ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

DataHub

User Guide

Product Version: v3.16.2

Document Version: 20220819

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.What is DataHub?

DataHub collects, stores, and processes streaming data, allowing you to analyze streaming data and build applications based on the streaming data.

DataHub is a platform designed to process streaming data. You can publish and subscribe to streaming data in DataHub and distribute the data to other platforms. DataHub allows you to analyze streaming data and build applications based on the streaming data.

DataHub collects, stores, and processes streaming data from mobile devices, applications, website services, and sensors. You can use your own applications or Apsara Stack Realtime Compute to process streaming data in DataHub, such as real-time website access logs, application logs, and events. The processing results such as alerts and statistics presented in graphs and tables are updated in real time.

Based on the Apsara system of Alibaba Cloud, DataHub features high availability, low latency, high scalability, and high throughput. DataHub is seamlessly integrated with Realtime Compute, allowing you to use SQL to analyze streaming data.

DataHub can also distribute streaming data to Apsara Stack services such as MaxCompute and Object Storage Service (OSS).

DataHub supports the following features:

- **Data queue**: DataHub automatically generates a cursor for each record in a shard, which can be considered as a logical data queue. The cursor is a unique sequence of numbers. You can improve the performance of a topic by increasing the number of shards in the topic.
- **Offset-based data consumption**: DataHub saves consumption offsets for applications. You can resume data consumption from a saved consumption offset when your application fails.
- **Data synchronization**: Data in DataHub can be automatically synchronized to other Apsara Stack services, including MaxCompute, OSS, AnalyticDB, ApsaraDB RDS for MySQL, Tablestore, and Elasticsearch.
- **Scalable topics**: DataHub allows you to scale in or out topics by splitting or merging shards.

2.Usage notes

Before you use DataHub, get familiar with the limits on specific features.

The following table describes the limits of DataHub.

Limits

Item	Limit	Description
Active shards per topic	(0,256]	Each topic can contain up to 256 active shards.
Shards	(0,512]	You can create up to 512 shards in each topic.
HTTP request body size	Up to 4 MB	The size of the HTTP request body cannot exceed 4 MB.
String size	Up to 1 MB	The size of a string cannot exceed 1 MB.
Merge and split operations on new shards	5s	You cannot merge a shard with another shard or split the shard within the 5s after the shard is created.
Queries per second (QPS)	Up to 5,000	The write QPS limit for each shard is 5,000. Multiple queries in one batch are considered as one query.
Throughput	Up to 5 MB/s	Each shard provides a throughput of up to 5 MB/s.
Projects	Up to 100	You can create up to 100 projects with each account.
Topics per project	Up to 1,000	You can create up to 1,000 topics in each project. Contact the administrator if you need to create more topics.
Time-to-live (TTL) of records	[1,7]	The TTL of each record in a topic ranges from one to seven days.

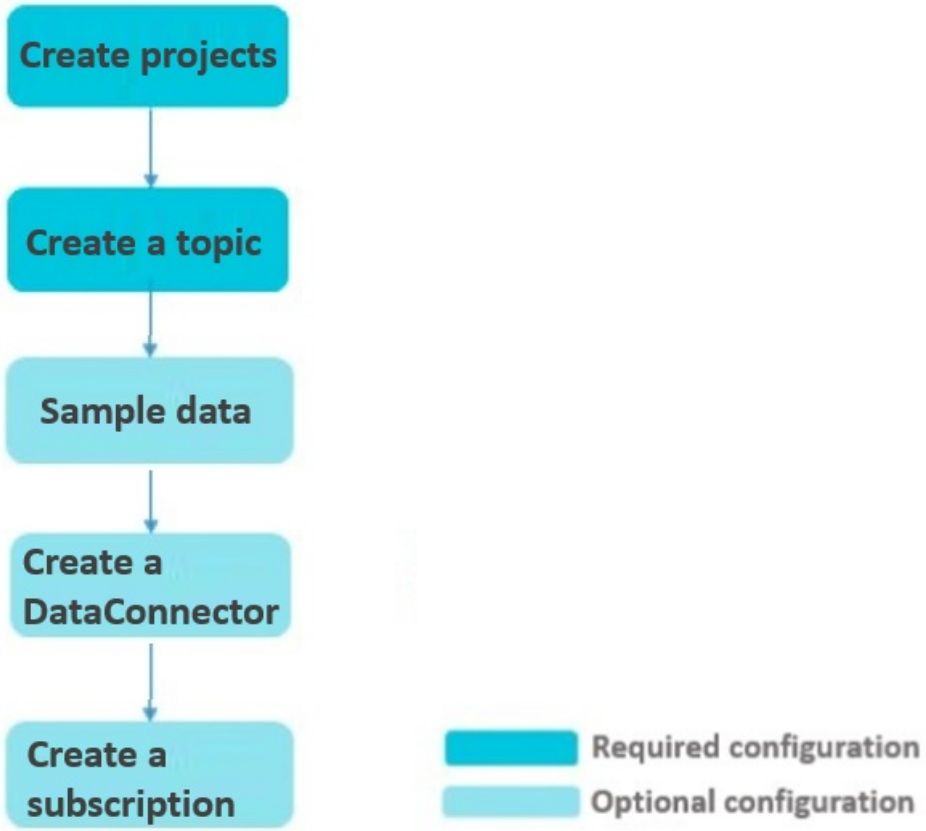
3.Quick Start

3.1. Overview

This topic describes the procedure of using DataHub.

Procedure shows the procedure of using DataHub.

Procedure



1. Create projects.

   A project is an organizational unit in DataHub and contains one or more topics. When you use DataHub, you must create a project first.

2. Create a topic.

   A topic is the smallest unit for data subscription and publication. You can use topics to distinguish different types of streaming data.

3. Optional. Sample data.

   DataHub supports data sampling. You can sample data of a specific shard.

4. Optional. Create a DataConnector.

   You can synchronize real-time data from DataHub to other data warehouses by using DataConnectors so that you can analyze and process historical data.

5. Optional. Create a subscription.

The subscription feature of DataHub supports saving consumption offsets on the server and allows applications to resume data consumption from saved consumption offsets. In addition, DataHub supports resetting offsets to ensure that data can be consumed at least once.

