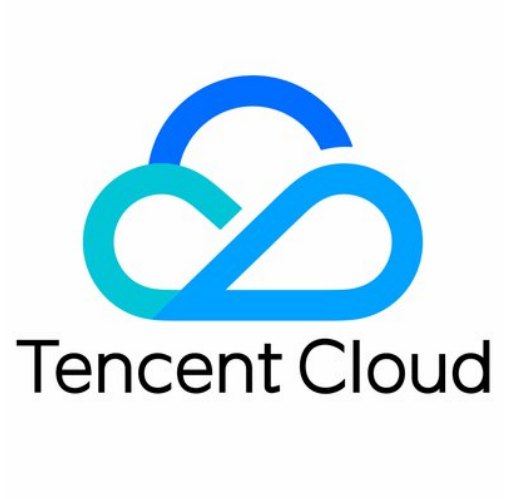
Data Lake Compute

Client Access

Product Documentation



Contents

Client Access
JDBC Access
DLC JDBC Access

Last updated：2024-08-07 17:36:25

Environment Preparation

Dependency: JDK 1.8

JDBC driver download: Click here to download the JDBC driver

Connecting to DLC

1. Load the JDBC driver for DLC.

```
Class.forName("com.tencent.cloud.dlc.jdbc.DlcDriver");
```

2. Create a connection using DriverManager.

```
Connection cnct = DriverManager.getConnection(url, secretId, secretKey);
```

URL Format

```
jdbc:dlc:<dlc_endpoint>?task_type=SQLTask&database_name=abc&datasource_connection_n
```

Description of parameters in the JDBC access URL:

Parameter	Required	Description
dlc_endpoint	Yes	Endpoint of the DLC service. The value is fixed at dlc.tencentcloudapi.com.
datasource_connection_name	Yes	Data source connection name, corresponding to the DLC data directory.
task_type	Yes	Task type Fill in SQLTask for the Presto engine. Fill in SparkSQLTask for the SparkSQL engine.

		Fill in BatchSQLTask for the Spark engine.
database_name	No	Database name
region	Yes	Region supported by the DLC service, including: ap-nanjing, ap-beijing, ap-guangzhou, ap-shanghai, ap-chengdu, ap-chongqing, na-siliconvalley, ap-singapore, and ap-hongkong.
data_engine_name	Yes	Data engine name
secretId	Yes	SecretId managed by Tencent Cloud KMS
secretKey	Yes	Secretkey managed by Tencent Cloud KMS
result_type	No	Default value: Service. You can set the value to COS if you demand a higher speed of obtaining results. Service: Obtain results through the DLC API. COS: Obtain results from the COS client.
read_type	No	Stream: Obtain results from COS via streams. DownloadSingle: Obtain results by downloading a single file to a local path. Default value: Stream. This value takes effect only when result_type is set to COS. The file is downloaded to the temporary directory /tmp. Make sure that the read and write permissions on this directory are granted. The data will be automatically deleted after it is read.

Querying

```
Statement stmt = cnct.createStatement();ResultSet rset = stmt.executeQuery("SELECT
while (rset.next())
 {// process the results
}
rset.close();
stmt.close();
conn.close();
}
rset.close();
stmt.close();
conn.close();
```

Statement Support

Currently, the statement that can be used by the JDBC driver is consistent with the standard statement of DLC.

Sample Code

Database and Table Operation

```
import java.sql.*;
public class MetaTest {
        public static void main(String[] args) throws SQLException {
            try {
                Class.forName("com.tencent.cloud.dlc.jdbc.DlcDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return;
    }
    Connection connection = DriverManager.getConnection(
                    "jdbc:dlc:<dlc_endpoint>?task_type=<task_type>&database_name=<d
                    "<secret_id>",
                    "secret_key");
    Statement statement = connection.createStatement();
    String dbName = "dlc_db1";
    String createDatabaseSql = String.format("CREATE DATABASE IF NOT EXISTS %s", db
    statement.execute(createDatabaseSql);
    String tableName = "dlc_t1";
    String wholeTableName = String.format("%s.%s", dbName, tableName);
    String createTableSql =
    String.format(
    "CREATE EXTERNAL TABLE %s (  "
            + "  id string ,   "
            + "  name string ,   "
            + "  status string ,   "
            + "  type string )  "
            + "ROW FORMAT SERDE   "
            + "  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'  "
            + "STORED AS INPUTFORMAT   "
            + "  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'   "
            + "OUTPUTFORMAT   "
            + "  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat' "
            + "LOCATION\\\\n"
            + "  'cosn://<bucket_name>/<path>'  ",
            wholeTableName);
    statement.execute(createTableSql);
```

```
    // get meta data
    DatabaseMetaData metaData = connection.getMetaData();
    System.out.println("product = " + metaData.getDatabaseProductName());
    System.out.println("jdbc version = "
    + metaData.getDriverMajorVersion() + ", "
    + metaData.getDriverMinorVersion());
    ResultSet tables = metaData.getTables(null, dbName, tableName, null);
    while (tables.next()) {
        String name = tables.getString("TABLE_NAME");
        System.out.println("table: " + name);
        ResultSet columns = metaData.getColumns(null, dbName, name, null);
        while (columns.next()) {
        System.out.println(
        columns.getString("COLUMN_NAME") + "\\\\t" +
        columns.getString("TYPE_NAME") + "\\\\t" +
        columns.getInt("DATA_TYPE"));
    }
    columns.close();
    }
    tables.close();
    statement.close();
    connection.close();
}
}
```

Data Query

```
import java.sql.*;
public class DataTest {
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName("com.tencent.cloud.dlc.jdbc.DlcDriver");
            } catch (ClassNotFoundException e) {
    e.printStackTrace();
    return;
    }
    Connection connection = DriverManager.getConnection(
    "jdbc:dlc:<dlc_endpoint>?task_type=<task_type>&database_name=<database_name>&da
    "<secret_id>",
    "secret_key");
    Statement statement = connection.createStatement();
    String sql = "select * from dlc_test";
    statement.execute(sql);
    ResultSet rs = statement.getResultSet();
    while (rs.next()) {
```

```
            System.out.println(rs.getInt(1) + ":" + rs.getString(2));
            }
            rs.close();
            statement.close();
            connection.close();
            }
    }
```

