

Data Lake Insight

Getting Started

Issue **01**
Date **2023-07-12**



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.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <https://www.huawei.com>

Email: support@huawei.com

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 Registering a Huawei ID and Enabling Huawei Cloud Services.....	1
2 Creating and Submitting a Spark SQL Job.....	4
3 Developing and Submitting a Spark SQL Job Using the TPC-H Sample Template.....	8
4 Creating and Submitting a Spark Jar Job.....	10
5 Creating and Submitting a Flink OpenSource SQL Job	14
6 Creating an Enhanced Datasource Connection to RDS.....	22
7 Practices.....	30
A Change History.....	32

1 Registering a Huawei ID and Enabling Huawei Cloud Services

Before using DLI, register a Huawei ID, enable Huawei Cloud services, and authorize DLI.

Registering a Huawei ID and Enabling Huawei Cloud Services

Skip over this procedure if you already have registered one.

Step 1 Visit and register with the [Huawei Cloud](#) official website.

Step 2 Click **Register** in the upper right corner to go to the registration page.



Step 3 Complete the registration as instructed. For details, see [Account Registration Process](#).

Step 4 Select the terms of service as prompted and click **Enable** to enable Huawei Cloud services.

----End

Service Authorization

After you log in to the public cloud and access the DLI management console, you are advised to set agency permissions to ensure that DLI can be used properly.

You do not need to set this parameter again after you log in to the system for the first time. If you need to change the value, choose **Global Configurations** > **Service Authorization** and change the value.

1. After login, click **Access Console** on the [DLI product page](#).
2. On the management console, choose **Global Configuration** > **Service Authorization**. On the page displayed, select permissions as needed by referring to [Table 1-1](#), and click **Update**.

 NOTE

Only the tenant account or a subaccount of user group **admin** can authorize access.

- Once service authorization has succeeded, an agency named **dli_admin_agency** on IAM will be created. Go to the agency list to view the details. Do not delete **dli_admin_agency**.

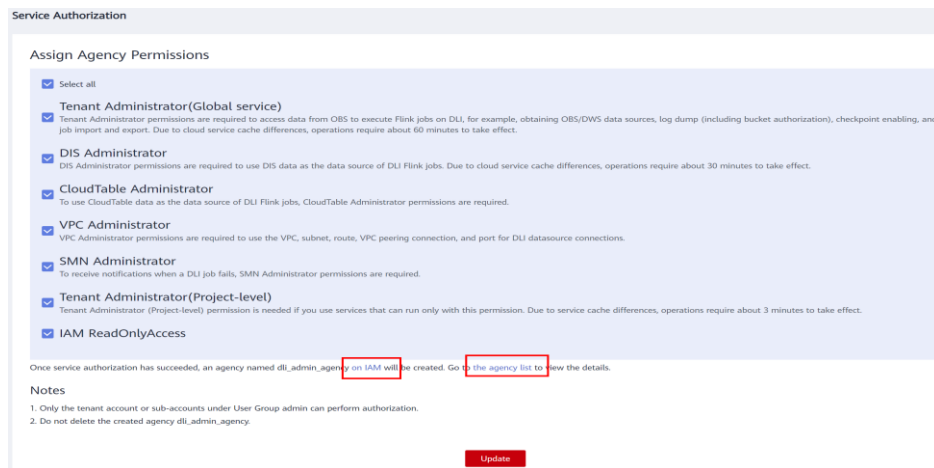


Table 1-1 DLI agency permissions

Permission	Details	Remarks
Tenant Administrator (global service)	Tenant Administrator permissions are required to access data from OBS to execute Flink jobs on DLI, for example, obtaining OBS/DWS data sources, log dump (including bucket authorization), checkpointing enabling, and job import and export.	Due to cloud service cache differences, permission setting operations require about 60 minutes to take effect.
DIS Administrator	DIS Administrator permissions are required to use DIS data as the data source of DLI Flink jobs.	Due to cloud service cache differences, permission setting operations require about 30 minutes to take effect.
CloudTable Administrator	To use CloudTable data as the data source of DLI Flink jobs, CloudTable Administrator permissions are required.	Due to cloud service cache differences, permission setting operations require about 3 minutes to take effect.
VPC Administrator	VPC Administrator permissions are required to use the VPC, subnet, route, VPC peering connection, and port for DLI datasource connections.	Due to cloud service cache differences, permission setting operations require about 3 minutes to take effect.
SMN Administrator	To receive notifications when a DLI job fails, SMN Administrator permissions are required.	Due to cloud service cache differences, permission setting operations require about

Permission	Details	Remarks
		3 minutes to take effect.
IAM ReadOnlyAccess	To authorize IAM users who have not logged in to DLI, you need to obtain their information. Therefore, the IAM ReadOnlyAccess permission is required.	To authorize IAM users who have not logged in to DLI, you need to obtain their information. So, the IAM ReadOnlyAccess permission is required.

