can be restarted again.

----End

Node Restart Statuses

Table 5-31 Restart statuses

Status	Description
Restart	The Restart button is highlighted, indicating either no restart operation has been initiated or that the restart has been completed successfully
Restarting	The cluster node is being restarted, and **Task Status** in **Cluster Information** is **Restarting**.
Restart failed	If a cluster node fails to be restarted and **Task Status** is **Failed to restart the node**, you can continue to restart the node.

5.8.4 Restarting a ClickHouse Cluster

If a cluster is in the unbalanced state or cannot work properly, you may need to restart it for restoration.

NOTE

- If your cluster is in arrears, this function may be unavailable. Please top up your account in time.
- The function is unavailable when the cluster status is subhealthy. Please contact technical support for assistance with restoring the cluster.

Impact on the System

- A cluster cannot provide services during the restart. Therefore, before the restart, ensure that no task is running and all data is saved.
- If a cluster is processing transactional data, for example, importing data, querying data, files may be damaged or the cluster may fail to be restarted once the cluster is restarted. You are advised to stop all cluster tasks before restarting a cluster.
- If the restart fails, the cluster may be unavailable. Try again later or contact technical support.

Procedure

Step 1 Log in to **the CloudTable console**.

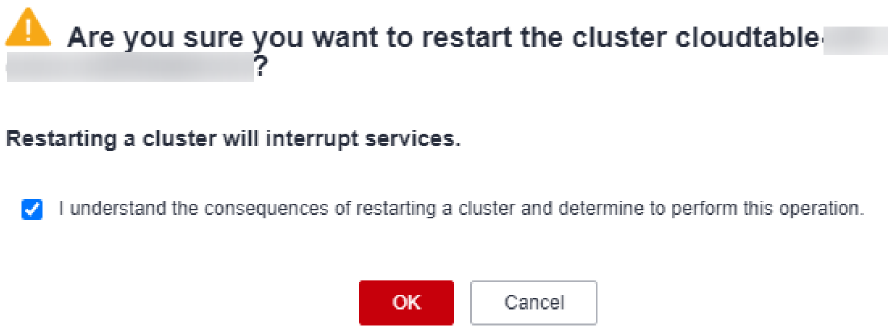
Step 2 Select a region in the upper left corner.

Step 3 Go to the **Cluster Management** page. In the upper right corner of the cluster list, enter the name of a cluster in the search box and click .

Step 4 In the **Operation** column of the cluster, click **More** > **Restart**.

Step 5 In the dialog box that is displayed, select the check box and click **OK** to restart the cluster.

Figure 5-6 Confirming the restart



----End

5.8.5 Deleting a ClickHouse Cluster

You can delete clusters you will no longer use. Deleting a CloudTable ClickHouse cluster will clear all resources and data related to the cluster. This operation cannot be undone. Exercise caution when deleting a cluster.

- Pay-per-use clusters can be directly deleted. For details, see **Deleting a Pay-per-Use ClickHouse Cluster**.
- Yearly/monthly clusters cannot be directly deleted. You need to unsubscribe from them (if they have not expired) or release them (if they have expired but have not been renewed). For details, see **Unsubscribing from or Releasing a Yearly/Monthly Cluster**.

Deleting a Pay-per-Use ClickHouse Cluster

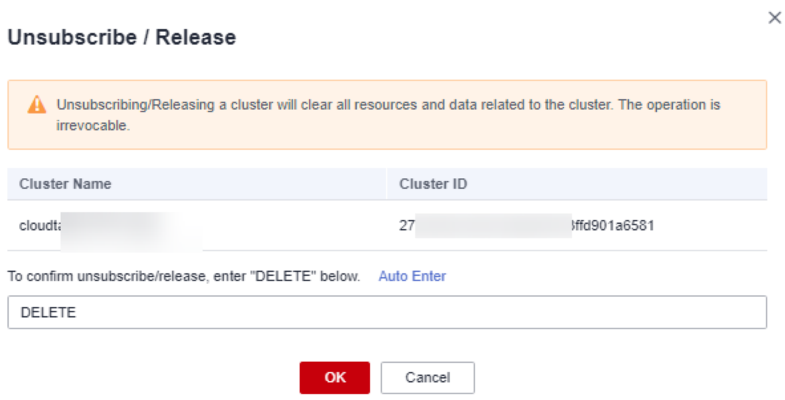
Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the **Operation** column of the cluster, click **More** > **Delete**.

Step 4 In the displayed dialog box, enter **DELETE** or click **Auto Enter**, and click **OK** to delete the cluster.

Figure 5-7 Confirming the deletion



----End

Unsubscribing from or Releasing a Yearly/Monthly Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the **Operation** column of the cluster, choose **More** > **Unsubscribe/Release**.

Step 4 In the displayed dialog box, enter **DELETE** or click **Auto Enter**, and click **OK** to unsubscribe from or release the cluster.

Step 5 On the **Unsubscribe** page, confirm the cluster information, select reasons for unsubscription, and confirm the unsubscription amount and related fees.

Step 6 Select "I've backed up the data or confirmed that the unsubscribed resources are no longer needed." and "I understand that only resources in the recycle bin can be restored after unsubscription". View the recycle bin description and click **Unsubscribe**.

Step 7 Return to the console and check whether the cluster has been unsubscribed from or released.

----End

5.8.6 Managing ClickHouse Cluster Tags

Tag Overview

A tag is a key-value pair customized by users and used to classify and search for cloud resources. A tag consists of a tag key and a tag value.

On CloudTable, after creating a cluster, you can add identifiers to items such as the project name, service type, and background information using tags. If you use tags in other cloud services, you are advised to create the same tags (same key-value pairs) for the cloud resources used by the same service to ensure consistency.

CloudTable supports the following tags:

- Resource tags

Non-global tags created in CloudTable.

- Predefined tags

  Predefined tags created on Tag Management Service (TMS). These tags are global tags.

  For details about predefined tags, see the *Tag Management Service User Guide*.

In CloudTable, tags can be added to the following resources:

- Clusters

  Tags can be added to a cluster when the cluster is being created or after it is successfully created. You can search for the cluster in the cluster list using tags.

  Each cluster can have a maximum of 20 tags.

  After you add tags to a cluster and then create a snapshot for the cluster, the tags cannot be restored if you use the snapshot to restore the cluster. Instead, you need to add tags again.

  When a cluster is deleted, non-predefined tags associated with the cluster are also deleted. Predefined tags need to be deleted on the TMS console.

  NOTE

  If you create a predefined tag that is identical to an existing predefined tag, the existing predefined tag will be overwritten. If you create a predefined tag that has the same key but different value, the new predefined tag will be created.

Adding a Tag to a Cluster

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Select the cluster to which you want to add a tag and choose *Cluster Name* > **Tag Management** > **Add Tag**. The **Add Tag** dialog box is displayed.

Step 4 Configure the tag parameters in the **Add Tag** dialog box.