Database Client

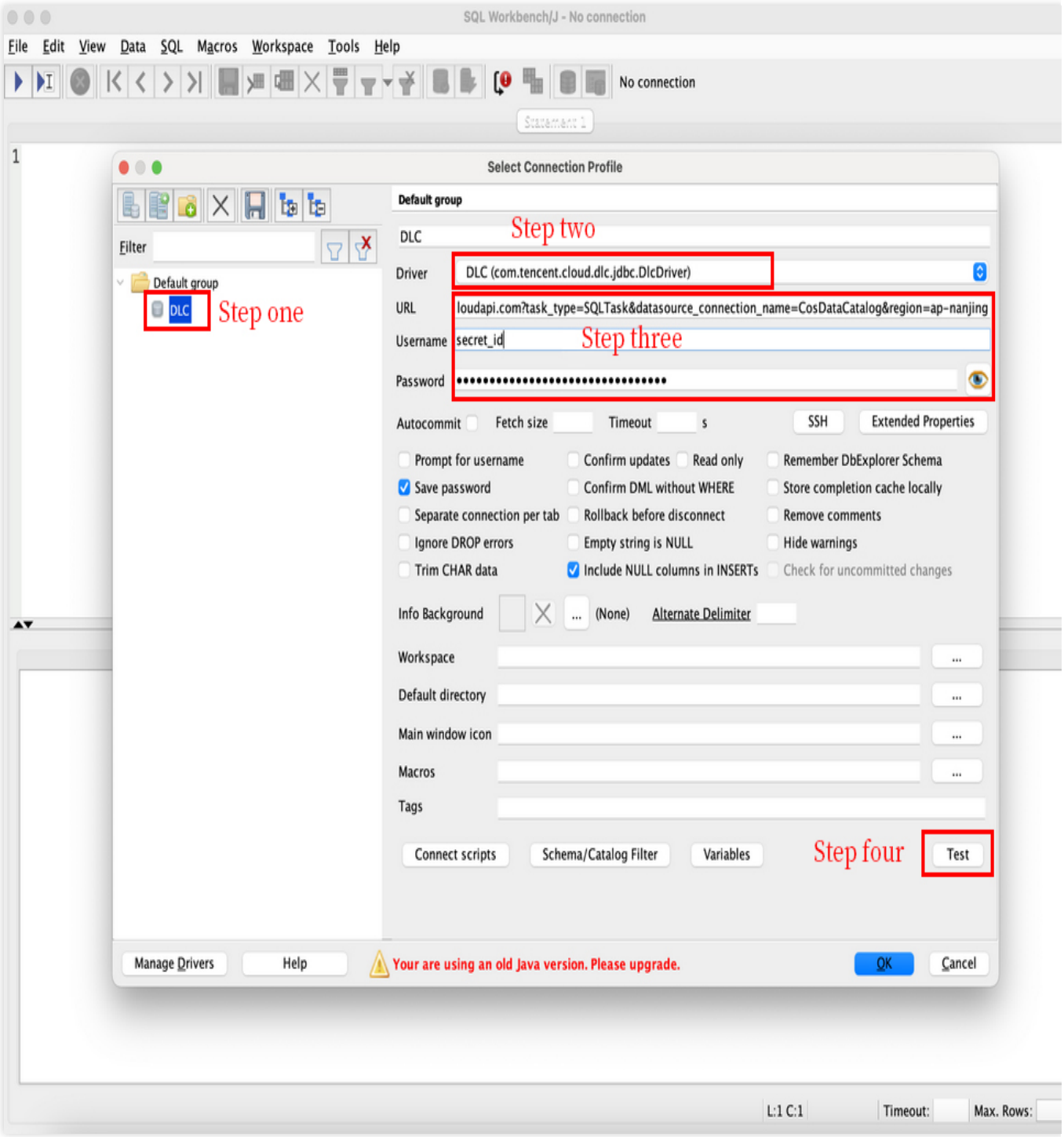
You can load the package of JDBC driver for DLC into the SQL client and then connect the client to the DLC service for querying.

Prerequisites

1. The DLC service has been activated.
2. The JDBC driver package mentioned above has been downloaded.
3. **SQL Workbench/J** has been downloaded and installed.

Directions

1. Create the driver for DLC based on the JDBC driver package.



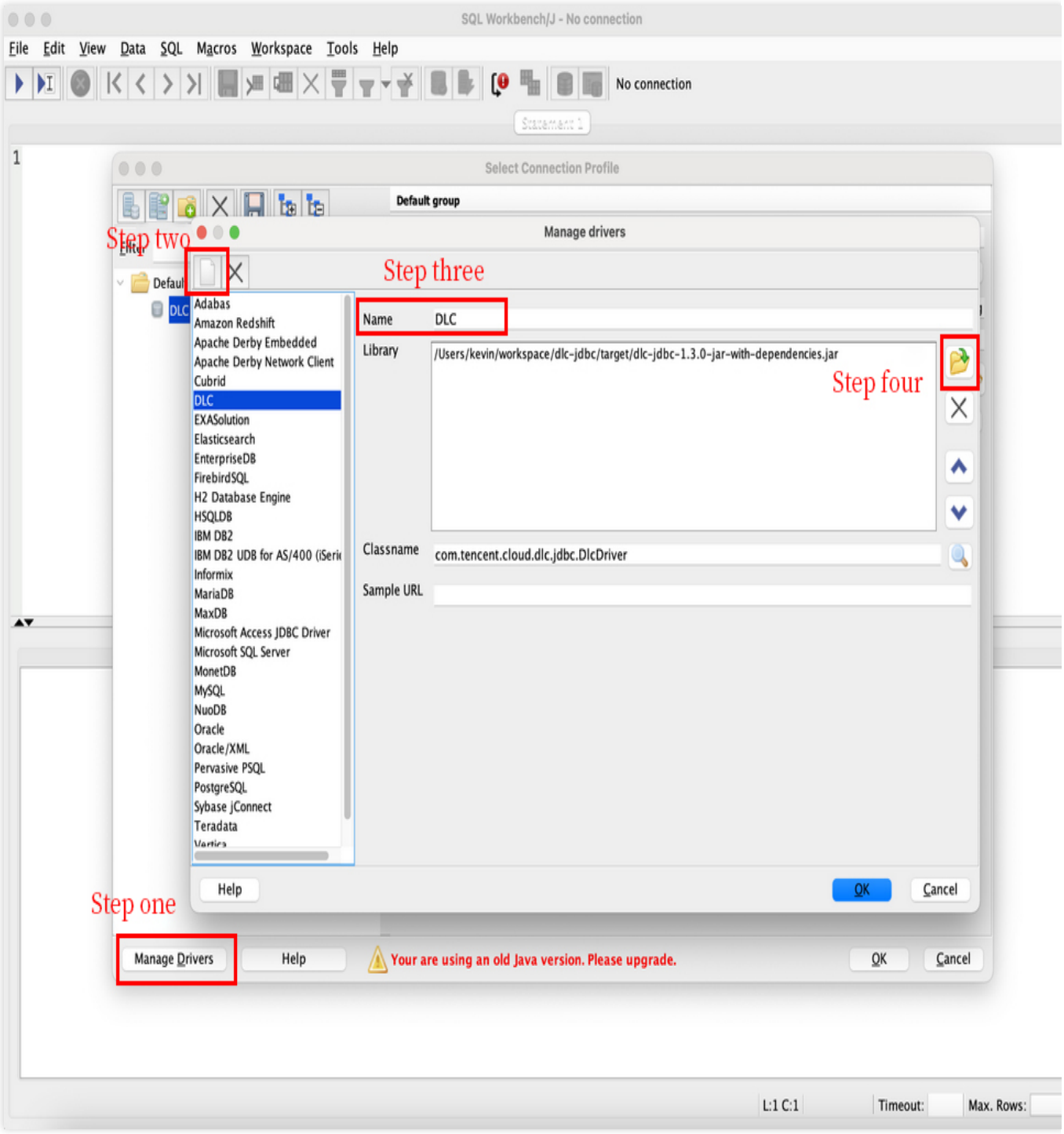
2. Connect to DLC, fill in the following parameters, and click **Test**. After the test passes, the connection to DLC is completed.

Name: Connection name, used to identify the connection to DLC.