3.2. Log on to the DataHub console

This topic describes how to log on to the DataHub console. Google Chrome is used in this example.

Prerequisites

Before you log on to the DataHub console, make sure that the following requirements are met:

- You have obtained the URL of the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password from the operations administrator.

   > **Note** If you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. To ensure the security of your account, the password must be 8 to 20 characters in length and must contain at least two types of the following characters:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters: exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

3. Click **Login** to go to the Apsara Uni-manager Management Console.
4. In the top navigation bar, choose **Products > Big Data > DataHub** to go to the DataHub console. The **Overview** page appears.

3.3. Create a project

A project is an organizational unit in DataHub and contains one or more topics. When you use DataHub, you must create a project first. This topic describes how to create a project and bind a virtual private cloud (VPC) to the project in the DataHub console.

Prerequisites

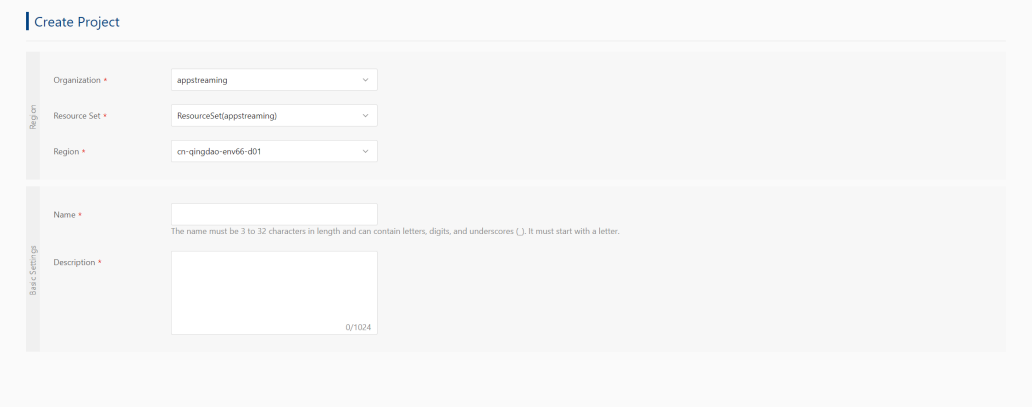
An Apsara Stack tenant account is created.

Considerations

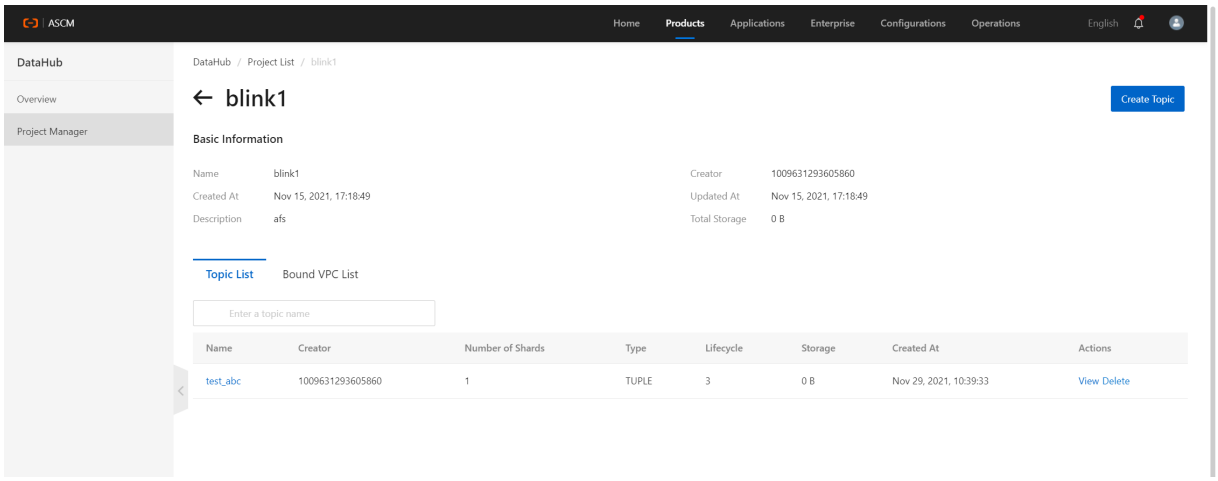
- DataHub projects are independent of MaxCompute projects. Projects you created in MaxCompute cannot be used in DataHub.
- You can create up to 100 projects with each account.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, click **Create Project**. On the Create Project page, set the parameters in the Region and Basic Settings sections and click **Submit**.



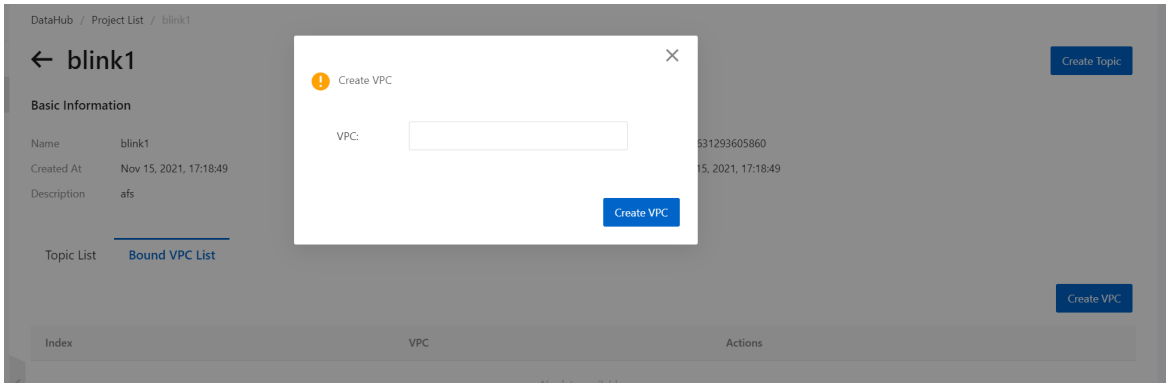
3. After the project is created, you can click **View** in the Actions column to view the details of the created project.



