

Data Lake Insight

Getting Started

Issue 01
Date 2024-08-16



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

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

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Security Declaration

Vulnerability

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

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

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

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

Contents

1 Using DLI to Submit a SQL Job to Query OBS Data.....	1
2 Using DLI to Submit a SQL Job to Query RDS for MySQL Data.....	7
3 Using DLI to Submit a Flink OpenSource SQL Job to Query RDS for MySQL Data	19
4 Using DLI to Submit a Flink Jar Job.....	32
5 Using DLI to Submit a Spark Jar Job.....	43
6 Practices.....	50

1 Using DLI to Submit a SQL Job to Query OBS Data

Scenario

DLI can query data stored in OBS. This section describes how to use DLI to submit a SQL job to query OBS data.

To illustrate, create a new file called **sampledata.csv** and upload it to an OBS bucket. Then, create an elastic resource pool, create queues within it, and use DLI to create a database and table. Finally, use DLI's SQL editor to query 1,000 data records from the table.

Procedure

Table 1-1 shows the process for submitting a SQL job to query OBS data.

Complete the preparations in **Preparations** before performing the following operations.

Table 1-1 Procedure for using DLI to submit a SQL job to query OBS data

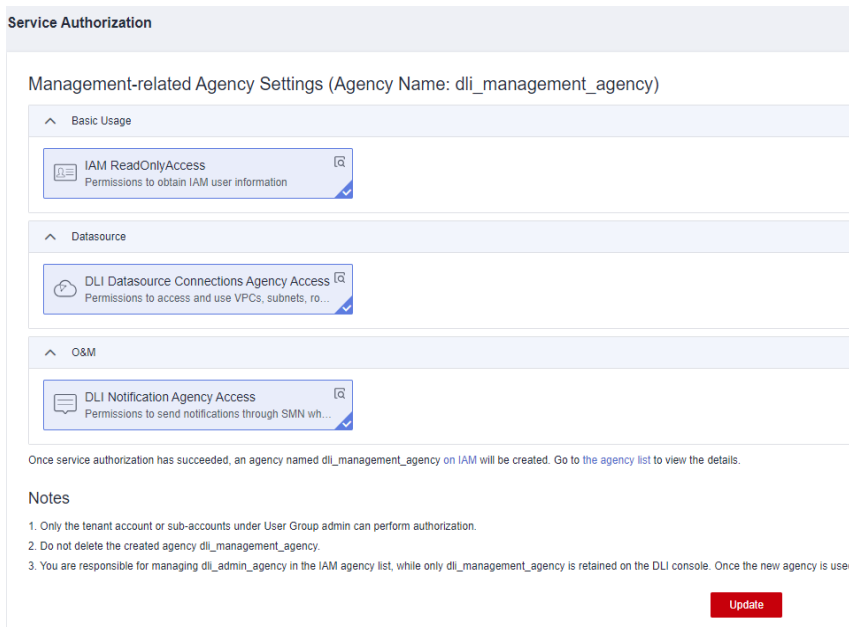
Procedure	Description
Step 1: Upload Data to OBS	Upload data files to OBS.
Step 2: Create an Elastic Resource Pool and Add Queues to the Pool	Create compute resources required for submitting jobs.
Step 3: Create a Database	DLI metadata forms the foundation for SQL job development. Before executing a job, you need to define databases and tables based on your business scenario.
Step 4: Create a Table	Use the sample data stored in OBS to create tables in the db1 database.

Procedure	Description
Step 5: Query Data	Use standard SQL statements to query and analyze data.

Preparations

- Register a Huawei ID and enable Huawei Cloud services. Make sure your account is not in arrears or frozen.
- Configure an agency for DLI.
To use DLI, you need to access services such as Object Storage Service (OBS), Virtual Private Cloud (VPC), and Simple Message Notification (SMN). If it is your first time using DLI, you will need to configure an agency to allow access to these dependent services.
 - a. Log in to the DLI management console using your account. In the navigation pane on the left, choose **Global Configuration > Service Authorization**.
 - b. On the agency settings page, select the agency permissions under **Basic Usage, Datasource, and O&M** and click **Update**.
 - c. Check and understand the notes for updating the agency, and click **OK**. The DLI agency permissions are updated.

Figure 1-1 Configuring an agency for DLI



- d. Once configured, you can check the agency **dli_management_agency** in the agency list on the IAM console.

Step 1: Upload Data to OBS

Upload data files to OBS.

1. Log in to the OBS management console.
2. Create a bucket. In this example, the bucket name is **obs1**.
 - a. Click **Create Bucket** in the upper right corner.
 - b. On the displayed **Create Bucket** page, specify **Region** and enter the **Bucket Name**. Retain the default values for other parameters or adjust them as needed.

 **NOTE**

Select a region that matches the location of the DLI console.

- c. Click **Create Now**.
3. Click **obs1** to access its **Objects** tab page.
 4. Click **Upload Object**. In the displayed dialog box, drag a desired file or folder, for example, **sampledata.csv** to the **Upload Object** area. Then, click **Upload**.
You can create a **sampledata.txt** file, copy the following content separated by commas (,), and save the file as **sampledata.csv**.

```
product_id,product_name
113,office_13
22,book_2
29,book_9
```

After the file is uploaded successfully, the file path is **obs://obs1/sampledata.csv**.

For more operations on the OBS console, see the *Object Storage Service User Guide*.

Step 2: Create an Elastic Resource Pool and Add Queues to the Pool

In this example, the elastic resource pool **dli_resource_pool** and queue **dli_queue_01** are created.

1. Log in to the DLI management console.
2. In the navigation pane on the left, choose **Resources > Resource Pool**.
3. On the displayed page, click **Buy Resource Pool** in the upper right corner.
4. On the displayed page, set the parameters.
5. In this example, we will buy the resource pool in the **CN East-Shanghai2** region. [Table 1-2](#) describes the parameters.

Table 1-2 Parameters

Parameter	Description	Example Value
Region	Select a region where you want to buy the elastic resource pool.	CN East-Shanghai2
Project	Project uniquely preset by the system for each region	Default
Name	Name of the elastic resource pool	dli_resource_pool

Parameter	Description	Example Value
Specifications	Specifications of the elastic resource pool	Standard
CU Range	The maximum and minimum CUs allowed for the elastic resource pool	64-64
CIDR Block	CIDR block the elastic resource pool belongs to. If you use an enhanced datasource connection, this CIDR block cannot overlap that of the data source. Once set, this CIDR block cannot be changed.	172.16.0.0/19
Enterprise Project	Select an enterprise project for the elastic resource pool.	default

6. Click **Buy**.
7. Click **Submit**.
8. In the elastic resource pool list, locate the pool you just created and click **Add Queue** in the **Operation** column.
9. Set the basic parameters listed below.

Table 1-3 Basic parameters for adding a queue

Parameter	Description	Example Value
Name	Name of the queue to add	dli_queue_01
Type	Type of the queue <ul style="list-style-type: none"> • To execute SQL jobs, select For SQL. • To execute Flink or Spark jobs, select For general purpose. 	–
Engine	SQL queue engine. The options include Spark and Trino .	–
Enterprise Project	Select an enterprise project.	default

10. Click **Next** and configure scaling policies for the queue.
Click **Create** to add a scaling policy with varying priority, period, minimum CUs, and maximum CUs.

Figure 1-2 shows the scaling policy configured in this example.

Figure 1-2 Configuring a scaling policy when adding a queue

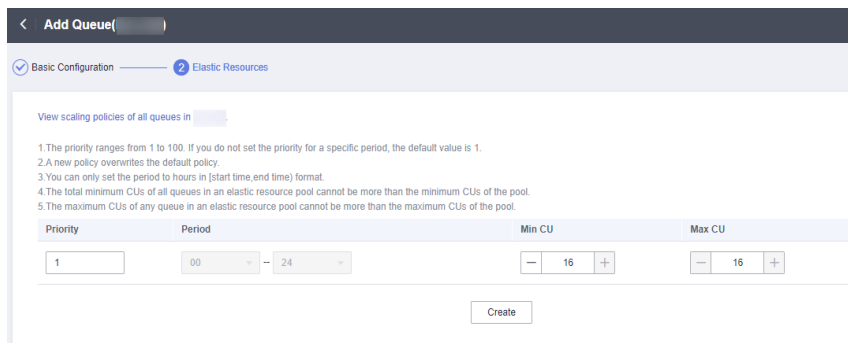


Table 1-4 Scaling policy parameters

Parameter	Description	Example Value
Priority	Priority of the scaling policy in the current elastic resource pool. A larger value indicates a higher priority. In this example, only one scaling policy is configured, so its priority is set to 1 by default.	1
Period	The first scaling policy is the default policy, and its Period parameter configuration cannot be deleted or modified. The period for the scaling policy is from 00 to 24.	00-24
Min CU	Minimum number of CUs allowed by the scaling policy	16
Max CU	Maximum number of CUs allowed by the scaling policy	64

11. Click **OK**.

Step 3: Create a Database


Before querying data, create a database, for example, **db1**.

 **NOTE**

The **default** database is a built-in database. You cannot create the **default** database.

1. In the left navigation pane of the DLI management console, choose **SQL Editor**.
2. In the editing window on the right of the **SQL Editor** page, enter the following SQL statement and click **Execute**. Read and agree to the privacy agreement, and click **OK**.

```
create database db1;
```

After the database is successfully created, click  in the middle pane to refresh the database list. The new database **db1** is displayed in the list.

 **NOTE**

When you execute a query on the DLI management console for the first time, you need to read the privacy agreement. You can perform operations only after you agree to the agreement. For later queries, you will not need to read the privacy agreement again.

Step 4: Create a Table

After database **db1** is created, create a table (for example, **table1**) containing data in the sample file **obs://obs1/sampledata.csv** stored on OBS in **db1**.

1. In the SQL editing window of the **SQL Editor** page, select the **default** queue and database **db1**.

2. Enter the following SQL statement in the job editor window and click

Execute:

```
create table table1 (product_id int, product_name string) using csv options (path 'obs://obs1');
```

When creating a table, you only need to specify the OBS storage path where the data file is located, without specifying the file name at the end of the directory.

After the table is successfully created, click the **Databases** tab then **db1**. The created table **table1** is displayed in the table list.

Step 5: Query Data

After performing the preceding steps, you can start querying data.

1. In the **Table** tab on the **SQL Editor** page, double-click the created table **table1**. The SQL statement is automatically displayed in the SQL job editing window in the right pane. Run following statement to query 1,000 records in the **table1** table:

```
select * from db1.table1 limit 1000;
```

2. Click **Execute**. The system starts the query.

After the SQL statement is successfully executed or fails to be executed, you can view the query result on the **View Result** tab under the SQL job editing window.

2 Using DLI to Submit a SQL Job to Query RDS for MySQL Data

Scenario

DLI can query data stored in RDS. This section describes how to use DLI to submit a SQL job to query RDS for MySQL data.

In this example, we will create an RDS for MySQL DB instance, create a database and table, create a DLI elastic resource pool and add a queue to it, create an enhanced datasource connection to connect the DLI elastic resource pool and the RDS for MySQL DB instance, and submit a SQL job to access the data in the RDS table.