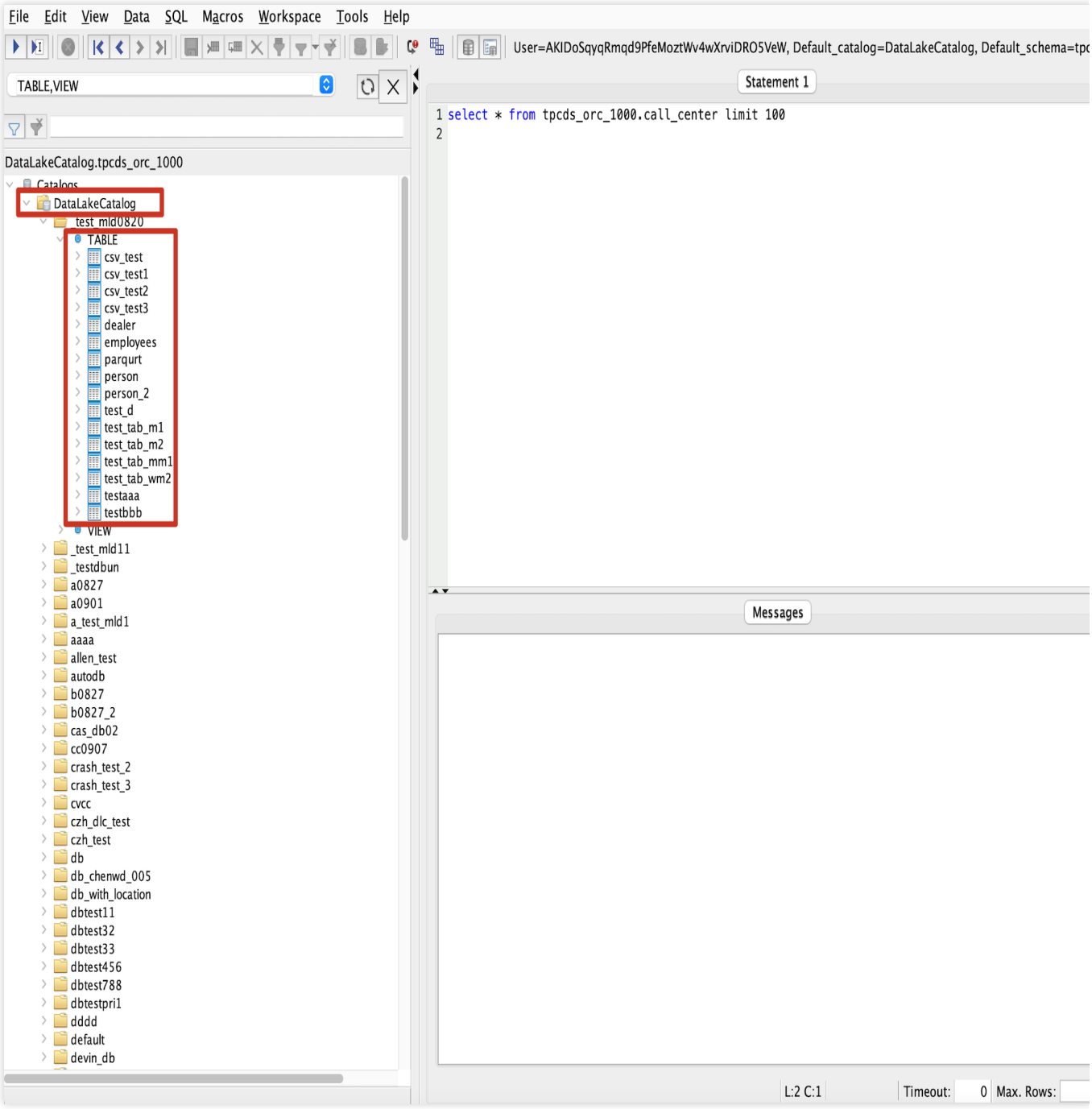
Username: Correspond to the secret_id of the Tencent Cloud user.

Password: Correspond to the secret_key of the Tencent Cloud user.

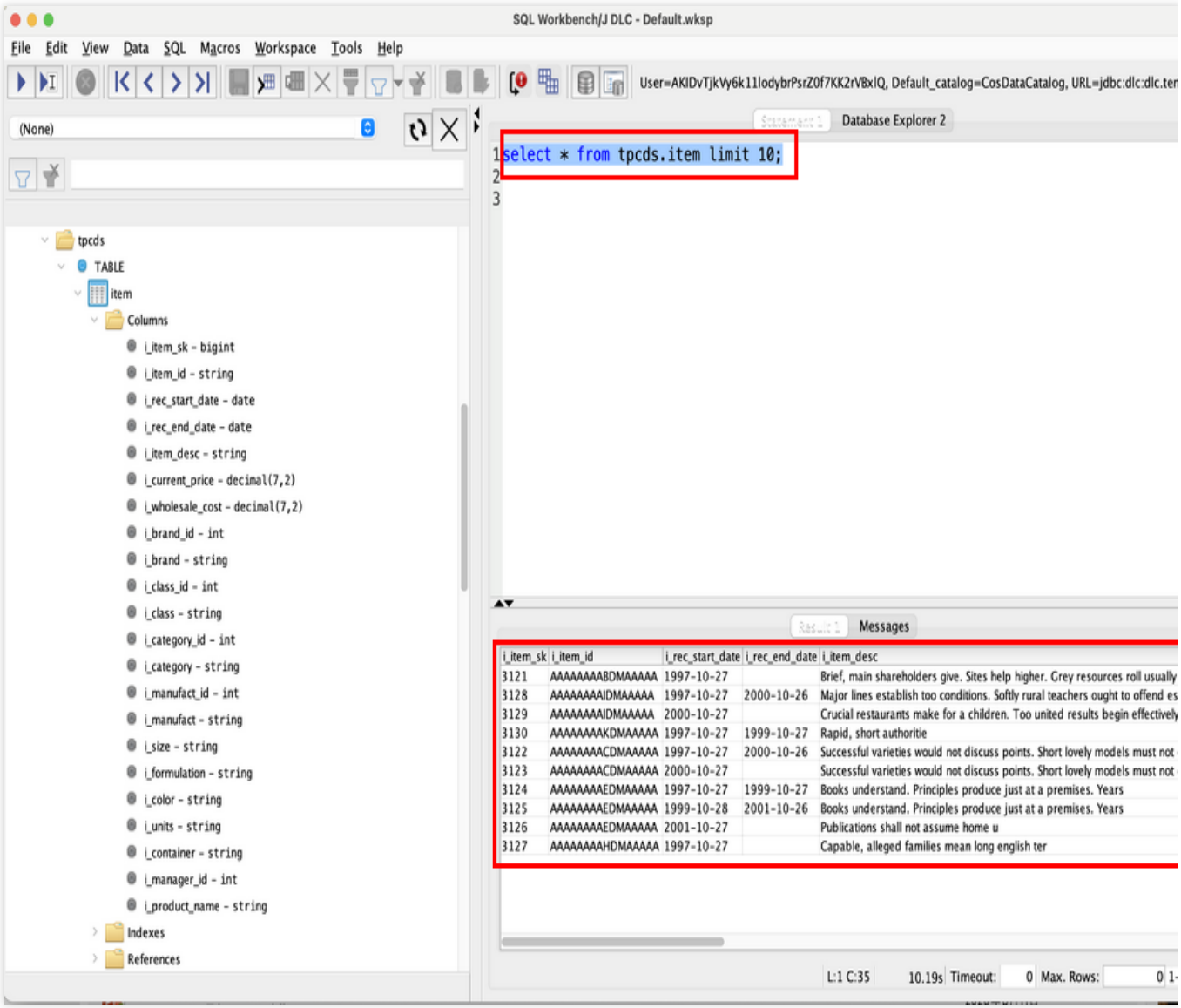
URL: URL used to connect to DLC. The format is the same as the URL for creating a connection through JDBC mentioned above.