Figure 5-8 Adding/Editing a tag

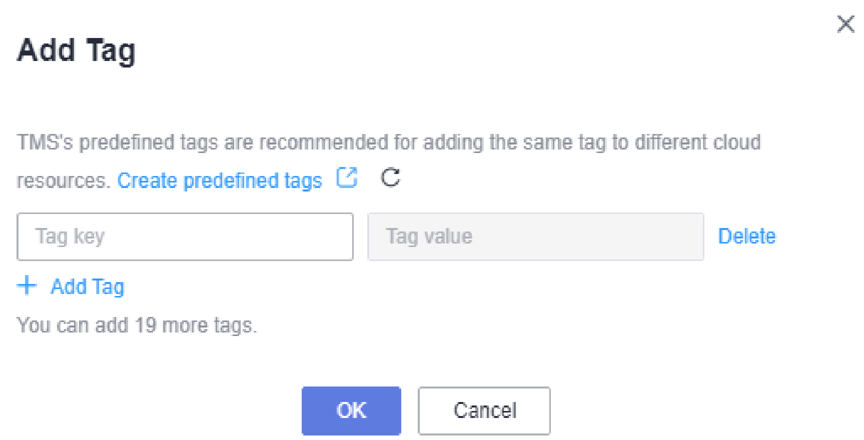


Table 5-32 Tag parameters

Parameter	Parameter description	Example Value
Key	You can: ● Select a predefined tag key or an existing resource tag key from the drop-down list of the text box. **NOTE** To add a predefined tag, you need to create one on TMS and then select it from the tag key drop-down list. You can click **Create Predefined Tags** to go to the **Predefined Tags** page of the TMS console. Then, click **Create Tag** in the upper corner of the page to create a predefined tag. ● Enter a tag key in the text box. **NOTE** A key must be unique in a given cluster. A tag key can contain letters, digits, spaces, and special characters (_.:=+-@), but cannot start or end with a space or start with **_sys_**.	key01
Value	You can specify the tag value in either of the following ways: ● Select a predefined tag value or resource tag value from the drop-down list of the text box. ● Enter a tag value in the text box. **NOTE** A tag value can contain letters, digits, spaces, and special characters (_.:=+-@).	value01

Step 5 Click **OK**. The new tag is displayed on the **Tag Management** page.

You can also add cluster tags when creating a cluster.

----End

5.9 ClickHouse Cluster O&M

5.9.1 Adjusting the Capacity of a ClickHouse Cluster

5.9.1.1 ClickHouse Cluster Node Scale-out

Add compute nodes in a ClickHouse cluster. The scale-out does not affect system running.

Constraints

- Scale-out cannot be performed if your account balance is insufficient.
- Scale-out cannot be performed if your quota is insufficient.
- ZooKeeper nodes cannot be scaled out.

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management**.

Step 3 In the cluster list, locate the row that contains the target cluster and click **Scale Out** in the **Operation** column.

NOTE

You can also click **Scale Out** on the cluster details page to go to the **Scale Out** page.

Step 4 Click **+** to add nodes.

Step 5 Confirm the fee and resource quota, and click **OK**.

----End

ClickHouse Scale-Out Statuses

Table 5-33 Scale-out statuses

Task Status	Description
Scaling out	Indicates that a cluster is being scaled out.
Scale-out failed	Indicates that the cluster fails to be scaled out.

Viewing Scaling Details

- After you click **OK**, the cluster task status changes to **Scaling out**. After the cluster scale-out is complete, the cluster status changes to **In service**.
- After you click **OK**, the cluster task status changes to **Scaling out**. If the scale-out fails, the cluster status is **In service** and the task status is **Scale-out failed**.

5.9.1.2 Expanding the Disk Capacity of a ClickHouse Cluster

The vertical capacity expansion is performed by expanding disk capacity. Data is usually stored on the compute nodes, and disk expansion is required when the disk capacity of the compute nodes is insufficient.

NOTE

- During disk capacity expansion, cluster services may jitter.
- You can perform disk capacity expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as node scale-out and parameter modification).

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management**.

Step 3 Select the cluster to be expanded and choose **Operation** > **Disk Expansion**.

NOTE

You can also click **Disk Expansion** on the cluster details page to go to the page for expanding disk capacity.

Step 4 Configure **Configured Node**.

Step 5 Press **+** to increase nodes.

Step 6 Confirm the fee and resource quota, and click **OK**.

----End

Disk Expansion Status

Table 5-34 Disk expansion status description

Status	Description
Disk expanding	Indicates that a cluster is being scaled out.
In service (cluster status)	This status is shown in **Cluster Status**, indicating that the scale-out is complete and the cluster can provide services.
Disk capacity expansion failed (task status)	This status is shown in **Task Status**, indicating that the cluster fails to be scaled out.

5.9.1.3 Changing the Specifications of a ClickHouse Cluster

Constraints

- You can only increase cluster specifications. To decrease cluster specifications, create a new cluster with the desired specifications and migrate data from the existing cluster.
- You can change the specifications of only one node type (the compute or ZooKeeper node) at a time. After the change, nodes in other types still maintain their original specifications.
- The entire cluster becomes temporarily unavailable during the specification change. Subsequent specification changes are not permitted until the current operation completes.
- You are advised to change the specifications during off-peak hours to avoid affecting services.

Prerequisites

You can perform specification expansion on a cluster when the cluster is in the **In service** state and no task is being performed (such as node scale-out and disk expansion).

Procedure

Step 1 Log in to **the CloudTable console**.

Step 2 Click **Cluster Management**.

Step 3 In the cluster list, locate the row that contains the target cluster and choose **More** > **Specification Expansion** in the **Operation** column.

NOTE

You can also click **Specification Expansion** on the cluster details page to switch to the **Specification Expansion** page.

Step 4 Configure **Configured Node**.

Step 5 Select the CPU and memory specifications from the drop-down list.

Step 6 Confirm the fee and resource quota, and click **OK**.

After you click **OK**, the cluster status changes to **Sub-health** and the task status changes to **Resizing Flavor**. After the cluster specifications are changed, the cluster status changes to **In service** and the task status is cleared.

----End

Specification Expansion Status

Table 5-35 Specification expansion status description

Status	Description
Resizing Flavor	Indicates that the specifications of the target cluster are being changed.
In service	Indicates that the specifications of the target cluster are changed and the cluster can provide services.
Resizing flavor failed	Indicates that the specifications of the target cluster failed to be changed.

5.9.2 Using Cloud Eye to Monitor a ClickHouse Cluster

5.9.2.1 ClickHouse Cluster Monitoring Metrics

Description

Monitoring is critical to ensure CloudTable reliability, availability, and performance. You can monitor the running status of CloudTable servers. This section describes the metrics that can be monitored by Cloud Eye as well as their namespaces and dimensions.