2 Creating and Submitting a Spark SQL Job

Scenario

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

Procedure

You can use DLI to submit a Spark SQL job to query data. The general procedure is as follows:

[Step 1: Upload Data to OBS](#)

[Step 2: Create a Queue](#)

[Step 3: Create a Database](#)

[Step 4: Create a Table](#)

[Step 5: Query Data](#)

Step 1: Upload Data to OBS

Before you use DLI to query and analyze data, upload data files to OBS.

1. Go to the DLI console.
2. In the service list, click **Object Storage Service** under **Storage**. The OBS console page is displayed.
3. 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

You must select the same region as the DLI management console.

- c. Click **Create Now**.
4. Click **obs1** to access its **Objects** tab page.
 5. 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**.

```
12,test
```

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

NOTE

- For more information about OBS operations, see the [Object Storage Service Console Operation Guide](#).
- For more information about the tool, see the [OBS Tool Guide](#).
- You are advised to use an OBS tool, such as OBS Browser+ or obsutil, to upload large files because OBS Console has restrictions on the file size and quantity.
- OBS Browser+ is a graphical tool that provides complete functions for managing your buckets and objects in OBS.
- obsutil is a command line tool for accessing and managing OBS resources. If you are familiar with command line interface (CLI), obsutil is recommended as an ideal tool for batch processing and automated tasks.

You can upload files to a bucket in the following ways. Then OBS stores these files as objects in the bucket.

Table 2-1 Access modes of objects uploaded to OBS

Access Mode	Upload Method
Console	Uploading an object using OBS Console
OBS Browser+	Uploading an object using OBS Browser+
obsutil	Uploading an object using obsutil
SDK	Uploading an object using SDKs . For details, see the section about object uploading in the developer guide of each language.
API	Uploading objects - PUT and Uploading objects - POST

Step 2: Create a Queue

A queue is the basis for using DLI. Before executing an SQL job, you need to create a queue.

- An available queue **default** is preset in DLI. If the **default** queue is used, you will be billed based on the amount of data scanned.
- You can also create queues as needed. If a self-built queue is used, you will be billed based on the used CUH.
 - a. Log in to the DLI management console.

NOTE

If this is your first time logging in to the DLI management console, you need to be authorized to access OBS.

For this example, you need at least the **Tenant Administrator (Global service)** permission.

- b. In the left navigation pane of the DLI management console, choose **SQL Editor**.


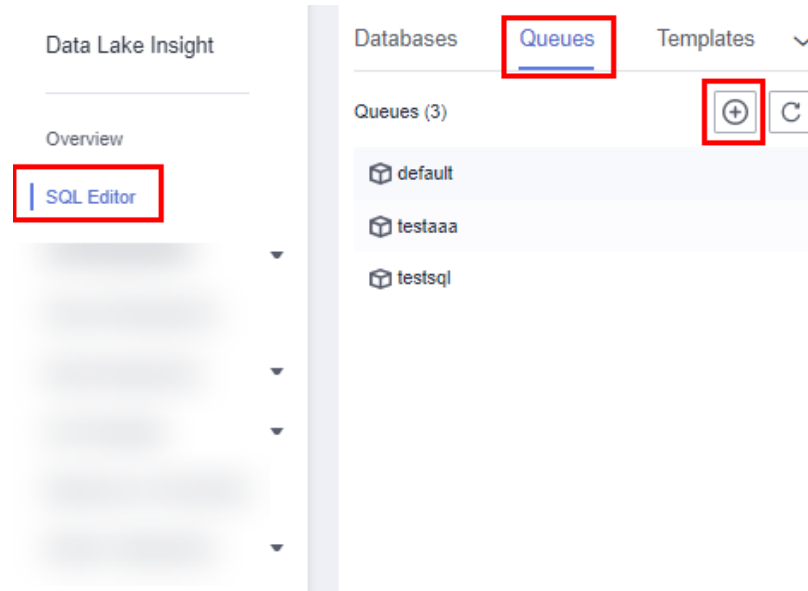
- c. On the left pane, select the **Queues** tab, and click  next to **Queues**.