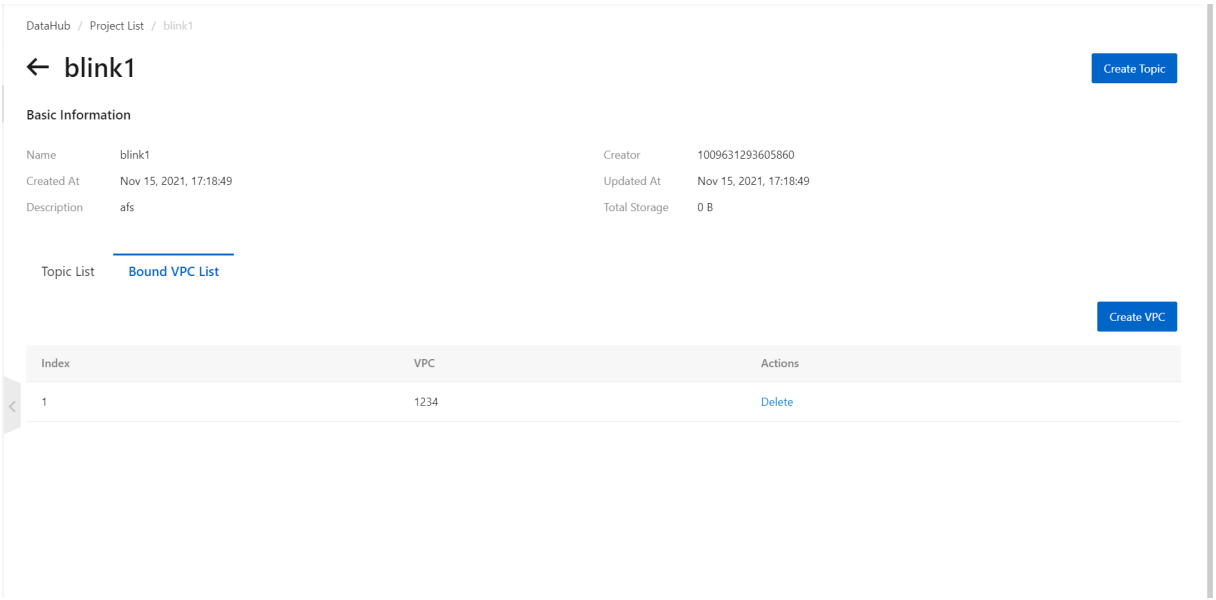
Bind a VPC to a DataHub project

You can bind a VPC to a DataHub project so that the DataHub project can be accessed only from IP addresses in this VPC. Perform the following operations to bind a VPC to a DataHub project:

1. On the project details page, click **Create VPC** on the **Bound VPC List** tab. In the dialog box that appears, enter the name of the VPC and click **Create VPC**.



2. To delete a VPC, click **Delete** in the Actions column.



3.4. Create a topic

A topic is the smallest unit for data subscription and publication. You can use topics to distinguish different types of streaming data. This topic describes how to create a topic in the DataHub console.

Prerequisites

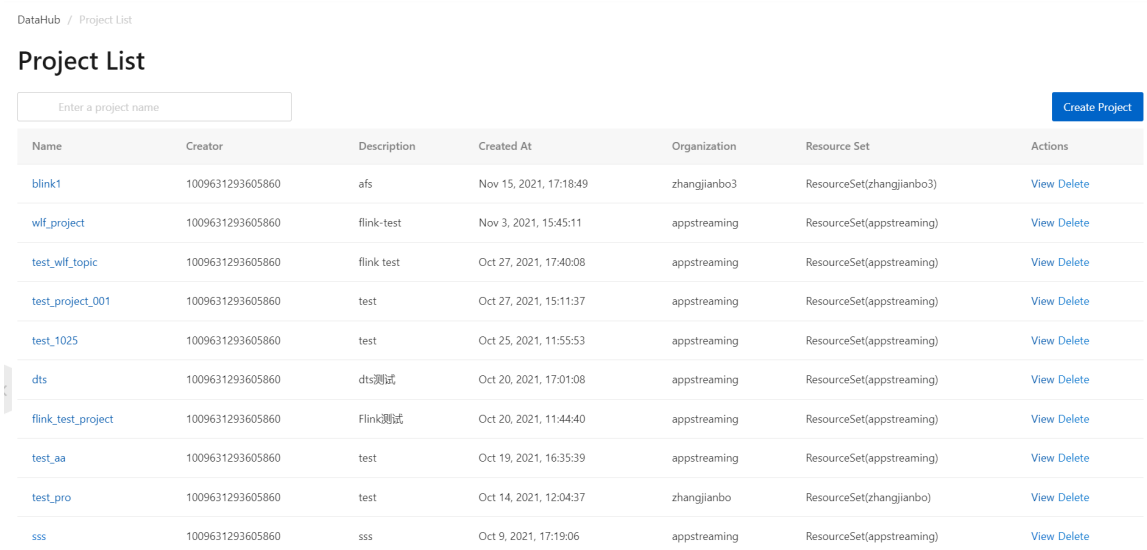
A project is created.

