TDMQ for CKafka

CKafka Connector

Product Documentation

Contents

CKafka Connector Introduction Overview

Last updated：2024-01-09 14:54:11

Overview

CKafka Connector is a SaaS tool that helps you integrate, process, and distribute your data all in one stop. It offers HTTP/TCP-based SDKs for quick data reporting and uses Change Data Capture (CDC) for subscribing to and storing changes in databases such as TencentDB for MySQL/PostgreSQL/MongoDB, making it easier for you to deliver logs across Tencent Cloud products. It also provides configurable ETL workflows and various data distribution channels, allowing you to build a low-cost data flow linkage from data sources to data processing systems.
By going SaaS, CKafka Connector allows you to connect data in and off the cloud in cross-cloud and hybrid cloud scenarios. It comes with low-cost data flow capabilities that are simple to configure, helping you create a reliable and stable data linkage.

Features

Data reporting

CKafka Connector offers client SDKs based on HTTP/TCP protocols to easily report data to various message queue services such as CKafka, TDMQ for Pulsar, and TDMQ for RocketMQ. It simplifies the data reporting process and completes data reporting in a SaaS manner, freeing you from developing and maintaining data reporting servers as well as learning the complex protocols of message queues.

Database change subscription

By using the CDC mechanism, CKafka Connector can subscribe to data changes in various databases such as binlog of TencentDB for MySQL, change stream of TencentDB for MongoDB, and row-level change of TencentDB for PostgreSQL/SQL Server. This makes it easier for your business to process, distribute, and query such data.

Diverse data sources

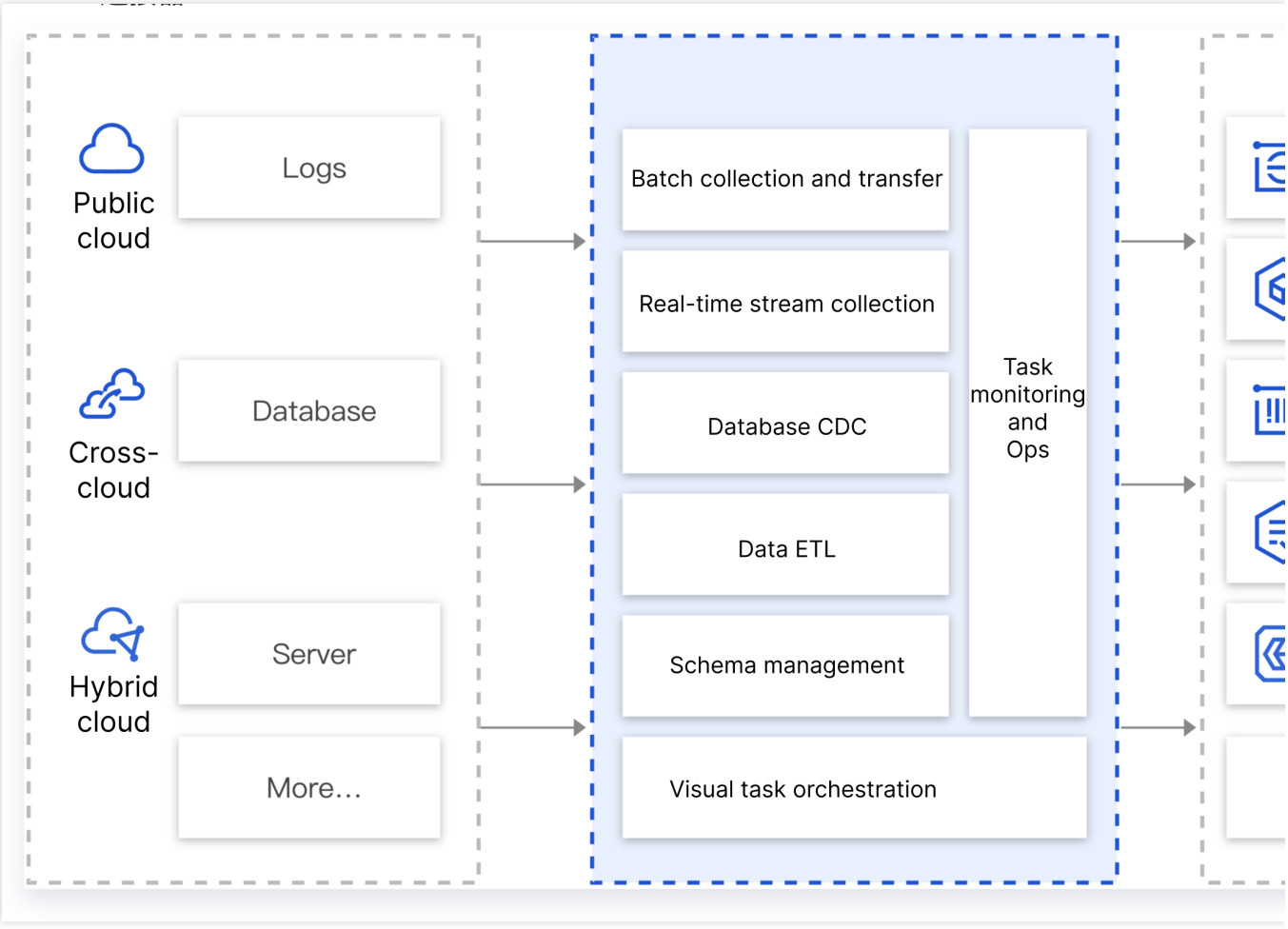
CKafka Connector opens up data connections in public cloud, cross-cloud, and hybrid cloud scenarios such as log, database, middleware, and HTTP. It can integrate logs from multiple Tencent Cloud services as well as data from your self-built sources and other clouds.

Data ETL and distribution

After data is reported to the message queue, CKafka Connector offers powerful visual data ETL configuration capabilities to conveniently and quickly ETL, format, and convert data and then dump the output data to downstream systems.

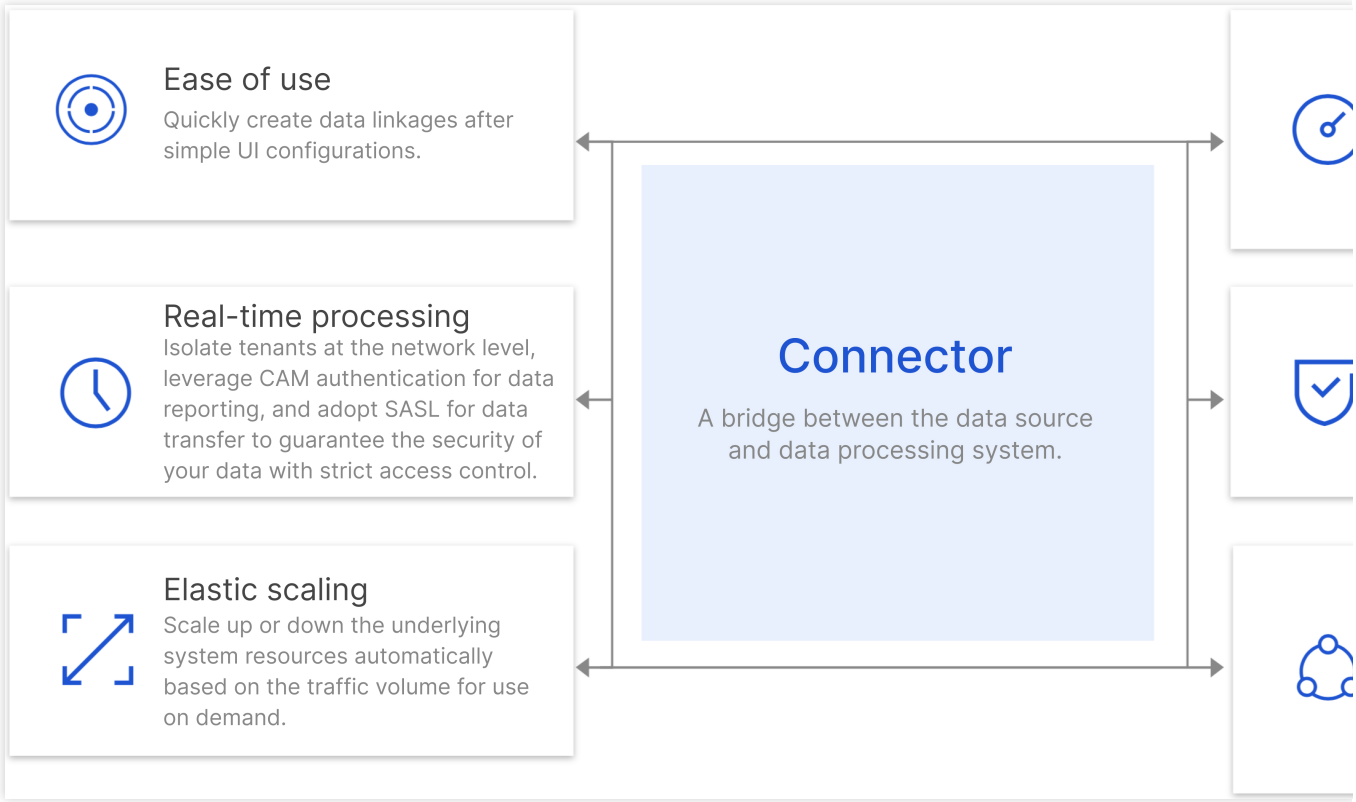
Custom serverless processing

By leveraging the strengths of the serverless platform and the pay-as-you-go billing and code customization capabilities of SCF, CKafka Connector allows you to write business logic based on functions for customized data processing and distribution.



Strengths

Last updated：2024-01-09 14:54:12



Ease of use

Quickly create data reporting, ETL, and storage linkages after simple UI configurations, freeing you from complex underlying system setup and component Ops.

Real-time processing

Instantly integrate, process, and distribute data to downstream systems throughout the entire linkage of data collection, reporting, and transfer. With CKafka Connector data linkage, you can query data in ES in just seconds after it is reported on the client.

Elastic scaling

Scale up or down automatically based on traffic volume to ensure high system availability during peak hours and eliminate the need to estimate your business capacity in advance. You can use CKafka Connector to integrate, process, and dump data in a serverless manner.

High availability

Leverage distributed cross-AZ deployment at the integration, processing, and distribution layers and automatic failover to ensure over 99.9% service availability. When a system fault occurs, data will be temporarily stored in CKafka to prevent it from being lost. After the system recovers, the data will continue to be processed and distributed to downstream systems.

High security

Isolate tenants at the network level, leverage CAM authentication for data reporting, and adopt SASL for data transfer to guarantee the security of your data with strict access control.

Rich environment support

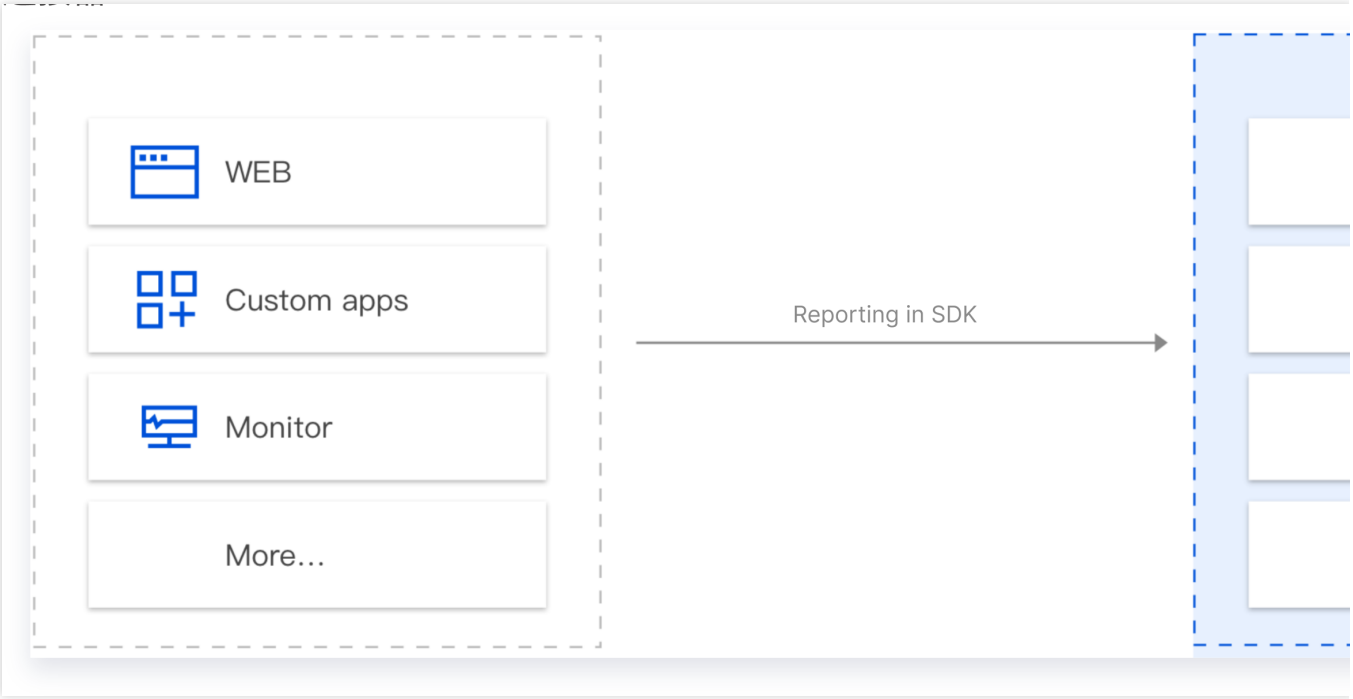
Support cross-cloud and hybrid cloud scenarios as well as self-built and Tencent Cloud service data connections. It integrates upstream log, database, and middleware data sources to interconnect over 15 Tencent Cloud products, including TKE, COS, ES, and TencentDB for MySQL/PostgreSQL. This easily enables one-stop data integration and transfer.

Use Cases

Last updated：2024-01-09 14:54:11

Data reporting and query

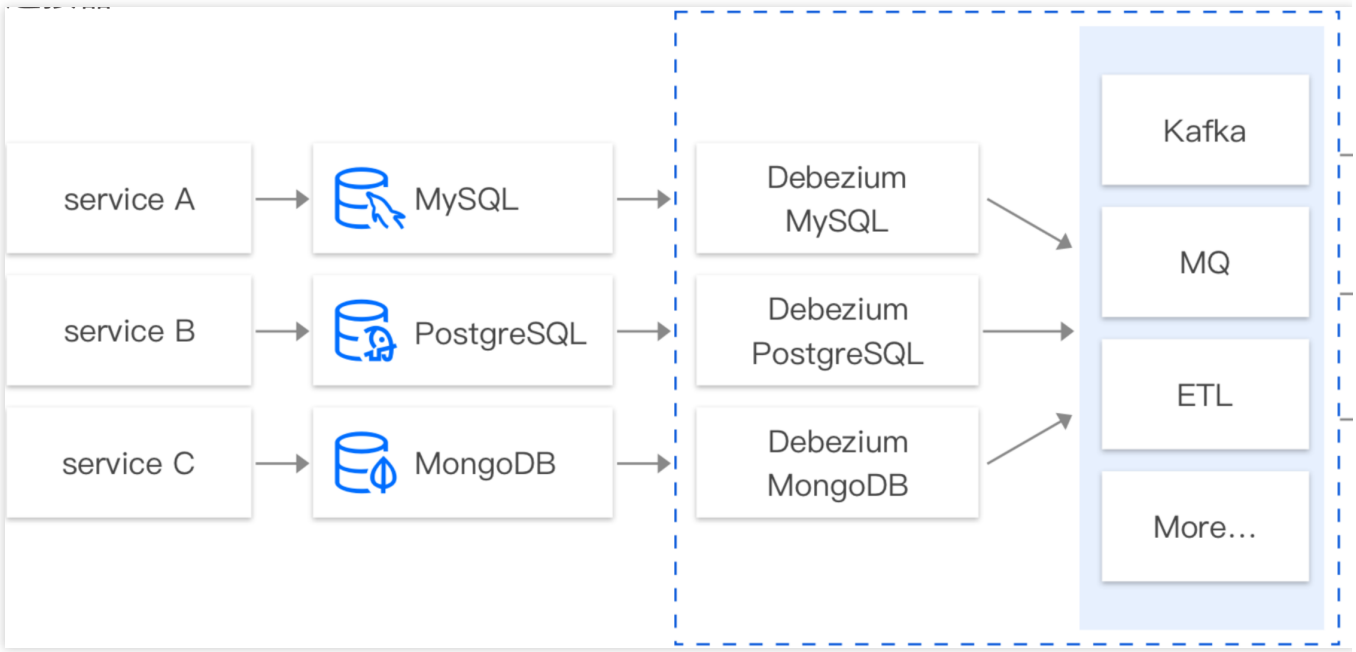
CKafka Connector supports a wide range of data reporting scenarios, such as operation behavior analysis on mobile applications, bug log reporting on frontend pages, and business data reporting. In general, reported data needs to be dumped to downstream storage and analysis systems like Elasticsearch and HDFS for processing. Traditionally, this requires setting up a server, purchasing a storage system, and performing data integration, processing, and dumping while customizing code along the way. This is cumbersome and costly in terms of long-term system Ops.
By going SaaS, CKafka Connector allows you to create a complete linkage in just two steps: configure in the console and report data in the SDK. It is designed to be serverless and pay-as-you-go, removing the need to estimate the capacity in advance and saving costs in development and use.