Procedure

Table 2-1 shows the process for submitting a SQL job to query RDS for MySQL data.

Complete the preparations in **Preparations** before performing the following operations.

Table 2-1 Procedure for using DLI to submit a SQL job to query RDS for MySQL data

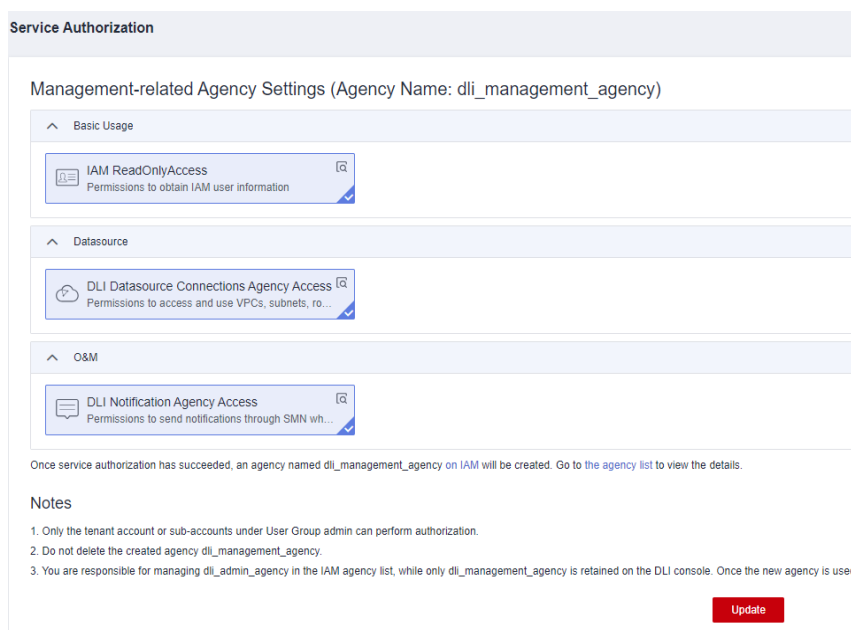
Procedure	Description
Step 1: Create an RDS MySQL Instance	Create an RDS for MySQL instance for the example scenario.
Step 2: Create an RDS Database Table	Log in to the RDS for MySQL DB instance and create a database and a table.
Step 3: Create an Elastic Resource Pool and Add Queues to the Pool	Create compute resources required for submitting jobs.

Procedure	Description
Step 4: Create an Enhanced Datasource Connection	Create an enhanced datasource connection to connect the DLI elastic resource pool and the RDS for MySQL DB instance.
Step 5: Create a Datasource Authentication	Create a datasource authentication to store the access credentials required by DLI to read from and write to RDS for MySQL data.
Step 6: Submit a SQL Job	Use standard SQL statements to query and analyze data.

Preparations

- Register a Huawei ID and enable Huawei Cloud services. Make sure your account is not in arrears or frozen.
- Configure an agency for DLI.
 To use DLI, you need to access services such as Object Storage Service (OBS), Virtual Private Cloud (VPC), and Simple Message Notification (SMN). If it is your first time using DLI, you will need to configure an agency to allow access to these dependent services.
 - a. Log in to the DLI management console using your account. In the navigation pane on the left, choose **Global Configuration > Service Authorization**.
 - b. On the agency settings page, select the agency permissions under **Basic Usage, Datasource, and O&M** and click **Update**.
 - c. Check and understand the notes for updating the agency, and click **OK**. The DLI agency permissions are updated.

Figure 2-1 Configuring an agency for DLI



- d. Once configured, you can check the agency **dli_management_agency** in the agency list on the IAM console.

Step 1: Create an RDS MySQL Instance

The sample job name is **JobSample**, and RDS is used as the data source. For details about how to create an RDS MySQL instance, see Getting Started with RDS for MySQL.

1. Log in to the RDS management console.
2. Select a region and a project in the upper left corner.
3. In the navigation pane on the left, choose **Instances**. On the displayed page, click **Buy DB Instance** in the upper right corner.
4. On the **Buy DB Instance** page, select a billing mode, set instance parameters, and click **Buy**.

Set the parameters listed below based on your service planning.

Table 2-2 RDS for MySQL instance parameters

Parameter	Description	Example Value
Billing Mode	Billing mode of the RDS for MySQL DB instance	Pay-per-use
Region	Region where you want to create the instance	CN East-Shanghai2
DB Instance Name	Instance name	rds-demo
DB Engine	MySQL	MySQL
DB Engine Version	If you select MySQL for DB Engine , select an engine version that best suits your service needs. You are advised to select the latest available version for more stable performance, higher security, and greater reliability.	8.0
DB Instance Type	Primary/standby mode of the instance	Single
Storage Type	Determines the instance read/write speed. The higher the maximum throughput, the faster the read and write speeds.	Cloud SSD
AZ	For a single DB instance, you only need to select a single AZ.	-

Parameter	Description	Example Value
Time Zone	Select a time zone based on the region you selected. You can change it after the DB instance is created.	Retain the default value.
Instance Class	vCPUs and memory. These instance classes support varying number of connections and maximum IOPS.	2 vCPUs 4 GB
Storage Space	If the storage type is cloud SSD or extreme SSD, you can enable storage autoscaling. If the available storage drops to a specified threshold, autoscaling is triggered.	40 GB
Disk Encryption	Determine whether to enable disk encryption.	Disable
VPC	Select an existing VPC. For how to recreate a VPC and subnet, refer to . NOTE In datasource scenarios, the CIDR block of the data source cannot overlap that of the elastic resource pool.	-
Database Port	Port 3306 is used by default.	3306
Security Group	Enhances security by providing rules that control access to RDS from other services. The security group where the data source is must allow access from the CIDR block of the DLI elastic resource pool.	-
Password	Set a password for logging in to the DB instance.	-
Administrator	root	root
Administrator Password	Administrator password	-

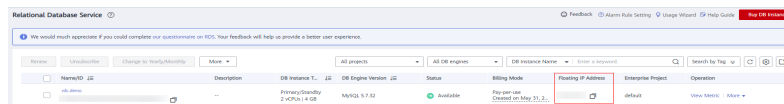
Parameter	Description	Example Value
Parameter Template	A template of parameters for creating an instance. The template contains engine configuration values that are applied to one or more instances.	Default-MySQL-5.7
Table Name	Determines whether the table name is case-insensitive.	Case insensitive
Enterprise Project	If the instance has been associated with an enterprise project, select the target project from the Enterprise Project drop-down list.	default
Quantity	Number of instances to buy	1

5. Click **Next**. The confirmation page is displayed.
6. Click **Submit**.
7. To view and manage the DB instance, go to the **Instance Management** page. During the creation process, the DB instance status is **Creating**. When the creation process is complete, the instance status will change to **Available**. You can view the detailed progress and result of the task on the **Task Center** page.

Step 2: Create an RDS Database Table

1. Log in to the RDS management console.
2. In the upper left corner of the management console, select the target region and project.
3. On the **Instances** page, locate the DB instance you just created and record its floating IP address.

Figure 2-2 Checking the floating IP address



4. Locate the RDS for MySQL DB instance you created, click **More** in the **Operation** column, and select **Log In**. On the displayed page, enter the username and password for logging in to the instance and click **Test Connection**. After **Connection is successful** is displayed, click **Log In**.

Figure 2-3 RDS console

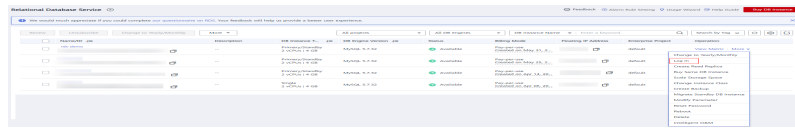


Figure 2-4 Logging in to an instance

Instance Login Information

DB Instance Name: **rds-demo** DB Engine Version: []

* Login Username:

* Password: Connection is successful.

Remember Password Select to remember your password in an encrypted form. Otherwise, the metadata collection function cannot be enabled.

Description:

Collect Metadata Periodically ? If not enabled, DAS can query the real-time structure information only from databases, which may affect the real-time performance of databases.

Show Executed SQL Statements ? If not enabled, the executed SQL statements cannot be viewed, and you need to input each SQL statement manually.

5. Click **Create Database**. In the displayed dialog box, enter database name **dli_demo**. Then, click **OK**.

6. Click **SQL Query** and run the following SQL statement to create a table:

```
CREATE TABLE `dli_demo`.`tabletest` (
  `id` VARCHAR(32) NOT NULL,
  `name` VARCHAR(32) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;
```

Step 3: Create an Elastic Resource Pool and Add Queues to the Pool

To execute SQL jobs in datasource scenarios, you must use your own SQL queue as the existing **default** queue cannot be used. In this example, create an elastic resource pool named **dli_resource_pool** and a queue named **dli_queue_01**.

1. Log in to the DLI management console.
2. In the navigation pane on the left, choose **Resources > Resource Pool**.
3. On the displayed page, click **Buy Resource Pool** in the upper right corner.
4. On the displayed page, set the parameters.
5. In this example, we will buy the resource pool in the **CN East-Shanghai2** region. **Table 2-3** describes the parameters.

Table 2-3 Parameters

Parameter	Description	Example Value
Region	Select a region where you want to buy the elastic resource pool.	CN East-Shanghai2
Project	Project uniquely preset by the system for each region	Default
Name	Name of the elastic resource pool	dli_resource_pool
Specifications	Specifications of the elastic resource pool	Standard
CU Range	The maximum and minimum CUs allowed for the elastic resource pool	64-64
CIDR Block	CIDR block the elastic resource pool belongs to. If you use an enhanced datasource connection, this CIDR block cannot overlap that of the data source. Once set, this CIDR block cannot be changed.	172.16.0.0/19
Enterprise Project	Select an enterprise project for the elastic resource pool.	default

6. Click **Buy**.
7. Click **Submit**.
8. In the elastic resource pool list, locate the pool you just created and click **Add Queue** in the **Operation** column.
9. Set the basic parameters listed below.

Table 2-4 Basic parameters for adding a queue

Parameter	Description	Example Value
Name	Name of the queue to add	dli_queue_01
Type	Type of the queue <ul style="list-style-type: none">• To execute SQL jobs, select For SQL.• To execute Flink or Spark jobs, select For general purpose.	–
Engine	SQL queue engine. The options include Spark and Trino .	–
Enterprise Project	Select an enterprise project.	default

10. Click **Next** and configure scaling policies for the queue.

Click **Create** to add a scaling policy with varying priority, period, minimum CUs, and maximum CUs.

Figure 2-5 shows the scaling policy configured in this example.