Figure 2-1 Creating a queue



For details, see [Creating a Queue](#).

For details, see [Data Lake Insight Billing](#).

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 (id int, name string) using csv options (path  
'obs://obs1/sampleddata.csv');
```

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.

3 Developing and Submitting a Spark SQL Job Using the TPC-H Sample Template

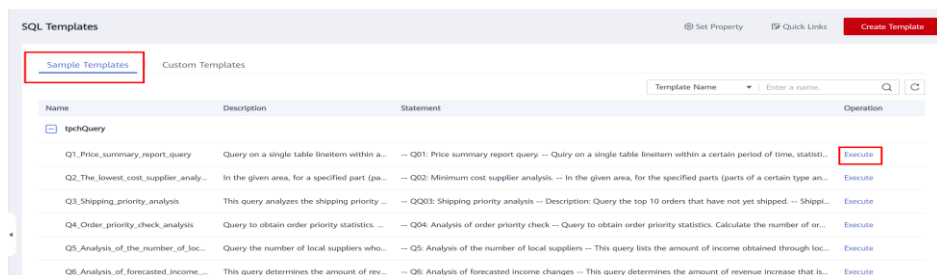
DLI allows you to customize query templates or save frequently used SQL statements as templates to facilitate SQL operations. After templates are saved, you do not need to write SQL statements. You can directly perform the SQL operations using the templates.

The current system provides various standard TPC-H query statement templates. You can select a template as needed. This example shows how to use a TPC-H template to develop and submit a Spark SQL job.

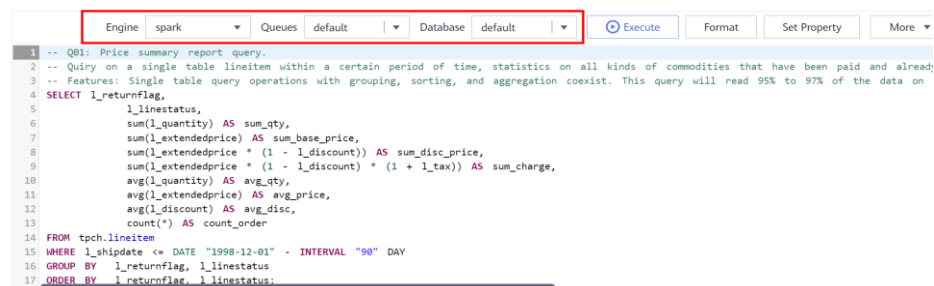
For details about the templates, see [SQL Template Management](#).

Procedure

1. Log in to the DLI management console.
2. On the DLI management console, choose **Job Templates > SQL Templates**, and click the **Sample Templates** tab. Locate the **Q1_Price_summary_report_query** template under **tpchQuery**, and click **Execute** in the **Operation** column. The **SQL Editor** page is displayed.



3. In the upper part of the editing window, set **Engine** to **spark**, **Queues** to **default**, and **Databases** to **default**, and click **Execute**.



4. View the query result in the **View Result** tab in the lower part of the SQL Editor page.

The screenshot shows the 'View Result' tab in the SQL Editor. It displays a table with the following data:

l_returnflag	l_linestatus	sum_qty	sum_base_price	sum_disc_price	sum_charge	avg_qty	avg_price	avg_disc	count_order
A	F	3774200	5320753880.6900215	5054096266.682835	5256751331.449283	25.53758711685...	36002.12382901429	0.0501445970634...	147790
N	F	95257	13373795.83999997	127152372.65119997	132286291.22944508	25.30066401062...	35521.5269163346...	0.0493944223107...	3765
N	O	7459297	10512270008.89968	9986238338.38475	10385578376.585411	25.54553767123...	36000.9246880132...	0.0500959589041...	292000
R	F	3785523	5337950526.469969	5071818532.941963	5274405503.048963	25.52594385742...	35994.0292140307...	0.0499892785618...	148301

This example uses the **default** queue and database preset in the system as an example. You can also run query statements on a self-created queue and database.

For details about how to create a queue, see [Creating a Queue](#). For details about how to create a database, see [Creating a Database](#).

4 Creating and Submitting a Spark Jar Job

Scenario

DLI can query data stored in OBS. This section describes how to use a Spark Jar job on DLI to query OBS data in real time.

Procedure

You can use DLI to submit Spark jobs for real-time computing. The general procedure is as follows:

[Step 1: Upload Data to OBS](#)

[Step 2: Create a Queue](#)

[Step 3: Create a Package](#)

[Step 4: Submit a Spark Job](#)

Step 1: Upload Data to OBS

Write a Spark Jar job program by referring to [Spark job sample code](#), and compile and pack it as `spark-examples.jar`. Perform the following steps to submit the job:

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**. The OBS console page is displayed.
3. Create a bucket. In this example, name it **dli-test-obs01**.
 - a. Click **Create Bucket**.
 - 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

When creating an OBS bucket, you must select the same region as the DLI management console.

- c. Click **Create Now**.
4. Click **dli-test-obs01** to switch to the **Objects** tab page.
 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**.

After the file is uploaded successfully, the file path is **obs://dli-test-obs01/spark-examples.jar**.

NOTE

- For more information about OBS operations, see the [Object Storage Service Console Operation Guide](#).
- For more information about the tool, see the [OBS Tool Guide](#).
- You are advised to use an OBS tool, such as OBS Browser+ or obsutil, to upload large files because OBS Console has restrictions on the file size and quantity.