Considerations

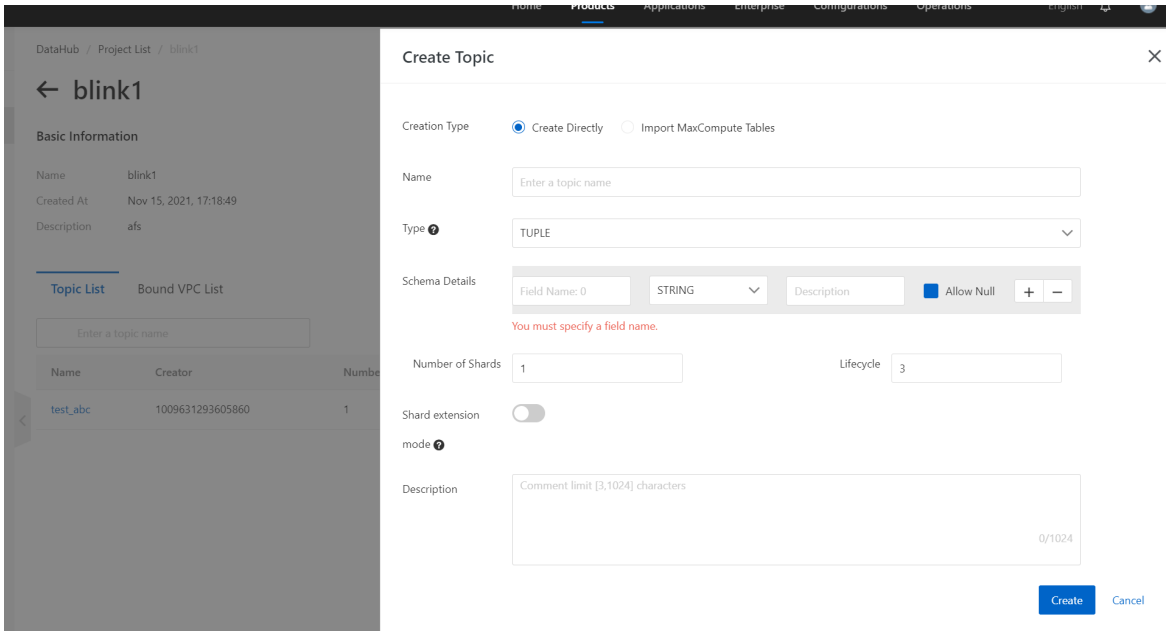
You can create up to 1,000 topics in each project. Contact the administrator if you need to create more topics.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the project for which you want to create a topic and click **View** in the Actions column.



3. On the project details page, click **Create Topic**.
4. In the Create Topic panel, set relevant parameters and click **Create**.



> **Note**
>
> - After a topic is created, you can click **View** in the Actions column to view the details of the created topic.
> - DataHub allows you to directly create a topic or create a topic by importing a table schema from MaxCompute.

3.5. Sample data

DataHub supports data sampling. You can sample data of a specific shard. This topic describes how to sample data in the DataHub console.

Prerequisites

A project and a topic are created and data is written to the topic.

Background information

Before you sample data, you must specify the start time and the maximum number of records that you want to sample.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the Shard List tab of the topic details page, find the target shard and click **Sample** in the Operate column.
4. In the dialog box that appears, specify the start time and the maximum number of records that you want to sample and click **Sample**. DataHub samples the records that are written to the shard after the specified time and displays the sampled records in the table below Sample.

3.6. Create a DataConnector

You can synchronize real-time data from DataHub to other data warehouses by using DataConnectors so that you can analyze and process historical data. This topic describes how to create a DataConnector in the DataHub console.

Prerequisites

A project and a topic are created and data is written to the topic.

Background information

You can synchronize data from DataHub to MaxCompute, AnalyticDB, ApsaraDB RDS for MySQL, Tablestore, OSS, and Elasticsearch.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Connector**. In the Create connector dialog box, select the data warehouse to which data is synchronized.
4. In the Create connector dialog box, set relevant parameters and click **Create**.

3.7. Create a subscription

The subscription feature of DataHub supports saving consumption offsets on the server and allows applications to resume data consumption from saved consumption offsets. In addition, DataHub supports resetting offsets to ensure that data can be consumed at least once. This topic describes how to create a subscription in the DataHub console.

Prerequisites

A project and a topic are created and data is written to the topic.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Subscription**. In the Create subscription dialog box, set relevant parameters and click **Create**.

4.Access Control

4.1. Overview

DataHub allows you to improve data security by granting different permissions to Apsara Stack tenant accounts and RAM user accounts.

DataHub uses Resource Access Management (RAM) for access control. Only users that have been granted the required permissions can access the resources in your department. By default, users do not have permission to access resources in your department. This topic describes how access control for DataHub is achieved by using RAM.

> **Note** An Apsara Stack tenant account is owned by a department and requires no authorization. A RAM user account must be granted permissions by the tenant account.

4.2. DataHub resources in RAM

RAM users can access the following resources in DataHub: projects, topics, and subscriptions. Subscription is the action that you specify an application to read and process the records in topics of a specific project. DataHub supports the RAM authentication of each project, topic, and subscription. RAM authentication is not supported at the shard level.

In RAM, each resource type has an Alibaba Cloud Resource Name (ARN) format to describe the specific object of the resource type. For example, the ARN format of a project that resides in a specific region is *acs:dhs:$region:$accountid:projects/$projectName*. The *$region, $accountid, and $projectName* fields indicate the region that the project resides, the user ID, and the project name.

ARN format for different resource types

Resource type	ARN format
SingleProject	acs:dhs:$region:$accountid:projects/$projectName
AllProject	acs:dhs:$region:$accountid:projects/*
SingleTopic	acs:dhs:$region:$accountid:projects/$projectName/topics/$topicName
AllTopic	acs:dhs:$region:$accountid:projects/$projectName/topics/*
SingleSubscription	acs:dhs:$region:$accountid:projects/$projectName/topics/$topicName/subscriptions/$subId
AllSubscription	acs:dhs:$region:$accountid:projects/$projectName/topics/$topicName/subscriptions/*

4.3. API

DataHub provides application programming interfaces (APIs) for projects, topics, shards, subscriptions, and records. Before you can call the API operations, you must grant corresponding permissions to the RAM user by using RAM authorization policies.