Figure 2-5 Configuring a scaling policy when adding a queue

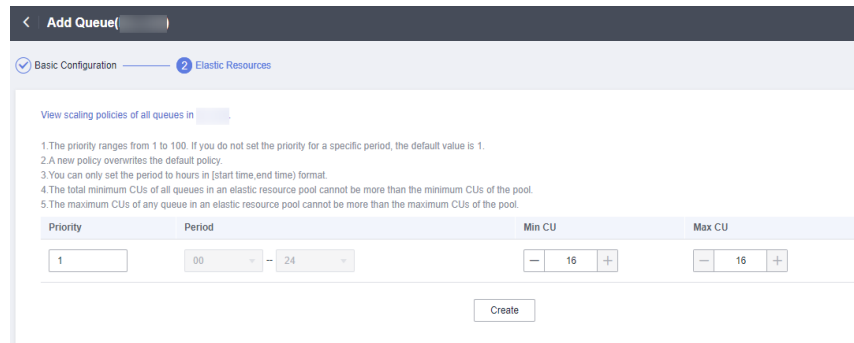


Table 2-5 Scaling policy parameters

Parameter	Description	Example Value
Priority	Priority of the scaling policy in the current elastic resource pool. A larger value indicates a higher priority. In this example, only one scaling policy is configured, so its priority is set to 1 by default.	1
Period	The first scaling policy is the default policy, and its Period parameter configuration cannot be deleted or modified. The period for the scaling policy is from 00 to 24.	00-24
Min CU	Minimum number of CUs allowed by the scaling policy	16
Max CU	Maximum number of CUs allowed by the scaling policy	64

11. Click **OK**.

Step 4: Create an Enhanced Datasource Connection

Step 1 Create a rule on the security group of the RDS DB instance to allow access from the CIDR block of the DLI queue.

1. Go to the RDS management console and choose **Instances** in the navigation pane on the left. In the instance list, click the name of the RDS for MySQL DB instance you created to access its basic information page.

- In the navigation pane on the left, choose **Connectivity & Security**. In the **Security Group Rules** area, find the **Inbound Rules** tab and click **Add Inbound Rule**.

For example, if the CIDR block of the queue is **172.16.0.0/19**, add the rule as follows:

- Set **Priority** to **1** and **Action** to **Allow**.
- Type**: Select **IPv4**.
- Protocol & Port**: Select **Protocols/TCP (Custom)** as the protocol and leave the port number blank.
- Source**: Select **IP Address** and enter **172.16.0.0/19**.

Click **OK**. The security group rule is added.

Step 2 Create an enhanced datasource connection between RDS for MySQL and DLI.

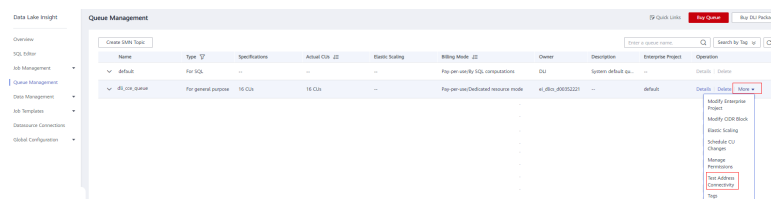
NOTE

The CIDR block of the elastic resource pool bound to a datasource connection cannot overlap that of the data source.

- Go to the DLI management console and choose **Datasource Connections** in the navigation pane on the left.
- On the displayed **Enhanced** tab, click **Create**. Set the following parameters:
 - Connection Name**: **dlirds**
 - Resource Pool**: Select the elastic resource pool created in [Step 3: Create an Elastic Resource Pool and Add Queues to the Pool](#).
 - VPC**: Select the VPC where the RDS for MySQL DB instance is, that is, the VPC selected in [Step 2: Create an RDS Database Table](#).
 - Subnet**: Select the subnet where the RDS for MySQL DB instance is, that is, the subnet selected in [Step 2: Create an RDS Database Table](#).

To check the subnet information, go to the RDS console, choose **Instances** in the navigation pane on the left, click the name of the RDS for MySQL DB instance you created, and find **Subnet** in the **Connection Information** area on the displayed page.
- Click **OK**.
- In the **Enhanced** tab, click the created connection **dlirds** to view its **VPC Peering ID** and **Connection Status**. If the connection status is **Active**, the connection is successful.
- Test if the queue can connect to the RDS for MySQL DB instance.
 - In the navigation pane on the left, choose **Resources > Queue Management**. On the displayed page, locate the queue added in [Step 3: Create an Elastic Resource Pool and Add Queues to the Pool](#), click **More** in the **Operation** column, and select **Test Address Connectivity**.

Figure 2-6 Testing address connectivity



- b. Enter the floating IP address of the RDS for MySQL DB instance recorded in [Step 2: Create an RDS Database Table](#).

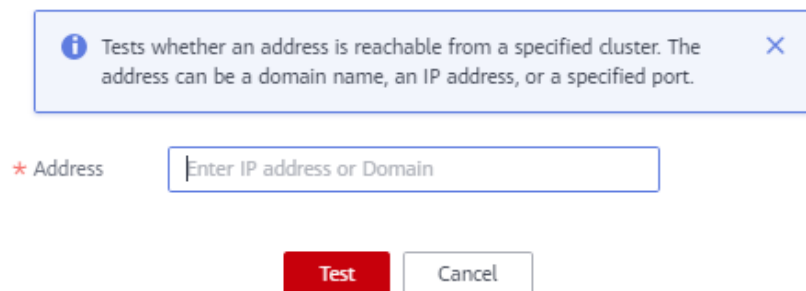
 NOTE

On the **Instance Management** page, click the target DB instance. On the displayed page, choose **Connection Information** > **Floating IP Address** to obtain the floating IP address.

- If the IP address is reachable, the DLI queue can connect to the RDS for MySQL DB instance through the created enhanced datasource connection.

Figure 2-7 Testing address connectivity

Test Address Connectivity



- If the IP address is unreachable, rectify the fault by referring to [Why Is a Datasource Connection Successfully Created But the Network Connectivity Test Fails?](#)

Once the fault is rectified, retest the network connectivity.

----End

Step 5: Create a Datasource Authentication

When analyzing across multiple sources, you are not advised to configure authentication information directly in a job as it can lead to password leakage. Instead, you are advised to use datasource authentication provided by DLI to securely store data source authentication information.

To connect a Spark SQL job to an RDS for MySQL data source, you can create a password-type datasource authentication.

1. Log in to the DLI management console.
2. In the navigation pane on the left, choose **Datasource Connections**. On the displayed page, click **Datasource Authentication**.
3. Click **Create**.

Set authentication parameters based on [Table 2-6](#).

Table 2-6 Datasource authentication parameters

Parameter	Description
Type	Select Password .
Authentication Certificate	Name of the datasource authentication to create <ul style="list-style-type: none">Only numbers, letters, and underscores (_) are allowed. The name cannot contain only numbers or start with an underscore (_).The value can contain a maximum of 128 characters.
Username	Username for logging in to the RDS for MySQL DB instance
Password	Password for logging in to the RDS for MySQL DB instance

Figure 2-8 Creating a Password-type datasource authentication

Create Authentication ×

Type

* Authentication Certificate

Username ?

* Password ?

Step 6: Submit a SQL Job

In this example, a SQL job accesses an RDS table using a datasource connection.

1. On the DLI management console, choose **SQL Editor** in the navigation pane on the left.
2. In the editing window on the right of the **SQL Editor** page, enter the following SQL statement to create database **db1** and click **Execute**.

```
create database db1;
```
3. On the top of the editing window, choose the **dli_queue_01** queue and the **db1** database. Enter the following SQL statements to create a table, insert data to the table, and query the data. Click **Execute**.

View the query result to verify that the query is successful and the datasource connection works.

```
CREATE TABLE IF NOT EXISTS rds_test USING JDBC OPTIONS (  
  'url' = 'jdbc:mysql://{ip}:{port}', // Private IP address and port of RDS  
  'driver' = 'com.mysql.jdbc.Driver',  
  'dbtable' = 'dli_demo.tabletest', // Name of the created DB instance and table name  
  'passwdauth'="xxxxx" // Name of the datasource authentication of the password type created on  
  DLI. If datasource authentication is used, you do not need to set the username and password for the  
  job.  
)  
  
insert into rds_test VALUES ('123','abc');  
  
SELECT * from rds_test;
```

3 Using DLI to Submit a Flink OpenSource SQL Job to Query RDS for MySQL Data

Scenario

DLI Flink jobs can use other cloud services as data sources and sink streams for real-time compute.

This example describes how to create and submit a Flink OpenSource SQL job that uses Kafka as the source stream and RDS as the sink stream.

Procedure

You need to create a Flink OpenSource SQL job that has an source stream and an sink stream. The source stream reads data from Kafka, and the sink stream writes data to RDS for MySQL. [Procedure](#) shows the process.

Complete the preparations in [Preparations](#) before performing the following operations.

Table 3-1 Procedure for using DLI to submit a Flink OpenSource SQL job to query RDS for MySQL data

Procedure	Description
Step 1: Prepare a Source Stream	In this example, a Kafka instance is created as the data source.
Step 2: Prepare a Sink Stream	In this example, an RDS for MySQL DB instance is created as the data destination.
Step 3: Create an OBS Bucket to Store Output Data	Create an OBS bucket to store checkpoints, job logs, and debugging test data for the Flink OpenSource SQL job.
Step 4: Create an Elastic Resource Pool and Add Queues to the Pool	Create compute resources required for submitting the Flink OpenSource SQL job.

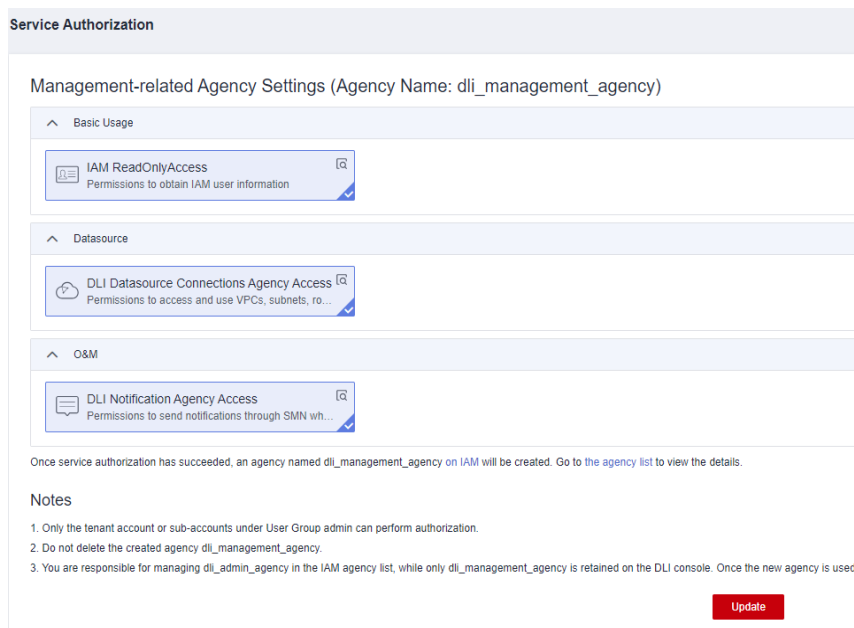
Procedure	Description
Step 5: Create an Enhanced Datasource Connection Between DLI and Kafka	Create an enhanced datasource connection to connect the DLI elastic resource pool and the Kafka instance.
Step 6: Create an Enhanced Datasource Connection Between DLI and RDS	Create an enhanced datasource connection to connect the DLI elastic resource pool and the RDS for MySQL DB instance.
Step 7: Use DEW to Manage Access Credentials and Create an Agency That Allows DLI to Access DEW	In cross-source analysis scenarios, use DEW to manage access credentials of data sources and create an agency that allows DLI to access DEW.
Step 8: Create a Flink OpenSource SQL Job	Once you have prepared a source stream and a sink stream, you can create a Flink OpenSource SQL job to analyze the data.