Database change subscription

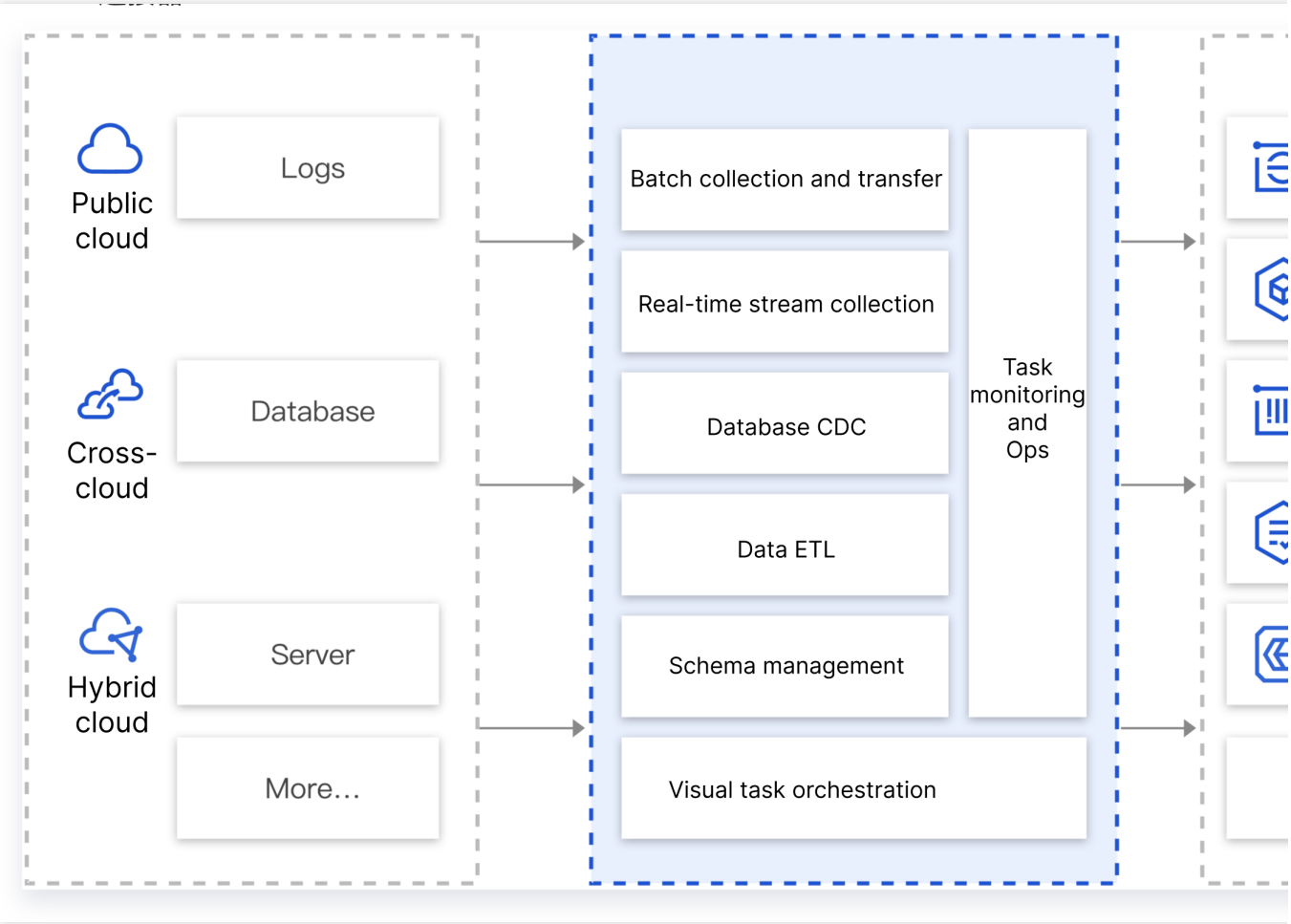
Using the CDC mechanism, CKafka Connector can subscribe to data changes in various databases such as binlog of TencentDB for MySQL, change stream of TencentDB for MongoDB, and row-level change of TencentDB for PostgreSQL/SQL Server. In real-world business scenarios, you often need to subscribe to MySQL binlogs to get the change history (INSERT, UPDATE, DELETE, DDL, DML, etc.), as well as perform business logic processing such as query, failure recovery, and analysis.

Generally, you have to customize a CDC-based database subscription component like Canal, Debezium, or Flink CDC to subscribe to data changes. These components are labor-intensive to build and maintain. You also need to have a complete monitoring system in place to ensure that the subscription component runs smoothly.
In contrast, CKafka Connector provides SaaS components that enable data subscription, processing, and dumping through simple UI configurations.



Data integration

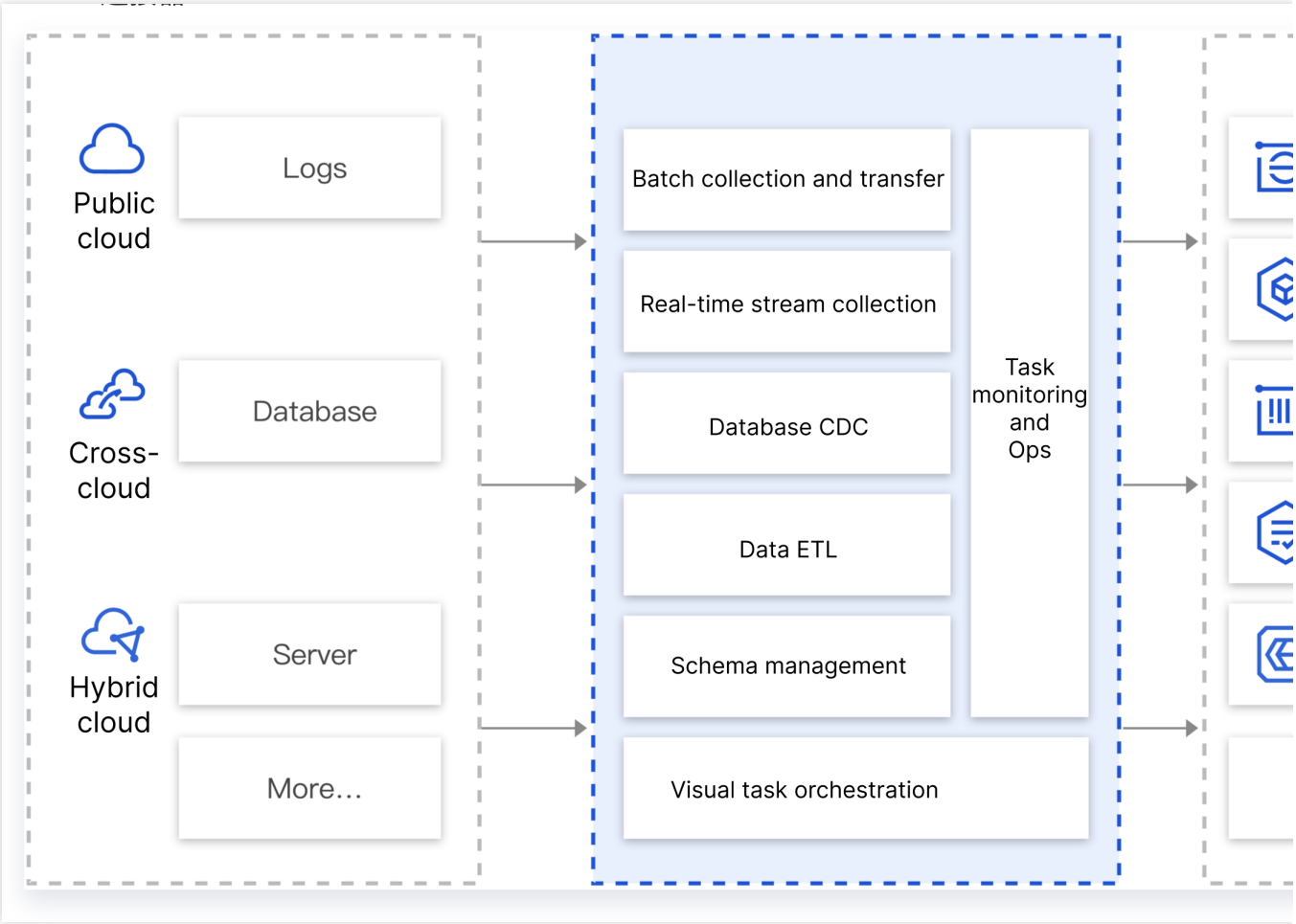
CKafka Connector can integrate data from different sources (database, middleware, log, application system, etc.) in different environments (Tencent public cloud, self-built IDC, cross-cloud environment, hybrid cloud, etc.) to CKafka for convenient processing and distribution. In practice, database data, business client data from an application, and log data often need to be aggregated into a message queue for unified dumping, analysis, and processing after ETL. CKafka Connector offers robust data aggregation, storage, processing, and dumping capabilities. In short, it can easily integrate data by connecting different data sources to downstream data targets.



Data ETL and dumping

In some use cases, data from a cache layer component such as Kafka needs to be stored in a downstream system such as CKafka, ES, or COS after ETL. The common practice is to process the data with Logstash, Flink, or custom code and monitor those components to ensure their stable operation. However, in order to operate and maintain the components, it requires learning their syntax, specifications, and technical principles. This incurs significant costs which are unnecessary if all you need is simple data processing.

CKafka Connector comes with lightweight, UI-based, data ETL and dumping capabilities that are simple to configure, making it easier for you to process and dump data to downstream storage systems.

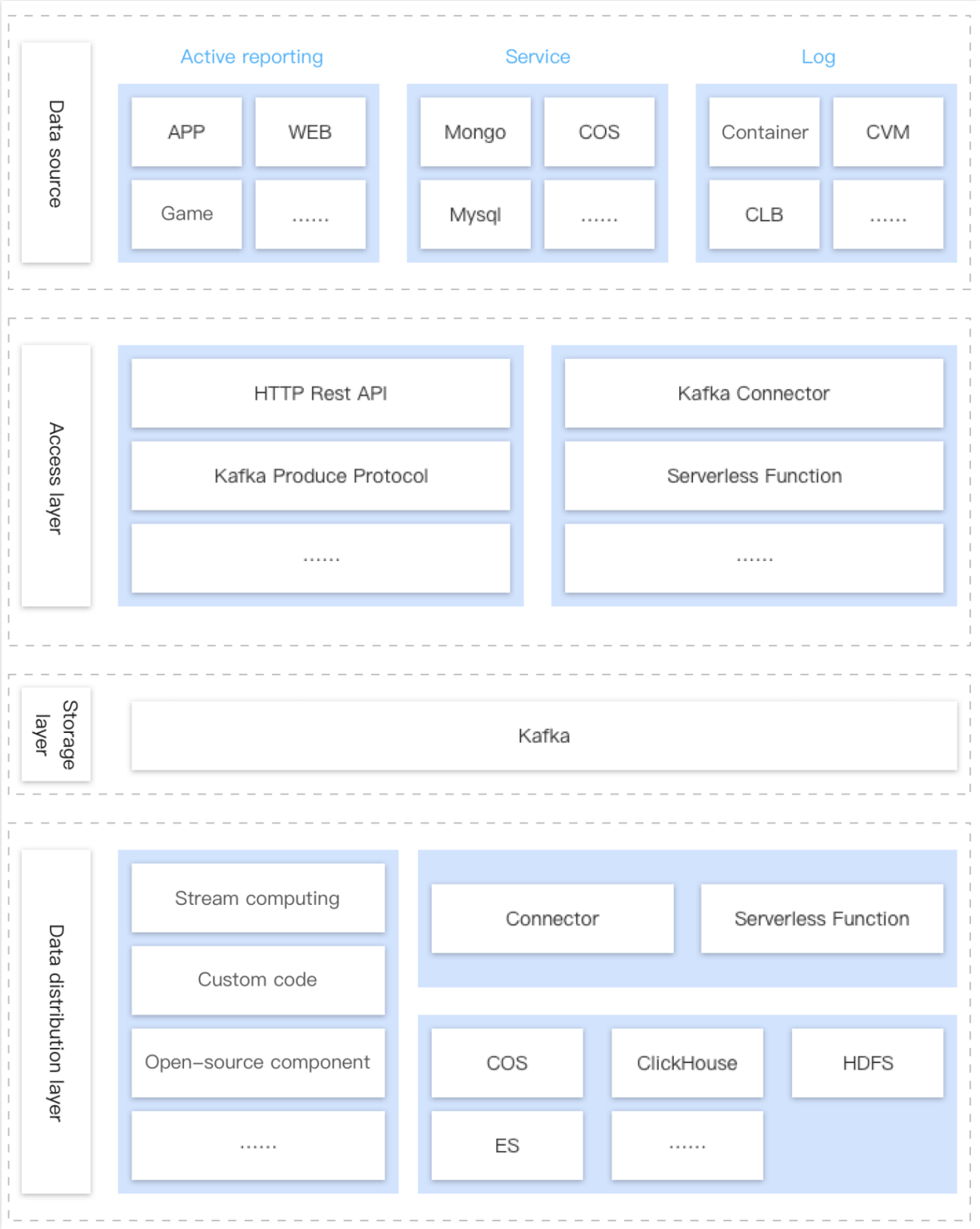


Technical Principles

Last updated：2024-01-24 17:20:44

System Architecture

The system architecture of the CKafka Connector is as shown below:



Primarily divided into four layers:

Data Source

The data source refers to the location of the customer's data source. The data source can be in the cloud, self-built IDC, cross cloud, or hybrid cloud. The data can be business data, log data, or data in the DB, etc.

Access Layer

The CKafka connector provides an access layer that adapts to various protocols, such as HTTP Rest, Kafka Protocol, Change Data Capture, etc. The access layer is deployed in a distributed manner, which features capabilities such as Auto Scaling and automatic retrying, ensuring the stability of data access.

Storage Layer

The storage layer of the CKafka connector is the Message Queue (MQ) on Tencent Cloud. By default, it supports Kafka, but it can also support other MQs such as Pulsar, RocketMQ. The MQ storage layer primarily serves to level peak loads, distribute data, and cache.

Data Distribution Layer

The CKafka Connector offers a data cleansing (ETL) engine that, based on the configuration, carries out data cleansing and provides the Connector with the dump capability. It can consume data from the data storage layer and guide the data into various different downstream storage engines.

Use Limits

Last updated：2024-01-09 14:54:11

This document lists the limits of certain metrics and performance in CKafka Connector. Be careful not to exceed the limits during use to avoid exceptions.

Limit	Description
Task parallelism	Data integration, processing, and dumping all run concurrently in multiple subtasks (workers) at the underlying layer. The worker parallelism is 1 by default. The system will automatically check whether there is a data heap, and if so, increase the number of running workers to improve the task processing capability. Currently, the maximum number of workers is equal to the number of partitions in the Kafka topic. You cannot set the number of workers; instead, the system will automatically adjust it.
Task throughput	CKafka Connector works in the form of tasks, and its task performance depends on the service capabilities of upstream and downstream components. For example, if the upstream is the Kafka > CKafka Connector > Elasticsearch linkage, then when there is no performance bottleneck in the upstream and downstream, CKafka Connector will improve the data processing capability by adjusting the task parallelism; if a performance bottleneck is hit, data flow will become slower. You can identify bottlenecks by viewing monitoring metrics and configuring heap alarms.
Number of tasks	The maximum number of tasks per account is 200 by default. If you need more, submit a ticket for application.
Number of connections	The maximum number of connections per account is 100 by default. If you need more, submit a ticket for application.
Number of topics	The maximum number of topics per account is 200 by default. If you need more, submit a ticket for application.
Number of schemas	The maximum number of schemas per account is 100 by default, each of which can contain up to 100 fields. If you need more, submit a ticket for application.
QPS for integration over HTTP	The maximum QPS per HTTP access point is 2,000 by default. If you need more, submit a ticket for application.
Batch size of reported data	The maximum size of data reported in each batch over HTTP is 5 MB, and an error will be reported if this limit is exceeded.
Number of data records	The maximum number of data records reported in each batch over HTTP is 500, and an error will be reported if this limit is exceeded.

reported per batch	
Task data loss in extreme cases	**Data processing** and **data distribution** tasks are essentially to create CKafka producers and consumers to produce and consume in the selected task data source topic. If the consumption is successful, that is, after the data is successfully delivered to the data target resource, the consumer group corresponding to the task (datahub-<task ID>) will submit the message offset corresponding to the data. However, if the task is restarted due to an exception before submitting the offset after successfully delivering the data, the data will be repeatedly delivered to the target resource. Therefore, we recommend you conduct idempotent processing for extreme cases in your business logic code. If the message has not been successfully consumed, but the configured maximum message retention period (including the dynamic message retention policy and the manually configured topic-level message retention policy) has elapsed, and the message is deleted, the task will not be able to consume the expired message, and the data contained in it will be missing in the data target.

Accessible IP Range

Last updated：2024-07-19 14:26:08

Overview

For some components, Ckafka Connector can be used only when their security groups allow their access to certain ports and IP ranges. This document lists the ports and IP ranges for such components in different regions.

Description

Accessible port configured for each component

MySQL

TCP port 3306 must be set to be accessible. You can also set other custom ports to be accessible as needed.

MariaDB

TCP port 3306 must be set to be accessible. You can also set other custom ports to be accessible as needed.

PostgreSQL

TCP port 5432 must be set to be accessible. You can also set other custom ports to be accessible as needed.

MongoDB

TCP port 27017 must be set to be accessible. You can also set other custom ports to be accessible as needed.

SQL Server

TCP port 1433 must be set to be accessible. You can also set other custom ports to be accessible as needed.

ClickHouse

TCP port 9000 must be set to be accessible. You can also set other custom ports to be accessible as needed.

Kafka (Self-Built)

Currently, only the YunTi environment/supporting environment (refers to Tencent's internally-developed environments) supports self-built Kafka instances. The unified port TCP:9092 is required to open. (If there are custom ports, please open ports based on the actual settings.) The IP ranges corresponding to the region in the list below are required to open based on the region to be used.

Accessible IP ranges in different regions

IP range that must be set to be accessible in all regions:

Please Submit Ticket to contact us to obtain the open network.

Connection Management

Last updated：2024-01-09 14:54:11

Overview

When a large number of data tasks share the same data source or target, the configuration information of the source or target needs to be entered repeatedly each time a task is created, which is time-consuming.
With CKafka Connector, you can create separate connections. After a connection is created, it can be directly associated with a specific data task as a data source or target with no repeated configurations needed, reducing your operating costs.
A connection can be associated with multiple data tasks. Currently supported connection types include ClickHouse, CTSDB, Doris, Data Transmission Service (DTS), Elasticsearch Service (ES), MongoDB, MySQL, PostgreSQL, MySQL, MariaDB, and SQL Server.
This document describes how to create, modify, and delete a connection in CKafka Connector.

Directions

Creating a connection

1. Log in to the CKafka console.
2. Select **Connector** > **Connection List** on the left sidebar, select the region, and click **Create Connection**.
3. Select the connection type in the pop-up window, click **Next**, and enter the connection configuration information.
ClickHouse
CTSDB
Doris
DTS
ES
MongoDB
MySQL
PostgreSQL
TDSQL-C
MariaDB
SQL Server
CDWPG
TDSQL for PostgreSQL
Connection Name: Enter the connection name.

Description: Enter the optional connection description.
Data Warehouse Type: Select **Cloud Data Warehouse** (CDWCH) or **Self-built ClickHouse**.
CDWCH: As an instance has been encapsulated with a private connection during creation, you can directly select the corresponding **CDWCH** instance in the console, and the data distribution feature will automatically connect to the instance's VPC.
Self-built ClickHouse: As the CKafka instance is a managed instance and EMR ClickHouse creates a public network route on the purchased CVM instance directly, you need to manually create a CLB instance to connect to the VPC. The following steps use EMR ClickHouse as an example to create a CLB instance:
1.1.1 Go to the EMR console, select the target cluster, click **Cluster Resource** > **Node Status**, and find the ClickHouse node IP on the status page.
1.1.2 Go to the CLB console, create a CLB instance, click **Listener Management** on the top, click **TCP/UDP/TCP SSL Listener** on the page, and enter the **port used during data distribution** as the port.
1.1.3 After creating a listener, click **Bind Backend Service** and enter the TCP port of ClickHouse, which is 9000 by default.
1.1.4 After the binding, you can select the created CLB instance and enter the port listened on by the **CLB** instance on the data distribution page in the CKafka console.
Note:
Currently, you can only create a data distribution to ClickHouse task in the same region as the CLB instance.
Username: ClickHouse username, which is `default` by default.
Password: ClickHouse password.
For security reasons, the password is required. Currently, the password may be empty after an instance is created, in which case you need to modify the password in the `user.xml` configuration file. For detailed directions, see the ClickHouse documentation.

Connection Name: Enter the connection name.
Description: Enter the optional connection description.
CTSDB Address: Enter the address of the connected CTSDB database.
Username: Enter the database username.
Password: Enter the database password.
Connection Name: Enter the connection name.
Description: Enter the optional connection description.
Source Database Type: The default option is **Self-built/EMR Doris**.
CLB Instance: Only private network CLB instance is supported, and the Doris FE and BE ports must be simultaneously mounted to the CLB instance.
FE Port: FE JDBC port, which defaults to 9030.
BE Port: BE HTTP port, which defaults to 8040.
Username: Enter the database username.
Password: Enter the database password.