3. View the database and table information.



4. Query the data.


SQL Workbench/J DLC - Default.wksp
File Edit View Data SQL Macros Workspace Tools Help
(None)
Database Explorer 2
select * from tpcds.item limit 10;
tpcds
TABLE
item
Columns
i_item_sk - bigint
i_item_id - string
i_rec_start_date - date
i_rec_end_date - date
i_item_desc - string
i_current_price - decimal(7,2)
i_wholesale_cost - decimal(7,2)
i_brand_id - int
i_brand - string
i_class_id - int
i_class - string
i_category_id - int
i_category - string
i_manufact_id - int
i_manufact - string
i_size - string
i_formulation - string
i_color - string
i_units - string
i_container - string
i_manager_id - int
i_product_name - string
Indexes
References
Messages
i_item_sk i_item_id i_rec_start_date i_rec_end_date i_item_desc
3121 AAAAAAAABDMAAAAA 1997-10-27 Brief, main shareholders give. Sites help higher. Grey resources roll usually
3128 AAAAAAAAIDMAAAAA 1997-10-27 2000-10-26 Major lines establish too conditions. Softly rural teachers ought to offend es
3129 AAAAAAAAIDMAAAAA 2000-10-27 Crucial restaurants make for a children. Too united results begin effectively
3130 AAAAAAAAKDMAAAAA 1997-10-27 1999-10-27 Rapid, short authoritie
3122 AAAAAAAACDMAAAAA 1997-10-27 2000-10-26 Successful varieties would not discuss points. Short lovely models must not
3123 AAAAAAAACDMAAAAA 2000-10-27 Successful varieties would not discuss points. Short lovely models must not
3124 AAAAAAAAEDMAAAAA 1997-10-27 1999-10-27 Books understand. Principles produce just at a premises. Years
3125 AAAAAAAAEDMAAAAA 1999-10-28 2001-10-26 Books understand. Principles produce just at a premises. Years
3126 AAAAAAAAEDMAAAAA 2001-10-27 Publications shall not assume home u
3127 AAAAAAAAHDMAAAAA 1997-10-27 Capable, alleged families mean long english ter
L:1 C:35
10.19s
Timeout: 0
Max. Rows: 0

Hive JDBC Access

Last updated：2024-09-04 11:16:44

Supported Engine Types

Standard Spark Engine

Environment Preparation

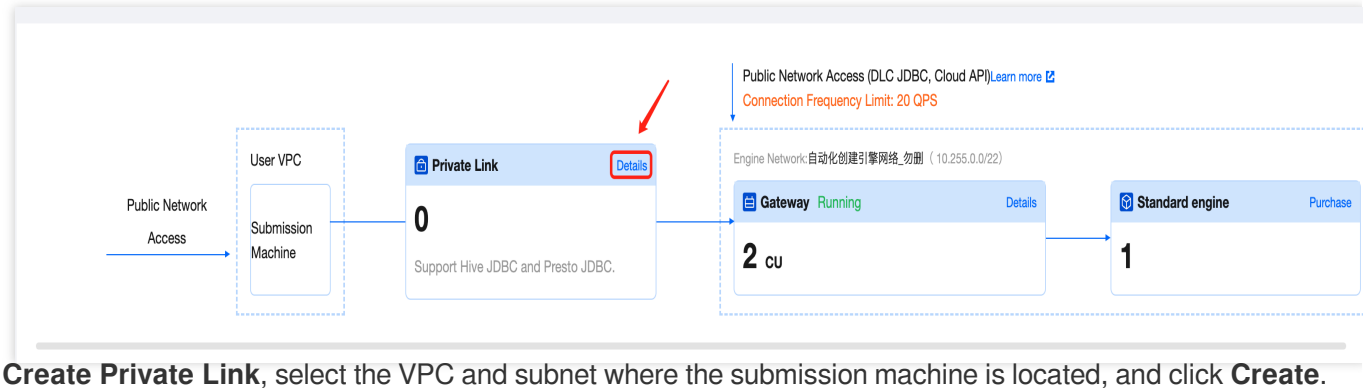
Dependency: JDK 1.8

JDBC download: Click to download hive-jdbc-3.1.2-standalone.jar

Connecting to the Standard Spark Engine

Creating a Service Access Link

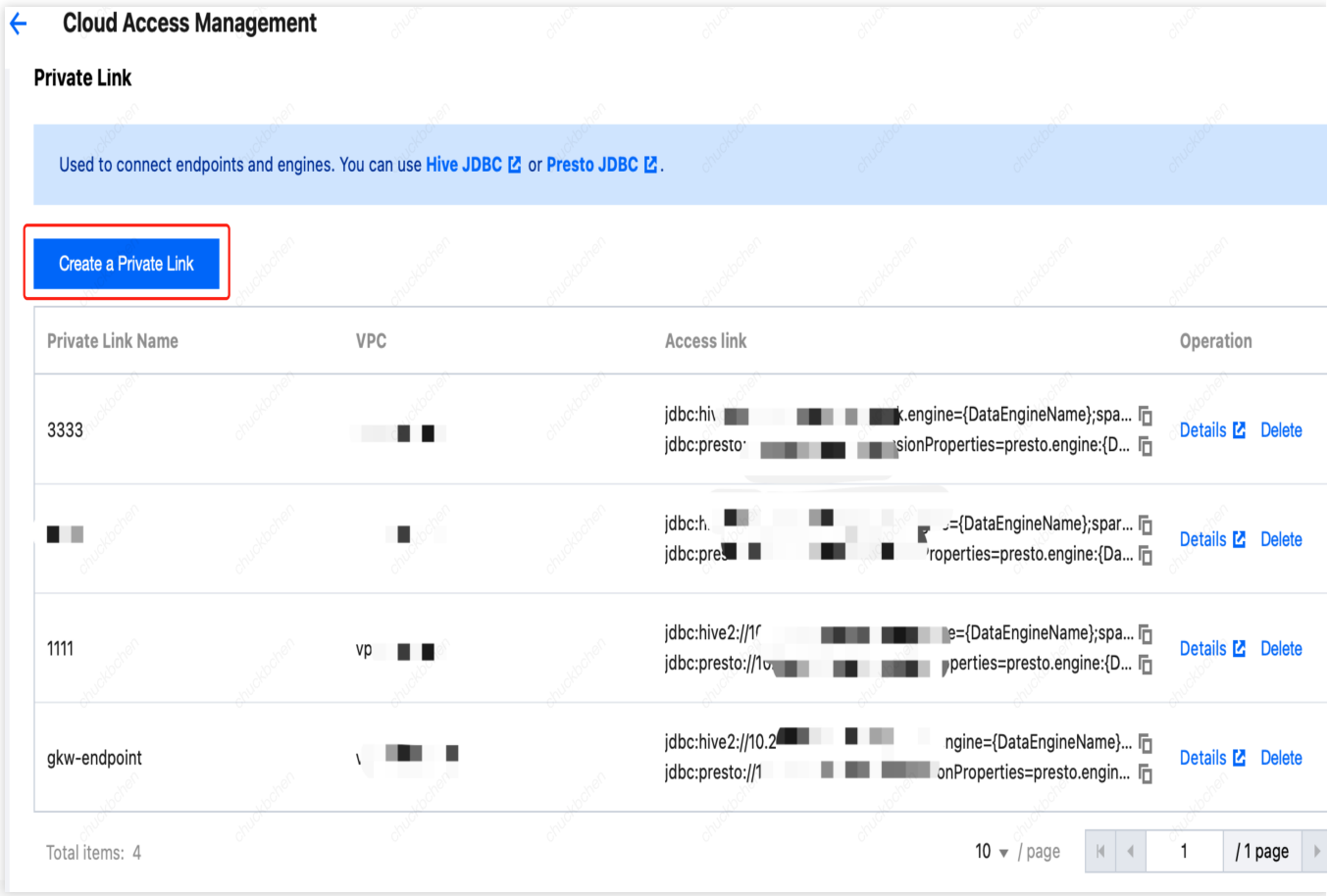
Go to the Standard engine page and click the Details button for the gateway to access the gateway details page:



Click **Create Private Link**, select the VPC and subnet where the submission machine is located, and click **Create**. This will generate two access links, one for the Hive2 protocol and one for the Presto protocol. Use the Hive2 protocol to connect to the Standard Spark Engine, as shown in the diagram below.

Note:

Creating a private connection will establish a network link between the engine network and the selected VPC. The submission machine can be any server within the selected VPC that is accessible and can be used for task submission. If there is no submission machine available in the selected VPC, you can create a new server to serve as the submission machine.




```
jdbc:hive2://{endpoint}:10009/?spark.engine={DataEngineName};spark.resourcegroup={R
```

Loading the JDBC Driver

```
Class.forName("org.apache.hive.jdbc.HiveDriver");
```

Creating a Connection Using DriverManager

```
jdbc:hive2://{endpoint}:10009/?spark.engine={DataEngineName};spark.resourcegroup={R
Properties properties = new Properties();
properties.setProperty("user", {AppId});
Connection cnct = DriverManager.getConnection(url, properties);
```

JDBC Connection String Parameter Descriptions

Parameter	Required	Description
spark.engine	Yes	The name of the Standard Spark Engine
spark.resourcegroup	No	The name of the Standard Spark Engine resource group. If it is not specified, temporary resources will be created.

secretkey	Yes	The SecretKey from Tencent Cloud API Key Management
secretid	Yes	The SecretId from Tencent Cloud API Key Management
region	Yes	The region. Currently, DLC services support the following: ap-nanjing, ap-beijing, ap-beijing-fsi, ap-guangzhou,ap-shanghai, ap-chengdu,ap-chongqing, na-siliconvalley, ap-singapore, ap-hongkong, na-ashburn, eu-frankfurt, ap-shanghai-fsi
kyuubi.engine.type	Yes	Must be set to: SparkSQLTask
kyuubi.engine.share.level	Yes	Must be set to: ENGINE
user	Yes	The user's APPID

Complete Example of Data Query

```
import org.apache.hive.jdbc.HiveStatement;
import java.sql.*;
import java.util.Properties;

public class TestStandardSpark {
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName("org.apache.hive.jdbc.HiveDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return;
        }
        String url = "jdbc:hive2://{endpoint}:10009/?spark.engine={DataEngineName};
        Properties properties = new Properties();
        properties.setProperty("user", {AppId});
        Connection connection = DriverManager.getConnection(url, properties);
        HiveStatement statement = (HiveStatement) connection.createStatement();
        String sql = "SELECT * FROM dlc_test LIMIT 100";
        statement.execute(sql);
        ResultSet rs = statement.getResultSet();
        while (rs.next()) {
            System.out.println(rs.getInt(1) + ":" + rs.getString(2));
        }
        rs.close();
        statement.close();
        connection.close();
}
}
```

After compilation is completed, you can upload the JAR file to the submission machine for execution.

Presto JDBC Access

Last updated：2024-09-04 11:16:59

Supported Engine Types

Standard Presto Engine

Environment Preparation

Dependency: JDK 1.8

JDBC download: Click to download presto-jdbc-0.284.jar

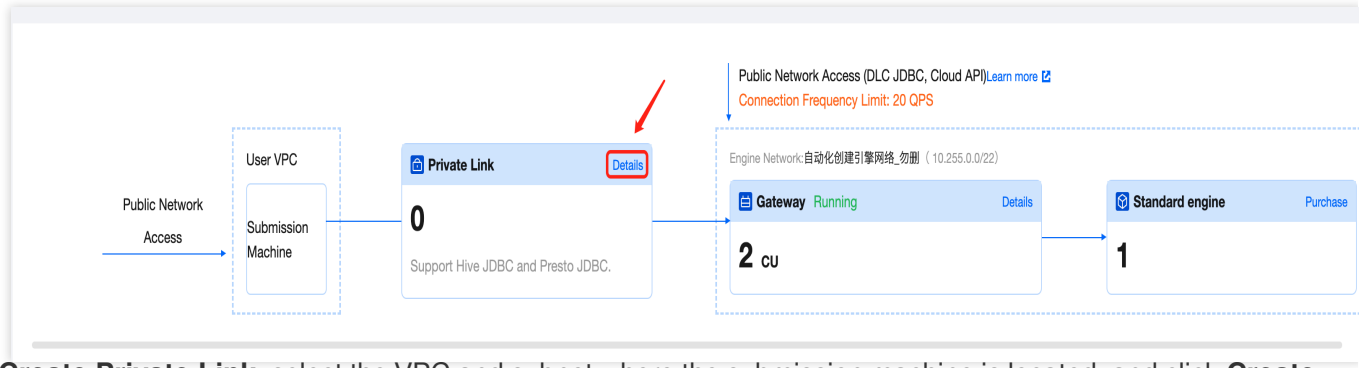
Note:

Currently, only the Standard Presto Engine supports access using the Presto JDBC. If you are using the SuperSQL Engine, see the DLC JDBC Access.

Connecting to the Standard Presto Engine

Creating a Service Access Link

Go to the Standard engine page, and click the Details button for the gateway to enter the gateway details page.