The RAM authorization policy and resource type for each API operation is described as follows:

API operations for projects

API operations for projects

Operation name	RAM authorization policy	Resource type
CreateProject	dhs:CreateProject	AllProject
ListProject	dhs:ListProject	AllProject
DeleteProject	dhs:DeleteProject	SingleProject
GetProject	dhs:GetProject	SingleProject
UpdateProject	dhs: UpdateProject	SingleProject

API operations for topics

API operations for topics

Operation name	RAM authorization policy	Resource type
CreateTopic	dhs:CreateTopic	AllTopic
ListTopic	dhs:ListTopic	AllTopic
DeleteTopic	dhs:DeleteTopic	SingleTopic
GetTopic	dhs:GetTopic	SingleTopic
UpdateTopic	dhs: UpdateTopic	SingleTopic

API operations for subscriptions

API operations for subscriptions

Operation name	RAM authorization policy	Resource type
CreateSubscription	dhs:CreateSubscription	AllSubscription
ListSubscription	dhs:ListSubscription	AllSubscription
DeleteSubscription	dhs:DeleteSubscription	SingleSubscription
GetSubscription	dhs:GetSubscription	SingleSubscription
UpdateSubscription	dhs: UpdateSubscription	SingleSubscription
CommitOffset	dhs:CommitOffset	SingleSubscription
GetOffset	dhs:GetOffset	SingleSubscription

API operations for shards

API operations for shards

Operation name	RAM authorization policy	Resource type
ListShard	dhs:ListShard	SingleTopic
MergeShard	dhs:MergeShard	SingleTopic
SplitShard	dhs:SplitShard	SingleTopic

API operations for shards

API operations for shards

Operation name	RAM authorization policy	Resource type
PutRecords	dhs:PutRecords	SingleTopic
GetRecords	dhs:GetRecords	SingleTopic
GetCursor	dhs:GetRecords	SingleTopic

4.4. Conditions

This section describes conditions that can be applied to the RAM authorization policies for DataHub.

Conditions that can be applied to the RAM authorization policies for DataHub are as follows:

RAM authorization policy conditions for DataHub

Condition keyword	Description	Valid value
acs:SourceIp	The IP address range that can access the specified object.	Any valid IP address. Wildcard masks are supported.
acs:SecureTransport	Indicates whether HTTPS is used to access the specified object.	true/false
acs:MFAPresent	Indicates whether the specified object can be accessed by multiple clients.	true/false
acs:CurrentTime	The time that the specified object can be accessed.	This keyword must be described in ISO 8601 format.

4.5. Sample RAM authorization policy content

4.5.1. AliyunDataHubFullAccess

This section describes how to set the AliyunDataHubFullAccess policy content.

The authorization policy content can be set as follows:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": "dhs:*",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

4.5.2. AliyunDataHubReadOnlyAccess

This section describes how to set the AliyunDataHubReadOnlyAccess policy content.

The authorization policy content can be set as follows:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": ["dhs:List*", "dhs:Get*"],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

5.Data Acquisition

5.1. Overview

In addition to SDK and local file uploads, DataHub supports various data acquisition tools to help you quickly collect data to DataHub.

This section describes how to acquire data by using Fluentd, Logstash, and Oracle GoldenGate (OGG).

5.2. Fluentd

This topic describes how to install and use the DataHub plug-in for Fluentd.

Developed based on the open-source data collector Fluentd, the DataHub plug-in for Fluentd is easy to install and is used to write the collected data to DataHub.

Install the DataHub plug-in for Fluentd

- Install the plug-in by using RubyGems

```
gem install fluent-plugin-datahub
```

> **Notice** We recommend that you change the gem source to https://ruby.taobao.org/.

- Install the plug-in by using a local installation package

  The agent must be installed in Linux. Before you install the agent, install Ruby. For users who have not installed Fluentd, a full installation package for installing both Fluentd and DataHub plug-in is provided. For users who have installed Fluentd, an installation package of the DataHub plug-in is provided.

  - If you have not installed Fluentd, download the full installation package and run the following commands to install Fluentd with the DataHub plug-in:

    > **Notice** Fluentd 0.12.23 is provided in the full installation package.

    ```
    $ tar -xzvf fluentd-with-datahub-0.12.23.tar.gz
    $ cd fluentd-with-dataHub
    $ sudo sh install.sh
    ```

  - If you have installed Fluentd, download the installation package of the DataHub plug-in for Fluentd and run the following command to install the plug-in.

    ```
    $ sudo gem install --local fluent-plugin-dataHub-0.0.2.gem
    ```

Use cases

Case 1: Collect CSV files

This example shows how to write the incremental content of a Comma-Separated Values (CSV) file to DataHub in quasi-real time by using the DataHub plug-in for Fluentd. The following CSV file is used in this example:

```
0,qe614c760fuk8judu01tn5x055rpt1,true,100.1,14321111111
1,znv1py74o8ynn87k66o32ao4x875wi,true,100.1,14321111111
2,7nm0mtpgo1q0ubuljjjx9b000ybltl,true,100.1,14321111111
3,10t0n6pvonnan16279w848ukko5f6l,true,100.1,14321111111
4,0ub584kw88s6dczd0mta7itmta10jo,true,100.1,14321111111
5,1ltfpf0jt7fhvf0oy4lo8m3z62c940,true,100.1,14321111111
6,zpqsfxqy9379lmcehd7q8kftntrozb,true,100.1,14321111111
7,ce1ga9aln346xcj761c3iytshyzuxg,true,100.1,14321111111
8,k5j2id9a0ko90cykl40s6ojq6gruyi,true,100.1,14321111111
9,ns2zcx9bdip5y0aqd1tdicf7bkdmsm,true,100.1,14321111111
10,54rs9cm1xau2fk66pzyz62tf9tsse4,true,100.1,14321111111
```

Each line is a record to be written to DataHub. Columns are separated by commas (,). Save the CSV file as /temp/test.csv on the local computer. The following table shows the schema of the DataHub topic to which the CSV file is written.

DataHub topic schema

Field	Data type
id	BIGINT
name	STRING
gender	BOOLEAN
salary	DOUBLE
my_time	TIMESTAMP

After you edit the Fluentd configuration file based on the CSV file and topic schema, run the following command to start Fluentd to write the CSV file to DataHub:

```
${FLUENTD_HOME}/fluentd-with-dataHub/bin/fluentd -c fluentd_test.conf
```

Use the following Fluentd configuration file in this example:

```
<source>
  @type tail
  path /xxx/yyy (Specify the path of the CSV file.)
  tag test1
  format csv
  keys id,name,gender,salary,my_time
</source>
<match test1>
  @type dataHub
  access_id your_app_id
  access_key your_app_key
  endpoint http://ip:port
  project_name test_project
  topic_name fluentd_performance_test_1
  column_names ["id", "name", "gender", "salary", "my_time"]
  flush_interval 1s
  buffer_chunk_limit 3m
  buffer_queue_limit 128
  dirty_data_continue true
  dirty_data_file /xxx/yyy (Specify the path of the dirty record file.)
  retry_times 3
  put_data_batch_size 1000
</match>
```