Connection Name: Enter the connection name.
Description: Enter the optional connection description.
DTS Instance: Select a DTS instance. The partition count of the topics that are subscribed to in DTS must be set to the same as that of the target Kafka topics.
DTS Consumer Group: Select a DTS consumer group.
Consumer Group Account: DTS consumer group account.
Consumer Group Password: DTS consumer group password.
Connection Name: Enter the connection name.
Description: Enter the optional connection description.
ES Instance Cluster: Select an ES instance cluster.
Instance Username: Enter the ES instance username, which cannot be modified once set. It is `elastic` by default.
Instance Password: Enter the ES instance password.
Connection Name: Enter the connection name.
Description: Enter the optional connection description.
Source Database Type:
TencentDB for MongoDB: Select a database instance.
Self-built MongoDB: Select your CLB instance and specify the port.
Username: Source MongoDB database username.
Password: Source MongoDB database password.
Connection Name: Enter the connection name.
Description: Enter the optional connection description.
Source Database Type:
TencentDB for MySQL: Select a database instance.
Self-built MySQL: Select your CLB instance and specify the port. Only private network CLB instance is supported. For CLB-based data integration, the CLB instance can only be mounted to one source database due to the restrictions of MySQL's sync mechanism.
Username: Enter the MySQL username.
Password: Enter the MySQL password.
Connection Name: Enter the connection name.
Description: Enter the optional connection description.
Source Database Type:
TencentDB for PostgreSQL: Select a database instance.
Self-built PostgreSQL: Select your CLB instance and specify the port. Only private network CLB instance is supported. For CLB-based data integration, the CLB instance can only be mounted to one primary database due to the restrictions of PostgreSQL's sync mechanism.
Username: Enter the PostgreSQL username.
Password: Enter the PostgreSQL password.

Connection Name: Enter the connection name.

Description: Enter the optional connection description.

Database Type: Select **PostgreSQL** or **MySQL**.

Database Instance: Currently, you can only select a running instance.

Username: Enter the database username.

Password: Enter the database password.

Connection Name: Enter the connection name.

Description: Enter the optional connection description.

Database Instance: Select a database instance.

Username: Enter the database username.

Password: Enter the database password.

Connection Name: Enter the connection name.

Description: Enter the optional connection description.

Database Instance: Select a database instance.

Username: Enter the database username.

Password: Enter the database password.

Connection Name: Enter the connection name.

Description: Enter the optional connection description.

Database Instance: Select a database instance.

Username: Enter the database username.

Password: Enter the database password.

Connection Name: Enter the connection name.

Description: Enter the optional connection description.

Database Instance: Select a database instance.

Username: Enter the database username.

Password: Enter the database password.

4. After entering the connection configuration information, click **Next** to verify the connection. After the verification is passed, the connection is created, and you can see it in the **Connection List**.

Editing the configuration

1. Log in to the CKafka console.
2. Select **Connector** > **Connection List** on the left sidebar and click the **ID** of the target connection to enter its **Basic Info** page.
3. Click **Edit Configuration** in the top-right corner of the **Basic Info** module to modify the connection configuration information. You can choose whether to enable **Update & Restart All Associated Tasks**. If it is enabled, all tasks associated with the connection will be updated and restarted.

Viewing associated tasks

1. Log in to the CKafka console.
2. Select **Connector** > **Connection List** on the left sidebar and click the **ID** of the target connection to enter its **Basic Info** page.
3. Select the **Associated Task** tab at the top of the page to view the list of tasks associated with the connection. You can filter tasks by **Data Source** or **Data Target**.

Deleting a connection

On the Connection List page, click **Delete** in the **Operation** column and click **OK** in the pop-up window to delete the connection.

Note:

A connection can be deleted only if it has no associated tasks.

Task Management
Creating Data Access Task
Reporting over HTTP

Last updated：2024-01-09 14:54:11

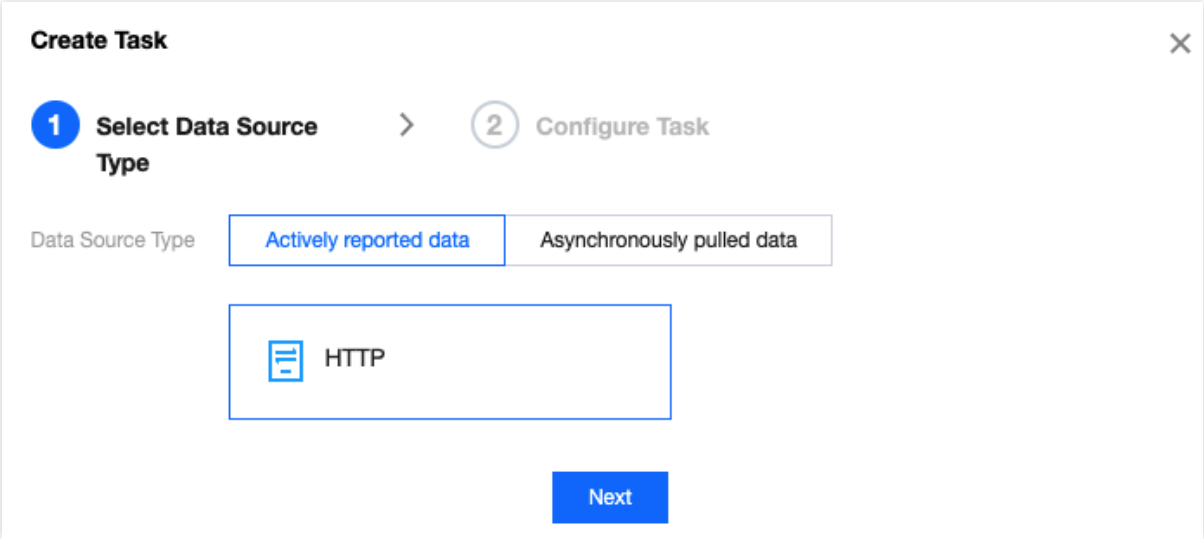
Overview

DataHub supports accessing different types of data generated by various data sources for unified management and distribution to downstream offline/online processing systems, forming a clear data flow channel.
This document takes HTTP data as an example to describe how to create an active data reporting task and modify the task configuration in the CKafka console.

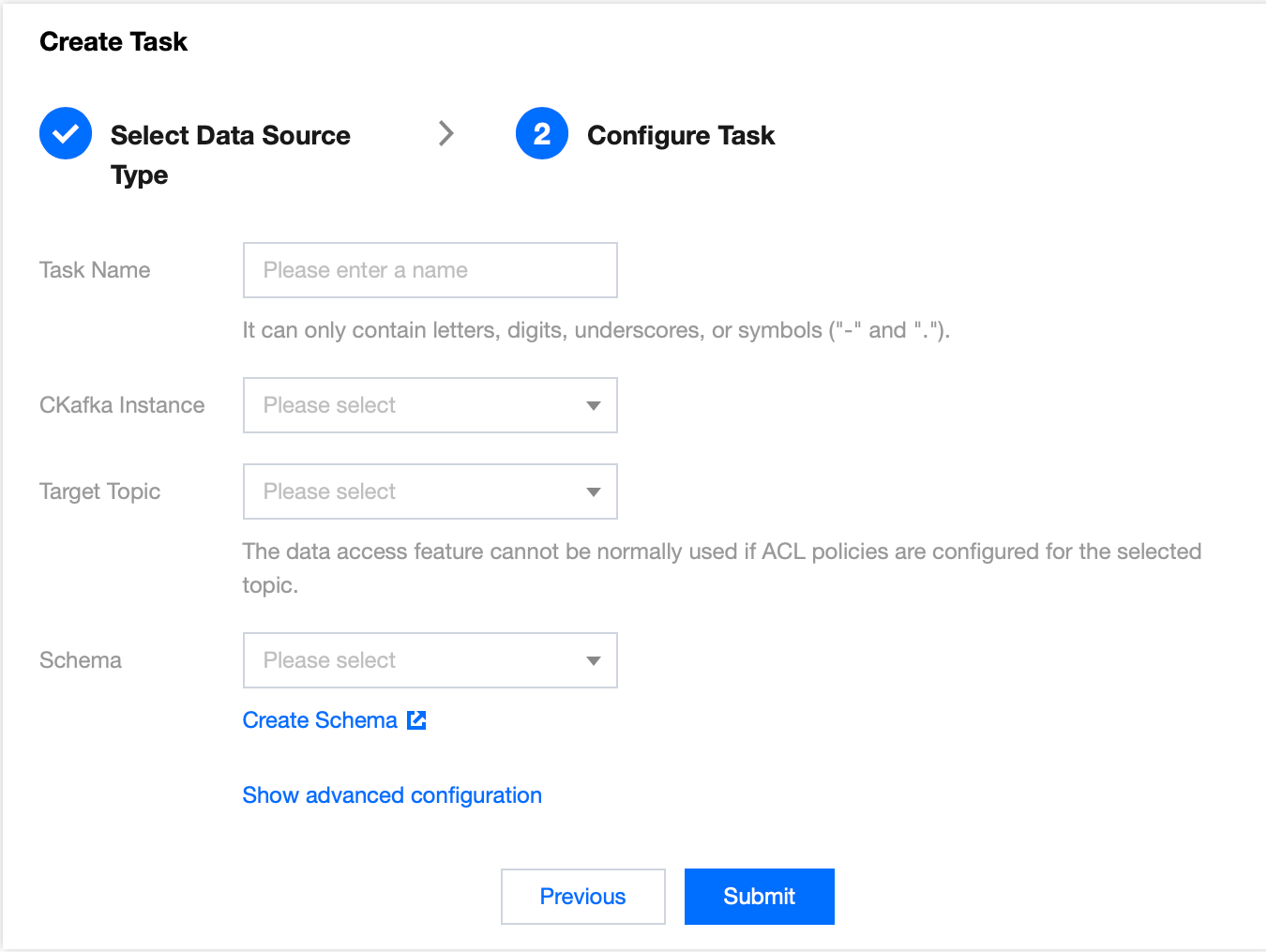
Directions

Creating data access task

Prerequisites: You have created a CKafka instance and a topic.
1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar, select the region, and click **Create Task**.
3. In the pop-up window, select **Actively reported data** for **Data Source Type**.



4. Click **Next**, enter the task name, and select the created CKafka instance and topic.



Task Name: Enter the task name. It can only contain letters, digits, underscores, or symbols ("-" and ".").

CKafka Instance: Select the target CKafka instance.

Target Topic: Select the target CKafka topic for data shipping. The data distribution feature cannot be normally used if ACL policies are configured for the selected topic.

Schema: After a schema is associated, it will be used to verify data format. If there is no appropriate schema, you can click Create Schema to enter the schema creation page.

QPS Limit: Enter the QPS limit.

5. Click **Submit**. After the task is created successfully, access point information will be generated.

6. Copy the access point information to the SDK to write data.

Note:

For more information, see Data Reporting SDK.

Modifying data target

1. Log in to the CKafka console.

2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.

3. Click **Change Data Target** in the top-right corner of the **Data Access** module to modify the data access target.

Note:

You can switch only the target CKafka topic. The target CKafka instance cannot be modified.

The new data target will take effect in about one minute.

The access point will not be generated again after the data target is modified.

4. Click **Submit**.

Associating/Disassociating schema

If you don't associate a schema during task creation, you can associate one later. Schemas can also be disassociated. The steps are as follows:

1. Log in to the CKafka console.

2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page. In the basic information module, you can associate/disassociate schemas.

Viewing monitoring data

1. Log in to the CKafka console.

2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.

3. Select the **Monitoring** tab to view the monitoring data of the target topic.

Pausing task

On the **Data Access** page, click **Pause** in the **Operation** column of the target task to pause the task.

Note:

If you find that the data access task affects the normal use of CKafka, you can pause it.

Resuming task

On the **Data Access** page, click **Resume** in the **Operation** column of the target task to resume the paused task.

Note:

A paused task can be resumed to continue dumping data.

Deleting task

On the **Data Access** page, click **Delete** in the **Operation** column of the target task and click **OK** in the pop-up window to delete the task.

Note:

Once the task is deleted, data access will be stopped and the task record will be deleted, but the previously dumped data and CKafka instance involved will not be affected.

A task cannot be recovered once deleted. Proceed with caution.

COS

Last updated：2024-01-09 14:54:12

Overview

DataHub supports accessing different types of data generated by various data sources for unified management and distribution to downstream offline/online processing systems, forming a clear data flow channel.
This document takes COS data as an example to describe how to create an async data pull task and modify the task configuration in the CKafka console.

Directions

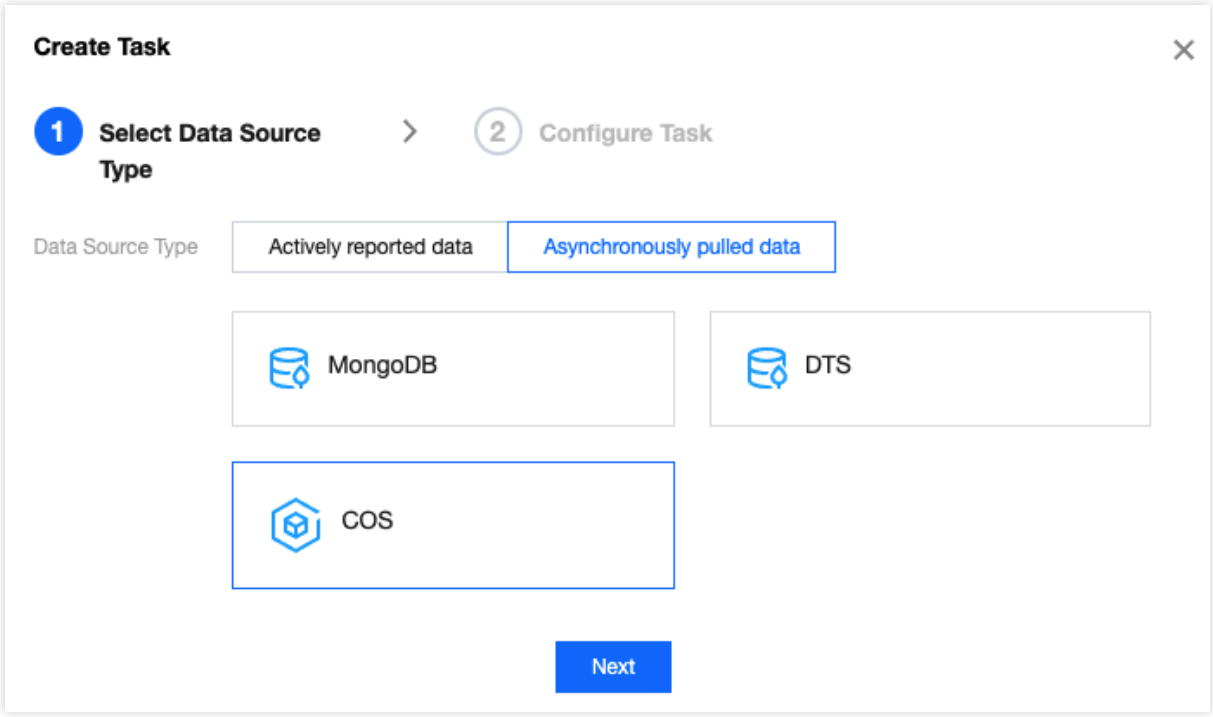
Creating data access task

Prerequisites

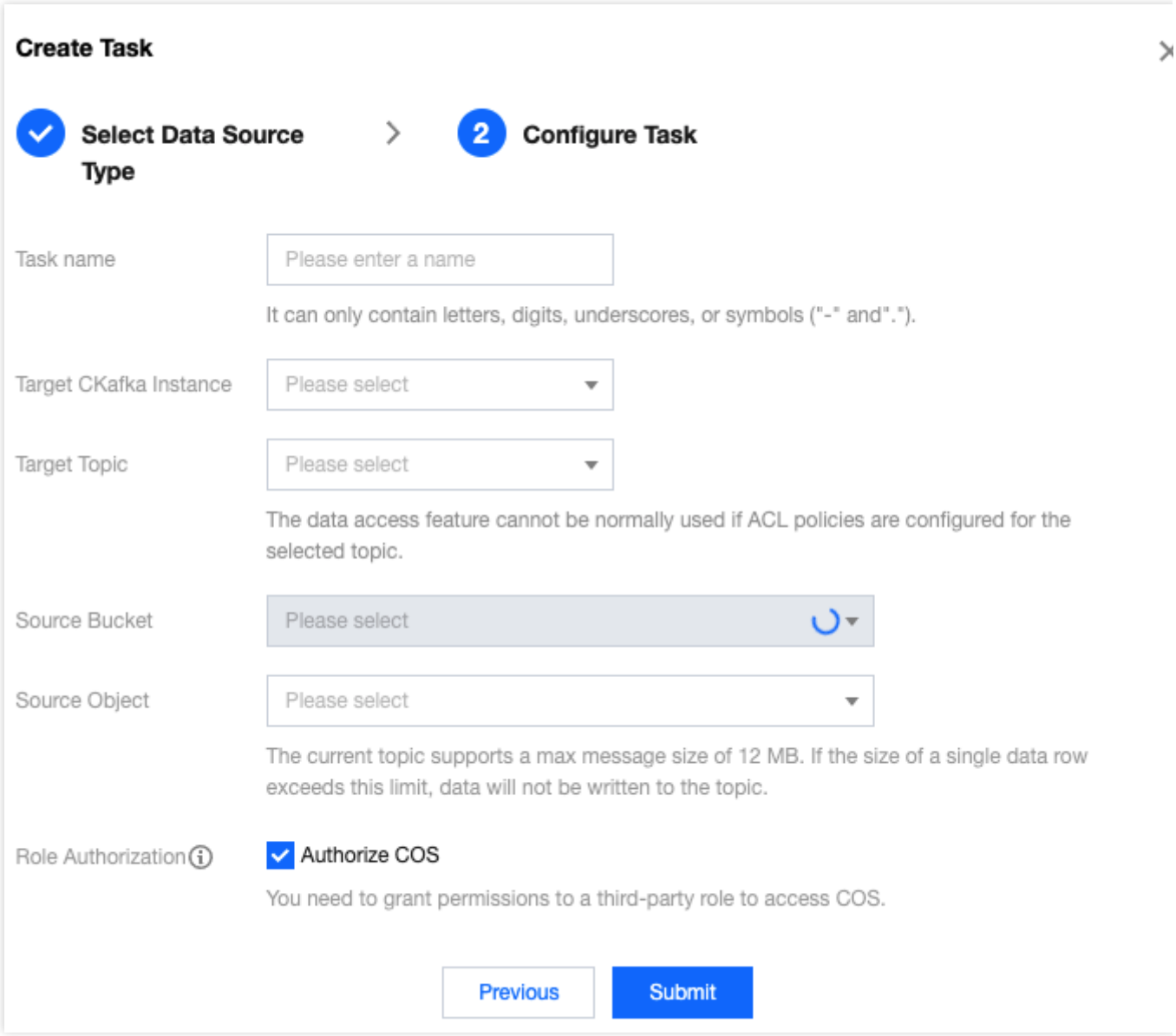
You have created a CKafka instance and a topic.
You have created a bucket and an object

Directions:

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar, select the region, and click **Create Task**.
3. In the pop-up window, select **Asynchronously pulled data** > **COS** for **Data Source Type**.