Preparations

- Register a Huawei ID and enable Huawei Cloud services. Make sure your account is not in arrears or frozen.
- Configure an agency for DLI.

To use DLI, you need to access services such as Object Storage Service (OBS), Virtual Private Cloud (VPC), and Simple Message Notification (SMN). If it is your first time using DLI, you will need to configure an agency to allow access to these dependent services.

 - a. Log in to the DLI management console using your account. In the navigation pane on the left, choose **Global Configuration > Service Authorization**.
 - b. On the agency settings page, select the agency permissions under **Basic Usage, Datasource, and O&M** and click **Update**.
 - c. Check and understand the notes for updating the agency, and click **OK**. The DLI agency permissions are updated.

Figure 3-1 Configuring an agency for DLI

- d. Once configured, you can check the agency **dli_management_agency** in the agency list on the IAM console.

Step 1: Prepare a Source Stream

In this example, Kafka is the source stream.

For more information about Flink OpenSource SQL job data, refer to [Preparing Flink Job Data](#).

Enable DIS to import Kafka data to DLI. For details, see "Buying a Kafka Instance" in the *Distributed Message Service Kafka User Guide*.

1. Create the dependent Kafka resources.

Before creating a Kafka instance, ensure the availability of resources, including a virtual private cloud (VPC), subnet, security group, and security group rules.

- For details about how to create a VPC and subnet, see "Creating a VPC and Subnet" in *Virtual Private Cloud User Guide*. For details about how to create and use a subnet in an existing VPC, see "Create a Subnet for the VPC" in *Virtual Private Cloud User Guide*.

NOTE

- The created VPC and the Kafka instance you will create must be in the same region.
- Retain the default settings unless otherwise specified.
- For details about how to create a security group, see "Creating a Security Group" in the *Virtual Private Cloud User Guide*. For details about how to add rules to a security group, see "Creating a Subnet for the VPC" in the *Virtual Private Cloud User Guide*.

For more information, see in *Distributed Message Service for Kafka User Guide*.

2. Create a Kafka premium instance as the job source stream.
 - a. Log in to the DMS for Kafka management console.
 - b. Select a region in the upper left corner.
 - c. On the **DMS for Kafka** page, click **Buy Instance** in the upper right corner and set related parameters. The required instance information is as follows:
 - **Region:** Select the region where DLI is located.
 - **Project:** Keep the default value.
 - **AZ:** Keep the default value.
 - **Instance Name:** **kafka-dliflink**
 - **Specifications:** **Default**
 - **Enterprise Project:** **default**
 - **Version:** Keep the default value.
 - **CPU Architecture:** Keep the default value.
 - **Broker Flavor:** Select a flavor as needed.
 - **Brokers:** Retain the default value.
 - **Storage Space:** Keep the default value.
 - **Capacity Threshold Policy:** Keep the default value.
 - **VPC and Subnet:** Select the VPC and subnet created in **1**.
 - **Security Group:** Select the security group created in **1**.
 - **Manager Username:** Enter **dliflink** (used to log in to the instance management page).
 - **Password:** **** (The system cannot detect your password.)
 - **Confirm Password:** ****
 - **More Settings:** Do not configure this parameter.
 - d. Click **Buy**. The confirmation page is displayed.
 - e. Confirm that the instance information is correct, read and agree to the **HUAWEI CLOUD Customer Agreement**, and click **Submit**. It takes about 10 to 15 minutes to create an instance.
3. Create a Kafka topic.
 - a. Click the name of the created Kafka instance. The basic information page of the instance is displayed.
 - b. Choose **Topics** in the navigation pane on the left. On the displayed page, click **Create Topic**. Configure the following parameters:

- **Topic Name:** For this example, enter **testkafkatopic**.
- **Partitions:** Set the value to **1**.
- **Replicas:** Set the value to **1**.

Retain the default values for other parameters.

Step 2: Prepare a Sink Stream

To use RDS as the sink stream, create an RDS MySQL instance. For details, see "Getting Started with RDS for MySQL" in [Getting Started with Relational Database Service](#) *Getting Started with Relational Database Service*.

1. Log in to the RDS management console.
2. Select a region in the upper left corner.
3. Click **Buy DB Instance** in the upper right corner of the page and set related parameters. Retain the default values for other parameters.
 - **Billing Mode:** Select **Pay-per-use**.
 - **Region:** Select the region where DLI is located.
 - **DB Instance Name:** Enter **rds-dliflink**.
 - **DB Engine:** Select **MySQL**.
 - **DB Engine Version:** Select **8.0**.
 - **DB Instance Type:** Select **Primary/Standby**.
 - **Storage Type:** Cloud SSD may be selected by default.
 - **Primary AZ:** Select a custom AZ.
 - **Standby AZ:** Select a custom AZ.
 - **Time Zone:** Select your current time zone.
 - **Instance Class:** Select a class as needed and choose **2 vCPUs | 8 GB**.
 - **Storage Space (GB):** Set it to **40**.
 - **VPC:** Select the VPC and subnet created in **1**.
 - **Database Port:** Enter **3306**.
 - **Security Group:** Select the security group created in **1**.
 - **Administrator Password:** **** (Keep the password secure. The system cannot retrieve your password.)
 - **Confirm Password:** ****
 - **Parameter Template:** Choose **Default-MySQL-8.0**.
4. Click **Next** and confirm the specifications.
5. Click **Submit**. The RDS DB instance is created.
6. Log in to the MySQL database and create table **orders** in database **flink**.

Log in to the MySQL instance, click the **flink** database. On the displayed page, click **SQL Window**. Enter the following table creation statement in the SQL editing pane to create a table.

```
CREATE TABLE `flink`.`orders` (  
  `order_id` VARCHAR(32) NOT NULL,  
  `order_channel` VARCHAR(32) NULL,  
  `order_time` VARCHAR(32) NULL,  
  `pay_amount` DOUBLE UNSIGNED NOT NULL,
```

```
`real_pay` DOUBLE UNSIGNED NULL,  
`pay_time` VARCHAR(32) NULL,  
`user_id` VARCHAR(32) NULL,  
`user_name` VARCHAR(32) NULL,  
`area_id` VARCHAR(32) NULL,  
PRIMARY KEY (`order_id`)  
) ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_general_ci;
```

Step 3: Create an OBS Bucket to Store Output Data

In this example, you need to enable OBS for job **JobSample** to provide DLI Flink jobs with the functions of checkpointing, saving job logs, and commissioning test data.

For details about how to create a bucket, see "Creating a Bucket" in the *Object Storage Service Console Operation Guide*.

1. In the navigation pane on the OBS management console, choose **Object Storage**.
2. In the upper right corner of the page, click **Create Bucket** and set bucket parameters.
 - **Region**: Select the region where DLI is located.
 - **Bucket Name**: Enter a bucket name. For this example, enter **obstest**.
 - **Default Storage Class**: **Standard**
 - **Bucket Policy**: **Private**
 - **Default Encryption**: **Do not enable**
 - **Direct Reading**: **Do not enable**
 - **Enterprise Project**: **default**
3. Click **Create Now**.

Step 4: Create an Elastic Resource Pool and Add Queues to the Pool

To create a Flink OpenSource SQL job, you must use your own queue as the existing **default** queue cannot be used. In this example, create an elastic resource pool named **dli_resource_pool** and a queue named **dli_queue_01**.

1. Log in to the DLI management console.
2. In the navigation pane on the left, choose **Resources > Resource Pool**.
3. On the displayed page, click **Buy Resource Pool** in the upper right corner.
4. On the displayed page, set the parameters.
5. In this example, we will buy the resource pool in the **CN East-Shanghai2** region. [Table 3-2](#) describes the parameters.

Table 3-2 Parameters

Parameter	Description	Example Value
Region	Select a region where you want to buy the elastic resource pool.	CN East-Shanghai2

Parameter	Description	Example Value
Project	Project uniquely preset by the system for each region	Default
Name	Name of the elastic resource pool	dli_resource_pool
Specifications	Specifications of the elastic resource pool	Standard
CU Range	The maximum and minimum CUs allowed for the elastic resource pool	64-64
CIDR Block	CIDR block the elastic resource pool belongs to. If you use an enhanced datasource connection, this CIDR block cannot overlap that of the data source. Once set, this CIDR block cannot be changed.	172.16.0.0/19
Enterprise Project	Select an enterprise project for the elastic resource pool.	default

- Click **Buy**.
- Click **Submit**.
- In the elastic resource pool list, locate the pool you just created and click **Add Queue** in the **Operation** column.
- Set the basic parameters listed below.

Table 3-3 Basic parameters for adding a queue

Parameter	Description	Example Value
Name	Name of the queue to add	dli_queue_01
Type	Type of the queue <ul style="list-style-type: none">To execute SQL jobs, select For SQL.To execute Flink or Spark jobs, select For general purpose.	–
Engine	SQL queue engine. The options include Spark and Trino .	–
Enterprise Project	Select an enterprise project.	default

- Click **Next** and configure scaling policies for the queue.
Click **Create** to add a scaling policy with varying priority, period, minimum CUs, and maximum CUs.

Figure 3-2 shows the scaling policy configured in this example.

Figure 3-2 Configuring a scaling policy when adding a queue

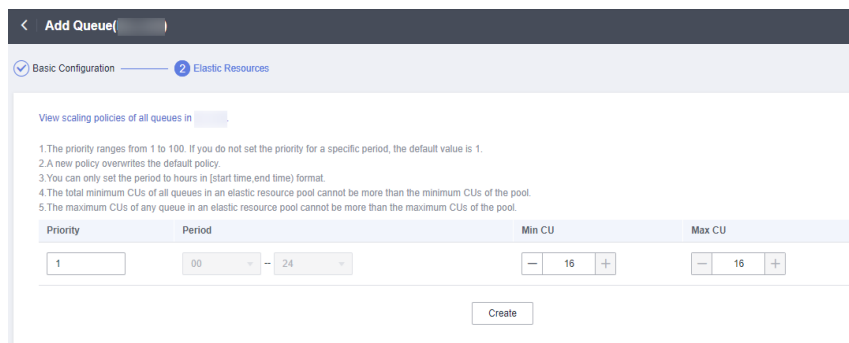


Table 3-4 Scaling policy parameters

Parameter	Description	Example Value
Priority	Priority of the scaling policy in the current elastic resource pool. A larger value indicates a higher priority. In this example, only one scaling policy is configured, so its priority is set to 1 by default.	1
Period	The first scaling policy is the default policy, and its Period parameter configuration cannot be deleted or modified. The period for the scaling policy is from 00 to 24.	00-24
Min CU	Minimum number of CUs allowed by the scaling policy	16
Max CU	Maximum number of CUs allowed by the scaling policy	64

11. Click **OK**.

Step 5: Create an Enhanced Datasource Connection Between DLI and Kafka

You need to create an enhanced datasource connection for the Flink OpenSource SQL job. For details, see "Creating an Enhanced Datasource Connection".