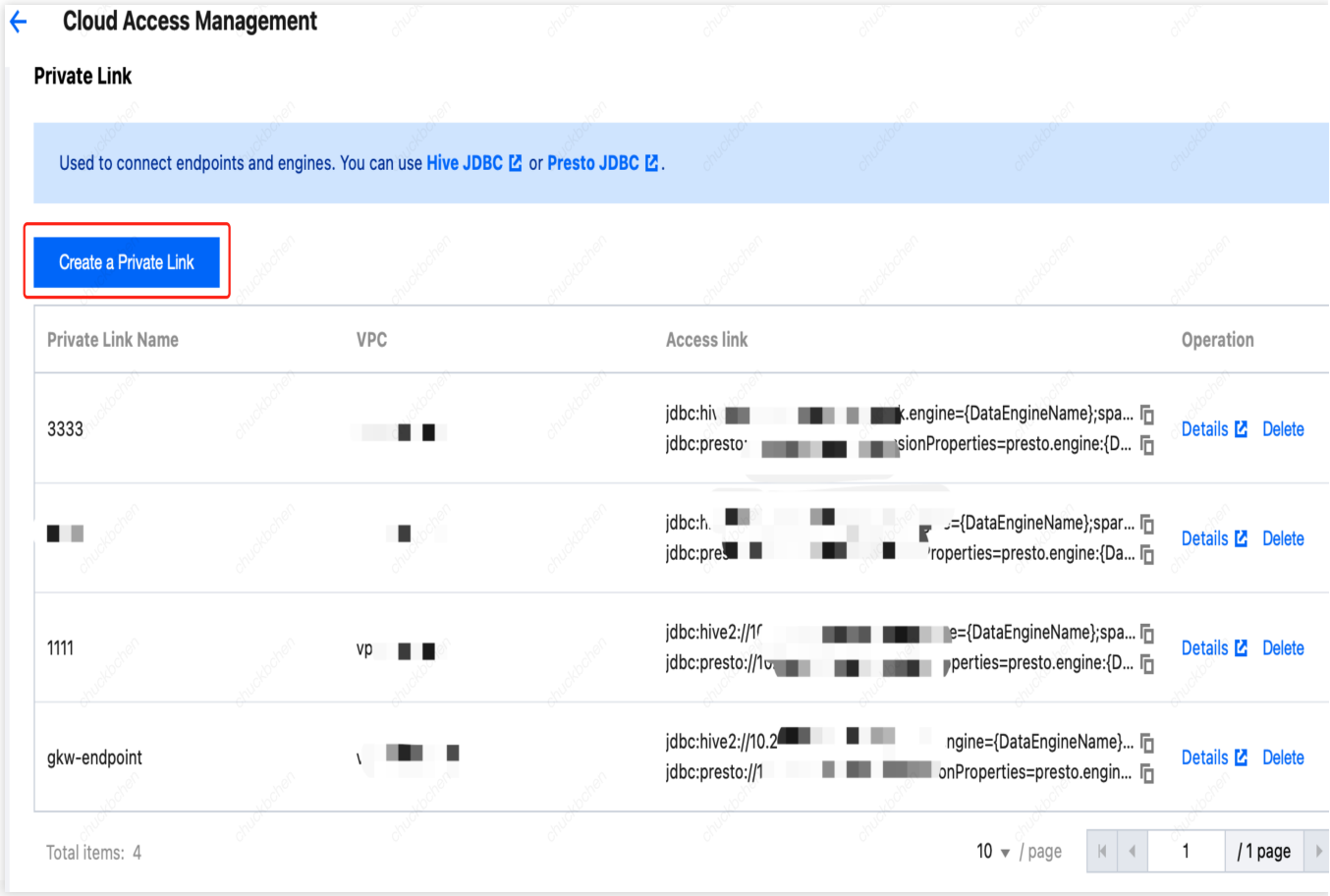


Click **Create Private Link**, select the VPC and subnet where the submission machine is located, and click **Create**. This will generate two access links, one for the Hive2 protocol and one for the Presto protocol. Use the Presto protocol to connect to the Standard Presto Engine, as shown in the diagram below.

Note:

Creating a private connection will establish a network link between the engine network and the selected VPC. The submission machine can be any server within the selected VPC that is accessible and used solely for task submission.

If there is no submission machine available in the selected VPC, you can create a new server to serve as the submission machine.




```
jdbc:presto://{endpoint}:10999/?sessionProperties=presto.engine:{DataEngineName};re
```

Loading the JDBC Driver

```
Class.forName("com.facebook.presto.jdbc.PrestoDriver");
```

Creating a Connection Using DriverManager

```
String url = "jdbc:presto://{endpoint}:10999/?sessionProperties=presto.engine:{Data
Properties properties = new Properties();
properties.setProperty("user", {AppId});
Connection connection = DriverManager.getConnection(url, properties);
```

JDBC Connection String Parameter Descriptions

Parameter	Required	Description

presto.engine	Yes	The name of the Standard Presto Engine
database	Yes	The name of the database
secretkey	Yes	The SecretKey from Tencent Cloud API Key Management
secretid	Yes	The SecretId from Tencent Cloud API Key Management.
region	Yes	The region. Currently, DLC services support the following: ap-nanjing, ap-beijing, ap-beijing-fsi, ap-guangzhou, ap-shanghai, ap-chengdu, ap-chongqing, na-siliconvalley, ap-singapore, ap-hongkong, na-ashburn, eu-frankfurt and ap-shanghai-fsi.
catalog	Yes	The name of the data catalog
extraCredentials	Yes	SecretKey: The SecretKey from Tencent Cloud API Key Management
		SecretId: The SecretId from Tencent Cloud API Key Management
user	Yes	The user's APPID

Complete Example of Data Query

```
import com.facebook.presto.jdbc.PrestoStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

public class TestJDBCKyuubiPresto {
    public static void main(String[] args) throws SQLException {
        try {
            Class.forName("com.facebook.presto.jdbc.PrestoDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return;
        }
        String url = "jdbc:presto://{endpoint}:10999/?sessionProperties=presto.engi
        Properties properties = new Properties();
        properties.setProperty("user", {AppId});
        Connection connection = DriverManager.getConnection(url, properties);
        PrestoStatement statement = (PrestoStatement) connection.createStatement();
        String sql = "show catalogs";
        statement.execute(sql);
        ResultSet rs = statement.getResultSet();
        while (rs.next()) {
```

```
            System.out.println(rs.getString(1));
        }
        rs.close();
        statement.close();
        connection.close();
    }
```

After compilation is completed, you can upload the JAR file to the submission machine for execution.

Configuring Public Access for Standard Engine

Last updated：2025-06-18 15:08:29

By binding a public IP to the executor, you can access the Standard Engine from the public network.

Step 1: Creating an Endpoint

1. Log in to the DLC console, click **Standard Engine** from the left-side menu to enter the Standard Engine interface, and then click the Gateway Details button to enter the details page.

On the CAM details page, click the Create Private Link button:

2. In the Create Private Link page, select the target VPC and target subnet. The target VPC and subnet should be chosen from a VPC within your account that is used for communication between users, access points, and the engine.

Step 2: Creating an Executor and Enabling Public Access

If you do not have an executor, first purchase a CVM to serve as the executor.
1. Go to the Tencent Cloud CVM. In the appropriate region, click **Create**.

2. The executor needs to be created within the subnet of the private network that was connected in Step 1:
Assign a public IP to the executor.
Ensure that the selected security group allows traffic from the specific subnet and ports required.