4. Click **Next** and enter the task details.



Task Name: It can only contain letters, digits, underscores, or symbols ("-" and ".").

Target CKafka Instance: Select a CKafka instance.

Target Topic: Select the target CKafka topic for data delivery.

Source Bucket: Select the source data bucket.

Source Object: Select the source data object. The current topic supports a max message size of 12 MB. If the size of a single data row exceeds this limit, data will not be written to the topic.

Role Authorization: You need to grant permissions to a third-party role to access COS.

5. Click **Submit**.

Changing data source and data target

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.
3. Click **Change Data Target** in the top-right corner of the **Data Target** module to modify the data target information.

Viewing monitoring data

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.
3. Select the **Monitoring** tab to view the monitoring data of the target topic.

Pausing task

On the **Data Access** page, click **Pause** in the **Operation** column of the target task to pause the task.

Note:

If you find that the data access task affects the normal use of CKafka, you can pause it.

Resuming task

On the **Data Access** page, click **Resume** in the **Operation** column of the target task to resume the paused task.

Note:

A paused task can be resumed to continue dumping data.

Deleting task

On the **Data Access** page, click **Delete** in the **Operation** column of the target task and click **OK** in the pop-up window to delete the task.

Note:

Once the task is deleted, data access will be stopped and the task record will be deleted, but the previously dumped data and CKafka instance involved will not be affected.

A task cannot be recovered once deleted. Proceed with caution.

DTS

Last updated：2024-01-09 14:54:11

Overview

DataHub supports accessing different types of data generated by various data sources for unified management and distribution to downstream offline/online processing systems, forming a clear data flow channel.
This document takes DTS data as an example to describe how to create an async data pull task and modify the task configuration in the CKafka console.

Directions

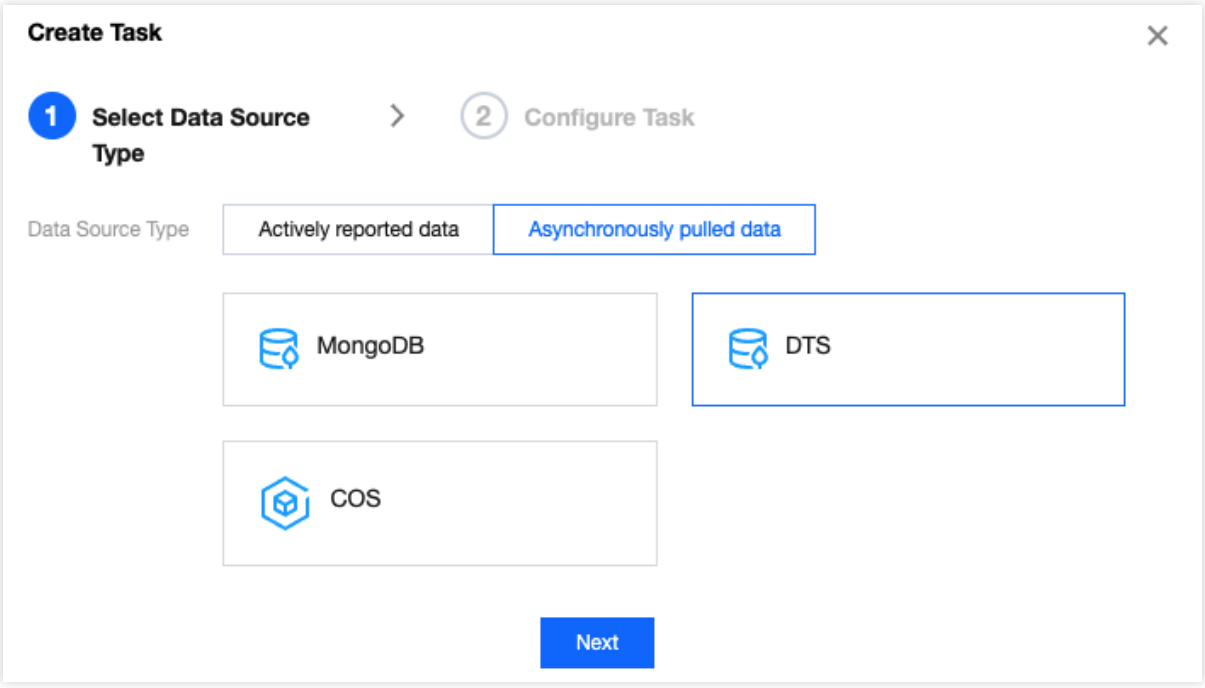
Creating data access task

Prerequisites

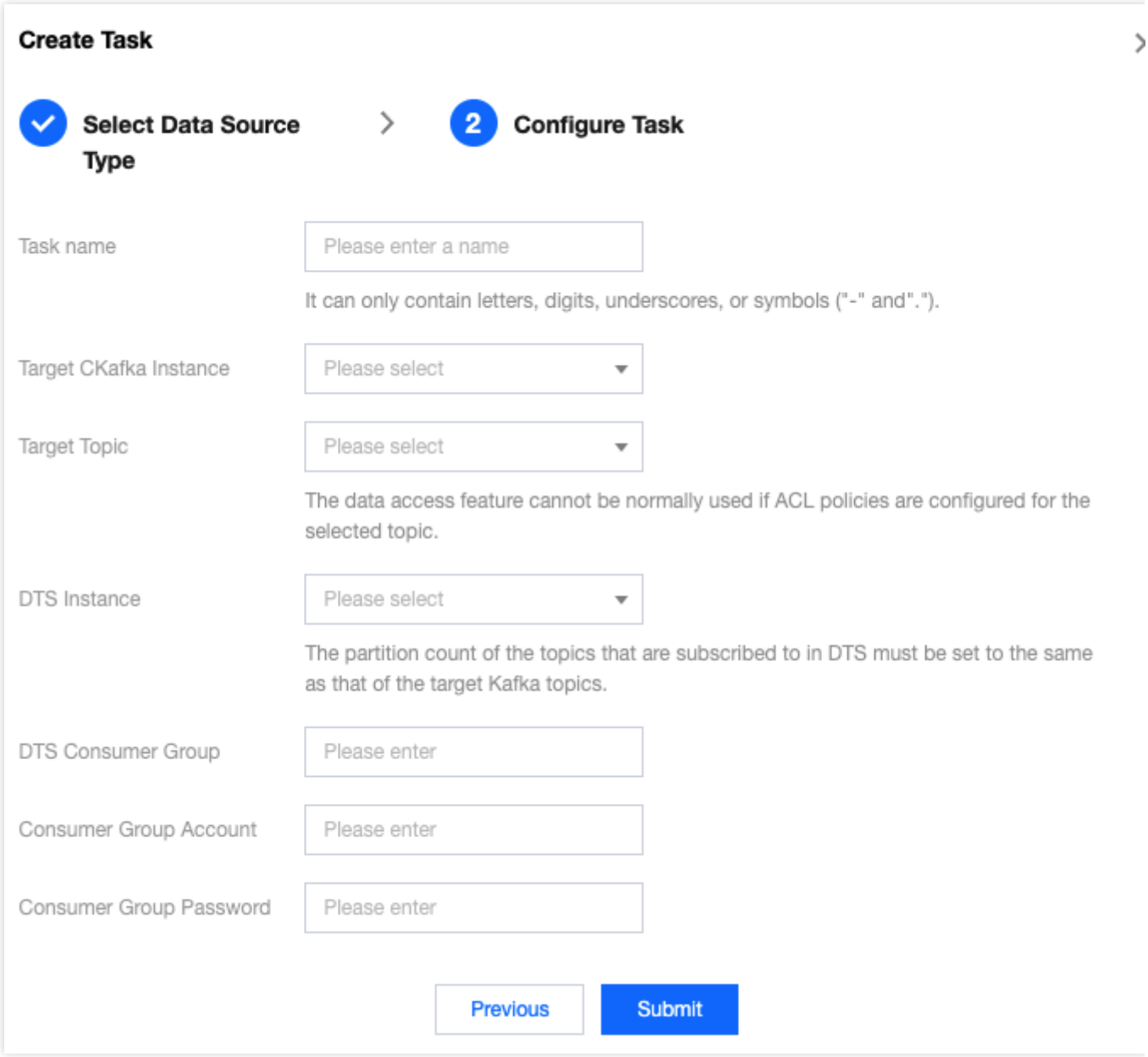
You have created a CKafka instance and a topic.
You have created a DTS instance and a consumer group.

Directions:

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar, select the region, and click **Create Task**.
3. In the pop-up window, select **Asynchronously pulled data** > **DTS** for **Data Source Type**.



4. Click **Next** and enter the task details.



Task Name: It can only contain letters, digits, underscores, or symbols ("-" and ".").

Target CKafka Instance: Select a CKafka instance.

Target Topic: Select the target CKafka topic for data delivery.

DTS Instance: Select a DTS instance. The partition count of the topics that are subscribed to in DTS must be set to the same as that of the target Kafka topics.

DTS Consumer Group: Select a DTS consumer group.

Consumer Group Account: DTS consumer group account.

Consumer Group Password: DTS consumer group password.

5. Click **Submit**.

Changing data source and data target

1. Log in to the CKafka console.

2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.
3. Click **Change Data Target** in the top-right corner of the **Data Target** module to modify the data target information.

Viewing monitoring data

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.
3. Select the **Monitoring** tab to view the monitoring data of the target topic.

Pausing task

On the **Data Access** page, click **Pause** in the **Operation** column of the target task to pause the task.

Note:

If you find that the data access task affects the normal use of CKafka, you can pause it.

Resuming task

On the **Data Access** page, click **Resume** in the **Operation** column of the target task to resume the paused task.

Note:

A paused task can be resumed to continue dumping data.

Deleting task

On the **Data Access** page, click **Delete** in the **Operation** column of the target task and click **OK** in the pop-up window to delete the task.

Note:

Once the task is deleted, data dumping will be stopped and the task record will be deleted, but the previously dumped data and CKafka instance involved will not be affected.

A task cannot be recovered once deleted. Proceed with caution.

MongoDB

Last updated：2024-01-09 14:54:11

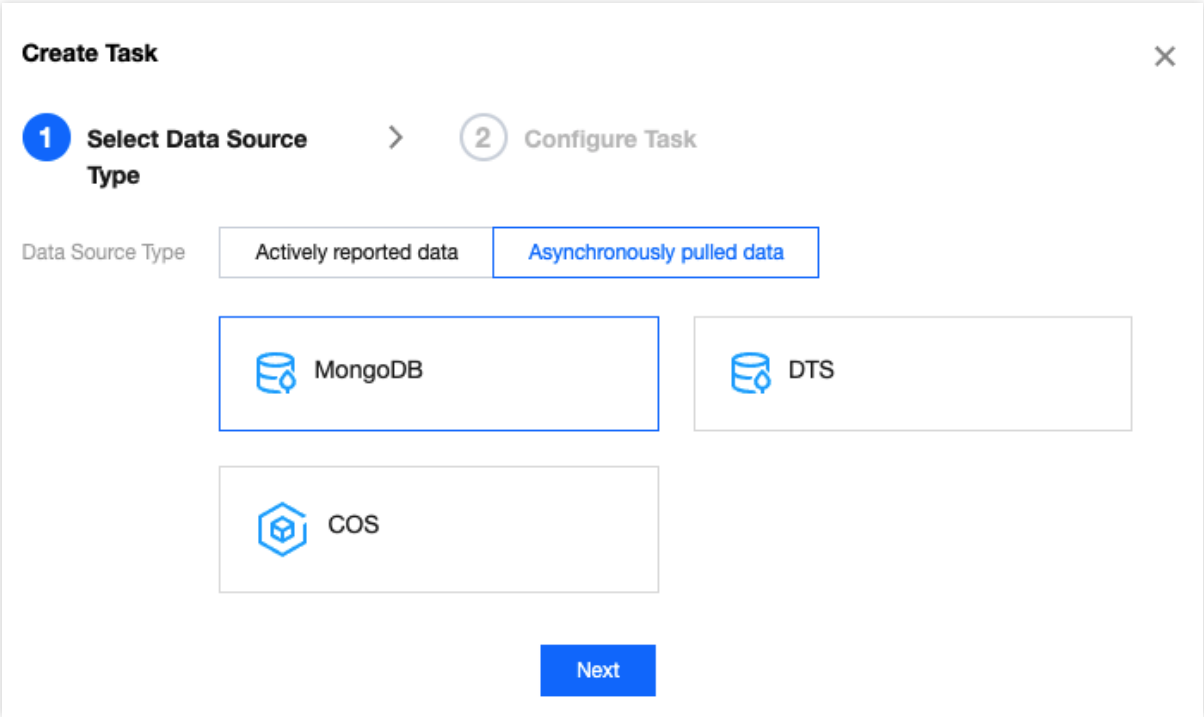
Overview

DataHub supports accessing different types of data generated by various data sources for unified management and distribution to downstream offline/online processing systems, forming a clear data flow channel.
This document takes MongoDB as an example to describe how to create an async data pull task and modify the task configuration in the CKafka console.

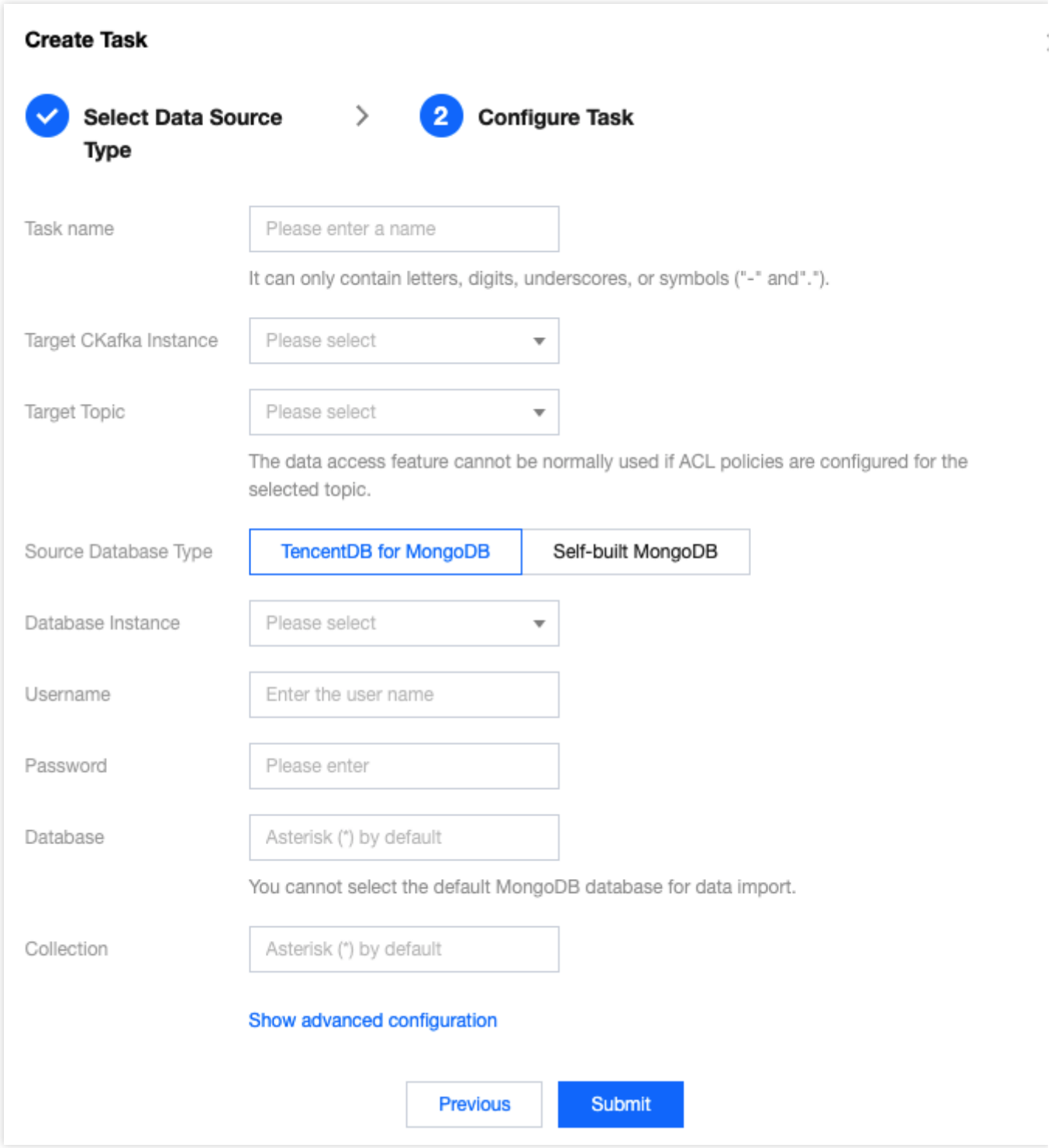
Directions

Creating data access task

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar, select the region, and click **Create Task**.
3. In the pop-up window, select **Asynchronously pulled data** > **MongoDB** for **Data Source Type**.



4. Click **Next** and enter the task details.



Task Name: It can only contain letters, digits, underscores, or symbols ("-" and ".").

Target CKafka Instance: Select a CKafka instance.

Target Topic: Select the target CKafka topic for data access.

Source Database Type:

TencentDB for MongoDB: Select a database instance.

Self-built MongoDB: Select your CLB instance and specify the port.

Username: Source MongoDB database username.

Password: Source MongoDB database password.
Database: Source MongoDB database name. You cannot select the default MongoDB database for data import.
Collection: Source MongoDB collection. You can keep the default setting, i.e., "", to listen on all collections, or specify a collection.
Copy Existing Data: Specify whether to replicate the existing data in the source MongoDB database.
5. Click **Submit**.

Changing data source and data target

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.
3. Click **Change Data Source** in the top-right corner of the **Data Source** module to modify the data source information.
4. Click **Change Data Target** in the top-right corner of the **Data Target** module to modify the data target information.

Viewing monitoring data

1. Log in to the CKafka console.
2. Click **Data Access** on the left sidebar and click the **ID** of the target task to enter its basic information page.
3. Select the **Monitoring** tab to view the monitoring data of the target topic.

Pausing task

On the **Data Access** page, click **Pause** in the **Operation** column of the target task to pause the task.
Note:
If you find that the data access task affects the normal use of CKafka, you can pause it.

Resuming task

On the **Data Access** page, click **Resume** in the **Operation** column of the target task to resume the paused task.
Note:
A paused task can be resumed to continue dumping data.

Deleting task

On the **Data Access** page, click **Delete** in the **Operation** column of the target task and click **OK** in the pop-up window to delete the task.
Note:
Once the task is deleted, data access will be stopped and the task record will be deleted, but the previously dumped data and CKafka instance involved will not be affected.
A task cannot be recovered once deleted. Proceed with caution.

Creating Data Distribution Task
Data Target
ClickHouse

Last updated：2024-09-09 21:49:50

Overview

CKafka Connector provides data distribution capabilities. You can distribute CKafka data to the data warehouse ClickHouse for storage, query, and analysis.