NOTE

- Enhanced datasource connections support only pay-per-use queues.
- The CIDR block of the DLI queue bound with a datasource connection cannot overlap with the CIDR block of the data source.
- Datasource connections cannot be created for the **default** queue.
- To access a table across data sources, you need to use a queue bound to a datasource connection.

Step 1 Create a Kafka security group rule to allow access from the CIDR block of the DLI queue.

1. On the Kafka management console, click an instance name on the **DMS for Kafka** page. Basic information of the Kafka instance is displayed.
2. In the **Connection** pane, obtain the **Instance Address (Private Network)**. In the **Network** pane, obtain the VPC and subnet of the instance.
3. Click the security group name in the **Network** pane. On the displayed page, click the **Inbound Rules** tab and add a rule to allow access from the DLI queue.

For example, if the CIDR block of the queue is **10.0.0.0/16**, set **Protocol** to **TCP**, **Type** to **IPv4**, **Source** to **10.0.0.0/16**, and click **OK**.

Step 2 Create an enhanced datasource connection to Kafka.

1. Log in to the DLI management console. In the navigation pane on the left, choose **Datasource Connections**. On the displayed page, click **Create** in the **Enhanced** tab.
2. In the displayed dialog box, set the following parameters: For details, see the following section:
 - **Connection Name**: Name of the enhanced datasource connection. For this example, enter **dli_kafka**.
 - **Resource Pool**: Select the elastic resource pool created in [Step 4: Create an Elastic Resource Pool and Add Queues to the Pool](#).
 - **VPC**: Select the VPC of the Kafka instance.
 - **Subnet**: Select the subnet of Kafka instance.
 - Set other parameters as you need.

Click **OK**. Click the name of the created datasource connection to view its status. You can perform subsequent steps only after the connection status changes to **Active**.

3. Choose **Resources > Queue Management** and locate the queue created in [Step 4: Create an Elastic Resource Pool and Add Queues to the Pool](#). In the **Operation** column, click **More** and select **Test Address Connectivity**.
4. In the displayed dialog box, enter *Kafka instance address (private network):port* in the **Address** box and click **Test** to check whether the instance is reachable. Note that multiple addresses must be tested separately.

----End

Step 6: Create an Enhanced Datasource Connection Between DLI and RDS

Step 1 Create an RDS security group rule to allow access from CIDR block of the DLI queue.

If the RDS DB instance and Kafka instance are in the same security group of the same VPC, skip this step. Access from the DLI queue has been allowed in [Step 1](#).

1. Go to the RDS console, click the name of the target RDS for MySQL DB instance on the **Instances** page. Basic information of the instance is displayed.
2. In the **Connection Information** pane, obtain the floating IP address, database port, VPC, and subnet.

3. Click the security group name. On the displayed page, click the **Inbound Rules** tab and add a rule to allow access from the DLI queue. For example, if the CIDR block of the queue is **10.0.0.0/16**, set **Priority** to **1**, **Action** to **Allow**, **Protocol** to **TCP**, **Type** to **IPv4**, **Source** to **10.0.0.0/16**, and click **OK**.

Step 2 Create an enhanced datasource connection to RDS.

If the RDS DB instance and Kafka instance are in the same VPC and subnet, skip this step. The enhanced datasource connection created in [Step 2](#) has connected the subnet.

If the two instances are in different VPCs or subnets, perform the following steps to create an enhanced datasource connection:

1. Log in to the DLI management console. In the navigation pane on the left, choose **Datasource Connections**. On the displayed page, click **Create** in the **Enhanced** tab.
2. In the displayed dialog box, set the following parameters: For details, see the following section:
 - **Connection Name**: Name of the enhanced datasource connection. For this example, enter **dli_rds**.
 - **Resource Pool**: Select the name of the queue created in [Step 4: Create an Elastic Resource Pool and Add Queues to the Pool](#).
 - **VPC**: Select the VPC of the RDS DB instance.
 - **Subnet**: Select the subnet of RDS DB instance.
 - Set other parameters as you need.

Click **OK**. Click the name of the created datasource connection to view its status. You can perform subsequent steps only after the connection status changes to **Active**.

3. Choose **Resources > Queue Management** and locate the queue created in [Step 4: Create an Elastic Resource Pool and Add Queues to the Pool](#). In the **Operation** column, click **More** and select **Test Address Connectivity**.
4. In the displayed dialog box, enter *Floating IP address:Database port* of the RDS for MySQL DB instance in the **Address** box and click **Test** to check if the instance is reachable.

----End

Step 7: Use DEW to Manage Access Credentials and Create an Agency That Allows DLI to Access DEW

In cross-source analysis scenarios, you need to set attributes such as the username and password in the connector. However, information such as usernames and passwords is highly sensitive and needs to be encrypted to ensure user data privacy. Flink 1.15 allows for the use of DEW to manage credentials. Before running a job, create a custom agency and configure agency information within the job.

Data Encryption Workshop (DEW) and Cloud Secret Management Service (CSMS) joint form a secure, reliable, and easy-to-use privacy data encryption and decryption solution. This example describes how a Flink OpenSource SQL job uses DEW to manage RDS access credentials.

1. Create an agency for DLI to access DEW and complete authorization.
2. Create a shared secret in DEW.
3. Log in to the DEW management console.
4. In the navigation pane on the left, choose **Cloud Secret Management Service > Secrets**.
5. Click **Create Secret**. On the displayed page, configure basic secret information.
 - **Secret Name:** Enter a secret name. In this example, the name is **secretInfo**.
 - **Secret Value:** Enter the username and password for logging in to the RDS for MySQL DB instance.
 - The key in the first line is **MySQLUsername**, and the value is the username for logging in to the DB instance.
 - The key in the second line is **MySQLPassword**, and the value is the password for logging in to the DB instance.

Figure 3-3 Secret Value

Secret Value

Secret key/value Plaintext

<input type="text" value="MySQLUsername"/>	<input type="text" value="Value"/>	Delete
<input type="text" value="MySQLPassword"/>	<input type="text" value="Value"/>	Delete

[+ Add](#)

6. Set other parameters as required and click **OK**.

Step 8: Create a Flink OpenSource SQL Job

After the source and sink streams are prepared, you can create a Flink OpenSource SQL job.

1. In the left navigation pane of the DLI management console, choose **Job Management > Flink Jobs**. The **Flink Jobs** page is displayed.
2. In the upper right corner of the **Flink Jobs** page, click **Create Job**. Set the following parameters:
 - **Type:** Flink OpenSource SQL
 - **Name:** **JobSample**
 - **Description:** Leave it blank.
 - **Template Name:** Do not select any template.
 - **Tags:** Leave it blank.
3. Click **OK** to enter the editing page.
4. Set job running parameters. The mandatory parameters are as follows:
 - **Queue:** **dli_queue_01**

- **Flink Version:** Select **1.12**.
- **Save Job Log:** Enable this function.
- **OBS Bucket:** Select an OBS bucket for storing job logs and grant access permissions of the OBS bucket as prompted.
- **Enable Checkpointing:** Enable this function.

You do not need to set other parameters.

5. Click **Save**.
6. Edit the Flink OpenSource SQL job.


In the SQL statement editing area, enter query and analysis statements as you need. The example statements are as follows. Note that the values of the parameters in bold must be changed according to the comments.

```
CREATE TABLE kafkaSource (  
  order_id string,  
  order_channel string,  
  order_time string,  
  pay_amount double,  
  real_pay double,  
  pay_time string,  
  user_id string,  
  user_name string,  
  area_id string  
) WITH (  
  'connector' = 'kafka',  
  'topic' = 'testkafkatopic', // Topic to be written to Kafka. Log in to the Kafka console, click the  
  name of the created Kafka instance, and view the topic name on the Topic Management page.  
  'properties.bootstrap.servers' = "192.168.0.237:9092,192.168.0.252:9092,192.168.0.137:9092", //  
  Replace it with the internal network address and port number of Kafka.  
  'properties.group.id' = 'GroupId',  
  'scan.startup.mode' = 'latest-offset',  
  'format' = 'json'  
);  
  
CREATE TABLE jdbcSink (  
  order_id string,  
  order_channel string,  
  order_time string,  
  pay_amount double,  
  real_pay double,  
  pay_time string,  
  user_id string,  
  user_name string,  
  area_id string  
) WITH (  
  'connector' = 'jdbc',  
  'url' = "jdbc:mysql://172.16.0.116:3306/rds-dliflink", // testrdsdb indicates the name of the  
  created RDS database. Replace the IP address and port number with those of the RDS for  
  MySQL instance.  
  'table-name' = 'orders',  
  'pwd_auth_name' = "xxxxx", // Name of the datasource authentication of the password type  
  created on DLI. If datasource authentication is used, you do not need to set the username and  
  password for the job.  
  'sink.buffer-flush.max-rows' = '1'  
);  
  
insert into jdbcSink select * from kafkaSource;
```

7. Click **Check Semantics**.
8. Click **Start**. On the displayed **Start Flink Job** page, confirm the job specifications and the price, and click **Start Now** to start the job.

After the job is started, the system automatically switches to the **Flink Jobs** page, and the created job is displayed in the job list. You can view the job

status in the **Status** column. After a job is successfully submitted, **Status** of the job will change from **Submitting** to **Running**.

If **Status** of a job is **Submission failed** or **Running exception**, the job fails to be submitted or fails to run. In this case, you can hover over the status icon to view the error details. You can click  to copy these details. Rectify the fault based on the error information and resubmit the job.

9. Connect to the Kafka cluster and send the following test data to the Kafka topics:

For details about how Kafka creates and retrieves data, visit [Connecting to an Instance Without SASL](#).

```
{"order_id":"202103241000000001", "order_channel":"webShop", "order_time":"2021-03-24 10:00:00", "pay_amount":"100.00", "real_pay":"100.00", "pay_time":"2021-03-24 10:02:03", "user_id":"0001", "user_name":"Alice", "area_id":"330106"}
```

```
{"order_id":"202103241606060001", "order_channel":"appShop", "order_time":"2021-03-24 16:06:06", "pay_amount":"200.00", "real_pay":"180.00", "pay_time":"2021-03-24 16:10:06", "user_id":"0001", "user_name":"Alice", "area_id":"330106"}
```

10. Run the following SQL statement in the MySQL database to view data in the table:

```
select * from orders;
```

The following is an example of the execution result copied from the MySQL database:

```
202103241000000001,webShop,2021-03-24 10:00:00,100.0,100.0,2021-03-24  
10:02:03,0001,Alice,330106  
202103241606060001,appShop,2021-03-24 16:06:06,200.0,180.0,2021-03-24  
16:10:06,0001,Alice,330106
```

4 Using DLI to Submit a Flink Jar Job

Scenario

Flink Jar jobs are suitable for data analysis scenarios that require custom stream processing logic, complex state management, or integration with specific libraries. You need to write and build a Jar job package. Before submitting a Flink Jar job, upload the Jar job package to OBS and submit it together with the data and job parameters to run the job.

This example introduces the basic process of submitting a Flink Jar job package through the DLI console. Due to different service requirements, the specific writing of the Jar package may vary. It is recommended that you refer to the sample code provided by DLI and edit and customize it according to your actual business scenario. Get [DLI Sample Code](#).