Case 2: Collect Log4j logs

The following format of Log4j logs is used in this example:

```
11:48:43.439 [qtp1847995714-17] INFO  AuditInterceptor - [c2un5sh7cu52ek6am1ui1m5h] end /we
b/v1/project/tefe4mfurtix9kwwyrvfqd0m/node/0m0169kapshvgc3ujskwkk8g/health GET, 4061 ms
```

Use the following Fluentd configuration file in this example:

```
<source>
  @type tail
  path bayes.log
  tag test
  format /(? <request_time>\d\d:\d\d:\d\d.\d+)\s+\[(? <thread_id>[\w\-]+)\]\s+(? <log_level
>\w+)\s+(? <class>\w+)\s+-\s+\[(? <request_id>\w+)\]\s+(? <detail>.+)/
</source>
<match test>
  @type dataHub
  access_id your_access_id
  access_key your_access_key
  endpoint http://ip:port
  project_name test_project
  topic_name dataHub_fluentd_out_1
  column_names ["thread_id", "log_level", "class"]
</match>
```

Parameter description

Input configuration

Parameter	Description
tag test1	The tag, which is mapped to the destination information by using the specified regular expression.
format csv	The format of the file from which data is collected.
keys id,name,gender,salary,my_time	The columns to be collected from the CSV file. The column names must be the same as those in the schema of the destination DataHub topic.

Output configuration

Parameter	Description
shard_id 0	The ID of the shard to which all records are written. By default, all records are written to the shard by polling. The default ID is 0.
shard_keys ["id"]	The column used as the shard key. Hashed shard key values are used as indexes for writing data.
flush_interval 1	The interval between data flushes. The default value is 60s.
buffer_chunk_limit 3m	The maximum size of a chunk. Unit: k or m, which indicates KB or MB. We recommend you set the maximum size to 3 MB.
buffer_queue_limit 128	The maximum length of the chunk queue. Both the buffer_chunk_limit and buffer_queue_limit parameters determine the size of the buffer. The default value is 128 MB.
put_data_batch_size 1000	The number of records to be written to DataHub at a time. In this example, 1,000 records are written to DataHub each time.
retry_times 3	The number of retries for writing data to DataHub. Default value: 3.
retry_interval 3	The retry interval at which data is written. Unit: seconds. Default value: 3.
dirty_data_continue true	Specifies whether to ignore dirty records. The value true indicates that the plug-in retries the operation for a specified number of times before it writes the dirty records to the dirty record file.
dirty_data_file /xxx/yyy	The directory where the dirty record file is stored.
column_names ["id"]	The name of the columns to be written to DataHub.

5.3. Logstash

This topic describes how to install and use Logstash to import data to DataHub and export data from DataHub.

Logstash is a distributed log collection framework. It is often used with Elasticsearch and Kibana, known as the ELK Stack, for log data analysis. To support a wider variety of data inputs, DataHub offers Output and Input plug-ins for data transfer with Logstash. By using Logstash, you can access more than 30 types of data sources in the Logstash open source community, such as files, Syslog logs, Redis logs, Log4j logs, Apache logs, and NGINX logs. Logstash also supports filter plug-ins for customizing the fields to be transferred. This topic demonstrates how to use Logstash with DataHub.

Install Logstash and DataHub plug-ins

Java Runtime Environment (JRE) 7 or later is required to run Logstash. If the JRE version does not meet the requirement, several features of Logstash are unavailable. You can install Logstash and DataHub plug-ins with one click by downloading and decompressing the software package or install Logstash and DataHub plug-ins separately.

- Install Logstash and DataHub plug-ins with one click: Download the software package.

  Run the following commands to decompress the package and go to the software directory:

  ```
  $ tar -xzvf logstash-with-datahub-2.3.0.tar.gz
  $ cd logstash-with-datahub-2.3.0
  ```

- Install Logstash and DataHub plug-ins separately
  - Install Logstash. For more information, see the documentation on the official website of Logstash.
  - Install the DataHub Output plug-in for Logstash. You can use this plug-in to import data to DataHub.
  - Install the DataHub Input plug-in for Logstash. You can use this plug-in to export data from DataHub.

Use cases

Case 1: Collect Log4j logs

This example shows how to collect unstructured Log4j logs and derive a structure out of the logs by using Logstash. The following format of Log4j logs is used in this example:

```
20:04:30.359 [qtp1453606810-20] INFO  AuditInterceptor - [13pn9kdr5tl84stzkmaa8vmg] end /we
b/v1/project/fhp4clxfbu0w3ym2n7ee6ynh/statistics? executionName=bayes_poc_test GET, 187 ms
```

In this example, you can derive a structure out of the logs and transfer the data to DataHub. The following table shows the schema of the DataHub topic to which the Log4j logs are written.

DataHub topic schema

Field	Data type
request_time	STRING
thread_id	STRING
log_level	STRING

Field	Data type
class_name	STRING
request_id	STRING
detail	STRING

Use the following configuration of the Logstash task in this example:

```
input {
    file {
        path => "${APP_HOME}/log/bayes.log"
        start_position => "beginning"
    }
}
filter{
    grok {
        match => {
           "message" => "(? <request_time>\d\d:\d\d:\d\d\.\d+)\s+\[(? <thread_id>[\w\-]+)\]
\s+(? <log_level>\w+)\s+(? <class_name>\w+)\s+\-\s+\[(? <request_id>\w+)\]\s+(? <detail>.+)
"
        }
    }
}
output {
    datahub {
        access_id => "Your accessId"
        access_key => "Your accessKey"
        endpoint => "Endpoint"
        project_name => "project"
        topic_name => "topic"
        #shard_id => "0"
        #shard_keys => ["thread_id"]
        dirty_data_continue => true
        dirty_data_file => "/Users/ph0ly/trash/dirty.data"
        dirty_data_file_max_size => 1000
    }
}
```

Case 2: Collect CSV files

This example shows how to use Logstash to collect CSV files. The following CSV file is used in this example:

```
1111,1.23456789012E9,true,14321111111000000,string_dataxxx0,
2222,2.23456789012E9,false,14321111111000000,string_dataxxx1
```

The following table shows the schema of the DataHub topic to which the CSV file is written.