Namespace

SYS.CloudTable

Monitoring metrics

Table 5-36 Monitoring metrics

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
cmdProcessCPU	CPU Utilization	CPU utilization of the monitored object	0~100	%	N/A	CloudTable instance node	60s
memory_usage	Memory Usage	Memory usage of each node.	[0, Node memory capacity]	MiB	1024(IEC)	CloudTable instance node	60s
cmdProcessMem	Memory Utilization	Memory utilization of the monitored object	0~100	%	N/A	CloudTable instance node	60s
disk_usage	Disk Usage	Disk space used by each node.	[0, Node disk capacity]	MiB	1024(IEC)	CloudTable instance node	60s
cmdForUsedStorageRate	Used Storage Rate	Ratio of the used storage space to the total storage space in the cluster	>=0	%	N/A	CloudTable instance node	60s
network_throughput_inbound_rate	Inbound Throughput	Inbound data volume over network of each node per second.	>=0	KiB/s	1024(IEC)	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
network_throughput_outgoing_rate	Outbound Throughput	Outbound data volume over network of each node per second.	>=0	KiB/s	1024(IEC)	CloudTable instance node	60s
number_of_mutations	Mutations	Number of running modification tasks on each node.	>=0	Count	N/A	CloudTable instance node	60s
number_of_run_merges	Running Merges	Number of running merge tasks.	>=0	Count	N/A	CloudTable instance node	60s
number_of_merges_thread	Merge Threads	Number of threads being merged.	[0, Number of CPU cores]	Count	N/A	CloudTable instance node	60s
number_of_zookeeper_requests	ZooKeeper Requests	Number of ZooKeeper requests on the node at the current time.	>=0	Count	N/A	CloudTable instance node	60s
number_of_zk_session	ZooKeeper Sessions	Number of connected ZooKeeper sessions on the node at the current time.	>=0	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_zookeeper_watches	ZooKeeper Watches	Number of connected ZooKeeper watches on the node at the current time.	>=0	Count	N/A	CloudTable instance node	60s
number_of_zookeeper_nodes	ZooKeeper Nodes	Number of temporary nodes created in ZooKeeper.	>=0	Count	N/A	CloudTable instance node	60s
number_of_parts	Parts	Number of parts.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_holding_read_locks	Threads Holding the Read Lock	Number of threads that hold read locks.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_waiting_to_be_read	Threads Waiting to Be Read	Number of threads waiting to be read.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_waiting_to_be_written	Number of Threads Waiting to Be Written	Number of threads waiting to be written.	>=0	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_threads_holding_write_locks	Threads Holding the Write Lock	Number of threads that hold write locks.	>=0	Count	N/A	CloudTable instance node	60s
maximum_number_of_active_blocks	Max. Active Blocks	Maximum number of active blocks of the ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_consistency_checkdata_blocks	Data Blocks for Consistency Check	Number of data blocks for consistency check on a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
extracting_the_number_of_data_chunks_from_the_replica	Data Blocks Being Extracted from a Replica	Number of data blocks that are being extracted from a replica by a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_data_chunks_being_sent_to_the_replica	Data Blocks Being Sent to a Replica	Number of data blocks on a ClickHouse instance that are being sent to a replica.	>=0	Count	N/A	CloudTable instance node	60s
number_of_readonly_replicas	Read-Only Replicas	Number of read-only replicas of a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_inserted_into_all_tables	Inserted Rows	Number of rows inserted into all tables in a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_open_read_files	Opened Read Files	Number of opened read files on a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_open_write_files	Opened Write Files	Number of opened write files on a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
total_file_opens	Opened Files	Number of opened files on a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_read_file_descriptors	Read File Descriptors	Number of file descriptors that have been read.	>=0	Count	N/A	CloudTable instance node	60s
number_of_written_file_descriptors	Write File Descriptors	Number of file descriptors that have been written.	>=0	Count	N/A	CloudTable instance node	60s
instance_process_running_duration	Instance Process Running Duration	Running duration of the process on a ClickHouse instance.	>=0	s	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_threads_running_in_global_thread_pool	Threads of Running Tasks in Global Thread Pool	Number of threads of running tasks in the global thread pool of a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_in_global_thread_pool	Threads in Global Thread Pool	Number of threads in the global thread pool of a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_in_local_thread_pool	Threads in Local Thread Pool	Number of threads in the local thread pool of a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_waiting_locked	Threads to Lock in Context	Number of threads waiting to be locked in context on a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s
number_of_threads_running_in_local_thread_pool	Threads of Running Tasks in Local Thread Pool	Number of threads of running tasks in the local thread pool of a ClickHouse instance.	>=0	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_tcp_connections	TCP Connections	Number of TCP connections.	[0,4096]	Count	N/A	CloudTable instance node	60s
number_of_http_connections	HTTP Connections	Number of HTTP connections.	[0,4096]	Count	N/A	CloudTable instance node	60s
number_of_databases	Databases	Number of databases on each node.	[0, 5000]	Count	N/A	CloudTable instance node	60s
number_of_tables	Tables	Number of local tables on each node.	[0, 5000]	Count	N/A	CloudTable instance node	60s
write_size_per_second	Write Size per Second	Amount of data written to each node per unit time.	>=0	Byte, KiB, MiB, GiB, TiB, and PiB	1024(IEC)	CloudTable instance node	60s
number_of_running_queries	Running Queries	Number of running queries on each node.	>=0	Count	N/A	CloudTable instance node	60s
number_of_query_threads	Query Threads	Number of running queries on each node.	>=0	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_distributed_ddls	Distributed DDLs	Number of distributed DDL statements.	>=0	Count	N/A	CloudTable instance node	60s
number_of_distributed_table_files	Distributed Tables	Number of tables to which data is to be inserted.	>=0	Count	N/A	CloudTable instance node	60s
number_of_concurrency	Concurrency of Each ClickHouse Node	Concurrency of each ClickHouse node.	[0,100]	Count	N/A	CloudTable instance node	60s
zookeeper_quota_percentage	ZooKeeper Quantity Percentage	Quota of the ZooKeeper nodes used by ClickHouse.	0~100	%	N/A	CloudTable instance node	60s
zookeeper_capacity_quota_percentage	ZooKeeper Capacity Percentage	Capacity quota of ZooKeeper directory used by ClickHouse.	0~100	%	N/A	CloudTable instance node	60s
zookeeper_process_is_alive	ZooKeeper Process Survival Status	ZooKeeper heartbeat check. **1** indicates that the ZooKeeper heartbeat is normal, and **0** indicates that it is abnormal.	[0, 1]	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_global_sessions	Global Sessions	Number of global sessions.	>=0	Count	N/A	CloudTable instance node	60s
number_of_rejected_connections	Refused Connections	Number of connections refused by Zookeeper per unit time.	>=0	Count	N/A	CloudTable instance node	60s
number_of_request_submission_queues	Requests to Submit in a Queue	Number of requests waiting to be submitted in a queue at the current time.	>=0	Count	N/A	CloudTable instance node	60s
waiting_time_of_the_preprocessing_queue	Wait Time of Request in Preprocessing Queue	Time that a request in the preprocessing queue waits to be processed per unit time.	>=0	s	N/A	CloudTable instance node	60s
number_of_zk_watches	ZooKeeper Watches	Number of ZooKeeper watches at the current time.	>=0	Count	N/A	CloudTable instance node	60s
zk_heapmemory_usage	Heap Memory Used	Heap memory size used by ZooKeeper process.	>=0	Byte, KiB, MiB, GiB, TiB, and PiB	1024(IEC)	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
zk_directmemory_usage	Used Direct Memory	Direct memory size used by ZooKeeper process.	>=0	Byte, KiB, MiB, GiB, TiB, and PiB	1024(IEC)	CloudTable instance node	60s
zk_heapmemory_usage_ratio	Heap Memory Usage	Heap memory usage of the ZooKeeper service.	0~100	%	N/A	CloudTable instance node	60s
zk_directmemory_usage_ratio	Direct Memory Usage	Direct memory usage of the ZooKeeper service.	0~100	%	N/A	CloudTable instance node	60s
cluster_number_of_sent_packets	ZooKeeper Sent Packets	Number of packets sent by a ZooKeeper node.	>=0	Count	N/A	CloudTable instance node	60s
cluster_number_of_received_packets	ZooKeeper Received Packets	Number of packets received by a ZooKeeper node.	>=0	Count	N/A	CloudTable instance node	60s
cluster_preprocessing_time	Preprocessing Duration	Preprocessing time.	>=0	s	N/A	CloudTable instance node	60s
rows_written_per_second	Rows Written per Second	Number of rows written to each node per unit time.	>=0	Count	N/A	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
number_of_failed_queries	Failed Queries	Number of failed query statements.	>=0	Count	N/A	CloudTable instance node	60s
number_of_failed_insert_queries	Failed Insert Queries	Number of failed insert statements.	>=0	Count	N/A	CloudTable instance node	60s
number_of_failed_select_query_messages	Failed Select Queries	Number of failed select statements.	>=0	Count	N/A	CloudTable instance node	60s
number_of_delayed_inserts	Delayed Inserts	Number of delayed insertions on each node.	>=0	Count	N/A	CloudTable instance node	60s
number_of_rows	Rows	Number of mergetree engine fields on each node.	>=0	Count	N/A	CloudTable instance node	60s
disk_throughput_read_rate	Disk Read Throughput	Disk read throughput.	>=0	Byte/s, KiB/s, MiB/s, GiB/s, TiB/s, and PiB/s	1024(IEC)	CloudTable instance node	60s

Metric Name	Name	Description	Value Range	Metric Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period
disk_throughput_write_rate	Disk Write Throughput	Disk write throughput.	>=0	Byte/s, KiB/s, MiB/s, GiB/s, TiB/s, and PiB/s	1024(IEC)	CloudTable instance node	60s

Dimension

Key	Value
cluster_id	CloudTable cluster ID. To obtain the value, go to the cluster management page, click the cluster name, access its details page, and obtain the cluster ID in the cluster information area.
instance_name	Name of a CloudTable cluster node. To obtain the value, go to the cluster management page, click the cluster name, access its details page, and obtain the value of **instance_name**.