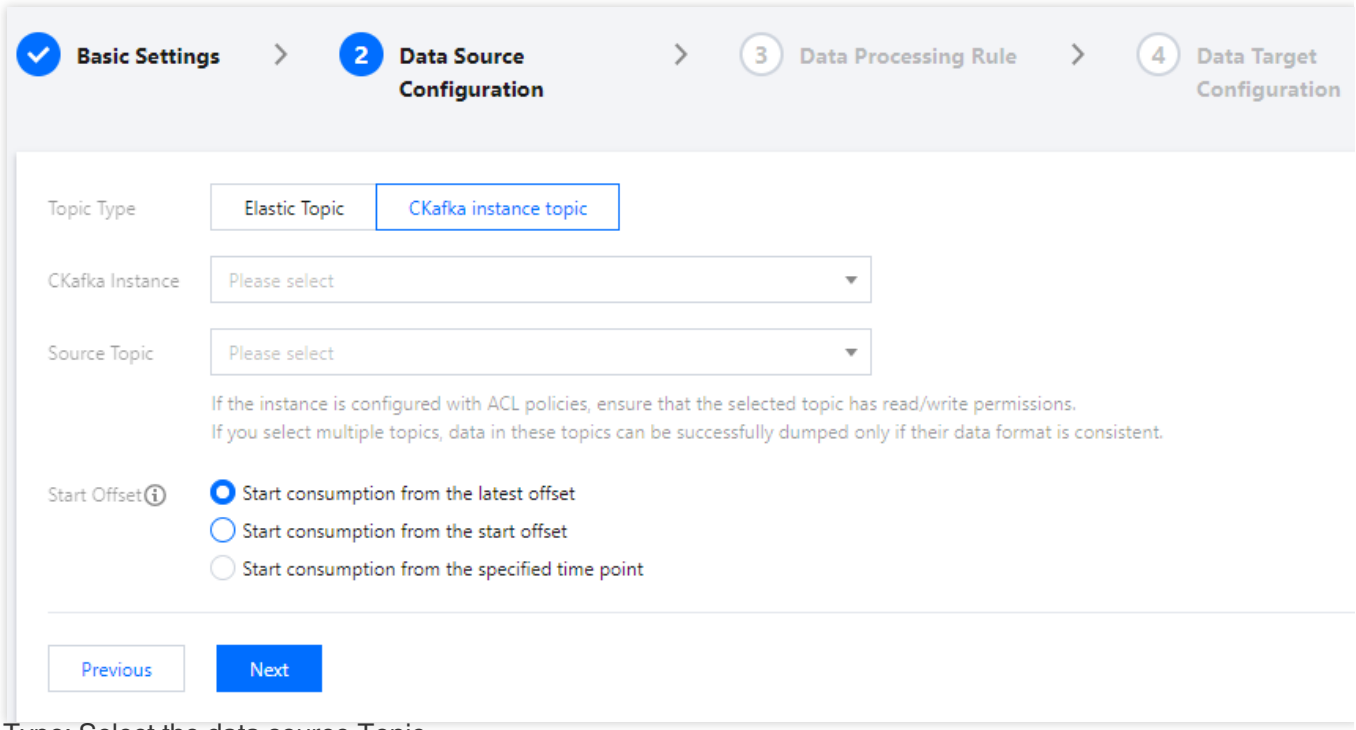
Prerequisites

If you are using Tencent Cloud's managed ClickHouse products, you need to enable the relevant product feature. It also supports data distribution to self-built ClickHouse.
Create a table in ClickHouse and specify a column and type during table creation.
A connection to the target ClickHouse for data distribution has been created.

Directions

Creating a Task

1. Log in to the CKafka Console .
2. In the left sidebar, click **Connectors** > **Task List** , select the right region, and then click **Create Task** .
3. Fill in the task name, select **Data Distribution** as the task type, and select **ClickHouse** as the data target type, click **Next** .
4. Configure data source information.



Topic Type: Select the data source Topic.

Elastic Topic: Select the pre-created elastic Topic. For details, see Topic Management.

CKafka Instance Topic: Select the instance and Topic created in CKafka. If the instance has ACL policies configured, ensure the selected topic has read and write permissions. For details, see Creating Topic.

Start Offset: Select how to handle historical messages during dump by setting the Topic offset.

5. After setting the above information, click **Next** , click **Preview Topic Message** , and the first message from the **source topic** will be obtained and parsed.

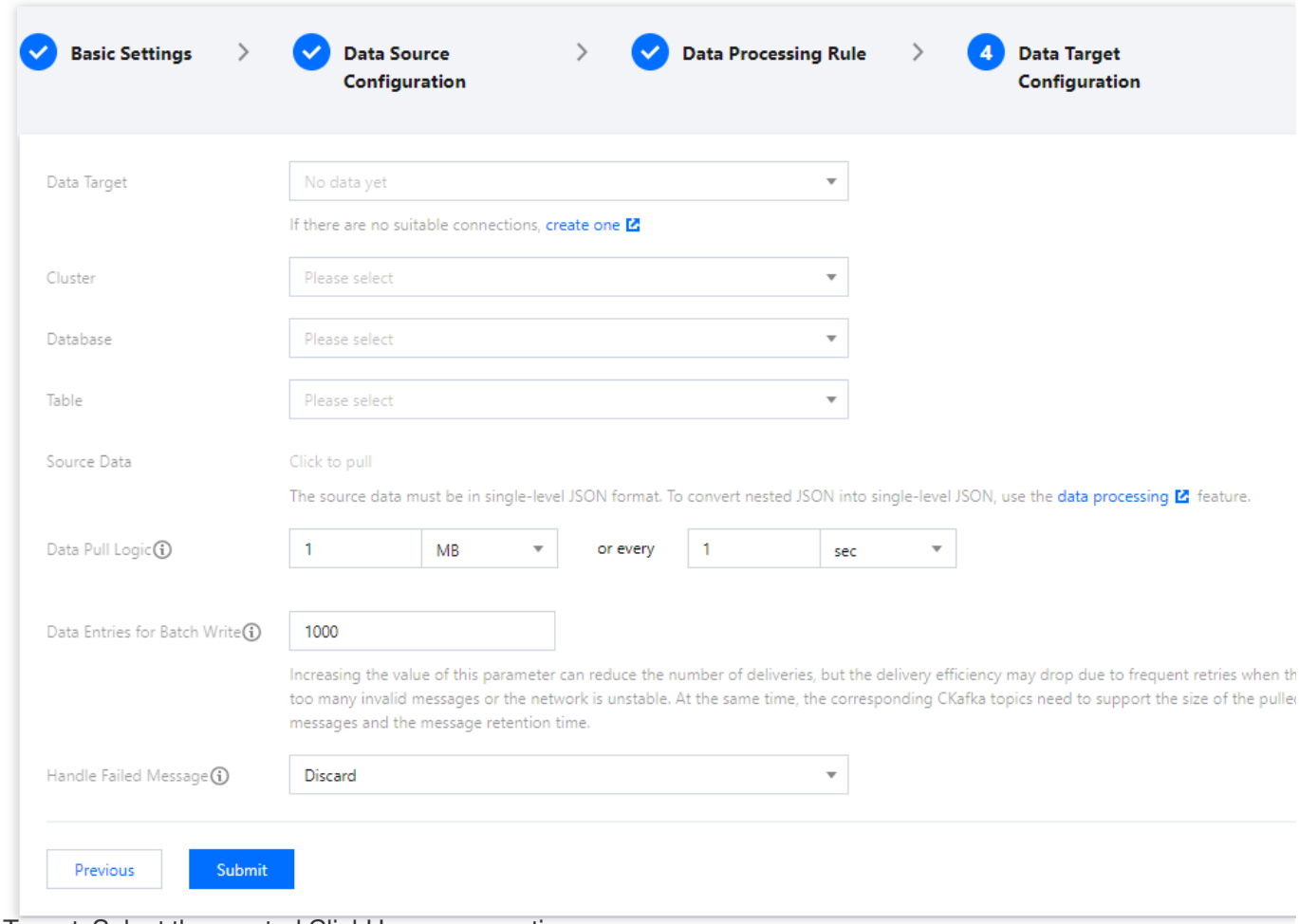
Currently, message parsing should meet the following requirements:

The message is a JSON string.

Source data should be in single-level JSON format. For nested JSON, you can use Data Processing to perform simple message format conversion.

6. (Optional) Enable the **Processing Source Data** button for source data. For detailed configuration, please see Simple Data Processing.

7. Click **Next** to configure the data target information.



Data Target: Select the created ClickHouse connection.

Cluster: The ClickHouse cluster name (Default is `default_cluster` .).

Database: The ClickHouse database name.

Table: The name of the table created in the database. Currently, no table will be created automatically during data distribution to ClickHouse. **You need to manually create the current ClickHouse target table.** .

Source Data: Click to pull data from the source topic. Source data should be in single-level JSON format. For nested JSON format, you can use Data Processing++ to perform conversion.+

Handle Failed Message: Select how to handle messages that failed to deliver. Supports **Discard** , **Retain** and **Ship to CLS** (requires specifying the target log set and log topic and granting access to CLS) three methods.

Retain: It is suitable for test environments. The task will terminate without retrying if it fails to run and will record the failure reason in the Event Center.

Discard: It is suitable for production environments. The task will ignore the current failure message if it fails to run. It is recommended to use the Retain mode for test without errors before editing the task to Discard mode.

Delivery to CLS: Suitable for strict production environments. If a task fails to run, the failed messages, metadata, and reasons for failure will be uploaded to the specified CLS topic.

8. Click **Submit**. You can see the created task in the task list and view the task creation progress in the status bar.

CLS

Last updated：2024-09-09 21:54:16

Overview

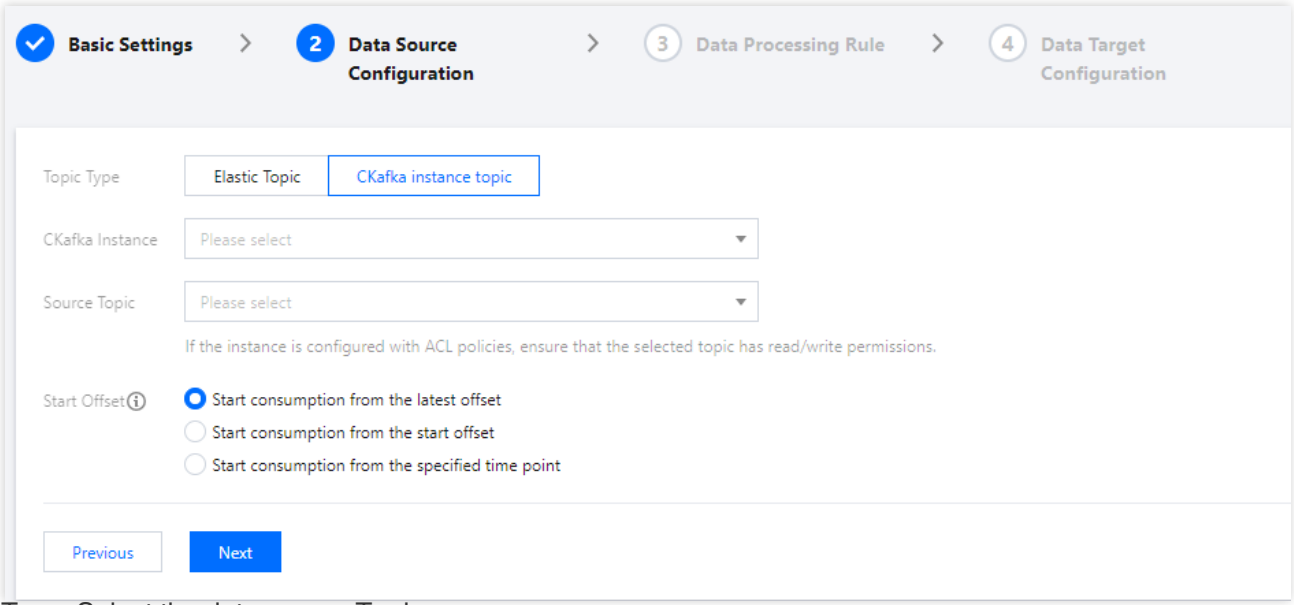
The CKafka Connector provides data distribution capabilities. You can distribute CKafka data to CLS for business troubleshooting, metric monitoring, security auditing, and other daily issues.

Prerequisites

Currently, this feature depends on the CLS service, and you should enable the relevant product features to use it.

Directions

1. Log in to the CKafka Console.
2. In the left sidebar, click **Connectors** > **Task List** , select the right region, and then click **Create Task** .
3. Fill in the task name, select **Data Distribution** as the task type, select **Cloud Log Service(CLS)** as the data target type, then Click **Next** .
4. Configure data source information.



Topic Type: Select the data source Topic.

Elastic Topic: Select the pre-created elastic Topic. For details, see Topic Management.

CKafka Instance Topic: Select the instance and Topic created in CKafka. If the instance has ACL policies configured, ensure the selected topic has read and write permissions. For details, see Creating Topic.

Start Offset: Select how to handle historical messages during dump by setting the Topic offset.

5. After setting the above information, click **Next** , click **Preview Topic Message** , and the first message from the **source topic** will be obtained and parsed.

Note

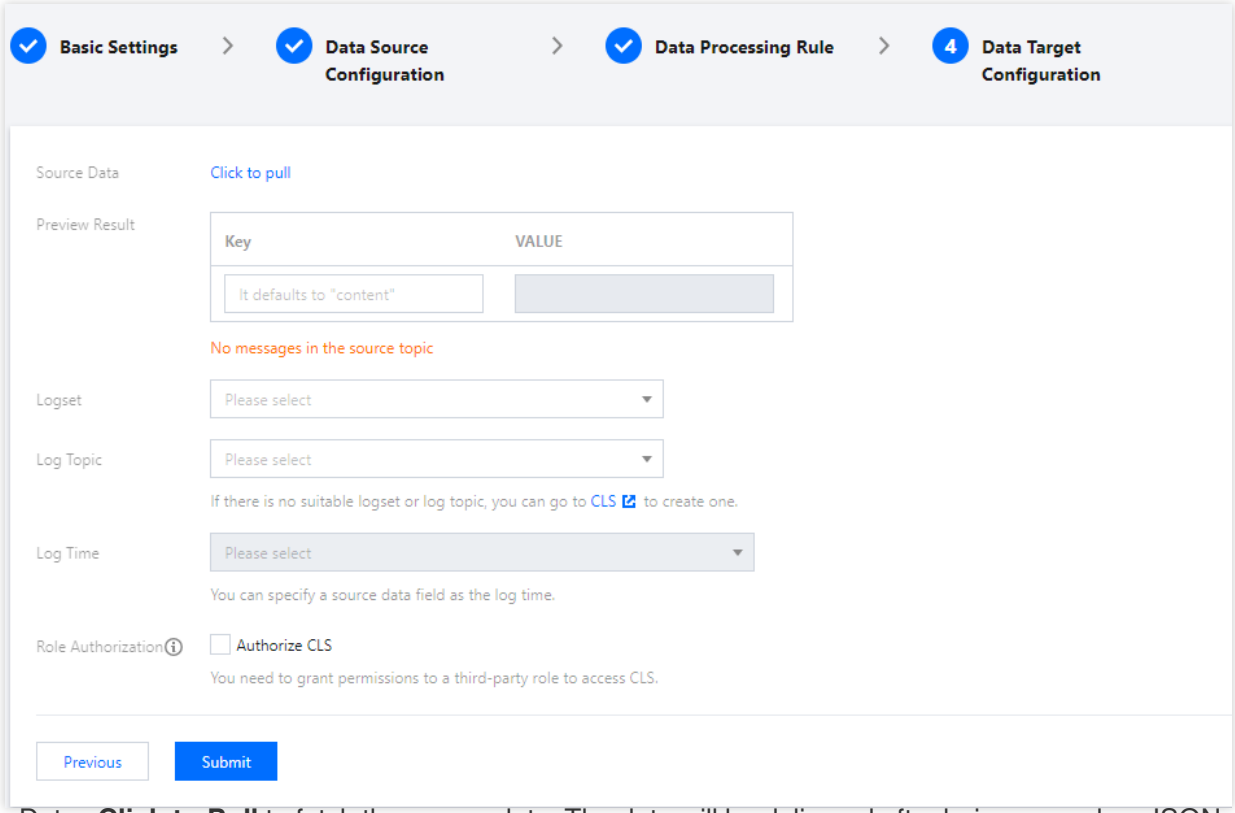
Currently, message parsing should meet the following requirements:

The message is a JSON string.

The source data should be in a single-layer JSON format. For nested JSON, you can useData Processing to perform simple message format conversion.

6. (Optional) Enable the **Process Source Data** button for source data. For detailed configuration, please see Simple Data Processing.

7. Click **Next** to configure the data target information.



Source Data: **Click to Pull** to fetch the source data. The data will be delivered after being parsed as JSON.

KEY: When the data in the source Topic is not in JSON format, you can specify a key to assemble it into JSON for delivery to CLS. The default is content.

Logset: Select a logset. A logset is a project management unit in CLS and is used to distinguish between logs from different projects.

Log Topic: Automatically create a log topic or select an existing one. A Log Set can contain multiple log topics, with each log topic corresponding to a type of application or service. It is recommended to collect similar logs from different machines into the same log topic.

Log Time: You can specify a field in the source data as the log time.
Role Authorization: To use CLS product features, you need to grant a third-party role the permissions to access the related products on your behalf.
8. Click **Submit** . You can see the newly created task in the task list and view the task creation progress in the status bar.

COS

Last updated：2024-09-09 21:57:25

Overview

The CKafka Connector provides data distribution capabilities, allowing you to distribute CKafka data to COS for analysis, download and other operations.

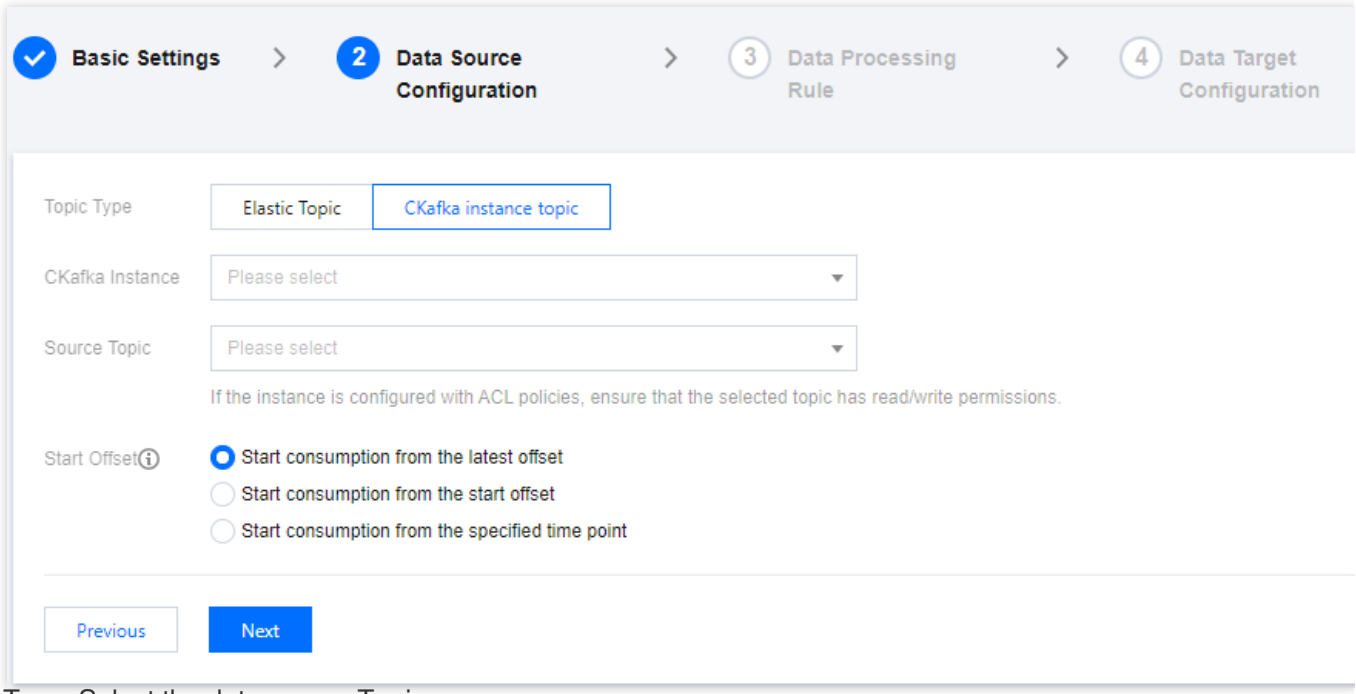
Prerequisites

Currently, this feature depends on the COS service, and you should enable the relevant product features to use it. Ensure that the target bucket for data distribution has been created.