- OBS Browser+ is a graphical tool that provides complete functions for managing your buckets and objects in OBS. You are advised to use this tool to create buckets or upload objects.
- obsutil is a command line tool for accessing and managing OBS resources. If you are familiar with command line interface (CLI), obsutil is recommended as an ideal tool for batch processing and automated tasks.

You can upload files to buckets in different ways. Then, OBS stores the files in the buckets as objects.

Table 4-1 Access modes of objects uploaded to OBS

Access Mode	Upload Method
Console	Uploading an object using OBS Console
OBS Browser+	Uploading an object using OBS Browser+
obsutil	Uploading an object using obsutil
SDK	Uploading an object using SDKs . For details, see the section about object uploading in the developer guide of each language.
API	Uploading objects - PUT and Uploading objects - POST

Step 2: Create a Queue

If you submit a Spark job for the first time, you need to create a queue first. For example, create a queue named **sparktest** and set **Queue Type** to **General Queue**.

1. Log in to the DLI management console.

NOTE

If this is your first time logging in to the DLI management console, you need to be authorized to access OBS.

For this example, you need at least the **Tenant Administrator (Global service)** permission.

2. In the navigation pane of the DLI management console, choose **Resources > Queue Management**.
3. In the upper right corner of the **Queue Management** page, click **Buy Queue** to create a queue.
4. Create a queue, name it **sparktest**, and set the queue usage to for general purpose. For details, see [Creating a Queue](#).

Billing Mode: Yearly/Monthly | Pay-per-use

Region: [Dropdown]

Name: sparktest

Queue Usage: For SQL | For general purpose

Dedicated Resource Mode: [Unselected]

CPU Architecture: x86 | Kunpeng

CU Specifications: 16 CUs (Test) | 64 CUs (Production) | 256 CUs (Production) | 512 CUs (Production)

Enterprise Project: default

Description: [Text Area]

Tags: [Form]

5. Click **Buy** to confirm the configuration.
6. Confirm the configuration and click **Submit**.

Step 3: Create a Package

Before submitting a Spark job, you need to create a package, for example, **spark-examples.jar**.

1. In the navigation pane on the left of the DLI console, choose **Data Management > Package Management**.
2. On the **Package Management** page, click **Create** in the upper right corner to create a package.
3. In the **Create Package** dialog box, set **Type** to **JAR**, **OBS Path** to the path of the spark-examples.jar package in [Step 1: Upload Data to OBS](#), and **Group** to **Do not use**.

Create Package

Type: JAR | PyFile | File | ModelFile

OBS Path: obs://dli-test-obs01/spark-examples.jar

Group: Use existing | Use new | Do not use

Tags: [Form]

OK | Cancel

4. Click **OK**.
You can view and select the package on the **Package Management** page.

For details about how to create a package, see [Creating a Package](#).

Step 4: 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**.
2. On the Spark job editing page, set **Queues** to the queue created in [Step 2: Create a Queue](#) and **Application** to the package created in [Step 3: Create a Package](#).

For details about other parameters, see the description of the Spark job editing page in [Creating a Spark Job](#).

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.

The screenshot shows the Spark job editing interface. At the top right, there is a red button labeled 'Execute'. Below it, the 'Select a Queue' dropdown menu is highlighted with a red box. Under the 'Job Configurations' section, the 'Application' dropdown menu is also highlighted with a red box. Other fields include 'Job Name (--name)', 'Main Class (--class)', 'Application Parameters', and 'Spark Arguments (--conf)'.

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.

5 Creating and Submitting a Flink OpenSource SQL Job

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 input stream and RDS as the output stream.

Procedure

You need to create a Flink OpenSource SQL job that has an input stream and an output stream. The input stream reads data from Kafka, and the output stream writes data into RDS. The procedure is as follows:

[Step 1: Prepare a Data Source](#)

[Step 2: Prepare a Data Output Channel](#)

[Step 3: Create an OBS Bucket to Store Output Data](#)

[Step 4: Create a Queue](#)

[Step 5: Create an Enhanced Datasource Connection Between DLI and Kafka](#)

[Step 6: Create an Enhanced Datasource Connection Between DLI and RDS](#)

[Step 7: Create a Flink OpenSource SQL Job](#)

Step 1: Prepare a Data Source

In this example, Kafka is the data source.

For more information about Flink job data, see [Preparing Flink Job Data](#).

Enable DIS to import Kafka data to DLI. For details, see [Create a Kafka Instance](#).

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 the *Virtual Private Cloud User Guide*. For details about how to create and use a

subnet in an existing VPC, see [Creating a subnet for the VPC](#) in the *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 [Preparing Required Resources](#) in *Distributed Message Service for Kafka User Guide*.

2. Create a DMS for Kafka Instance for job input streams.
 - a. Log in to the DMS for Kafka 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 default values for other parameters.

Step 2: Prepare a Data Output Channel

To use RDS as the data output channel, create an RDS MySQL instance. For details, see "Getting Started with RDS for MySQL" in [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:** Keep the default value.
 - **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**.
 - **Read Replica:** Select **Skip**.
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 step, you need to enable OBS for **JobSample** and enable checkpointing for the DLI Flink job to save job logs and test data storage.

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**
 - **Tags:** Leave it blank.
3. Click **Create Now**.

Step 4: Create a Queue

Flink OpenSource SQL jobs cannot run on the default queue. You need to create a queue, for example, **Flinktest**. For details, see [Creating a Queue](#).

1. Log in to the DLI management console. On the **Overview** page, click **Buy Queue** in the upper right corner.

If this is your first time logging in to the DLI management console, you need to be authorized to access OBS.
2. Configure the following parameters:
 - **Billing Mode:** **Pay-per-use**
 - **Region:** Retain the default value.
 - **Project:** Retain the default value or select one as you need.
 - **Name:** **Flinktest**
 - **Type:** **For general purpose**. Select **Dedicated Resource Mode**.
 - **AZ Mode:** **Single AZ**
 - **Specifications:** **16 CUs**
 - **Enterprise Project:** **default**
 - **Description:** Leave it blank.
 - **Advanced Settings:** **Custom**

- **CIDR Block:** Set a CIDR block that does not conflict with the Kafka instance's CIDR block.
 - **Tags:** Leave it blank.
3. Click **Buy** and confirm the configuration.
 4. Submit the request.

It takes 10 to 15 minutes to bind the queue to a cluster after the queue is created.

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 **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 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 name of the queue created in [Step 4: Create a Queue](#).
 - **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 a Queue](#). In the **Operation** column, choose **More > 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 a Queue](#).
 - **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 a Queue](#). In the **Operation** column, choose **More > Test Address Connectivity**.
4. In the displayed dialog box, enter *floating IP address:database port* of the RDS DB instance in the **Address** box and click **Test** to check whether the database is reachable.

----End

Step 7: Create a Flink OpenSource SQL Job

After the data source and data output channel 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: Flinktest**
 - **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 move the cursor over the status icon to view the error details. You can click  to copy these details. After handling the fault based on the provided information, 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 order;
```