5.9.2.2 Setting ClickHouse Cluster Monitoring Rules

To monitor the usage of cloud service resources or key operations on them, you can create an alarm rule. After an alarm rule is created, if a metric reaches the specified threshold or there is a specified event, Cloud Eye immediately informs you of the exception through Simple Message Notification (SMN).

You can set CloudTable ClickHouse alarm rules to customize the monitored objects and notification policies. Then, you can learn about ClickHouse running status in a timely manner. The ClickHouse alarm rules include alarm rule name, instance, metric, threshold, monitoring interval and whether to send notification. This section describes how to set alarm rules.

Setting a ClickHouse Cluster Alarm Rule

Step 1 Log in to **the CloudTable console**.

Step 2 In the navigation pane, choose **Alarm Management** > **Alarm Rules**.

Step 3 In the **Resource Type** column, select CloudTable Service to filter alarm rules.

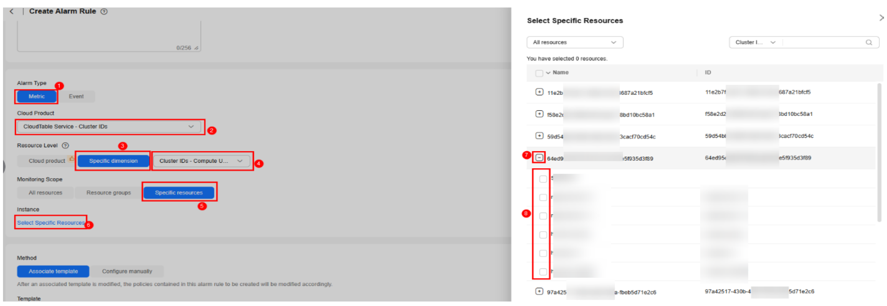
If there are no alarm rules that meet your requirements, create a CloudTable alarm rule by referring to **Creating an Alarm Rule and Notifications**. You need to set the following parameters and retain the default values of other parameters.

Table 5-37 Alarm rule configuration parameters

Parameter	Description
Alarm Type	Select **Metric**.
Cloud Product	Select CloudTable Service.
Resource Level	Select a level as required. ● **Cloud product**: Alarm rules are specified by product. Metrics across dimensions can be configured in the same alarm rule. ● **Specific dimension** > **Cluster IDs - Compute Units**: Alarm rules are specified by cluster node.

Figure 5-9 Configuring cluster alarm rules

Figure 5-10 Configuring alarm rules for cluster nodes



Step 4 After the configuration is complete, click **Next**.

After the alarm rule is created, if the metric data reaches the specified threshold, Cloud Eye will immediately inform you that an exception has occurred.

----End

Viewing an Alarm Rule

Step 1 Log in to **the CloudTable console**.

Step 2 On the **Cluster Management** page, locate the target cluster and click **View Metric** in the **Operation** column to go to the **Details** page.

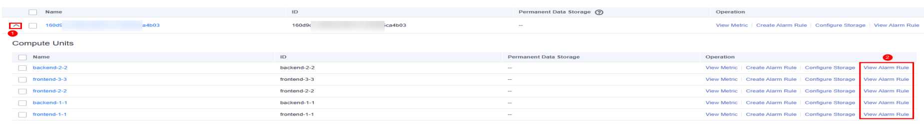
Step 3 Viewing alarm rules for clusters or cluster nodes.

1. View cluster alarm rules. Click **View Alarm Rule** in the **Operation** of the target cluster. In the displayed dialog box, you can view all alarm rules of the cluster.

Figure 5-11 Viewing cluster alarm rules

2. View alarm rules for cluster nodes. Click to expand cluster nodes. Click **View Alarm Rule** in the **Operation** of the target node. In the displayed dialog box, you can view all alarm rules of the cluster node.

Figure 5-12 Viewing alarm rules for cluster nodes



----End

5.9.2.3 Viewing ClickHouse Cluster Monitoring Information

Scenario

Cloud Eye monitors the running status of ClickHouse clusters. You can view the monitoring metrics of ClickHouse on the management console. According to the monitoring information, you can quickly learn about cluster health status and key system information.

Monitoring Function

The cluster monitoring function consists of compute node and ZooKeeper metric monitoring. You can adjust the time range of monitoring data in each module to view historical data at different time. You can also adjust the time granularity of monitoring data to view data in different dimensions. If you want to view monitoring data in real time, enable automatic refresh. The platform can automatically refresh monitoring graphs at intervals of 1 hour, 3 hours, 12 hours, 24 hours, and 7 days.

If you are interested in a specific metric graph, you can click the zoom-in button to view the graph or export the graph.

Viewing ClickHouse Cluster Monitoring Information on the Cloud Eye Console

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 In the cluster list on the **Cluster Management** page, locate the row where the target cluster resides, click **View Metric** in the **Operation** column. The Cloud Eye console is displayed.

The status of the cluster to be viewed must be **In service**.

Step 4 On the **Cloud Service Monitoring** page, click on the left of the cluster ID to expand the compute unit list, and select the corresponding node to view the monitoring information.

Step 5 Set the metrics to be viewed if there are too many metrics on the monitoring page.

1. If there are too many metrics, delete them on the **Select Metric** page.
2. If the metrics displayed on the page do not contain the desired metrics, add the metrics on the **Select Metric** page.
3. Select at least one metric. You can drag a selected metric and drop it to a desired location to sort the metrics.

NOTE

- If you want to view monitoring data in real time, enable automatic refresh. The platform can automatically refresh monitoring graphs at intervals of 1 hour, 3 hours, 12 hours, 24 hours, and 7 days.
- If you want to zoom in on a single metric view, click in the upper right corner of the metric view to view its details.

----End

Viewing ClickHouse Cluster Monitoring Information on the Cluster Details Page

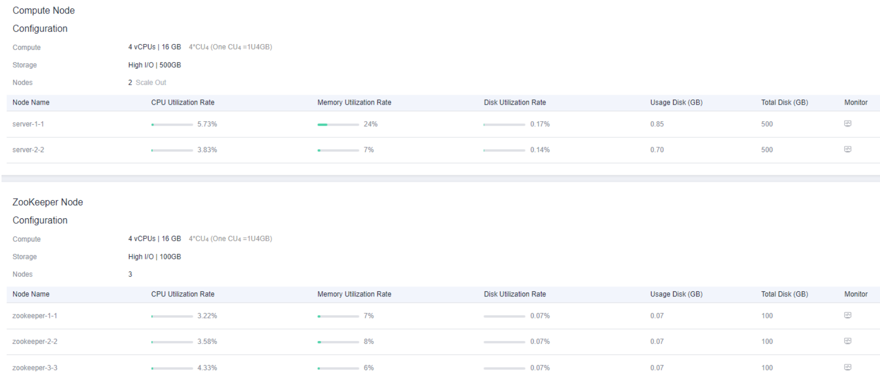
The cluster details page displays the CPU, memory, disk utilization rate, as well as the disk usage and total disk capacity of each ClickHouse node.