Procedure

[Table 4-1](#) describes the procedure for submitting a Flink Jar job using DLI.

Complete the preparations in [Preparations](#) before performing the following operations.

Table 4-1 Procedure for submitting a Flink Jar job using DLI

Procedure	Description
Step 1: Develop a JAR File and Upload It to OBS	Prepare a Flink Jar job package and upload it to OBS.
Step 2: Buy an Elastic Resource Pool and Add Queues to the Pool	Create compute resources required for submitting the Flink Jar job.
Step 3: Use DEW to Manage Access Credentials	In cross-source analysis scenarios, use DEW to manage access credentials of data sources.

Procedure	Description
Step 4: Create a Custom Agency to Allow DLI to Access DEW and Read Credentials	Create an agency to allow DLI to access DEW.
Step 5: Create a Flink Jar Job and Configure Job Information	Create a Flink Jar job to analyze data.

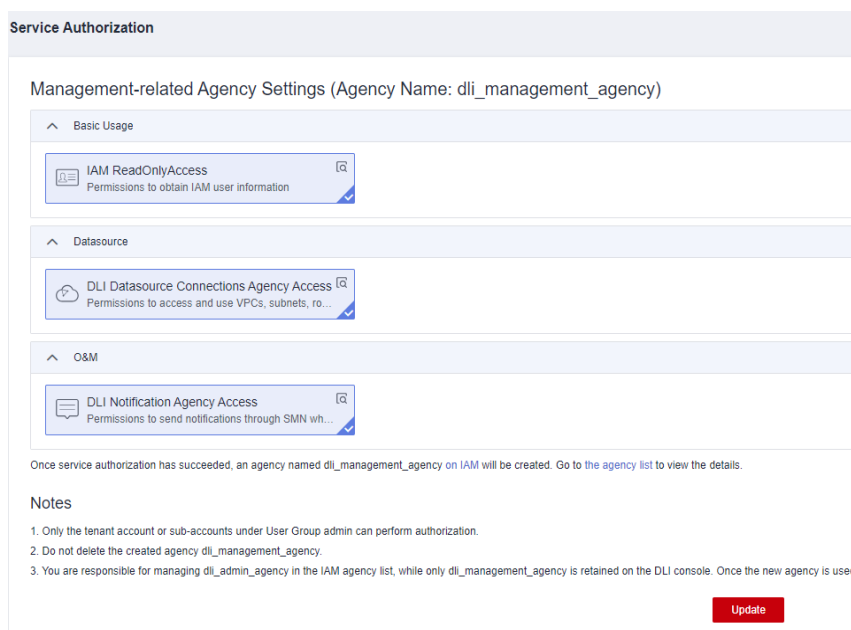
Preparations

- Register a Huawei ID and enable Huawei Cloud services. Make sure your account is not in arrears or frozen.
- Configure an agency for DLI.

To use DLI, you need to access services such as Object Storage Service (OBS), Virtual Private Cloud (VPC), and Simple Message Notification (SMN). If it is your first time using DLI, you will need to configure an agency to allow access to these dependent services.

 - a. Log in to the DLI management console using your account. In the navigation pane on the left, choose **Global Configuration > Service Authorization**.
 - b. On the agency settings page, select the agency permissions under **Basic Usage, Datasource, and O&M** and click **Update**.
 - c. Check and understand the notes for updating the agency, and click **OK**. The DLI agency permissions are updated.

Figure 4-1 Configuring an agency for DLI



- d. Once configured, you can check the agency **dli_management_agency** in the agency list on the IAM console.

Step 1: Develop a JAR File and Upload It to OBS

Develop a JAR file offline as the DLI console does not support this capability. For development examples, refer to "Flink Jar Job Examples".

Develop a Flink Jar job program by referring to [Flink Job Sample Code](#), compile it, and pack it into **flink-examples.jar**. Perform the following steps to upload the program:

Before submitting a Flink job, upload data files to OBS.

1. Log in to the DLI console.
2. In the service list, click **Object Storage Service** under **Storage**.
3. Create a bucket. In this example, name it **dli-test-obs01**.
 - a. On the displayed **Buckets** page, click **Create Bucket** in the upper right corner.
 - b. On the displayed **Create Bucket** page, set **Region** and **Bucket Name**. Retain the default values for other parameters or modify them as needed.

NOTE

Select a region that matches the location of the DLI console.

- c. Click **Create Now**.
4. In the bucket list, click the name of the **dli-test-obs01** bucket you just created to access its **Objects** tab.
5. Click **Upload Object**. In the displayed dialog box, drag or add files or folders, for example, **flink-examples.jar**, to the upload area. Then, click **Upload**.

In this example, the path after upload is **obs://dli-test-obs01/flink-examples.jar**.

For more operations on the OBS console, see the *Object Storage Service User Guide*.

Step 2: Buy an Elastic Resource Pool and Add Queues to the Pool

To execute SQL jobs in datasource scenarios, you must use your own SQL queue as the existing **default** queue cannot be used. In this example, create an elastic resource pool named **dli_resource_pool** and a queue named **dli_queue_01**.

1. Log in to the DLI management console.
2. In the navigation pane on the left, choose **Resources > Resource Pool**.
3. On the displayed page, click **Buy Resource Pool** in the upper right corner.
4. On the displayed page, set the parameters.
5. In this example, we will buy the resource pool in the **CN East-Shanghai2** region. [Table 4-2](#) describes the parameters.

Table 4-2 Parameters

Parameter	Description	Example Value
Region	Select a region where you want to buy the elastic resource pool.	CN East-Shanghai2
Project	Project uniquely preset by the system for each region	Default
Name	Name of the elastic resource pool	dli_resource_pool
Specifications	Specifications of the elastic resource pool	Standard
CU Range	The maximum and minimum CUs allowed for the elastic resource pool	64-64
CIDR Block	CIDR block the elastic resource pool belongs to. If you use an enhanced datasource connection, this CIDR block cannot overlap that of the data source. Once set, this CIDR block cannot be changed.	172.16.0.0/19
Enterprise Project	Select an enterprise project for the elastic resource pool.	default

6. Click **Buy**.
7. Click **Submit**.
8. In the elastic resource pool list, locate the pool you just created and click **Add Queue** in the **Operation** column.
9. Set the basic parameters listed below.

Table 4-3 Basic parameters for adding a queue

Parameter	Description	Example Value
Name	Name of the queue to add	dli_queue_01
Type	Type of the queue <ul style="list-style-type: none"> • To execute SQL jobs, select For SQL. • To execute Flink or Spark jobs, select For general purpose. 	–
Engine	SQL queue engine. The options include Spark and Trino .	–
Enterprise Project	Select an enterprise project.	default

- Click **Next** and configure scaling policies for the queue.
Click **Create** to add a scaling policy with varying priority, period, minimum CUs, and maximum CUs.

Figure 4-2 shows the scaling policy configured in this example.

Figure 4-2 Configuring a scaling policy when adding a queue

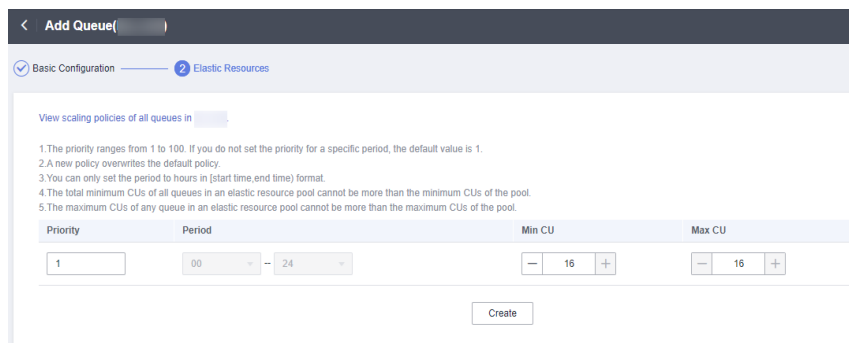


Table 4-4 Scaling policy parameters

Parameter	Description	Example Value
Priority	Priority of the scaling policy in the current elastic resource pool. A larger value indicates a higher priority. In this example, only one scaling policy is configured, so its priority is set to 1 by default.	1
Period	The first scaling policy is the default policy, and its Period parameter configuration cannot be deleted or modified. The period for the scaling policy is from 00 to 24.	00-24
Min CU	Minimum number of CUs allowed by the scaling policy	16
Max CU	Maximum number of CUs allowed by the scaling policy	64

- Click **OK**.

Step 3: Use DEW to Manage Access Credentials

In cross-source analysis scenarios, you need to set attributes such as the username and password in the connector. However, information such as usernames and passwords is highly sensitive and needs to be encrypted to ensure user data privacy.

Data Encryption Workshop (DEW) is a secure, reliable, and easy-to-use solution for encrypting and decrypting private data while ensuring its security.

This example introduces how to create a shared secret in DEW.

1. Log in to the DEW management console.
2. In the navigation pane on the left, choose **Cloud Secret Management Service > Secrets**.
3. Click **Create Secret**. On the displayed page, configure basic secret information.
 - In this example, the key in the first line is the user's access key ID (AK).
 - In this example, the key in the second line is the user's secret access key (SK).

Figure 4-3 Configuring access credentials in DEW

Secret Value

Secret key/value Plaintext

USER_AK_CSMS_KEY_obstest1 Value Delete

USER_SK_CSMS_KEY_obstest1 Value Delete

+ Add

4. Set access credential parameters on the DLI Flink Jar job editing page.

```
flink.hadoop.fs.obs.bucket.USER_BUCKET_NAME.dew.access.key=USER_AK_CSMS_KEY_obstest1
flink.hadoop.fs.obs.bucket.USER_BUCKET_NAME.dew.secret.key=USER_SK_CSMS_KEY_obstest1
flink.hadoop.fs.obs.security.provider=com.dli.provider.UserObsBasicCredentialProvider
flink.hadoop.fs.dew.csms.secretName=obsAksKflink.hadoop.fs.dew.endpoint=kmsendpoint
flink.hadoop.fs.dew.csms.version=v6flink.hadoop.fs.dew.csms.cache.time.second=3600flink.dli.job.agency.name=agencyname
```

Step 4: Create a Custom Agency to Allow DLI to Access DEW and Read Credentials

1. Log in to the management console.
2. In the upper right corner of the page, hover over the username and select **Identity and Access Management**.
3. In the navigation pane of the IAM console, choose **Agencies**.
4. On the displayed page, click **Create Agency**.
5. On the **Create Agency** page, set the following parameters:
 - **Agency Name:** Enter an agency name, for example, **dli_dew_agency_access**.
 - **Agency Type:** Select **Cloud service**.
 - **Cloud Service:** This parameter is available only when you select **Cloud service** for **Agency Type**. Select **Data Lake Insight (DLI)** from the drop-down list.
 - **Validity Period:** Select **Unlimited**.
 - **Description:** You can enter **Agency with OBS OperateAccess permissions**. This parameter is optional.
6. Click **Next**.
7. Click the agency name. On the displayed page, click the **Permissions** tab. Click **Authorize**. On the displayed page, click **Create Policy**.