Directions

Creating a Task

1. Log in to the CKafka Console .
2. In the left sidebar, click **Connectors** > **Task List** , select the right region, and then click **Create Task** .
3. Fill in the task name, select **Data Distribution** as the task type, select **Cloud Object Storage (COS)**as the data target type,then click **Next** .++
4. Configure the data source information.



Topic Type: Select the data source Topic.

Elastic Topic: Select the pre-created elastic Topic, for details see Topic Management.

CKafka Instance Topic: Select the instance and Topic created in CKafka. If the instance has ACL policies configured, ensure the selected topic has read and write permissions. For details see Creating Topic.

Start Offset: Select how to handle historical messages during the dump by setting the Topic offset.

5. After setting the above information, click **Next** , click **Preview Topic Message** , and the first message from the **source topic** will be obtained and parsed.

Note

Currently, message parsing should meet the following requirements:

The message is a JSON string.

The source data should be in a single-layer JSON format. For nested JSON, you can useData Processing to perform simple message format conversion.

6. (Optional) Toggle on the **Process Source Data** for source data. For detailed configuration, see Simple Data Processing.

7. Click **Next** to configure the data target information.

Target Bucket: Select the target bucket for data delivery.

Directory Prefix: Supports custom directory prefixes. Log files will be delivered to this directory in the COS Bucket.

Partition Format: Automatically generates directories based on the task creation time using the syntax of strftime, where the slash `/` indicates a first-level COS directory.

Fill out the partition format according to the strftime format requirements. Different partition formats will affect the file path delivered to COS. Below is an example of how to use the partition format, e.g., if the data is delivered to the bucket_test bucket, the directory prefix is `logset/` ; for a delivery time of 2018/7/31 17:14, the corresponding delivery file path is as follows:

Bucket Name	Directory Prefix	Partition Format	COS File Path
bucket_test	logset/	%Y/%m/%d	bucket_test:logset/2018/7/31_{random}_{index}
bucket_test	logset/	%Y%m%d/%H	bucket_test:logset/20180731/14_{random}_{index}
bucket_test	logset/	%Y%m%d/log	bucket_test:logset/20180731/log_{random}_{index}

Aggregation Method:

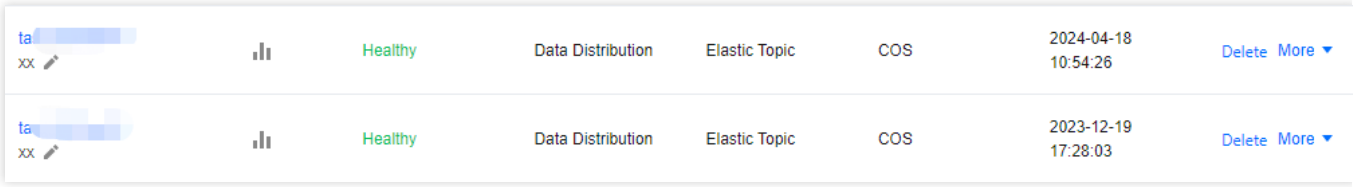
Single File Size: Specify the maximum size of an uncompressed delivery file within the delivery interval. If the log file exceeds this limit within the interval, it will be split into multiple log files.

By Time: Select an aggregation interval based on message volume, ranging from 1 minute to 24 hours.

Storage Format: Supports CSV and JSON formats.

Role Authorization: To use COS product features, you need to grant a third-party role the permission to access to related product on your behalf.

8. Click **Submit** . You can see the created task in the task list and check its creation progress in the status bar.



ES

Last updated：2024-09-09 21:58:05

Overview

The CKafka connector offers data distribution capabilities, allowing you to distribute CKafka data to Elasticsearch Service (ES) for massive data storage and search, real-time log analysis, and other operations.

Note:

Only ES 7.0 and later are supported.

Prerequisites

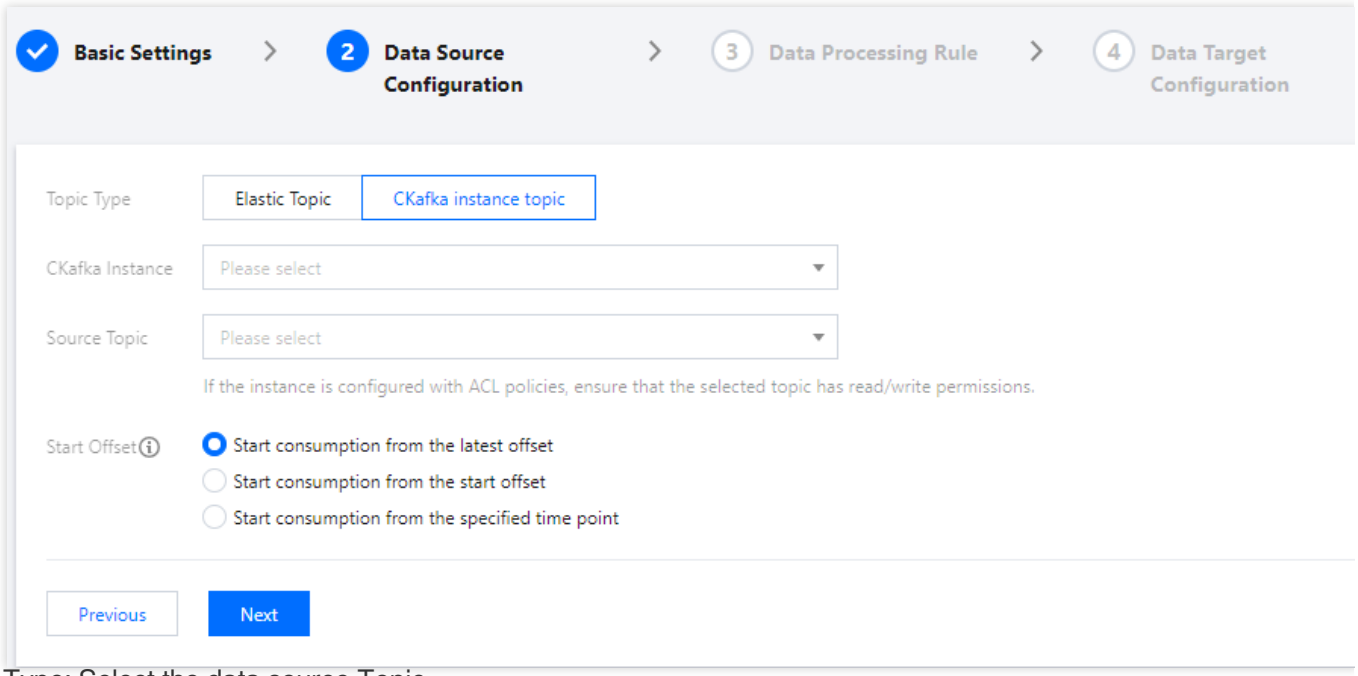
Currently, this feature depends on the ES service, and you should enable the relevant product features to use it.

A data distribution target ES connection has been successfully established.

Directions

Creating Data Distribution Task

1. Log in to the CKafka Console .
2. In the left sidebar, click **Connectors** > **Task List** , select the right region, and then click **Create Task** .
3. Fill in the task name, select **Data Distribution** as the task type, select **ES** as the data target type, and click **Next** .
4. Configure the data source information.



Topic Type: Select the data source Topic.

Elastic Topic: Select the pre-created elastic Topic. For details, see Topic Management.

CKafka Instance Topic: Select the instance and Topic created in CKafka. If the instance has ACL policies configured, ensure the selected topic has read and write permissions. For details, see Creating Topic.

Start Offset: Select how to handle historical messages during dump by setting the Topic offset.

5. After setting the above information, click **Next** , click **Preview Topic Message** , and the first message from the **source topic** will be obtained and parsed.

Note

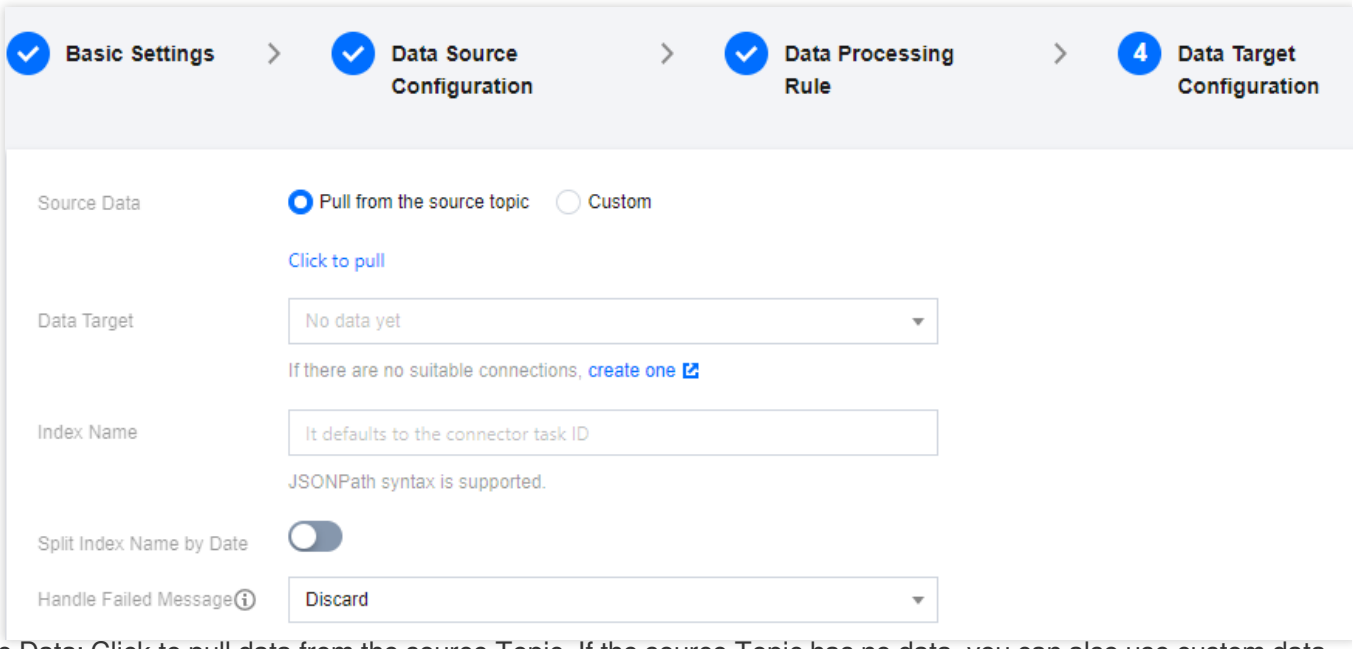
Currently, message parsing should meet the following requirements:

The message is a JSON string.

The source data should be in a single-layer JSON format. For nested JSON, you can use Data Processing to perform simple message format conversion.

6. (Optional) Toggle on the **Process Source Data** for source data. For detailed configuration, please see Simple Data Processing.

7. Click **Next** to configure the data target information.



Source Data: Click to pull data from the source Topic. If the source Topic has no data, you can also use custom data.

Data Target: Select the target of the pre-created data stream ES connection.

Index Name: Enter the index name. The index name should be in lowercase and support jsonpath syntax.

Split Index Name by Date: Optional. If enabled, you need to choose a date format. The index written to ES will be %(index_name)_%(date).

Handle Failed Message: Select how to handle messages that failed to deliver. Supports **Discard** , **Retain** and **Ship to CLS** (requires specifying the target log set and log topic and granting access to CLS) three methods.

Retain: It is suitable for testing environments. The task will terminate without retrying if it fails to run and will record the failure reason in the Event Center.

Discard: It is suitable for production environments. If the task fails to run, the current failure message will be ignored. It is recommended to use the Retain mode for testing. Once the test is error-free, switch to Discard mode for production.

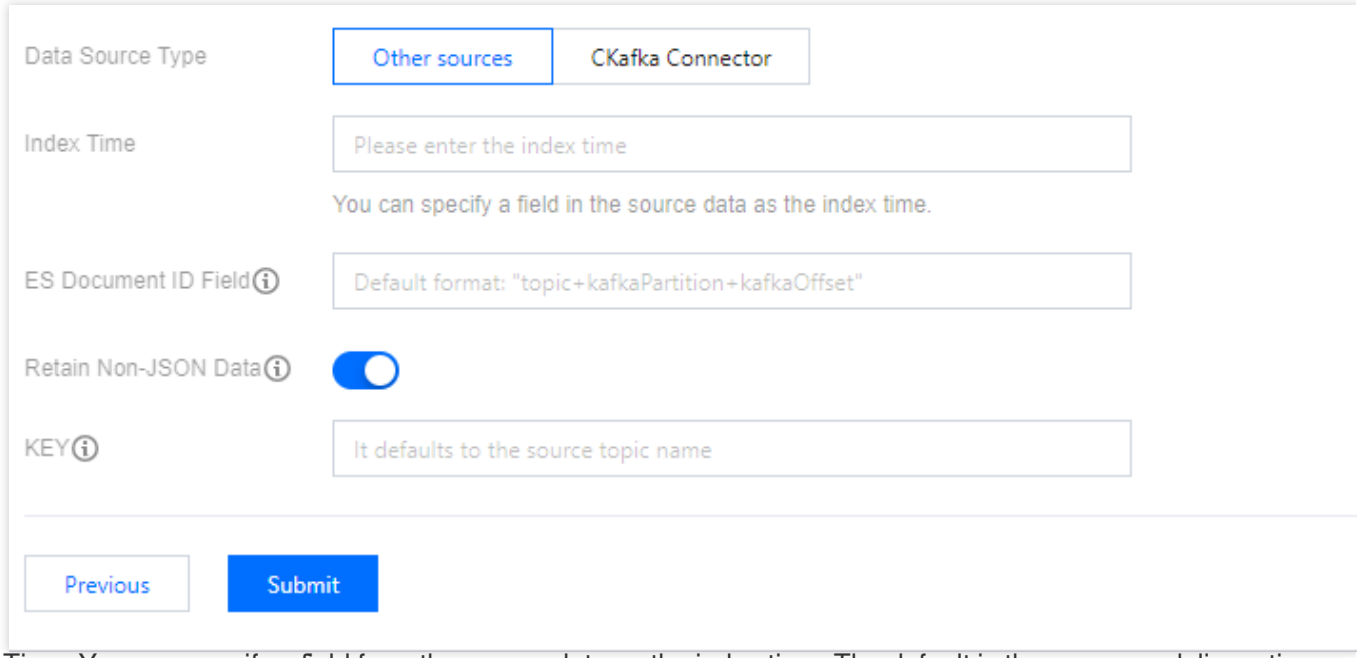
Delivery to CLS: It is suitable for strict production environments. If the task fails to run, the failure message, metadata, and failure reason will be uploaded to the specified CLS topic.

Dead Letter Queue: It is suitable for strict production environments. If the task fails to run, the failure message, metadata, and failure reason will be sent to the specified CKafka Topic.

Data Source Type

Other Data

Data Inside Connector Data Subscription Task

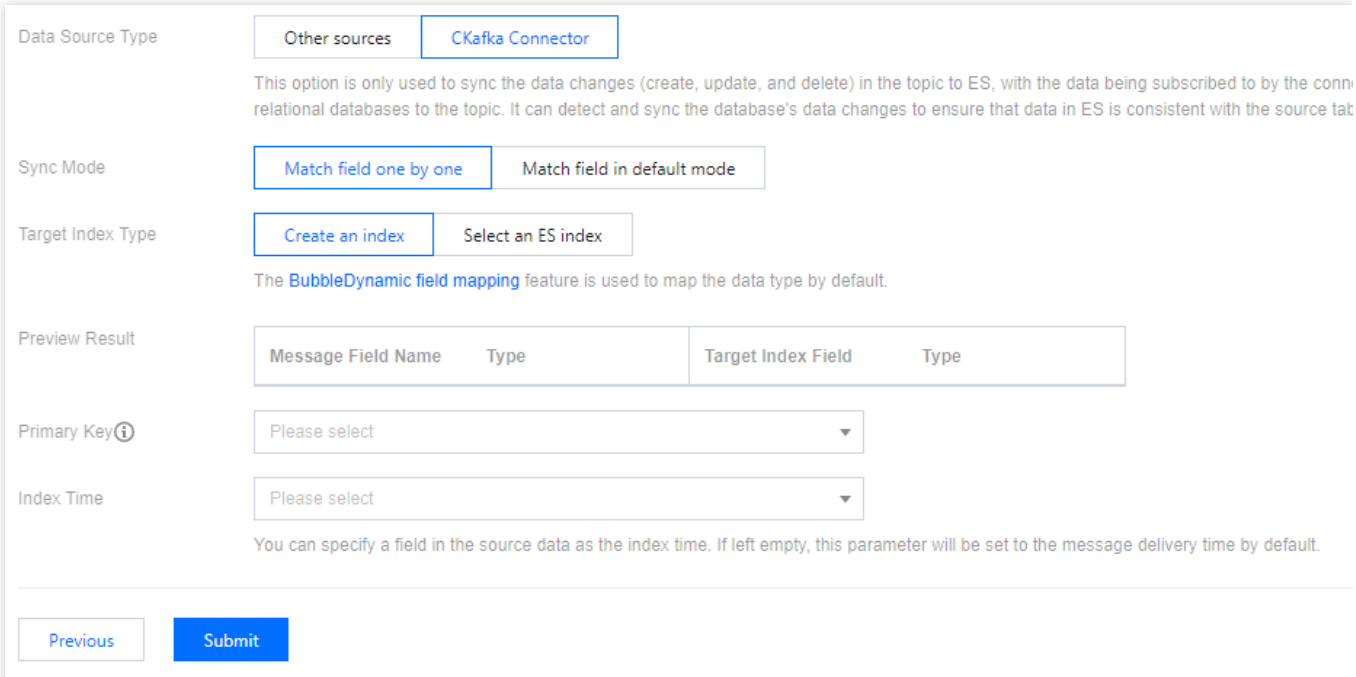


Index Time: You can specify a field from the source data as the index time. The default is the message delivery time.