Step 1 Log in to **the CloudTable console**.

Step 2 Go to the **Cluster Management** page, select the target ClickHouse cluster, and click the cluster name to go to the cluster details page.

Step 3 View the monitoring metrics on the cluster details page.

Figure 5-13 ClickHouse monitoring page



Step 4 To view all node metrics, navigate to the Cloud Eye monitoring page from the details page by clicking the monitoring icon . This page provides detailed monitoring metrics for individual ClickHouse cluster nodes.

----End

5.9.3 Managing ClickHouse Cluster Logs

5.9.3.1 Viewing ClickHouse Cluster Logs with LTS

Cluster logs are collected and sent to Log Tank Service (LTS). You can check or dump the collected cluster logs on LTS.

NOTE

Currently, the following log types are supported:

Path for storing ClickHouse cluster logs: /var/log/clickhouse/

- clickhouse-server.log
- clickhouse-server.err.log
- clickhouse-server-audit.log

Enabling LTS

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Click the toggle button in the upper left corner of the page to enable LTS.

NOTE

- If this function is enabled for the first time, the **Create Agency** dialog box is displayed. Click **OK** to authorize the agency.
- If LTS has been enabled and authorized to create an agency, no authorization is required when LTS is enabled again.

----End

Checking Cluster Logs

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Select **View Logs** in the **Operation** column. The LTS console is displayed.

----End

Disable Logging

Step 1 Log in to **the CloudTable console**.

Step 2 Select a region in the upper left corner.

Step 3 Click **Clusters**. The cluster list is displayed.

Step 4 In the cluster list, click a cluster name to go to the cluster details page. In the navigation pane on the left, click **Logs**. The **Logs** page is displayed.

Step 5 Toggle off the LTS switch.

Step 6 Click **OK** in the dialog box.

----End

5.9.3.2 Viewing ClickHouse Cluster Logs with CTS

CloudTable uses CTS to record operations associated with CloudTable for later query, audit, and backtrack operations. After you enable CTS, the system starts recording CloudTable operation logs. Operation records of the last 7 days can be viewed on the CTS management console.

The following key operation traces of CloudTable are recorded in audit logs. For details, see **Table 5-38**.

Table 5-38 CloudTable operation traces supported by CTS

Operation	Trace Name	Resource Type
Creating a cluster	createCloudTableClusterV3	cluster
Scaling out a node	growCloudTableCluster	cluster

Operation	Trace Name	Resource Type
Restarting a cluster	rebootCloudTableCluster	cluster
Setting the storage quota	storageClusterAction	cluster
Modifying a feature	modifyClusterFeatures	cluster
Creating a data migration task	copierCreateTask	cluster
Enumerating database information	copierListDatabaseInfo	cluster
Enumerating cluster node information	copierListNodeInfo	cluster
Querying the migration task details	copierTaskDetail	cluster
Creating a ClickHouse user	createCloudTableAccount	cluster
Creating a role	createRole	cluster
Deleting a cluster	deleteCloudTableClusterV2	cluster
Deleting a role	deleteRole	cluster
Disabling cluster logs	disableLTSAccess	cluster
Enabling cluster logs	enableLTSAccess	cluster
Obtaining cluster information	getClusterInfo	cluster
Obtaining database information	getDatabases	cluster
Obtaining role information	getRoles	cluster
Obtaining table information	getTables	cluster
Accessing the disk expansion page	growCloudTableDisk	cluster
Expanding specifications	growCloudTableFlavor	cluster
Restarting a node	restartInstance	cluster
Restart	REBOOTING	cluster

Operation	Trace Name	Resource Type
Capacity expansion	GROWING	cluster
Deletion	DELETING	cluster
Changing the specifications of a yearly/monthly-billed cluster	changeCloudTableCluster	cluster
Enabling both secure and non-secure streams for ClickHouse	enableBothSslAndNone	cluster
Scanning and killing SQL statements	killQueryBySqlId	cluster
Deleting a ClickHouse user	deleteCloudTableAccount	cluster
Updating a ClickHouse user	updateCloudTableAccount	cluster

Enabling CTS

A tracker will be automatically created after CTS is enabled. All traces recorded by CTS are associated with a tracker. Currently, only one tracker can be created for each account.

Step 1 Log in to the **CTS console**.

Step 2 Enabling CTS

If you are a first-time CTS user and do not have any created tackers in the tracker list, refer to **Overview**.

If you have enabled CTS, the system has automatically created a management tracker. Only one management tracker can be created and it cannot be deleted. You can also manually create a data tracker. For details, see **Creating a Tracker** in the *Cloud Trace Service User Guide*.

----End

Disabling the Audit Log Function

If you want to disable the audit log function, disable the tracker in CTS.

Step 1 Log in to the **CTS console**.

Step 2 Disable the audit log function by disabling the tracker. To enable the audit log function again, you only need to enable the tracker.

For details about how to enable or disable a tracker, see **Disabling or Enabling a Tracker** in the *Cloud Trace Service Getting Started*.

----End

Viewing CTS Logs of CloudTable

Step 1 Log in to the **CTS console**.

Step 2 In the navigation pane on the left, choose **Trace List**.

Step 3 In the upper right corner of the trace list, click **Filter** to set the search criteria.

The following four filter criteria are available:

- **Trace Source**, **Resource Type**, and **Search By**
  - **Trace Source**: Select **CloudTable**.
  - **Resource Type**: Select **All resource types** or specify a resource type.
  - **Search By**: Select **All** or any of the following options:
    - **Trace name**: If you select this option, you also need to select a specific trace name.
    - **Resource ID**: If you select this option, you also need to select or enter a specific resource ID.
    - **Resource name**: If you select this option, you also need to select or enter a specific resource name.
- **Operator**: Select a specific operator (at user level rather than tenant level).
- **Trace Status**: Available options include **All trace statuses**, **normal**, **warning**, and **incident**. You can only select one of them.
- **Start Date** and **End Date**: You can specify the time period to query traces.

Step 4 Click **Query**.

Step 5 Click  on the left of the trace to be queried to extend its details.

Figure 5-14 Trace

Step 6 Locate the row containing the target trace and click **View Trace** in the **Operation** column.

Figure 5-15 Viewing a trace


```
{
    "service_type": "CloudTable",
    "user": {
        "name": "[illegible]",
        "id": "2f210e62179a425c96e5b57f25d1d492",
        "domain": {
            "name": "[illegible]",
            "id": "760dfba8429649e7a477445e1033a839"
        }
    },
    "time": "05/29/2018 14:50:16 GMT+08:00",
    "code": 200,
    "resource_type": "cluster",
    "resource_name": "cloudtable-[illegible]",
    "resource_id": "689ae304-086b-4fa9-984e-16c58e3d75bf",
    "source_ip": "[illegible]",
    "trace_name": "rebootCloudTableCluster",
    "trace_type": "ConsoleAction",
    "api_version": "v1.0",
    "record_time": "05/29/2018 14:50:16 GMT+08:00",
    "trace_id": "8b253930-[illegible]3",
    "trace_status": "normal"
}
```


For details about key fields in the CTS trace structure, see the **Trace Structure** in the *Cloud Trace Service User Guide*.

----End

5.10 Common SQL Commands for Storage-Compute Coupled ClickHouse

5.10.1 Data Types

This section describes the data types that are supported by ClickHouse.

ClickHouse does not support the JSON and Object('json') data types.