Step 3: Binding a Public IP to the Executor

If you are using an existing CVM as the executor but it does not have an Elastic Public IP (EIP), you can create a EIP and bind it to the executor created in Step 2:

Step 4: Modifying the Executor's iptables

1. Log in to the executor and execute the following commands on the machine:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
iptables -t nat -A PREROUTING -p tcp --dport 10999 -j DNAT --to
$endPointIp:10999
iptables -t nat -A PREROUTING -p tcp --dport 10009 -j DNAT --to
$endPointIp:10009
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -t nat -L -n -v
```

2. $endPointIP refers to the internal IP generated in the DLC console after establishing the private connection, as shown below:

3. After completing the configuration, you can access the Standard Engine and submit tasks through the EIP bound to the NAT gateway created in Step 2.

Note:

For specific access methods, see the documentation for Hive JDBC Access and Presto JDBC Access.

You only need to replace the internal IP in the access link with the EIP bound to the NAT gateway.

TDLC Command Line Interface Tool Access

Last updated：2024-07-31 17:33:04

TDLC is the Client Command Tool provided by Tencent Cloud Data Lake Computing (DataLake Compute, DLC). With the TDLC tool, you can submit SQL, Spark tasks to the DLC data engine.
TDLC is written in Go, based on the Cobra Framework, and supports configuring multiple buckets and cross-bucket operations. You can view the usage of TDLC by using `./tdlc [command] --help` .

Download and Installation

TDLC TCCLI offers binary packages for Windows, Mac, Linux operating systems. You can use them after simple installation and configuration. You can choose to download according to the type of operating system on the client.

Operating system	TDLC Binary Packages Download Address
Windows	tdlc-windows-v0.1.1
Mac	tdlc-macos-v0.1.1
Linux	tdlc-linux-v0.1.1

Rename the downloaded file to `tdlc` . Open the command line on your client, switch to the download path, and if you are using a Mac/Linux system, you need to grant file execution permission with the `chmod +x tdlc` command. After executing `./tdlc` , if the following content is displayed successfully, the installation is successful and it can be used.

```
Tencentcloud DLC command tools is used to play around with DLC.
With TDLC user can manger engines, execute SQLs and submit Spark Jobs.

Usage:
  tdlc [flags]
  tdlc [command]

Available Commands:
  config
  help        Help about any command
  spark       Submit spark app to engines.
  sql         Executing SQL.
  version

Flags:
```

```
      --endpoint string     Endpoint of Tencentcloud account. (default
"dlc.tencentcloudapi.com")
      --engine string       DLC engine. (default "public-engine")
  -h, --help                help for tdlc
      --region string       Region of Tencentcloud account.
      --role-arn string     Required by spark jar app.
      --secret-id string    SecretId of Tencentcloud account.
      --secret-key string   SecretKey of Tencentcloud account.
      --token string        Token of Tencentcloud account.

Use "tdlc [command] --help" for more information about a command.
```

Use Instructions

Global Parameters

TDLC provides the following global parameters.

Global Parameters	Description
--endpoint string	Service Connection Address, the default is dlc.tencentcloudapi.com
--engine string	The DLC Data Engine name, with a default value of public-engine. It is recommended that you use a Dedicated Data Engine
--region string	Use Region, such as ap-nanjing, ap-beijing, ap-guangzhou, ap-shanghai, ap-chengdu, ap-chongqing, na-siliconvalley, ap-singapore, ap-hongkong
--role-arn string	When you submit a Spark job, you need to specify the permissions to access COS files. For this, specify the role's rolearn. Details on rolearn can be referred to in Configuring Data Access Policy.
--secret-id string	The Tencent Cloud account's secretId
--secret-key string	The Tencent Cloud account's secretKey
--token string	(Optional) Tencent Cloud Account Temporary Token

CONFIG Command

The config can be used to set commonly used parameters, which will be provided with default values. Command line parameters will override the parameters set in the config.

Command	Description
list	List the current default configuration

set	Adjusting configuration
unset	Reset Configuration

Example:

```
./tdlc config list
./tdlc config set secret-id={1} secret-key={2} region={b}
./tdlc config unset region
```

SQL Subcommand

SQL subcommands currently only support Presto or SparkSQL clusters. Below are the parameters supported by SQL subcommands.

Parameter	Description
-e, --exec	Execute SQL Statement
-f, --file	Execute SQL file, if there are multiple SQL files, please use `;` to split
--no-result	No result retrieval after execution
-p, --progress	Display Execution Progress
-q, --quiet	Quiet Mode, submit the task without waiting for the execution status

Example:

```
./tdlc sql -e "SELECT 1" --secret-id aa --secret-key bb --region ap-beijing --
engine public-engine
./tdlc sql -f ~/biz.sql --no-result
```

SPARK Subcommand

Spark Subcommands include the following commands which can be used to submit Spark jobs, view running logs, and terminate tasks.

Command	Description
submit	Submit tasks via spark-submit
run	Execute Spark job
log	Viewing Execution Logs

list	View Spark job list
kill	Terminating Task

Below are the parameters supported by **Spark submit subcommand**, parameters related to files in the list support using local files or COSN protocol.

Parameter	Description
--driver-size	Driver Specification, defaults to small, medium, large, xlarge, for memory-optimized clusters use m.xmall, m.medium, m.large, m.xlarge
--executor-size	Executor Specification, defaults to small, medium, large, xlarge, for memory-optimized clusters use m.xmall, m.medium, m.large, m.xlarge
--executor-num	Number of Executors
--files	View Spark job list
--archives	Dependencies in Compressed Files
--class	Main Function for Java/Scala Execution
--jars	Dependent JAR Packages, use `,` to separate
--name	Program Name
--py-files	Dependent Python Files, Supports .zip, .egg, .py Formats
--conf	Additional Configuration

Example:

```
./tdlc spark submit --name spark-demo1 --engine sparkjar --jars /root/sparkjar-dep.jar --class com.demo.Example /root/sparkjar-main.jar arg1
./tdlc spark submit --name spark-demo2 cosn://bucket1/abc.py arg1
```

Third-party Software Linkage

Last updated：2024-07-31 17:33:21

Tencent Cloud DLC adheres to the principles of agility and openness in its product positioning, supporting the integration of numerous third-party software, including Scheduling Tools, Interactive Development Tools, BI Tools, etc. It is continuously supporting and testing other third-party software integrations.

Note

The third-party software listed below has been tested by the DLC product and research team on mainstream versions' core features, but not all version numbers are covered.

If you encounter any issues with integrating third-party software with DLC, or have needs for other third-party software integrations, you are welcome to Submit Ticket to contact us.

Scheduling Tool

If you have already deployed/own the following third-party software, you can connect it to DLC for managing and scheduling jobs on DLC.

Third-Party Software	Description
Apache Airflow	Airflow is a Workflow Management Platform that can be used for management, scheduling, and execution.
Apache DolphinScheduler	DolphinScheduler is a Visual DAG Workflow Task Scheduling Platform.

Interactive Computing

If you have already deployed/own the following third-party software, you can connect it to DLC to leverage DLC's capabilities for computation and analysis.

Third-Party Software	Description
Yanagishima	Yanagishima is an open-source web application for visualizing access to Trino, Hive, Spark.
Apache Zeppelin	Zeppelin is a web-based interactive computing environment.
Jupyter Notebook	Jupyter Notebook is a web-based interactive computing platform.

Database Tools

If you have already deployed/own the following third-party database tools, you can connect them to DLC to query DLC data.

Third-Party Software	Description
SQL Workbench	SQL Workbench is a cross-platform SQL query tool. You can access DLC using your own deployed SQL Workbench.
DBeaver	DBeaver is a universal database tool.

Business Intelligence

If you have already deployed/own the following third-party business intelligence software, you can connect it to DLC to conduct business intelligence analysis and generate reports to support your business decisions.

Third-Party Software	Description
Metabase	Metabase is an easy-to-use Report System. You can use your own deployed Metabase to access DLC and generate reports.
Redash	Redash is an open-source BI Tool, offering web-based Database Query and Data Visualization features.
FineBI	FineBI is a Business Intelligence Product launched by Fanruan Software Co., Ltd.
Tableau	Tableau is a Visual Analysis Platform, providing a powerful, secure, and flexible End-to-end Analysis Platform, offering a complete set of features from connection to Collaboration.
CBoard	CBoard is a Self-service BI Data Analysis Product, open-sourced and led by Shanghai Chuguo Information Technology Co., Ltd.
Apache Superset	Apache Superset is a modern Business Intelligence Web Application, open-sourced by Airbnb.

Python Access

Last updated：2024-07-31 17:33:57

DLC offers tools compliant with the DBAPI 2.0 standard. You can connect to DLC's Presto/Spark engine via Python, allowing for convenient SQL operations on DLC database tables.

Environment preparations

1. Python 3.9 or higher version.
2. Install `tencentcloud-dlc-connector.`