DataHub topic schema

Field	Data type
col1	BIGINT
col2	DOUBLE
col3	BOOLEAN
col4	TIMESTAMP
col5	STRING

Use the following configuration of the Logstash task in this example:

```
input {
    file {
        path => "${APP_HOME}/data.csv"
        start_position => "beginning"
    }
}
filter{
    csv {
        columns => ['col1', 'col2', 'col3', 'col4', 'col5']
    }
}
output {
    datahub {
        access_id => "Your accessId"
        access_key => "Your accessKey"
        endpoint => "Endpoint"
        project_name => "project"
        topic_name => "topic"
        #shard_id => "0"
        #shard_keys => ["thread_id"]
        dirty_data_continue => true
        dirty_data_file => "/Users/ph0ly/trash/dirty.data"
        dirty_data_file_max_size => 1000
    }
}
```

Case 3: Consume data from DataHub

Use the following configuration of the Logstash task in this example:

```
input {
    datahub {
        access_id => "Your accessId"
        access_key => "Your accessKey"
        endpoint => "Endpoint"
        project_name => "test_project"
        topic_name => "test_topic"
        interval=> 5
        #cursor => {
        #    "0"=>"20000000000000000000000003110091"
        #    "2"=>"20000000000000000000000003110091"
        #    "1"=>"20000000000000000000000003110091"
        #    "4"=>"20000000000000000000000003110091"
        #    "3"=>"20000000000000000000000003110000"
        #}
        shard_ids => []
        pos_file => "/home/admin/logstash/logstash-2.3.0/pos_file"
    }
}
output {
    file {
        path => "/home/admin/logstash/logstash-2.3.0/output"
    }
}
```

Start Logstash

Run the following command to start Logstash:

```
$LOGSTASH_HOME/bin/logstash -f <The preceding configuration file> -b 256
```

Note -f is followed by the path of the configuration file. -b is followed by the number of records transferred to DataHub at a time. Default value: 125.

Parameters

The following table describes the parameters of the DataHub Output plug-in.

Parameters of the DataHub Output plug-in

Parameter	Description
access_id	Required. The AccessKey ID of your Apsara Stack tenant account.
access_key	Required. The AccessKey secret of your Apsara Stack tenant account.
endpoint	Required. The endpoint used to access DataHub.
project_name	Required. The name of the DataHub project.
topic_name	Required. The name of the DataHub topic.

Parameter	Description
retry_times	Optional. The maximum number of retries. The value -1 indicates unlimited retries. The value 0 indicates no retries. A value greater than 0 indicates the specified number of retries. Default value: -1.
retry_interval	Optional. The interval between retries. Unit: seconds. Default value: 5.
shard_keys	Optional. The key of the shard. The hash of the key value is used to map the ID of the shard to which the records are written. If the shard_keys and shard_id parameters are not specified, the system polls the shards to decide which shard the records are written to.
shard_id	Optional. The ID of the shard where records are written. If the shard_keys and shard_id parameters are not specified, the system polls the shards to decide which shard the records are written to.
dirty_data_continue	Optional. Specifies whether to ignore dirty records. The value true indicates that dirty records are to be ignored. Default value: false. If you set the value to true, you must specify the dirty_data_file parameter.
dirty_data_file	Optional. The name of the dirty record file. The dirty record file is divided into .part 1 and .part 2. The most recent records are stored in .part 2.
dirty_data_file_max_size	Optional. The maximum size of the dirty record file. This value is for reference only.

The following table describes the parameters of the DataHub Input plug-in.

Parameters of the DataHub Input plug-in

Parameter	Description
access_id	Required. The AccessKey ID of your Apsara Stack tenant account.
access_key	Required. The AccessKey secret of your Apsara Stack tenant account.
endpoint	Required. The endpoint used to access DataHub.
project_name	Required. The name of the DataHub project.
topic_name	Required. The name of the DataHub topic.
retry_times	Optional. The maximum number of retries. The value -1 indicates unlimited retries. The value 0 indicates no retries. A value greater than 0 indicates the specified number of retries. Default value: -1.
retry_interval	Optional. The interval between retries. Unit: seconds. Default value: 5.
shard_ids	Optional. The shards in which records are to be consumed. If this parameter is not specified, records in all the shards are consumed.
cursor	Optional. The sequence number of the record from which the consumption begins. The consumption starts from the earliest record by default.

Parameter	Description
pos_file	Required. The checkpoint file, which is used to reset the consumption offset.

5.4. Oracle GoldenGate

This topic describes how to install and use Oracle GoldenGate (OGG).

OGG is a tool for log-based structured data replication across heterogeneous environments. It is used for data backup between primary and secondary Oracle databases. It is also used to synchronize data from Oracle databases to other databases such as IBM Db2 and MySQL databases. OGG must be deployed in the source and destination databases. It is composed of the following components: Manager, Extract, data pump, Collector, and Replicat.

- Manager is the control process of OGG. A Manager process must be running on the source and destination databases. It is responsible for starting, stopping, and monitoring other processes.
- Extract is a process that captures data from the source database or transaction logs. You can configure the Extract process for initial data loads and incremental data synchronization. For initial data loads, Extract captures a set of data directly from their source objects. To keep source data synchronized to the destination database, Extract captures incremental DML and DDL operations after the initial data loading has taken place. This topic describes incremental data synchronization.
- A data pump is a secondary Extract group within the source OGG configuration. In a typical configuration with a data pump, the primary Extract group writes to a trail on the source database. The data pump reads the trail and sends the DML or DDL operations over the network to a remote trail on the destination database.
- Collector is a process on the destination database, which receives data from the source database and generates trail files.
- Replicat is a process that reads the trail on the destination database, reconstructs the DML or DDL operations, and then applies them to the destination database.

The DataHub agent for OGG offers the Replicat feature that applies the updated data to DataHub by analyzing the trail. The data in DataHub is processed in real time by using Realtime Compute and can be archived into MaxCompute.