8. Configure policy information.
 - a. Enter a policy name, for example, **dli-dew-agency**.
 - b. Select **JSON**.
 - c. In the **Policy Content** area, paste a custom policy.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "csms:secretVersion:get",
        "csms:secretVersion:list",
        "kms:dek:decrypt"
      ]
    }
  ]
}
```

- d. Enter a policy description as required.
 9. Click **Next**.
 10. On the **Select Policy/Role** page, select **Custom policy** from the first drop-down list and select the custom policy created in **8**.
 11. Click **Next**. On the **Select Scope** page, set the authorization scope. In this example, select **All resources**.
- For details about authorization operations, see [Creating a User Group and Assigning Permissions](#).
12. Click **OK**.

It takes 15 to 30 minutes for the authorization to be in effect.

Step 5: Create a Flink Jar Job and Configure Job Information

Step 1 Create a Flink Jar job.

1. In the navigation pane on the left of the DLI management console, choose **Job Management > Flink Jobs**.
2. On the displayed page, click **Create Job** in the upper right corner.
In this example, set **Type** to **Flink Jar** and **Name** to **Flink_Jar_for_test**.

Figure 4-4 Creating a Flink Jar job

Create Job

Type

* Name

Description

Tags

It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. [View predefined tags](#)

To add a tag, enter a tag key and a tag value below.

20 tags available for addition.

3. Click **OK**.

Step 2 Configure basic job information.

Configure basic job information based on [Table 4-5](#).

Table 4-5 Parameters

Parameter	Mandato ry	Description
Queue	Yes	Select a queue where you want to run your job.
Application	Yes	Select the custom package in Step 1: Develop a JAR File and Upload It to OBS .

Parameter	Mandatory	Description
Main Class	Yes	<p>Class name of the JAR file to load</p> <p>This parameter specifies the entry for the Flink job, that is, the class that contains the main method. This is the class that is executed first when a Flink job is started.</p> <p>If the application program is of type .jar, the main class name must be provided.</p> <p>The main class name is case-sensitive and must be correct.</p> <ul style="list-style-type: none"> • Default: Specified based on the Manifest file in the JAR file. • Manually assign: You must enter the class name and confirm the class arguments (separated by spaces). <p>NOTE When a class belongs to a package, the package path must be carried, for example, packageName.KafkaMessageStreaming.</p>
Flink Version	Yes	<p>Flink version used for job running</p> <p>If you choose to use Flink 1.15, make sure to configure the agency information for the cloud service that DLI is allowed to access in the job.</p>
Agency	No	<p>If you choose to use Flink 1.15, configure the agency name yourself to ensure smooth job running.</p>

Step 3 Configure advanced settings for the Flink Jar job.

Configure the Flink Jar job based on [Table 4-6](#).

Table 4-6 Advanced settings for the Flink Jar job

Parameter	Mandatory	Description
CUs	Yes	<p>One CU consists of one vCPU and 4 GB of memory. The number of CUs ranges from 2 to 400.</p>
Job Manager CUs	Yes	<p>Number of CUs allowed for the job manager. The value ranges from 1 to 4. The default value is 1.</p>

Parameter	Mandatory	Description
Parallelism	Yes	<p>Maximum number of parallel operators in a job</p> <p>NOTE</p> <ul style="list-style-type: none"> The value cannot exceed four times the number of compute units (CUs – Job Manager CUs). You are advised to set this parameter to a value greater than that configured in the code. Otherwise, job submission may fail.
Task Manager Config	No	<p>Whether Task Manager resource parameters are set</p> <p>If this option is selected, you need to set the following parameters:</p> <ul style="list-style-type: none"> CU(s) per TM: Number of resources occupied by each Task Manager. Slot(s) per TM: Number of slots contained in each Task Manager.
Save Job Log	No	<p>Whether job running logs are saved to OBS</p> <p>If this option is selected, you need to set the following parameters:</p> <p>OBS Bucket: Select an OBS bucket to store job logs. If the OBS bucket you selected is unauthorized, click Authorize.</p>
Alarm on Job Exception	No	<p>Whether to notify users of any job exceptions, such as running exceptions or arrears, via SMS or email.</p> <p>If this option is selected, you need to set the following parameters:</p> <p>SMN Topic</p> <p>Select a custom SMN topic. For how to create a custom SMN topic, see "Creating a Topic" in the <i>Simple Message Notification User Guide</i>.</p>

Parameter	Mandatory	Description
Auto Restart on Exception	No	<p>Whether automatic restart is enabled. If enabled, jobs will be automatically restarted and restored when exceptions occur.</p> <p>If this option is selected, you need to set the following parameters:</p> <ul style="list-style-type: none"> Max. Retry Attempts: maximum number of retries upon an exception. The unit is times/hour. <ul style="list-style-type: none"> Unlimited: The number of retries is unlimited. Limited: The number of retries is user-defined. Restore Job from Checkpoint: Restore the job from the latest checkpoint. If you select this parameter, you also need to set Checkpoint Path. <p>Checkpoint Path: Select a path for storing checkpoints. This path must match that configured in the application package. Each job must have a unique checkpoint path, or, you will not be able to obtain the checkpoint.</p>


Step 4 Click **Save** in the upper right of the page.

Step 5 Click **Start** in the upper right corner.

Step 6 On the displayed **Start Flink Job** page, confirm the job specification and fee and click **Start Now** to start the job.

Once the job is started, the system automatically switches to the **Flink Jobs** page. Locate the job you created and check its status in the **Status** column.

Once a job is successfully submitted, its status changes from **Submitting** to **Running**. After the execution is complete, the status changes to **Completed**.

If the job status is **Submission failed** or **Running exception**, the job fails to submit or run. In this case, you can hover over the status icon in the **Status** column of the job list to view the error details. You can click  to copy these details. Rectify the fault based on the error information and resubmit the job.

----End

5 Using DLI to Submit a Spark Jar Job

Scenario

DLI allows you to submit Spark jobs compiled as JAR files, which contain the necessary code and dependency information for executing the job. These files are used for specific data processing tasks such as data query, analysis, and machine learning. Before submitting a Spark Jar job, upload the package to OBS and submit it along with the data and job parameters to run the job.

This example introduces the basic process of submitting a Spark Jar job package through the DLI console. Due to different service requirements, the specific writing of the Jar package may vary. It is recommended that you refer to the sample code provided by DLI and edit and customize it according to your actual business scenario. Get [DLI Sample Code](#).

Procedure

[Table 5-1](#) describes the procedure for submitting a Spark Jar job using DLI.

Complete the preparations in [Preparations](#) before performing the following operations.

Table 5-1 Procedure for submitting a Spark Jar job using DLI

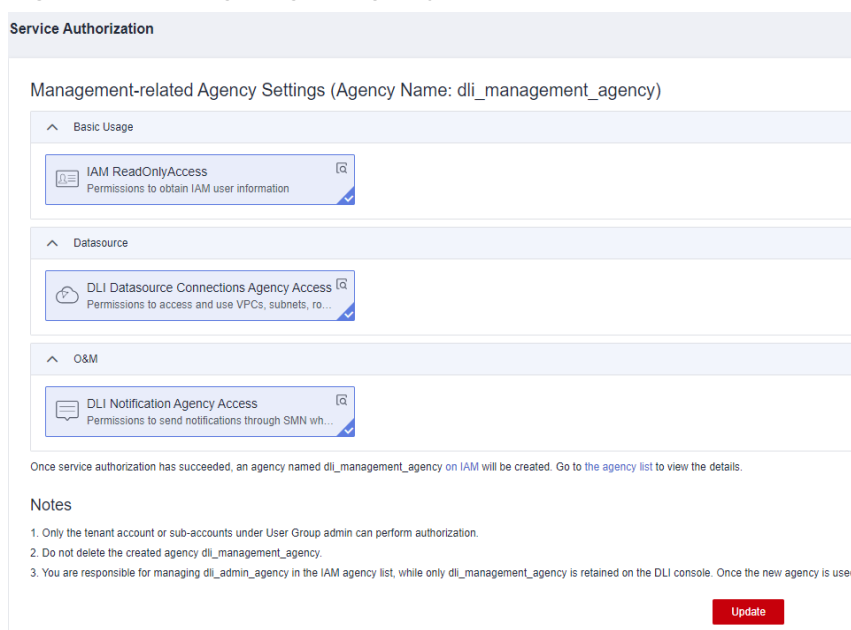
Procedure	Description
Step 1: Upload Data to OBS	Prepare a Spark Jar job package and upload it to OBS.
Step 2: Create an Elastic Resource Pool and Add Queues to the Pool	Create compute resources required for submitting the Spark Jar job.
Step 3: Use DEW to Manage Access Credentials	In cross-source analysis scenarios, use DEW to manage access credentials of data sources and create an agency that allows DLI to access DEW.

Procedure	Description
Step 4: Create a Custom Agency to Allow DLI to Access DEW and Read Credentials	Create an agency to allow DLI to access DEW.
Step 5: Submit a Spark Job	Create a Spark Jar job to analyze data.

Preparations

- Register a Huawei ID and enable Huawei Cloud services. Make sure your account is not in arrears or frozen.
- Configure an agency for DLI.
To use DLI, you need to access services such as Object Storage Service (OBS), Virtual Private Cloud (VPC), and Simple Message Notification (SMN). If it is your first time using DLI, you will need to configure an agency to allow access to these dependent services.
 - a. Log in to the DLI management console using your account. In the navigation pane on the left, choose **Global Configuration > Service Authorization**.
 - b. On the agency settings page, select the agency permissions under **Basic Usage, Datasource, and O&M** and click **Update**.
 - c. Check and understand the notes for updating the agency, and click **OK**. The DLI agency permissions are updated.

Figure 5-1 Configuring an agency for DLI



- d. Once configured, you can check the agency **dli_management_agency** in the agency list on the IAM console.

Step 1: Upload Data to OBS

Develop a Spark Jar job program by referring to [Spark Job Sample Code](#), compile it, and pack it into `spark-examples.jar`. Perform the following steps to upload the program:

Before submitting Spark Jar jobs, upload data files to OBS.

1. Log in to the DLI console.
2. In the service list, click **Object Storage Service** under **Storage**.
3. Create a bucket. In this example, name it `dli-test-obs01`.
 - a. On the displayed **Buckets** page, click **Create Bucket** in the upper right corner.
 - b. On the displayed **Create Bucket** page, specify **Region** and enter the **Bucket Name**. Retain the default values for other parameters or set them as required.

NOTE

Select a region that matches the location of the DLI console.

- c. Click **Create Now**.
4. In the bucket list, click the name of the `dli-test-obs01` bucket you just created to access its **Objects** tab.
 5. Click **Upload Object**. In the dialog box displayed, drag or add files or folders, for example, `spark-examples.jar`, to the upload area. Then, click **Upload**.

In this example, the path after upload is `obs://dli-test-obs01/spark-examples.jar`.

For more operations on the OBS console, see the *Object Storage Service User Guide*.

Step 2: Create an Elastic Resource Pool and Add Queues to the Pool

In this example, the elastic resource pool `dli_resource_pool` and queue `dli_queue_01` are created.

1. Log in to the DLI management console.
2. In the navigation pane on the left, choose **Resources** > **Resource Pool**.
3. On the displayed page, click **Buy Resource Pool** in the upper right corner.
4. On the displayed page, set the parameters.
5. In this example, we will buy the resource pool in the **CN East-Shanghai2** region. [Table 5-2](#) describes the parameters.

Table 5-2 Parameters

Parameter	Description	Example Value
Region	Select a region where you want to buy the elastic resource pool.	CN East-Shanghai2
Project	Project uniquely preset by the system for each region	Default

Parameter	Description	Example Value
Name	Name of the elastic resource pool	dli_resource_pool
Specifications	Specifications of the elastic resource pool	Standard
CU Range	The maximum and minimum CUs allowed for the elastic resource pool	64-64
CIDR Block	CIDR block the elastic resource pool belongs to. If you use an enhanced datasource connection, this CIDR block cannot overlap that of the data source. Once set, this CIDR block cannot be changed.	172.16.0.0/19
Enterprise Project	Select an enterprise project for the elastic resource pool.	default

6. Click **Buy**.
7. Click **Submit**.
8. In the elastic resource pool list, locate the pool you just created and click **Add Queue** in the **Operation** column.
9. Set the basic parameters listed below.

Table 5-3 Basic parameters for adding a queue

Parameter	Description	Example Value
Name	Name of the queue to add	dli_queue_01
Type	Type of the queue <ul style="list-style-type: none"> • To execute SQL jobs, select For SQL. • To execute Flink or Spark jobs, select For general purpose. 	–
Engine	SQL queue engine. The options include Spark and Trino .	–
Enterprise Project	Select an enterprise project.	default

10. Click **Next** and configure scaling policies for the queue.
Click **Create** to add a scaling policy with varying priority, period, minimum CUs, and maximum CUs.

Figure 5-2 shows the scaling policy configured in this example.

Figure 5-2 Configuring a scaling policy when adding a queue

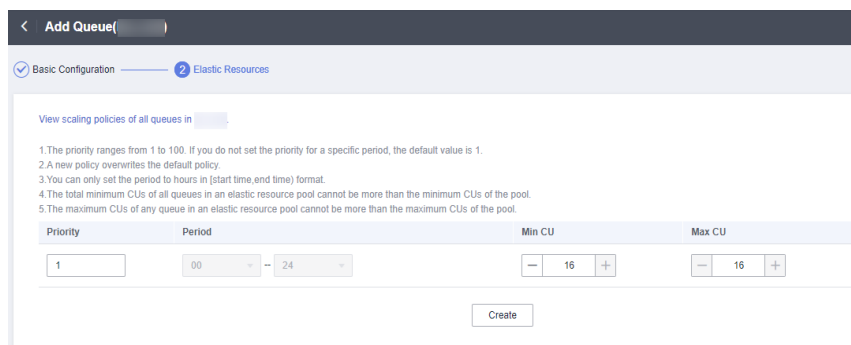


Table 5-4 Scaling policy parameters

Parameter	Description	Example Value
Priority	Priority of the scaling policy in the current elastic resource pool. A larger value indicates a higher priority. In this example, only one scaling policy is configured, so its priority is set to 1 by default.	1
Period	The first scaling policy is the default policy, and its Period parameter configuration cannot be deleted or modified. The period for the scaling policy is from 00 to 24.	00-24
Min CU	Minimum number of CUs allowed by the scaling policy	16
Max CU	Maximum number of CUs allowed by the scaling policy	64

11. Click **OK**.

Step 3: Use DEW to Manage Access Credentials

To write the output data of a Spark Jar job to OBS, AK/SK is required for accessing OBS. To ensure the security of AK/SK data, you can use Data Encryption Workshop (DEW) and Cloud Secret Management Service (CSMS) for unified management of AK/SK, effectively avoiding sensitive information leakage and business risks caused by hard-coded or plaintext configuration of programs.

This part introduces how to create a shared secret in DEW.

1. Log in to the DEW management console.
2. In the navigation pane on the left, choose **Cloud Secret Management Service > Secrets**.
3. On the displayed page, click **Create Secret**. Set basic secret information. Set the AK and SK credential key-value pairs.

- In this example, the key in the first line is the user's access key ID (AK).
- In this example, the key in the second line is the user's secret access key (SK).

Figure 5-3 Configuring access credentials in DEW

Secret Value

Secret key/value Plaintext

USER_AK_CSMS_KEY_obstest1	Value	Delete
USER_SK_CSMS_KEY_obstest1	Value	Delete

[+ Add](#)

4. Set access credential parameters on the DLI Spark Jar job editing page.