```
pip install -i https://mirrors.tencent.com/pypi/simple/ tencentcloud-dlc-connector
```

Usage Examples

Step 1: Connect to the engine

Code:

```
import tdlc_connector
import datetime
from tdlc_connector import constants

conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    token=None,
    endpoint=None,
    catalog=constants.Catalog.DATALAKECATALOG,
    engine="<ENGINE>",
    engine_type=constants.EngineType.AUTO,
    result_style=constants.ResultStyles.LIST,
    download=False,
    mode=constants.Mode.LASY,
    database='',
    config={},
    callback=None,
    callback_events=None,
    )
```

Parameter description:

Parameter	Description
region	Engine Region, such as ap-nanjing, ap-beijing, ap-guangzhou, ap-shanghai, ap-chengdu, ap-chongqing, na-siliconvalley, ap-singapore, ap-hongkong
secret_id	Tencent Cloud SecretID
secret_key	Tencent Cloud SecretKey
token	(Optional) Temporary Secret Token
endpoint	(Optional) Connect to the service node
engine	Engine name used, for example "test_python"
engine_type	(Optional) Engine type: corresponding to the engine type of the engine name, default value constants.EngineType.AUTO For example: AUTO, PRESTO, SPARK, SPARK_BATCH
result_style	(Optional) Format of the returned result, options are LIST/DICT
download	(Optional) Whether to download the data directly True/False, see Download Mode Description
mode	(Optional) Mode. Supports ALL/LAZY/STREAM
database	(Optional) Default database
config	(Optional) Submit to cluster configuration
callback	(Optional) Callback function, function signature def cb(statement_id, status)
callback_events	(Optional) Callback trigger event, used in conjunction with callback, see callback mechanism description for details
driver_size	(Optional) Driver node size, default value constants.PodSize.SMALL (Only valid for SPARK_BATCH clusters) Optional values: SMALL, MEDIUM, LARGE, XLARGE, M_SMALL, M_MEDIUM, M_LARGE, M_XLARGE
executor_size	(Optional) Executor node size, default value constants.PodSize.SMALL (Only valid for SPARK_BATCH clusters) Optional values: SMALL, MEDIUM, LARGE, XLARGE, M_SMALL, M_MEDIUM, M_LARGE, M_XLARGE
executor_num	(Optional) Number of Executor nodes, default value 1 (Only valid for SPARK_BATCH clusters)

executor_max_num	(Optional) Maximum number of Executor nodes, if not equal to executor_num, then enable Dynamic Resource Allocation (Only valid for SPARK_BATCH clusters)

Download Mode Explanation:

Serial number	download	mode	Description
1	False	ALL	Fetch all data from the interface, can only fetch data after completion
2	False	LASY	Fetch data from the interface, delay fetching data to the server based on the amount fetched
3	False	STREAM	Same as LASY mode
4	True	ALL	Download all results from COS (requires COS read permission) using local temporary storage, recommended for large data volumes
5	True	LASY	Download results from COS (requires COS read permission), delay downloading files based on fetch data volume
6	True	STREAM	Read result stream from COS in real-time (requires COS read permission), slower performance, extremely low local memory disk usage ratio

Step 2: Execute SQL

Code:

```
# Basic Operations

cursor = conn.cursor()
count = cursor.execute("SELECT 1")
print(cursor.fetchone())                    # Read one line of data
for row in cursor.fetchall():               # Read the remaining multiple lines of
    print(row)

# Use the pyformat format

cursor.execute("SELECT * FROM dummy WHERE date < %s", datetime.datetime.now())
cursor.execute("SELECT * FROM dummy WHERE status in %s", (('SUCCESS', 'INIT', 'FAIL
cursor.execute("SELECT * FROM dummy WHERE date < %(date)s AND status = %(status)s",

# Use BULK method
```

```
cursor.executemany("INSERT INTO dummy VALUES(%s, %s)", [('Zhang San', 18), ('Li Si'
```

Basic Operation Procedure

The process of the aforementioned code is as follows:

1. A cursor object is created with `conn.cursor()` .
2. A SQL query statement is executed with `cursor.execute("SELECT 1")` , and the result is assigned to the variable `count.`
3. A line of data is read through the `cursor.fetchone()` method and printed out.

Characteristic function

Description of the callback mechanism

```
import tdlc_connector
import datetime
from tdlc_connector import constants


def tdlc_connector_callback(statement_id, state):
    '''
    parmas: statement_id Quest id
    params: state Task status. The enumeration value is constants.TaskStatus
    '''
    print(statement_id, state)


conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    engine="<ENGINE>",
    engine_type=constants.EngineType.SPARK,
    result_style=constants.ResultStyles.LIST,
    callback=tdlc_connector_callback,
    callback_events=[constants.CallbackEvent.ON_INIT, constants.CallbackEvent.ON_SU
    )

cursor = conn.cursor()
cursor.execute("SELECT 1")
cursor.fetchone()

# The callback function is called when the task is initialized and the task is succ
```

```
```

Submit the task to the job cluster

Currently, you can submit tasks to the Spark job cluster. For details, see the following example.

```
from tdlc_connector import constants

conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    engine="<ENGINE>",                              # Select the spark job engine
    result_style=constants.ResultStyles.LIST,
    driver_size=constants.PodSize.SMALL,            # Select Driver Specifications
    executor_size=constants.PodSize.SMALL,          # Select the Executor specificati
    executor_num=1,                                 # Set the number of Executors
    executor_max_num=1,                             # Set the maximum number of execu
)
```

Note：

Upgrade the connector to >= 1.1.0 to use this feature.

Automatically infer engine type

You do not need to specify the engine type. The connector will automatically infer the engine type. For details, see the following example.

```
from tdlc_connector import constants

conn = tdlc_connector.connect(region="<REGION>",
secret_id="<SECRET_ID>",
secret_key="<SECRET_KEY>",
engine="<ENGINE>",
engine_type=constants.EngineType.AUTO  # This parameter can be set to AUTO or not t
)
```

Note：

Upgrade the connector to >= 1.1.0 to use this feature.

Null conversion

The current result set is stored in CSV format, the engine will convert the null value into an empty string by default, if you need to distinguish the null value, please specify the null value symbol, such as "\\1", the engine query result will convert the null value into "\\1", while the driver will convert the "\\1" field into None, please refer to the following example.

```
from tdlc_connector import constants, formats

formats.FORMAT_STRING_NULL = '\\1'

conn = tdlc_connector.connect(region="<REGION>",
    secret_id="<SECRET_ID>",
    secret_key="<SECRET_KEY>",
    engine="<ENGINE>",
    result_style=constants.ResultStyles.LIST
)
```

Note：

Null conversion currently only supports SparkSQL clusters. Upgrade connector to >= 1.1.3.