Supported Data Types

Table 5-39 Supported data types

Type	Keyword	Data Type	Description
Integer	Int8	Int8	Value range: [–128, 127]
	Int16	Int16	Value range: [–32768, 32767]
	Int32	Int32	Value range: [–2147483648, 2147483647]
	Int64	Int64	Value range: [–9223372036854775808, 9223372036854775807]

Type	Keyword	Data Type	Description
Floating point	Float32	Single-precision floating point	Similar to the Float type in the C programming language. A single-precision floating point number occupies four bytes in storage of a computer and is represented in 32-bit binary.
	Float64	Double-precision floating point	Similar to the Double type in the C programming language. A double-precision floating point number occupies eight bytes in storage of a computer and is represented in 64-bit binary.
Decimal	Decimal	Decimal	A signed fixed-point number that can ensure precision during addition, subtraction, and multiplication operations. Decimal values can be in the following formats: ● Decimal(P, S) ● Decimal32(S) ● Decimal64(S) ● Decimal128(S) **NOTE** P stands for precision. The valid range is [1:38]. It determines the number of decimal digits (including fractions) that can be contained. S stands for scale. The valid range is [0:P]. It determines the number of decimal places of a number.
String	String	String	A string can be of a random length. It can contain any set of bytes, including null bytes. Therefore, the String type can replace the VARCHAR, BLOB, and CLOB types in other database management systems.

Type	Keyword	Data Type	Description
	FixedString	Fixed-length string	When the length of the data happens to be N bytes, using the FixedString type is more efficient than other types. In other cases, efficiency may be impaired. The following values can be effectively stored in columns of the FixedString type: ● IP addresses represented in binary (FixedString (16) for IPv6) ● Language codes ● Currency codes ● Binary representation of hash values (FixedString (16) for MD5 and FixedString (32) for SHA256)
Date and time	Date	Date	A Date value takes up two bytes, indicating the date value from 1970-01-01 (unsigned) to the current time. Date values are stored without the time zone.
	DateTime	Timestamp	A Unix timestamp value takes up four bytes (unsigned). Value range of this type is the same as the Date type. The minimum value is 1970-01-01 00:00:00. Timestamp values are accurate to seconds. Leap seconds are not supported. The system time zone will be used when the client or server is started.
	DateTime64	DateTime64	This type allows you to store both the date and time of a specific point in time.
Boolean	Boolean	Boolean	ClickHouse does not support the Boolean type. You can use the UInt8 type for Boolean values. Valid values are 0 and 1.

Type	Keyword	Data Type	Description
Array	Array	Array	An Array value is a collection of elements of the same data type. The elements can be of a random data type, even the Array type itself. However, multi-dimensional arrays are not recommended, because ClickHouse supports multi-dimensional arrays only to a limited extent. For example, you cannot store multi-dimensional arrays in MergeTree tables.
Tuple	Tuple	Tuple	A Tuple value is a collection of elements of different data types. Tuple values cannot be stored in tables, except for memory tables. You can use Tuple values to group temporary columns. In queries, you can use **IN** expressions and **lambda** functions with specific parameters to group temporary columns.
Domains	Domains	Domains	The implementation of the Domains type varies based on different values: ● If the values are IPv4 addresses, the Domains type is binary compatible with the UInt32 type. Compared with the UInt32 type, the Domains type saves the binary storage space and supports more readable input and output formats. ● If the values are IPv6 addresses, the Domains type is binary compatible with the FixedString (16) type. Compared with the FixedString (16) type, the Domains type saves the binary storage space and supports more readable input and output formats.

Type	Keyword	Data Type	Description
Enumeration	Enum8	Enum8	Value range: [–128, 127] An Enum value stores the mapping of 'string'= integer, for example, **Enum8('hello' = 1, 'world' = 2)**.
	Enum16	Enum16	Value range: [–32768, 32767]
Nullable	Nullable	Nullable	Unless otherwise stated in ClickHouse server configurations, the default value of the NULLABLE type is NULL. Nullable values cannot be included in table indexes. Nullable values can be stored together with the normal values of TypeName. For example, columns of the Nullable(Int8) type can store values of the Int8 type, while rows without values store NULL.
Nested	nested	nested	A nested data structure is similar to a table inside a cell. You can specify the parameters of a nested data structure, such as field name and data type, the same way that you specify parameters in a **CREATE TABLE** statement. Each row in a **CREATE TABLE** statement can correspond to a random number of rows in a nested data structure. Example: **Nested (Name1 Type1,Name2 Type2, ...)**

5.10.2 CREATE DATABASE

This section describes the syntax of a **CREATE DATABASE** statement that is used to create a database in ClickHouse.

CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER default_cluster];
```

Table 5-40 Parameter description

Parameter	Description
db_name	Database
IF NOT EXISTS	You can add the **IF NOT EXISTS** keyword phrase to a **CREATE DATABASE** statement. In this case, if the name specified in the statement has been used by an existing database, no database is created and no error is returned.
ON CLUSTER default_cluster	The specified cluster name. The default value is **default_cluster**.

NOTE

You can run the following statement to obtain the cluster name from the **cluster** field:

```
select cluster,shard_num,replica_num,host_name from system.clusters;
```

Usage Example

- Create a database named **demo**.
```
create database demo ON CLUSTER default_cluster;
```
- View the created database.
```
host-172-16-30-9 :) show databases;
SHOW DATABASES
Query id: ced1af23-0286-40cc-9c7a-ccbca41178d8
┌─name───────────────┐
│ INFORMATION_SCHEMA │
│ default            │
│ demo               │
│ information_schema │
│ system             │
└────────────────────┘
5 rows in set. Elapsed: 0.002 sec.
```

5.10.3 CREATE TABLE

This section describes how to create a table.

Creating a Local Table

```
CREATE TABLE [IF NOT EXISTS] [database_name.]table_name [ON CLUSTER default_cluster]
(
name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
name2[type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
...
) ENGINE = engine_name()
[PARTITION BY expr_list]
[ORDER BY expr_list]
```

Table 5-41 Parameters

Parameter	Description
database_name	Database name. The default value is the name of the selected database.

Parameter	Description
table_name	Name of the local table.
ON CLUSTER default_cluster	This parameter specifies that a local table is created on each node. The parameter format is fixed at **ON CLUSTER** *default_cluster*.
name1,name2	Column names.
ENGINE = engine_name()	Table engine type. When creating a table in a dual-replica cluster, you must use the Replicated* engine that supports data replication in the MergeTree series. Otherwise, data is not replicated between replicas, resulting in inconsistent data query results. When using this engine to create a table, set the parameters as follows: ● **ReplicatedMergeTree('/clickhouse/tables/{database}/{table}/{shard}', '{replica}')** (The parameters are fixed and do not need to be modified.) ● **ReplicatedMergeTree()**, which is equivalent to **ReplicatedMergeTree('/clickhouse/tables/{database}/{table}/{shard}', '{replica}')**.
ORDER BY expr_list	Sorting key, which is mandatory. It can be a tuple of a set of columns or any expression.
[PARTITION BY expr_list]	Partition key. The table is usually partitioned by date. You can also use other fields or field expressions.

Example:

- Create a database. For details, see **CREATE DATABASE**.
- Use the database.

```
use demo;
```

- Create a table named **demo.test**.

```
CREATE TABLE demo.test ON CLUSTER default_cluster(`EventDate` DateTime, `id` UInt64)ENGINE =
ReplicatedMergeTree('/clickhouse/tables/{shard}/default/test', '{replica}') PARTITION BY
toYYYYMM(EventDate) ORDER BY id;
```

Creating a Table by Copying the Structure of an Existing Table

You can use the following syntax to create a table by copying the structure of a source table:

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name2 ON CLUSTER default_cluster
 AS [db.]table_name1 [ENGINE = engine_name];
```

Table 5-42 Parameters

Parameter	Description
db	Database name. The default value is the name of the selected database.
table_name1	Name of the source table from which the structure is copied.
table_name2	Name of the table that you want to create.
ON CLUSTER default_cluster	This parameter specifies that a table is created on each node. The parameter format is fixed at **ON CLUSTER** *default_cluster*.
[ENGINE = engine_name]	Table engine type. If you do not specify a table engine when you create a table, the table engine of the source table is used by default.