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
```

6 Creating an Enhanced Datasource Connection to RDS

Scenario

This example creates an enhanced datasource connection for a Spark SQL job to access RDS database tables.

Procedure

Create an enhanced datasource connection to access RDS. In this example, you need to create an RDS MySQL instance, an RDS database table, and a DLI enhanced datasource connection, and then access RDS database tables through a Spark SQL job. The procedure is as follows:

[Step 1: Create an RDS MySQL Instance](#)

[Step 2: Create an RDS Database Table](#)

[Step 3: Create a Queue](#)

[Step 4: Create an Enhanced Datasource Connection](#)

[Step 5: Submit a SQL Job](#)

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 console.
2. In the upper left corner of the management console, select the target region and project.
3. On the **Instance Management** page, click **Buy DB Instance**.
4. On the displayed page, select a billing mode and configure information about your DB instance. Then, click **Next**.

Set basic information as follows:

- **Billing Mode:** Pay-per-use
- **Region:** The region where your RDS resources will be located. You can change it on the creation page, or go back to the **Instance Management** page and change it in the upper left corner.
- **DB Instance Name:** Retain the default value.

- **DB Engine:** MySQL
- **DB Engine Version:** 8.0
- **DB Instance Type:** Single
- **Storage Type:** Cloud SSD
- **AZ:** default value
- **Time Zone:** default value
- **Instance Class:** default value
- **Storage Space:** default value
- **Disk Encryption:** Disable
- **VPC:** A dedicated virtual network in which your RDS DB instances are located. For details about how to create a VPC and subnet, see **Creating a VPC** in the *Virtual Private Cloud User Guide*. If you need to create and use a subnet in an existing VPC, see **Creating a Subnet for the VPC** in the *Virtual Private Cloud User Guide*.

NOTE

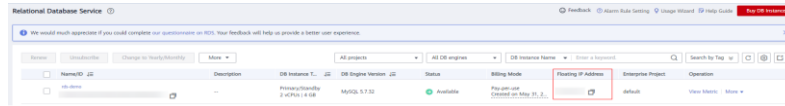
- The created VPC and the DB instance must be in the same region.
 - Retain the default settings unless otherwise specified.
 - **Database Port:** The database port of the read replica and the primary DB instance must be the same.
 - **Security Group:** For details about how to create a security group, see [Creating a Security Group](#). For details about how to add rules to a security group, see [Adding a Security Group Rule](#).
 - **Password: Configure**
 - **Administrator: root**
 - **Administrator Password:** The password can contain 8 to 32 characters and contain at least three types of the following: uppercase letters, lowercase letters, digits, and special characters (~!@#\$%^_+=()?,&).
 - **Confirm Password:** Must be the same as **Administrator Password**.
 - **Parameter Template:** default values
 - **Table Name:** Set the table names case insensitive.
 - **Enterprise Project:** default
 - **Tag:** Skip this parameter.
 - **Required Duration:** Select a duration as needed.
 - **Quantity:** Set to 1.
 - **Read Replica:** Skip
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 console.
2. In the upper left corner of the management console, select the target region and project.

- On the **Instance Management** page, locate the created DB instance and view its floating IP address.

Figure 6-1 Viewing the floating IP address



- Click **More > Log In** in the **Operation** column. 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 6-2 RDS console



Figure 6-3 Logging in to an Instance

Instance Login Information

DB Instance Name **rds-demo** DB Engine Version

* Login Username

* Password ✔ Connection is successful.

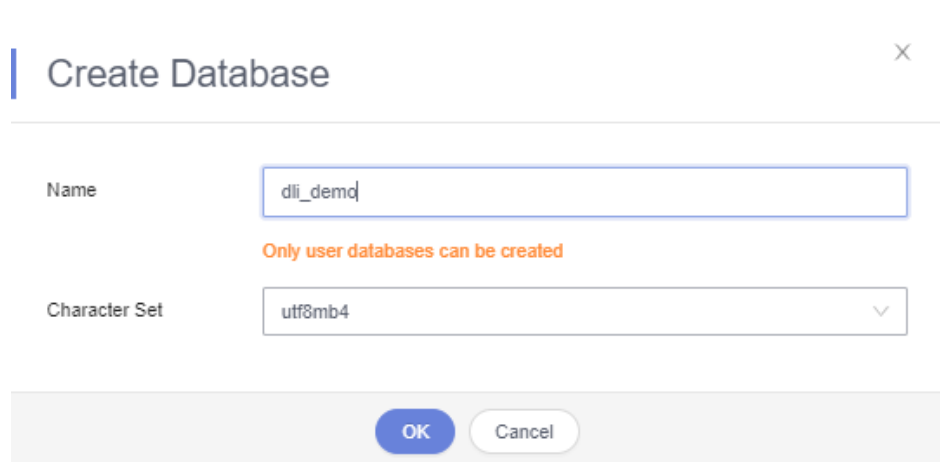
Remember Password Select to remember your password in an encrypted form. Otherwise, the metadata collection function cannot be enabled.

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.

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

Figure 6-4 Creating a database

The screenshot shows a 'Create Database' dialog box with the following fields and options:

- Name:** dli_demo
- Character Set:** utf8mb4
- Warning:** Only user databases can be created
- Buttons:** OK, Cancel

6. Choose **SQL Operation > SQL Query** and run the following SQL statement to create a MySQL table for test:

```
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 a Queue

To run a DLI SQL job for datasource connections, you cannot use the existing default queue. Instead, you need to create a SQL queue, for example, a queue named **test**. For details, see [Creating a Queue](#).

1. Go to the DLI console.
2. On the **Overview** page of the DLI management console, click **Buy Queue** in the upper right corner.
3. Configure parameters.
 - **Billing Mode:** Pay-per-use
 - **Region:** Retain the default value.
 - **Project:** Retain the default value or select one as you need.
 - **Name:** test
 - **Queue Usage:** Select **For SQL** and select **Dedicated Resource Mode**.
 - **AZ Mode:** Single AZ
 - **Specifications:** 16 CUs
 - **Enterprise Project:** default
 - **Description:** Leave it blank.
 - **Advanced Settings:** Custom
 - **CIDR Block:** The configured CIDR block cannot conflict with the RDS subnet CIDR block.
 - **Queue Type:** Basic

- **Tags:** Leave it blank.
4. Click **Buy** to confirm the configuration.
 5. Submit the request.

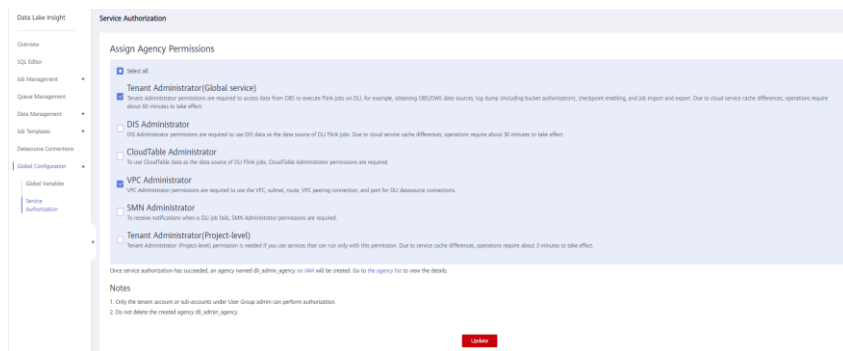
Step 4: Create an Enhanced Datasource Connection

For details about datasource connections, see **Datasource Connections > Enhanced Datasource Connections** in the *Data Lake Insight User Guide*.

NOTE

- The enhanced datasource connection function supports 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 datasource connection table, you need to use the queue for which a datasource connection has been created.
1. In the navigation pane of the DLI management console, choose **Resources > Queue Management**. The created SQL queue **test** is displayed in the queue list.
 2. In the navigation pane of the DLI management console, choose **Global Configuration > Service Authorization**. On the displayed page, select **VPC Administrator**, and click **Update** to grant the DLI user the permission to access VPC resources. The permission is used to create a VPC peering connection.

Figure 6-5 Updating agency permissions



3. In the navigation pane of the DLI management console, choose **Datasource Connections**.
4. Click the **Enhanced** tab and click **Create** in the upper left corner. Set the following parameters:
 - **Connection Name:** dlirds
 - **Resource Pool:** test

NOTE

If you are not sure about the queue to be bound when creating an enhanced datasource connection, you do not need to bind it. After you create the datasource connection, click **More** in its **Operation** column and select **Bind Queue** to bind it to a queue. For details, see [Binding a Queue](#).

- **VPC:** Select the VPC where MySQL DB instance is located.

NOTE

On the **Instance Management** page of the RDS console, click the target instance. On the displayed page, choose **Connection Information > VPC** to obtain the VPC information.

- **Subnet:** Select the subnet where the MySQL DB instance is located.
- **Host Information:** (Optional) When connecting to the HBase cluster of MRS, enter the host name and IP address of the ZooKeeper instance.