ES Document ID Field: You can specify the value of this field as the value of ES Document ID. The default is `topic+kafkaPartition+kafkaOffset` .

Retain Non-JSON Data: If enabled, for non-JSON data, a KEY will be designated for assembly and delivery. If disabled, non-JSON data will be discarded.

This option is only used to synchronize updates of data (add, delete, modify) from relational databases to Topic. It will synchronize with ES. It will detect the addition, deletion, and modification of actions in the database to keep the ES data consistent with the source table data.



Sync Mode: If you select **match field one by one**, you can define the mapping relationship between the custom message field names and target index fields. If you select **match field in default mode**, the message key will be

used as the field name in the ES index mapping.

Target Index Type: You can select to create a index or select from an existing ES index.

Primary Key: Specify the primary key of the database table as the value of the ES Document ID.

Index Time: You can specify a field from the source data as the index time. The default is the message delivery time.

8. Click **Submit** . You can see the created task in the task list and view the task creation progress in the status bar.

TDSQL-C PostgreSQL

Last updated：2024-01-09 14:54:11

Overview

CKafka Connector offers data distribution capabilities. You can distribute CKafka data to TDSQL-C for PostgreSQL for further storage, query, and analysis.

Prerequisites

Currently, this feature relies on the TDSQL-C for PostgreSQL service, which should be activated first.

Directions

1. Log in to the CKafka console.
2. Click **Connector** > **Task Management** > **Task List** on the left sidebar, select the region, and click **Create Task**.
3. Enter the task name, select **Data Distribution** as the **Task Type**, select **TDSQL-C for PostgreSQL** as the **Data Target Type**, and click **Next**.
4. Configure the data source information.

Source Topic: Select the data source topic.

Elastic Topic: Select the created elastic topic. For more information, see Topic Management.

CKafka Instance Topic: Select the created CKafka instance and topic. If the instance is configured with ACL policies, ensure that the selected topic has read/write permissions. For more information, see Creating Topic.

Start Offset: Select the topic offset of historical messages when dumping.

5. Click **Next**, click **Preview Data**, and the first message from the specified **Source Topic** will be obtained and parsed.

Note:

Currently, message parsing must meet the following requirements:

The message is a JSON string.

The source data must be in single-level JSON format. To convert nested JSON into single-level JSON, see Data Processing Rule Description.

6. (Optional) Enable **Data Processing Rule**. For more information, see Data Parsing.
7. Click **Next** to configure the data target.

Data Target: Select the created connection to TDSQL-C.

Database: Select the source database.
Table: Select the source table.
Database Sync Mode
Default field match: This option is only used for the following:
The source topic data is the binlog/row-level changes data (insertion, deletion, or modification) of a single table subscribed by CKafka Connector from MySQL/PostgreSQL;
The source topic data must have a primary key and contain a schema.
Field match one by one:
Source Data: Click to pull the source topic data. You need to select the matching fields in the target table for the message fields one by one.
Insertion Mode: This option supports **INSERT** or **UPSERT**. If you select **UPSERT**, you need to select the **primary key** (when the inserted rows conflict, the task will update the columns of the conflicting rows except the primary key).
Upstream Data Format: **JSON** and **Debezium** are supported.
Note:
When the upstream MySQL binlog/PostgreSQL row-level changes data table structure changes, the changes can be synced to the downstream PostgreSQL.
Handle Failed Message: Specify the method of handling failed messages. You can select **Discard**, **Retain**, or **Deliver to CLS** (for this option, you need to specify the target logset and log topic and grant the CLS access).
Retain: This mode is suitable for the test environment. When a task fails, it will be terminated without retry, and the cause of failure will be recorded in the event center.
Discard: This mode is suitable for the production environment. When a task fails, the current failure message will be ignored. We recommend you use the **Retain** mode to conduct a test first and then change the task to **Discard** mode for production.
Deliver to CLS: This mode is suitable for a strict production environment. When a task fails, the failure message, metadata, and cause of failure will be uploaded to the specified CLS topic.
8. Click **Submit**, and you can see the created task in the **Task List**. You can also check the task creation progress on the status bar.

Data Distribution to TDW

Last updated：2024-09-09 21:59:23

Overview

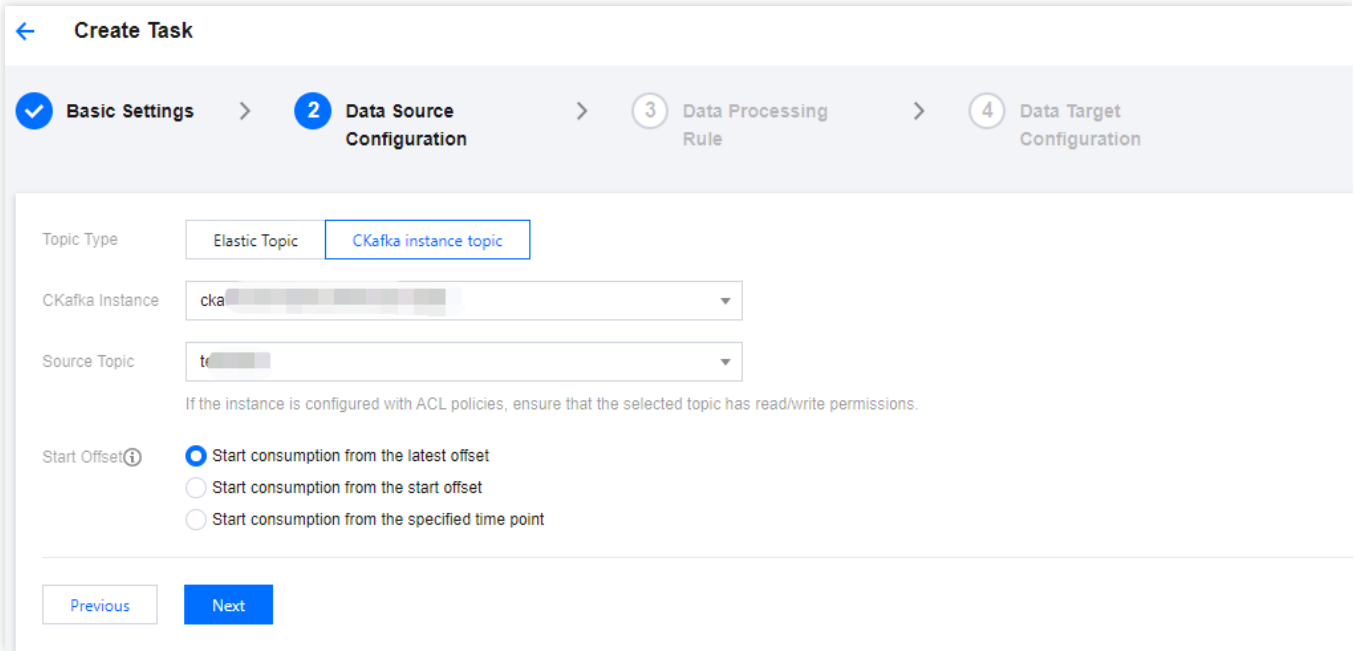
DataHub offers data distribution capabilities. You can distribute CKafka data to TDW for data storage, query, and analysis.

Prerequisites

Currently, this feature depends on the TDW service, and you need to enable the relevant product features to use it.

Directions

1. Log in to the CKafka Console.
2. In the left sidebar, click **Connector** > **Task List** , select the right region, and then click **Create Task** .
3. Fill in the task name, select **Data Distribution** as the task type, select **Data Warehouse (TDW)** as the data target type, then click **Next** .
4. Configure data source information.



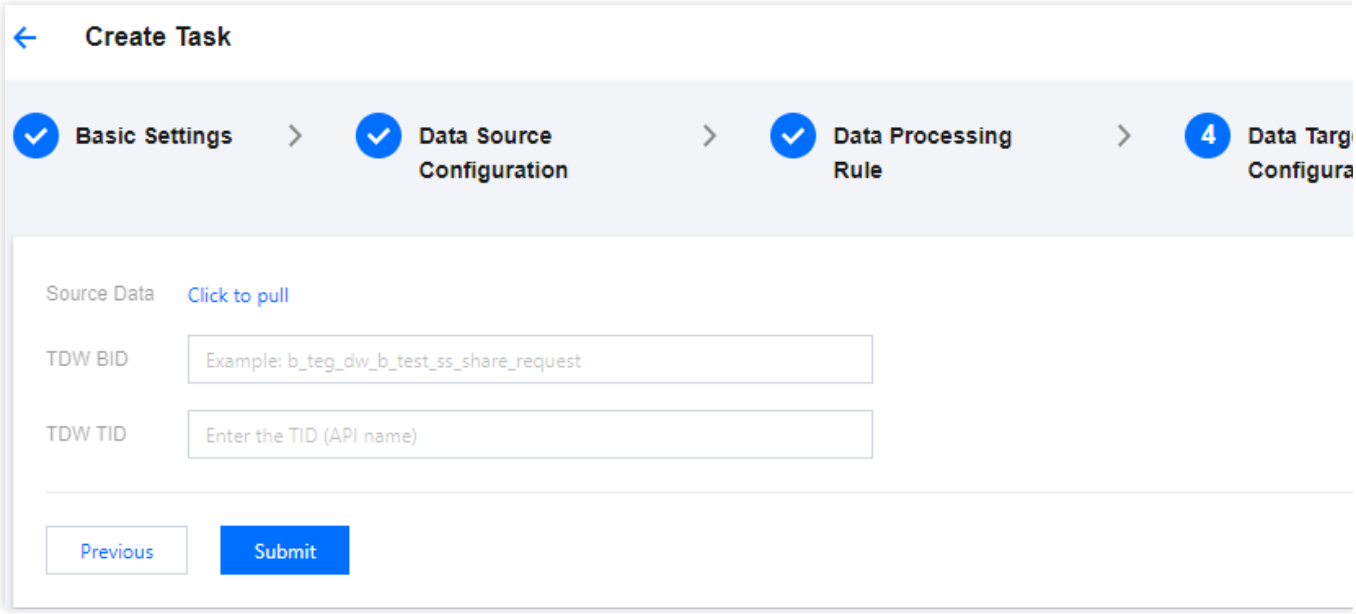
Topic Type: Select the data source Topic

Elastic Topic: Select the pre-created elastic Topic, for details see Topic Management.

CKafka Instance Topic: Select the instance and Topic created in CKafka. If the instance has ACL policies, you need to ensure the selected topic has read and write permissions. For details, please seeTopic Management.
Start Offset: Select how to handle historical messages during dump by setting the Topic offset.
5. After setting the above information, click **Next** , click **Preview Topic Message** , and the first message from the **source topic** will be obtained and parsed.

Note

Currently, message parsing should meet the following requirements:
The message is in JSON string format. JSON keys matching the TDW field names will correspond to the table structure in TDW.
The source data should be in a single-layer JSON format. For nested JSON, you can useData Processing to perform simple message format conversion.
6. (Optional) Toggle on the **Data Processing** button for source data, for detailed configuration, please see Simple Data Processing.
7. Click **Next** to configure the data target information.



Source Data: Click to pull data from the source Topic.
TDW BID: Enter the TDW BID.
TDW TID: Enter the TDW TID.
8. Click **Submit** . You can see the newly created task in the task list and view the task creation progress in the status bar.

Simple Data Processing

Simple Data Processing

Last updated：2024-01-09 14:54:11

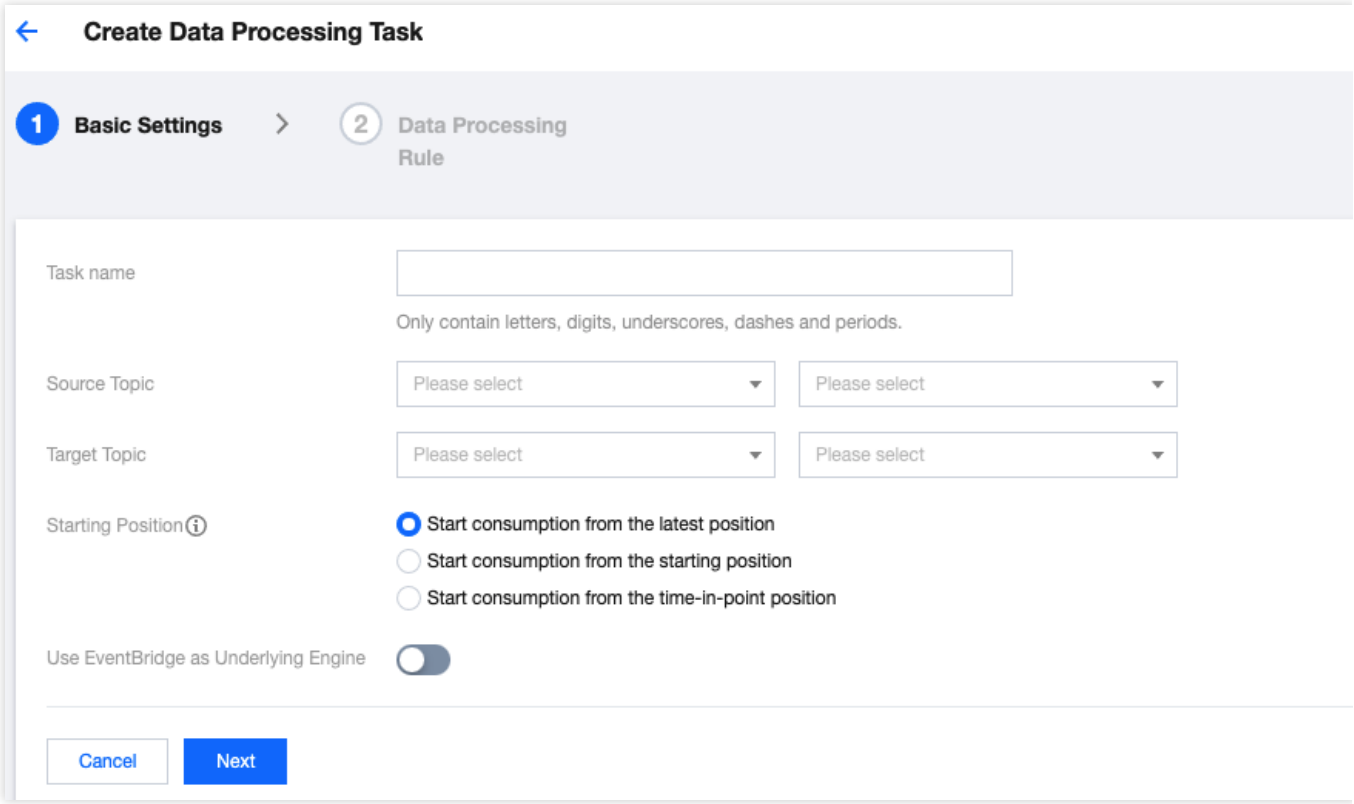
Overview

DataHub provides the simple data processing feature. You only need to pass in data and configuration items, and this feature can format the data, return the processed structured data, and distribute it to offline/online processing systems, connecting data sources to data processing systems.

Directions

Creating rule

1. Log in to the CKafka console.
2. Click **Data Processing** on the left sidebar, select the region, and click **Create Task**.
3. Enter the basic task information.



Task Name: It can only contain letters, digits, underscores, or symbols ("-" and ".").

Source Topic: CKafka topic of the source data.

Target Topic: CKafka topic of the target data.

Starting Position: Select the topic offset of historical messages when dumping.

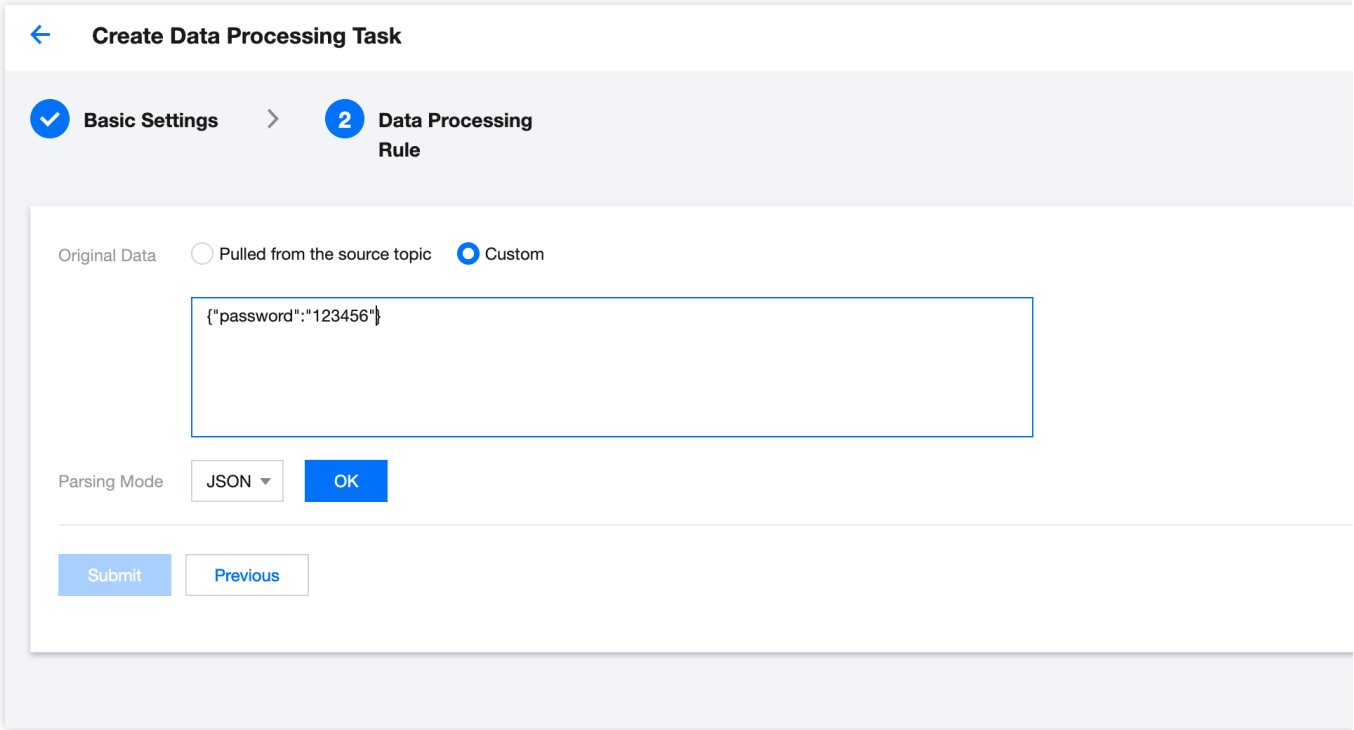
Use EventBridge as Underlying Engine: You can select to use EventBridge as the underlying engine.

Note:

This is supported only in Beijing, Shanghai, and Guangzhou regions.

Role Authorization: To use the EventBridge service as the underlying engine, you need to grant a third-party role to perform access to related products for you.

4. Click **Next** to set a data processing rule.



Original Data: You can select **Pulled from the source topic** or **Custom**.

Parsing Mode: You can select **JSON**, **Separator**, or **Regex**.

JSON.

Separator: You can select a `Space` , `Tab` , `,` , `;` , `|` , or `Custom` .