Example:

- Create a database.
```
create database demo;
```
- Use the database.
```
use demo;
```
- Create a data table.
```
create table demo_t(uid Int32,name String,age UInt32,gender String)engine = TinyLog;
```
- Copy the table structure.
```
create table demo_t2 as demo_t;
```
- **View the table structure**.

Creating a Table by Specifying a SELECT Clause in a CREATE TABLE Statement

You can use a specified table engine to create a table that has the same structure as the query result of the SELECT clause. The query result of the SELECT clause is populated to the table.

```
CREATE TABLE [IF NOT EXISTS] [database_name.]table_name ENGINE = engine_name AS SELECT ...
```

Table 5-43 Parameters

Parameter	Description
database_name	Database name. The default value is the name of the selected database.
table_name	Table created using the SELECT statement.
ENGINE = engine_name()	Table engine type.
SELECT ...	**SELECT** clause.

Example:

- Create a table.
```
CREATE TABLE default.demo1 ON CLUSTER default_cluster( `EventDate` DateTime, `id`
UInt64)ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/default/demo1', '{replica}')
PARTITION BY toYYYYMM(EventDate) ORDER BY id;
```
- Create a table using the SELECT statement.
```
create table t3 ON CLUSTER default_cluster ENGINE =MergeTree() order by EventDate as select *
from default.demo1;
```
- Query the structures of tables **demo1** and **t3**.
```
desc demo1;
```
The query result shows that the structures of the two tables are the same.
```
cloudtable-wlr-click-20230730-06-server-1-1 :) desc demo1;
DESCRIBE TABLE demo1
Query id: 712f6b91-668d-4f70-b160-aac8e52f63a4
┌─name──────┬─type─────┬─default_type─┬─default_expression─┬─comment─┬─codec_expre
ssion─┬─ttl_expression─┐
│ EventDate │ DateTime │              │                    │         │
│ id        │ UInt64   │              │                    │         │
└───────────┴──────────┴──────────────┴────────────────────┴─────────┴──────────
────┴────────────────┘
2 rows in set. Elapsed: 0.001 sec.
cloudtable-wlr-click-20230730-06-server-1-1 :) desc t3;
DESCRIBE TABLE t3
Query id: 11b67532-26f0-49c5-b36d-439d45c279bf
┌─name──────┬─type─────┬─default_type─┬─default_expression─┬─comment─┬─codec_expre
ssion─┬─ttl_expression─┐
│ EventDate │ DateTime │              │                    │         │
│ id        │ UInt64   │              │                    │         │
└───────────┴──────────┴──────────────┴────────────────────┴─────────┴──────────
────┴────────────────┘
2 rows in set. Elapsed: 0.001 sec.
```

5.10.4 DESC|DESCRIBE TABLE

This section describes the basic syntax and usage of the SQL statement for querying the table structure in ClickHouse.

Basic Syntax

```
DESC|DESCRIBE TABLE [database_name.]table [INTO OUTFILE filename] [FORMAT format]
```

Query the structures of tables **demo_t** and **demo_2** by referring to **Creating a Table by Copying the Structure of an Existing Table**.

```
cloudtable-wlr-click-20230730-06-server-1-1 :) desc demo_t;
DESCRIBE TABLE demo_t
Query id: 27a38d90-9459-430f-962e-881817789fc9
┌─name───┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_
expression─┐
│ uid    │ Int32  │              │                    │         │                  │
│ name   │ String │              │                    │         │                  │
│ age    │ UInt32 │              │                    │         │                  │
│ gender │ String │              │                    │         │                  │
└────────┴────────┴──────────────┴────────────────────┴─────────┴──────────────────┴─────
───────────────┘
4 rows in set. Elapsed: 0.001 sec.
cloudtable-wlr-click-20230730-06-server-1-1 :) desc demo_t2;
DESCRIBE TABLE demo_t2
Query id: 60054fe3-794c-410a-be13-cd0b204a9129
┌─name───┬─type───┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_
```

```
expression─┐
│ uid     │ Int32  │           │                │       │             │            │
│ name    │ String │           │                │       │             │            │
│ age     │ UInt32 │           │                │       │             │            │
│ gender  │ String │           │                │       │             │            │
└─────────┴────────┴───────────┴────────────────┴───────┴─────────────┴────────────┴──────────────┴───
──────────────────────┴───────────────────────────┘

4 rows in set. Elapsed: 0.001 sec.
```

5.10.5 CREATE VIEW

This section describes how to create a normal view in ClickHouse.

Creating a View

```
CREATE VIEW [IF NOT EXISTS] [db.]view_name [ON CLUSTER default_cluster] AS SELECT ...
```

Table 5-44 Parameter description

Parameter	Description
db	Database name. The default value is the name of the selected database.
view_name	View name.
[ON CLUSTER default_cluster]	This parameter specifies that a view is created on each node. The parameter format is fixed at **ON CLUSTER** *default_cluster*.
SELECT ...	**SELECT** clause. When you insert data into the source table that is specified in the **SELECT** clause in the view, the inserted data is transformed by the **SELECT** query and the final result is inserted into the view.

Example:

1. Create a source table.
```
create table DB.table1 ON CLUSTER default_cluster (id Int16,name String) ENGINE = MergeTree()
ORDER BY (id);
```
2. Create a view.
```
CREATE VIEW test_view ON CLUSTER default_cluster AS SELECT * FROM DB.table1;
```
3. Insert data to the source table.
```
insert into DB.table1 values(1,'X'),(2,'Y'),(3,'Z');
```
4. Query a view.
```
SELECT * FROM test_view;
```
5. Delete a view.
```
drop table test_view ON CLUSTER default_cluster;
```

NOTE

- If the table creation statement contains **ON CLUSTER** *ClickHouse cluster name*, run the following command to delete the table:
```
drop table Table name ON CLUSTER default_cluster;
```
- If the table creation statement does not contain **ON CLUSTER** *ClickHouse cluster name*, run the following command to delete the table:
```
drop table Table name;
```

5.10.6 CREATE MATERIALIZED VIEW

This section describes how to create a materialized view in ClickHouse.