```
spark.hadoop.fs.obs.bucket.USER_BUCKET_NAME.dew.access.key= USER_AK_CSMS_KEY_obstest1
spark.hadoop.fs.obs.bucket.USER_BUCKET_NAME.dew.secret.key= USER_SK_CSMS_KEY_obstest1
spark.hadoop.fs.obs.security.provider=com.dli.provider.UserObsBasicCredentialProvider
spark.hadoop.fs.dew.csms.secretName=obsAkSkspark.hadoop.fs.dew.endpoint=kmsendpoint
spark.hadoop.fs.dew.csms.version=v3spark.dli.job.agency.name=agencyname
```

Step 4: Create a Custom Agency to Allow DLI to Access DEW and Read Credentials

1. Log in to the management console.
2. In the upper right corner of the page, hover over the username and select **Identity and Access Management**.
3. In the navigation pane of the IAM console, choose **Agencies**.
4. On the displayed page, click **Create Agency**.
5. On the **Create Agency** page, set the following parameters:
 - **Agency Name:** Enter an agency name, for example, **dli_dew_agency_access**.
 - **Agency Type:** Select **Cloud service**.
 - **Cloud Service:** This parameter is available only when you select **Cloud service** for **Agency Type**. Select **Data Lake Insight (DLI)** from the drop-down list.
 - **Validity Period:** Select **Unlimited**.
 - **Description:** You can enter **Agency with OBS OperateAccess permissions**. This parameter is optional.
6. Click **Next**.
7. Click the agency name. On the displayed page, click the **Permissions** tab. Click **Authorize**. On the displayed page, click **Create Policy**.
8. Configure policy information.
 - a. Enter a policy name, for example, **dli-dew-agency**.
 - b. Select **JSON**.
 - c. In the **Policy Content** area, paste a custom policy.

```
{
  "Version": "1.1",
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": [
    "csms:secretVersion:get",
    "csms:secretVersion:list",
    "kms:dek:decrypt"
  ]
}
```

- d. Enter a policy description as required.
9. Click **Next**.
10. On the **Select Policy/Role** page, select **Custom policy** from the first drop-down list and select the custom policy created in [8](#).
11. Click **Next**. On the **Select Scope** page, set the authorization scope. In this example, select **All resources**.
For details about authorization operations, see [Creating a User Group and Assigning Permissions](#).
12. Click **OK**.
It takes 15 to 30 minutes for the authorization to be in effect.

Step 5: Submit a Spark Job

1. On the DLI management console, choose **Job Management > Spark Jobs** in the navigation pane on the left. On the displayed page, click **Create Job** in the upper right corner.
2. Set the following Spark job parameters:
 - **Queue**: Select the queue created in [Step 2: Create an Elastic Resource Pool and Add Queues to the Pool](#).
 - **Spark Version**: Select a Spark engine version. In this example, version 3.3.1 is selected.
 - **Application**: Select the package created in [Step 1: Upload Data to OBS](#).
 - **Agency**: Select the agency created in [Step 4: Create a Custom Agency to Allow DLI to Access DEW and Read Credentials](#), which is used to allow DLI to access the credentials stored in DEW.

For other parameters, refer to the description about the Spark job editing page in "Creating a Spark Job" in the *Data Lake Insight User Guide*.

3. Click **Execute** in the upper right corner of the Spark job editing window, read and agree to the privacy agreement, and click **OK**. Submit the job. A message is displayed, indicating that the job is submitted.
4. (Optional) Switch to the **Job Management > Spark Jobs** page to view the status and logs of the submitted Spark job.

NOTE

When you click **Execute** on the DLI management console for the first time, you need to read the privacy agreement. Once agreed to the agreement, you will not receive any privacy agreement messages for subsequent operations.

6 Practices

You can better use DLI for big data analytics and processing by following the scenario-specific instructions and best practices provided in this section.

Table 6-1 Common DLI development instructions and best practices

Scenario	Instructions	Description
Connecting a queue to an external data source	Configuring the Connection Between a DLI Queue and a Data Source in a Private Network	When creating and running a job on a DLI queue, you need to connect the DLI queue to external data sources. This section describes how to connect DLI queues to external data sources. For example, to connect a DLI queue to MRS, RDS, CSS, Kafka, or GaussDB(DWS), you need to configure the connection between the queue and the external data source.
	Configuring the Connection Between a DLI Queue and a Data Source in the Internet	Connect a DLI queue to a data source on the Internet. You can configure SNAT rules and add routes to the public network to enable communications between a queue and the Internet.
Spark SQL job development	Using Spark SQL Jobs to Analyze OBS Data	Use a Spark SQL job to create OBS tables, and import, insert, and query OBS table data.
Flink OpenSource SQL job development	Reading Data from Kafka and Writing Data to RDS	Use a Flink OpenSource SQL job to read data from Kafka and write the data to RDS.
	Reading Data from Kafka and Writing Data to GaussDB(DWS)	Use a Flink OpenSource SQL job to read data from Kafka and write the data to GaussDB(DWS).

Scenario	Instructions	Description
	Reading Data from Kafka and Writing Data to Elasticsearch	Use a Flink OpenSource SQL job to read data from Kafka and write the data to Elasticsearch.
	Reading Data from MySQL CDC and Writing Data to GaussDB(DWS)	Use a Flink OpenSource SQL job to read data from MySQL CDC and write the data to GaussDB(DWS).
	Reading Data from PostgreSQL CDC and Writing Data to GaussDB(DWS)	Use a Flink OpenSource SQL job to read data from PostgreSQL CDC and write the data to GaussDB(DWS).
Flink Jar job development	Flink Jar Job Examples	Create a custom Flink Jar job to interact with MRS.
	Using Flink Jar to Write Data to OBS	Write Kafka data to OBS.
	Using Flink Jar to Connect to Kafka with SASL_SSL Authentication Enabled	Use Flink OpenSource SQL to connect to Kafka with SASL_SSL authentication enabled.
Spark Jar job development	Using Spark Jar Jobs to Read and Query OBS Data	Write a Spark program to read and query OBS data, compile and package your code, and submit a Spark Jar job.
Data migration	Migrating Data from Hive to DLI	Migrate data from MRS Hive to DLI using the CDM data synchronization function.
	Migrating Data from Kafka to DLI	Migrate data from MRS Kafka to DLI using the CDM data synchronization function.
	Migrating Data from Elasticsearch to DLI	Migrate data from a CSS Elasticsearch cluster to DLI using the CDM data synchronization function.
	Migrating Data from RDS to DLI	Migrate data from an RDS database to DLI using the CDM data synchronization function.
	Migrating Data from GaussDB(DWS) to DLI	Migrate data from GaussDB(DWS) to DLI using the CDM data synchronization function.