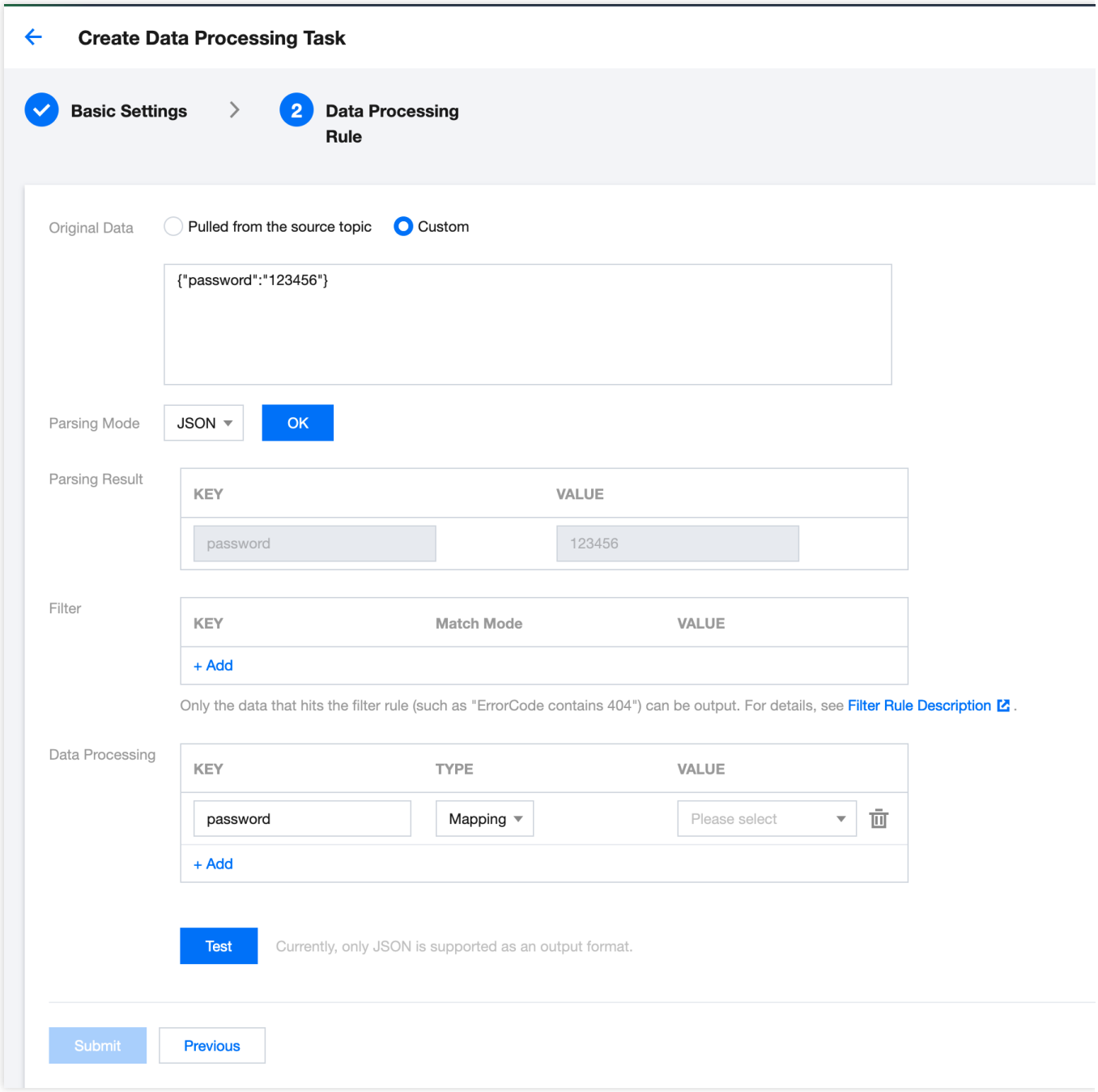
Regex: You need to enter a regex.

5. After selecting the parsing mode, click **OK** to start parsing data.

6. After data parsing is completed, set the filtering rule and data processing mode.

Note:

Currently, only JSON is supported as an output format.



Filter: It outputs only data meeting the filter rules. Supported filter match modes include **By prefix**, **By suffix**, **Contains**, **Except**, **By value**, and **By IP**. For more information, see Filter Rule Description.

Data Processing: Valid values of `TYPE` are **Default**, **Preset**, **Mapping**, **Custom**, and **JSONPATH**.

TYPE = Default: `VALUE` is mapped based on the parsing result and cannot be modified.

TYPE = Preset: You can select a system preset value for `VALUE` . Currently, `DATE` (timestamp) is supported.

TYPE = Mapping: You can select an existing key. The final output value of `VALUE` is mapped by the specified key.

TYPE = Custom: You can enter a custom value for `VALUE` .

TYPE = JSONPATH: Parse nested JSON data. Use `$` as the first character and `.` as the last character to locate a specific field in nested JSON data.

7. Click **Test** to check the test result.
8. Set the failure handling rule.
Retry Interval: Specify the interval for retry upon failure, which can range from 60 seconds to 1 hour.
Retry Count: Specify the maximum number of retries upon failure. After this limit is exceeded, the message will be considered a failed message. The result of `Retry Count` multiplied by `Retry Interval` should be equal to or less than 6 hours.
Handle Failed Message: Specify the method of processing failed messages. You can choose to **discard** or **retain** failed messages, or deliver them to the **dead letter queue** (in this case, you need to specify the dead letter topic).
9. Click **Submit**.

Editing rule

1. In the task list on the **Data Processing** page, click the **ID** of the target task to enter its basic information page.
2. Click **Edit Rule** in the top-right corner of the **Processing Rule** module to modify the data processing rule.

Changing configuration

1. In the task list on the **Data Processing** page, click the **ID** of the target task to enter its basic information page.
2. Click **Change Configurations** in the top-right corner of the **Configuration Information** module to modify the configuration of the data processing rule.

Viewing monitoring data

1. In the task list on the **Data Processing** page, click the **ID** of the target task to enter its basic information page.
2. Enter the basic task information page.
3. At the top of the task details page, click **Monitoring**, select the resource to be viewed, and set the time range to view the corresponding monitoring data.

Pausing task

On the **Data Processing** page, click **Pause** in the **Operation** column of the target task to pause the task.

Note:

As this operation is an async task with a delay, the task status may not change immediately.

Resuming task

On the **Data Processing** page, click **Resume** in the **Operation** column of the target task to resume the paused task.

Note:

A paused task can be resumed to continue processing data.

Deleting task

On the **Data Processing** page, click **Delete** in the **Operation** column of the target task and click **OK** in the pop-up window to delete the task.

Note:

Once the task is deleted, data processing will be stopped and the task record will be deleted, but the previously processed data and CKafka instance involved will not be affected.

A task cannot be recovered once deleted. Proceed with caution.

Data Conversion

Last updated：2024-01-09 14:54:11

CKafka Connector supports data transformation in multiple ways during data processing as described below:

Data transformation

Enter the raw data. Below is an example.

```
{
    "@timestamp": "2022-02-26T22:25:33.210Z",
    "beat": {
        "hostname": "test-server",
        "ip": "6.6.6.6",
        "version": "5.6.9"
    },
    "input_type": "log",
    "message": "{\\"userId\\":888,\\"userName\\":\\"testUser\\"}",
    "offset": 3030131
}
```

CKafka Connector processes the data in the following ways:

Option 1: Quickly define a rule by selecting the corresponding **Process Value** option.

Option 2: Quickly change the data type of a field by selecting the target data format.

Option 3: Implement the join feature through the JSONPath syntax.

For example, you can use the `$.concat($.data.Response.SubnetSet[0].VpcId,"#",$.data.Response.SubnetSet[0].SubnetId,"#",$.data.Response.SubnetSet[0].CidrBlock))` syntax to concatenate the attributes of VPC and subnet and separate them by `#` .

Filter Rule Description

Last updated：2024-01-09 14:54:11

The filter allows you to configure filtering rules such as field sizes to filter data. Only data that meets the specified rules will be retained.

Notes

Filter matching is case-sensitive and accurate down to the character. During matching, no standardized operations will be performed on strings.
Values to be matched must be in JSON format, which include strings and numeric values enclosed in quotation marks as well as keywords not enclosed in quotation marks (`true` , `false` , and `null`).

Prefix Match

You can perform key value matching by comparing a specified prefix with the prefix in data.
For example, for data `{"password":"topicname"}` , you can specify `top` as the prefix of the `password` value so that `{"password":"topicname"}` can be normally matched.

Suffix Match

You can perform key value matching by comparing a specified suffix with the suffix in data.
For example, for data `{"password":"topicname"}` , you can specify `name` as the suffix of the `password` value so that `{"password":"topicname"}` can be normally matched.

Inclusion Match

You can specify a field to be included in data as a match condition.
For example, for data `{"password":"topicname"}` , you can specify `na` to be included in the `password` value so that `{"password":"topicname"}` can be normally matched.

Exclusion Match

You can specify a field to be excluded from data as a match condition.
For example, for data `{"password":"topicname"}`, you can specify `topicname` to be excluded from the `password` value so that `{"password":"topicname"}` cannot be normally matched.

Numeric Match

You can specify the value or value range of a certain field as a match condition.
For example, for data `{ "numeric": 10}`, you can specify the value of `numeric` to be less than 15 (`<15`) as a match condition so that `{ "numeric": 10}` can be normally matched.
The following are examples of value match rules:
Greater than 10: Enter `>10`
Greater than or equal to 10: Enter `>=10`
Greater than or equal to 10, and less than or equal to 20: Enter `>=10&<=20`
Greater than or equal to 10, or less than or equal to 5: Enter `>=10|<=5`

IP Match

You can specify an IP in CIDR notation as a match condition. For example, you can enter `1.2.3.4/24` to match IPs whose leading 24 bits start with "1.2.3.".

Task Management

Last updated：2024-01-09 14:54:12

This document describes how to pause, start, restart, reconfigure, copy, and delete a task in the Ckafka console.

Editing a data source

1. Log in to the CKafka console.
2. On the Task List page, page, click the **ID** of the target task to enter its basic information page.
3. 3. Click **Change Data Source** in the top-right corner of the **Data Source** module to modify the data source information. MySQL and TDSQL for MySQL database subscription tasks support adding subscribed tables. When the user checks the table for adding subscription:
If the original task has selected **Distribute to Multiple Topics** when configuring the data target, after adding a subscribed table, you need to edit the data target and assign the distributed topic to the newly subscribed table.
If the newly subscribed table needs to copy the existing data, you can toggle on **Copy Existing Data**. You need to set the required information as follows:
Copy Existing Data: Select whether to enable it. This option only applies to the newly added tables. For tables that have already been listened on, the original collection logic will apply.
Database for Signaling Table Storage: Select a database to store the signaling table when enabling **Copy Existing Data**. Make sure the user configured on the connection management page has permission to create, modify, and delete a signaling table.

Editing a data target

1. Log in to the CKafka console.
2. On the Task List page, page, click the **ID** of the target task to enter its basic information page.
3. Click **Change Data Target** in the top-right corner of the **Data Target** module to modify the data target information.

Pausing a task

On the Task List page, click **More** > **Pause Task** in the **Operation** column of the target task and click **OK** to pause the task.

Note:

If you find that the data task has already affected the normal use of CKafka, you can pause it.

Starting a task

On the Task List page, click **More** > **Start Task** in the **Operation** column of the target task and click **OK** to start the task.

Note:

A paused task can be resumed to continue dumping data.

Restarting a task

On the Task List page, click **More** > **Restart Task** in the **Operation** column of the target task and click **OK** to restart the task.

Note:

Abnormal tasks can be restarted. The previously dumped data and CKafka instance involved will not be affected.

</td>

Reconfiguring a task

Task creation failures may be caused by incorrect configurations. In this case, you can manually reconfigure a task.

1. On the Task List page, click **More** > **Reconfigure Task** in the **Operation** column of the target task to enter the **Configure Task** page.
2. Specify the new task name, edit the data target, and click **Submit** to reconfigure the task.

Copying a task

When you have a large number of tasks with similar configurations, after creating the first task successfully, you can create more tasks quickly with the task copy feature.

Note:

When creating or copying a task, you are not allowed to create a data task with the same data source and data target.

1. On the Task List page, click **More** > **Copy Task** in the **Operation** column of the target task to enter the **Configure Task** page.
2. Specify the new task name, edit the data target, and click **Submit** to reconfigure the task.

Deleting a task

On the Task List page, click **Delete** in the **Operation** column of the target task and click **OK** to delete the task.

Note:

Once the task is deleted, data dumping will be stopped and the task record will be deleted, but the previously dumped data and CKafka instance involved will not be affected.

A task cannot be recovered once deleted. Proceed with caution.

Schema Management

Last updated：2024-01-09 14:54:11

Overview

With this feature, you can associate a created schema to a specific data access task to verify the format of the accessed data according to the schema.

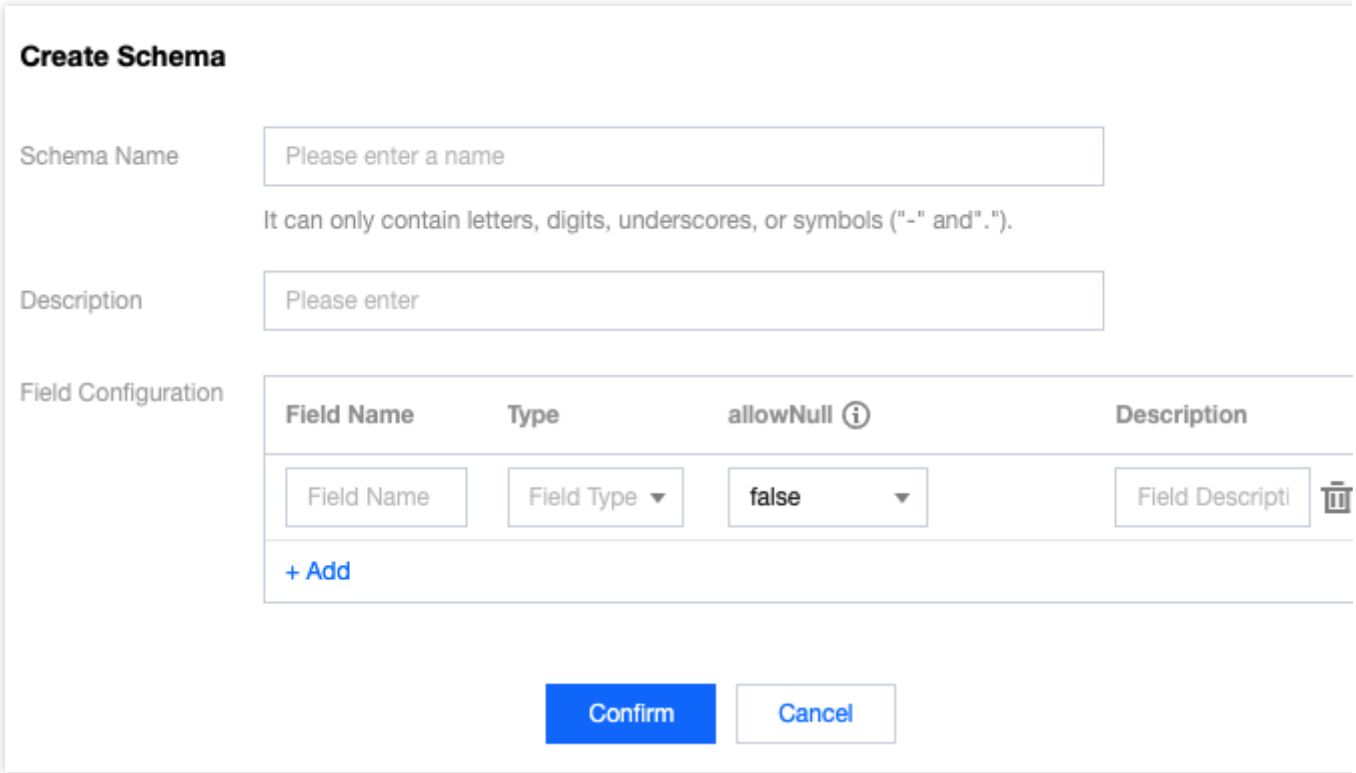
Directions

Creating a schema

The following describes how to create a linkage with **subscribing MySQL data to ES** as an example.

1. Log in to the CKafka console.

2. Select **Connector** > **Schema Management** on the left sidebar, select the region, click **Create Schema**, and enter the schema name.



Schema Name: Enter the schema name. It can only contain letters, digits, underscores, or symbols ("-" and ".").

Description: Enter the optional schema description.

Field Configuration: Add up to 100 fields.

Field Name: Enter the field name.

Type: Eight types are supported, namely, BOOLEAN, INT8, INT16, INT32, INT64, FLOAT32, FLOAT64, and STRING.
allowNull: Verify whether this field exists in the upstream. The value "true" indicates if this field doesn't exist in the upstream, the default value field you specify here will be automatically added instead.
Note:
Detailed description of whether the field configuration is NULL:
true
The upstream field exists and meets the requirements: Write.
The upstream field exists but doesn't meet the requirements: Do not write.
The upstream field doesn't exist: Write the default value.
false
The upstream field exists and meets the requirements: Write.
The upstream field exists but doesn't meet the requirements: Do not write.
The upstream field doesn't exist: Do not write.
Description: Enter the field description.
3. Click **OK**.

Deleting a schema

On the schema management page, click **Delete** in the **Operation** column and click **OK** in the pop-up window.

Associating a task

After a schema is created, it can be associated with a specific data integration task.
1. On the schema management list page, click the ID of the target schema to enter its basic information page.
2. Select the **Associated Task** tab at the top of the page, click **Associate Task**, and select the specific data integration task.

Disassociating a task

1. On the schema management list page, click the ID of the target schema to enter its basic information page.
2. Select the **Associated Task** tab at the top of the page and click **Disassociate** in the **Operation** column of the target associated task.

Note
Once disassociated, the schema no longer applies format verification to the data.

Event Center

Last updated：2024-01-09 14:54:12

The event center manages, stores, analyzes, and displays the event data generated by CKafka Connector in a unified manner. From there you can easily view the event data details. You can also configure alarm policies for events to discover and handle problems promptly.
The events currently supported by CKafka Connector include:
Data source change
Data target change
Data processing rule change
Connection refresh
Task pause
Task resumption
Task restart
This document describes how to view event details and configure event notification rules in CKafka Connector.

Entering the event center

1. Log in to the CKafka console.
2. Click the **ID** of the target task on the **Connector** > **Task List** page to enter the **Basic Info** page.
3. Select the **Event Center** tab at the top of the page, set the time range (last 7 days, last 30 days, or a custom time range), and select the target event type.
Click **View Details** in the **Operation** column to view the event details on the right.

Configuring an event alarm rule

1. On the **Event Details** page, click **Configure Alarm Policy** in the top-right corner to enter the event rule configuration page.
2. Click **Create Event Rule** and configure the event pattern information.
Rule Name: Enter the name of the event rule, which cannot be modified once created.
Event Pattern: Select **Preset Tencent Cloud service events**.
Tencent Cloud Service Type: Select **CKafka Connector**.
Event Type: Select the event type for which you want to configure the alarm rule.
3. Click **Next** to enter the delivery target information.
Trigger Method: Select **Message push**.

Message Template: Two types of push templates are provided based on the event type, and you can choose an appropriate one as needed. For alarm events, we recommend you select the alarm notification template uniformly.
User Notification: Select the notification methods and alarm recipients as needed.
4. Click **Complete**.