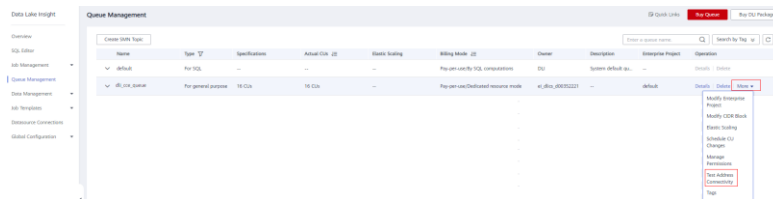
NOTE

On the **Instance Management** page of the RDS console, click the target DB instance. On the displayed page, choose **Connection Information > Subnet** to obtain the subnet information.

- **Tags:** Tags are used to identify cloud resources. A tag includes the tag key and tag value.

5. Click **OK**.
6. 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.
7. Test the connectivity between the queue and the DB instance.
 - a. On the **Queue Management** page, locate the target queue. In the **Operation** column, click **More > Test Address Connectivity**.

Figure 6-6 Testing address connectivity



- b. Enter the floating IP address of the DB instance.

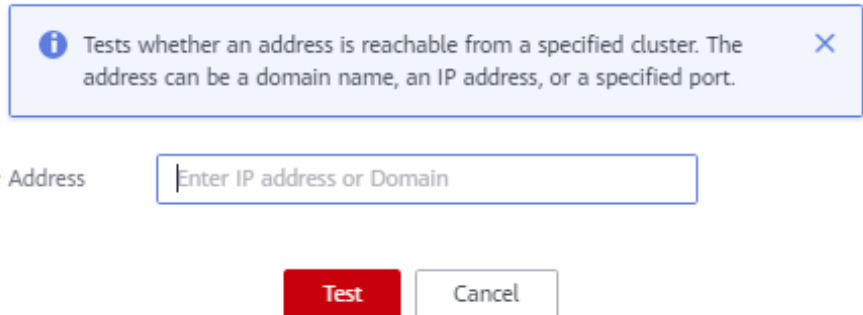
NOTE

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 address is reachable, the queue and the DB instance are successfully connected over the network.

Figure 6-7 Test result

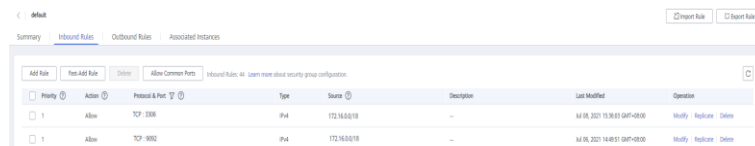
Test Address Connectivity



If the test result shows that the instance is unreachable, modify the security group rules of the VPC to which the instance belongs. The operations are as follows:

- On the DLI management console, click **Resources > Queue Management**, select the bound queue, and click the arrow next to the queue name to view the network segment information of the queue.
- On the **Instance Management** page of the RDS console, click the instance name. In the **Connection Information** area, locate **Database Port** to obtain the port number of the RDS DB instance.
- In the **Connection Information** area, locate the **Security Group** and click the security group name to go to the management page. Click the **Inbound Rules** tab and click **Add Rule**. Set the protocol to **TCP**, the port to the RDS DB instance port, and **Source** to the CIDR block of the DLI queue. Click **OK**.

Figure 6-8 VPC security group rule



- After the configuration is complete, test the network connectivity again.

Step 5: Submit a SQL Job

This section uses a SQL job as an example to describe how to access an RDS table using a datasource connection. For sample code, see **Using the Spark Job to Access Data Sources of Datasource Connections > Connecting to RDS** in the *Data Lake Insight Development Guide*.

For details, see [Creating and Submitting a Spark SQL Job](#).

1. On the DLI management console, click **SQL Editor** in the navigation pane on the left. The **SQL Editor** page is displayed.
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 **test** 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  
'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.  
)  
  
insert into rds_test VALUES ('123','abc');  
  
SELECT * from rds_test;
```

7 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 7-1 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).
	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	Use a Flink OpenSource SQL job to read data from MySQL CDC and write the data to

Scenario	Instructions	Description
	Writing Data to GaussDB(DWS)	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.
	Using Flink Jar to Read and Write Data from and to DIS	Use a Flink Jar job to read and write data from and to DIS.
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.

A Change History

Released On	What's New
2023-07-12	<ul style="list-style-type: none">Added Practices.
2023-01-10	<ul style="list-style-type: none">Adjusted the document structure and added Registering a Huawei ID and Enabling Huawei Cloud Services.
2022-07-28	<ul style="list-style-type: none">Added Creating and Submitting a Flink OpenSource SQL Job.Deleted "Creating and Submitting a Flink SQL Job" for Flink SQL EOS.