Creating a Materialized View

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [db.]Materialized_name [TO[db.]name] [ON CLUSTER
default_cluster]
ENGINE = engine_name()
ORDER BY expr
[POPULATE]
AS SELECT ...
```

Table 5-45 Parameter description

Parameter	Description
db	Database name. The default value is the name of the selected database.
Materialized_name	Name of the materialized view.
TO[db.]name	This parameter specifies that the data of the materialized view is inserted into a new table.
[ON CLUSTER default_cluster]	This parameter specifies that a materialized view is created on each node. The parameter format is fixed at **ON CLUSTER** *default_cluster*.
ENGINE = engine_name()	Table engine type.
[POPULATE]	**POPULATE** keyword. If you specify the **POPULATE** keyword when you create a materialized view, the data in the source table that is specified in the **SELECT** clause is inserted into the materialized view when the materialized view is being created. Otherwise, the materialized view contains only the data that is inserted into the source table after the materialized view is created. **NOTE** The **POPULATE** keyword is not recommended because the data that is written to the source table when the materialized view is being created is not inserted into the materialized view.
SELECT ...	**SELECT** clause. When you insert data into the source table that is specified in the **SELECT** clause in the materialized view, the inserted data is transformed by the **SELECT** query and the final result is inserted into the materialized view. **NOTE** A **SELECT** query can contain **DISTINCT**, **GROUP BY**, **ORDER BY**, and **LIMIT**. The corresponding transformations are performed independently on each block of the data that is inserted.

Example:

1. Create a source table.
```
create table DB.table1 ON CLUSTER default_cluster (id Int16,name String) ENGINE = MergeTree()
ORDER BY (id);
```
2. Insert data.
```
insert into DB.table1 values(1,'X'),(2,'Y'),(3,'Z');
```
3. Create a materialized view based on the source table.
```
CREATE MATERIALIZED VIEW demo_view ON CLUSTER default_cluster ENGINE = MergeTree() ORDER
BY (id) AS SELECT * FROM DB.table1;
```
4. Query the materialized view.
```
SELECT * FROM demo_view;
```

NOTE

If the query result is empty, the data that is written to the source table before the materialized view is created cannot be queried if the **POPULATE** keyword is not specified.

5. Insert data to the **DB.table1** table.
```
insert into demo_view values(4,'x'),(5,'y'),(6,'z');
```
6. Query the materialized view.
```
SELECT * FROM demo_view;
```
Query the result.
```
┌─id─┬─name─┐
│  4 │ x    │
│  5 │ y    │
│  6 │ z    │
└────┴──────┘
```

5.10.7 INSERT INTO

This section describes how to insert data.

Basic Syntax

- Insert data in standard format.
```
INSERT INTO [db.]table [(c1, c2, c3)] VALUES (v11, v12, v13), (v21, v22, v23), ...
```

NOTE

For columns that exist in the table structure but not in the list to be inserted, data is filled as follows:

- If the DEFAULT expression is defined, the value is calculated according to the DEFAULT expression.
- If no DEFAULT expression is defined, the value is filled with zero or an empty string.

Insert data by referring to **Creating a Table by Copying the Structure of an Existing Table**.
```
insert into demo_t values(1,'Candy','23','M'),(2,'cici','33','F');
```
- Insert data using the **SELECT** result.
```
INSERT INTO [db.]table [(c1, c2, c3)] SELECT ...
```

NOTE

The columns to insert data are mapped to the columns in the SELECT statement based on their positions. The columns can have different names in the SELECT statement and in the INSERT statement. The system converts the data types of the columns.

Except for the VALUES type, all other data types do not support expressions such as **now()** and **1+2**. The VALUES type allows you to use these expressions to a limited extent. However, you are not advised doing so because executing these expressions is inefficient.

5.10.8 SELECT

This section describes how to execute a **SELECT** statement to query data.

Basic Syntax

```
SELECT [DISTINCT] expr_list
[FROM [database_name.]table | (subquery) | table_function] [FINAL]
[SAMPLE sample_coeff]
[ARRAY JOIN ...]
[GLOBAL] [ANY|ALL|ASOF] [INNER|LEFT|RIGHT|FULL|CROSS] [OUTER|SEMI|ANTI] JOIN (subquery)|table
(ON <expr_list>)|(USING <column_list>)
[PREWHERE expr]
[WHERE expr]
[GROUP BY expr_list] [WITH TOTALS]
[HAVING expr]
[ORDER BY expr_list] [WITH FILL] [FROM expr] [TO expr] [STEP expr]
[LIMIT [offset_value, ]n BY columns]
[LIMIT [n, ]m] [WITH TIES]
[UNION ALL ...]
[INTO OUTFILE filename]
[FORMAT format]
```

Example:

- View the ClickHouse cluster information.
```
select * from system.clusters;
```
- View the macros set for the node.
```
select * from system.macros;
```
- Check the database capacity.
```
select sum(rows) as "Total number of rows", formatReadableSize(sum(data_uncompressed_bytes)) as
"Original size", formatReadableSize(sum(data_compressed_bytes)) as "Compression size",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100, 0) "Compression rate"
from system.parts;
```
- Query the capacity of the **test** table. Add and modify the **WHERE** clause as required.
```
select sum(rows) as "Total number of rows", formatReadableSize(sum(data_uncompressed_bytes)) as
"Original size", formatReadableSize(sum(data_compressed_bytes)) as "Compression size",
round(sum(data_compressed_bytes) / sum(data_uncompressed_bytes) * 100, 0) "Compression rate"
from system.parts where table in ('test') and partition like '2020-11-%' group by table;
```

5.10.9 ALTER TABLE

This section describes the basic syntax and usage of the SQL statement for modifying a table structure in ClickHouse.

Basic Syntax

```
ALTER TABLE [database_name].name [ON CLUSTER default_cluster] ADD|DROP|CLEAR|COMMENT|MODIFY
COLUMN ...
```


NOTE

ALTER supports only *MergeTree, Merge, and Distributed engine tables.

Example:

1. Create a table named **DB_table1**.
```
CREATE TABLE DB_table1 ON CLUSTER default_cluster(Year UInt16,Quarter UInt8,Month UInt8,DayofMonth UInt8,DayOfWeek UInt8,FlightDate Date,FlightNum String,Div5WheelsOff String,Div5TailNum String)ENGINE = MergeTree() PARTITION BY toYYYYMM(FlightDate) PRIMARY KEY (intHash32(FlightDate)) ORDER BY (intHash32(FlightDate),FlightNum) SAMPLE BY intHash32(FlightDate) SETTINGS index_granularity= 8192;
```
2. Add the **test** column to table **DB_table1**.
```
ALTER TABLE DB_table1 ADD COLUMN test String DEFAULT 'defaultvalue';
```
Query the table.
```
desc DB_tables;
```
3. Change the type of the **Year** column in the **DB_table1** table to **UInt8**.
```
ALTER TABLE DB_table1 MODIFY COLUMN Year UInt8;
```
View the table structure.
```
desc DB_tables;
```
4. Delete the **test** column from the **DB_table1** table.
```
ALTER TABLE DB_table1 DROP COLUMN test;
```
Query the table.
```
desc DB_tables;
```
5. Change the name of the **Month** column in the **DB_table1** table to **Month_test**.
```
ALTER TABLE DB_table1 RENAME COLUMN Month to Month_test;
```
Query the table.
```
desc DB_tables;
```

5.10.10 DROP

This section describes the basic syntax and usage of the SQL statement for deleting a ClickHouse table.

Basic Syntax

```
DROP [TEMPORARY] TABLE [IF EXISTS] [database_name.]name [ON CLUSTER default_cluster] [SYNC]
```

Example:

Delete the **t1** table.
```
drop table t1 SYNC;
```

NOTE

- When you delete a replication table, create a path on ZooKeeper to store related data. The default library engine of ClickHouse is the atomic database engine. After a table in the atomic database is deleted, it is not deleted immediately but deleted 24 hours later. To resolve this issue, when deleting a table, add the **SYNC** field to the deletion command, for example, **drop table** *t1* **SYNC;**
- This issue does not occur when a local or distributed table is deleted. The **SYNC** field is not required in your deletion command, for example, **drop table** *t1***;**
- If the table creation statement contains **ON CLUSTER** *ClickHouse cluster name*, run the following command to delete the table:
  ```
  drop table Table name ON CLUSTER default_cluster;
  ```
- If the table creation statement does not contain **ON CLUSTER** *ClickHouse cluster name*, run the following command to delete the table:
  ```
  drop table Table name;
  ```
- Before deleting a data table, check whether the data table is in use to avoid unnecessary troubles. After a data table is deleted, it can be restored within 24 hours. The restoration command is as follows:
  ```
  set allow_experimental_undrop_table_query = 1;
  UNDROP TABLE Data table name;
  ```

5.10.11 SHOW

This section describes the basic syntax and usage of the SQL statement for displaying information about databases and tables in ClickHouse.

Basic Syntax

```
show databases;
show tables;
```

Examples:

- Query the database.
  ```
  show databases;
  ```
- Query the table information.
  ```
  show tables;
  ```