The following example shows how to synchronize incremental data from an Oracle database to DataHub and process the data in DataHub.

Install OGG

Prerequisites:

- You have installed the Oracle database client.
- You have obtained the OGG installation package for the source database. We recommend that you use OGG V12.1.2.1.
- You have obtained the OGG Adapters installation package for the destination database. We recommend that you use OGG Application Adapters 12.1.2.1.
- You have installed Java 7.

Follow these steps to install OGG:

1. Install OGG for the source database.

i. Extract the OGG installation package for the source database and the following directories appear:

```
drwxr-xr-x install
drwxrwxr-x response
-rwxr-xr-x runInstaller
drwxr-xr-x stage
```

ii. Install dependencies in response/oggcore.rsp. The OGG response file template is as follows:

```
oracle.install.responseFileVersion=/oracle/install/rspfmt_ogginstall_response_schem
a
#The installation option, which must reflect the installed Oracle version. Specify
ORA11g for installing OGG for Oracle Database 11g.
INSTALL_OPTION=ORA11g
#The location in which OGG is installed.
SOFTWARE_LOCATION=/home/oracle/u01/ggate
#Indicates whether to start the Manager after installation.
START_MANAGER=false
#The port number of the Manager process.
MANAGER_PORT=7839
#The location of the Oracle database.
DATABASE_LOCATION=/home/oracle/u01/app/oracle/product/11.2.0/dbhome_1
#The location that stores the inventory files. This parameter is not required to be
configured.
INVENTORY_LOCATION=
#The UNIX group of the inventory directory. In this example, OGG is installed by us
ing the ogg_test Oracle account. You can also create a dedicated account for OGG as
necessary.
UNIX_GROUP_NAME=oinstall
```

iii. Run the following command to install OGG:

```
runInstaller -silent -responseFile {YOUR_OGG_INSTALL_FILE_PATH}/response/oggcore.rs
p
```

> **Note**
>
> In this example, OGG is installed in */home/oracle/u01/ggate* and the installation logs are stored in */home/oracle/u01/ggate/cfgtoollogs/oui*. The OGG installation is complete when the following message appears in the silentInstall{time}.log file:
>
> ```
> The installation of Oracle GoldenGate Core was successful.
> ```

iv. Run the following command and enter `CREATE SUBDIRS` as required to create OGG directories:

```
/home/oracle/u01/ggate/ggsci
```

2. Perform Oracle configurations in the source database.

   Navigate to *sqlplus: sqlplus / as sysdba* as the database administrator and complete the following configurations:

```
#Create a tablespace.
create tablespace ATMV datafile '/home/oracle/u01/app/oracle/oradata/uprr/ATMV.dbf' siz
e 100m autoextend on next 50m maxsize unlimited;
 #Create a user named ogg_test. The password is also set to ogg_test.
create user ogg_test identified by ogg_test default tablespace ATMV;
 #Grant required privileges to ogg_test.
grant connect,resource,dba to ogg_test;
 #Check whether supplemental logging is enabled for the database.
Select SUPPLEMENTAL_LOG_DATA_MIN, SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI, S
UPPLEMENTAL_LOG_DATA_FK, SUPPLEMENTAL_LOG_DATA_ALL from v$database;
 #If the result is NO, enable supplemental logging.
alter database add supplemental log data;
alter database add supplemental log data (primary key, unique,foreign key) columns;
 #Enable rollback.
alter database drop supplemental log data (primary key, unique,foreign key) columns;
alter database drop supplemental log data;
 #Enable all column logging at the database level. Note: Even when all column logging i
s enabled, only primary key columns are logged for a delete operation.
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
 #Enable the forced logging mode.
alter database force logging;
 #Run the marker_setup.sql script.
@marker_setup.sql
 #Run the ddl_setup.sql script.
@ddl_setup.sql
 #Run the role_setup.sql script.
@role_setup.sql
 #Grant the GGS_GGSUSER_ROLE to ogg_test.
grant GGS_GGSUSER_ROLE to ogg_test;
 #Run the ddl_enable.sql script to enable the DDL trigger.
@ddl_enable.sql
 #Run the ddl_pin script to improve the performance of the DDL trigger.
@ddl_pin ogg_test
 #Run the sequence.sql script.
@sequence.sql
#
alter table sys.seq$ add supplemental log data (primary key) columns;
```

3. Configure the Manager process on the source database.

   Start the Oracle GoldenGate Software Command Interface (GGSCI) and perform the following steps:

i. Run the following command to configure the Manager process:

```
edit params mgr
PORT 7839
DYNAMICPORTLIST  7840-7849
USERID ogg_test, PASSWORD ogg_test
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 7
LAGREPORTHOURS 1
LAGINFOMINUTES 30
LAGCRITICALMINUTES 45
PURGEDDLHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
PURGEMARKERHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
```

ii. Run the following command to start the Manager process. The logs are stored in ggate/dirrpt.

```
start mgr
```

iii. Run the following command to check whether the Manager process is running:

```
info mgr
```

iv. Run the following command to view the Manager parameter file:

```
view params mgr
```

4. Configure the Extract process on the source database.

   Start the GGSCI and perform the following steps:

   i. Run the following command to configure the Extract process. In the following example, the group name of the process is extract.

```
edit params extractEXTRACT extract
SETENV (NLS_LANG="AMERICAN_AMERICA.AL32UTF8")
DBOPTIONS   ALLOWUNUSEDCOLUMN
USERID ogg_test, PASSWORD ogg_test
REPORTCOUNT EVERY 1 MINUTES, RATE
NUMFILES 5000
DISCARDFILE ./dirrpt/ext_test.dsc, APPEND, MEGABYTES 100
DISCARDROLLOVER AT 2:00
WARNLONGTRANS 2h, CHECKINTERVAL 3m
EXTTRAIL ./dirdat/st, MEGABYTES 200
DYNAMICRESOLUTION
TRANLOGOPTIONS CONVERTUCS2CLOBS
TRANLOGOPTIONS RAWDEVICEOFFSET 0
DDL &
INCLUDE MAPPED OBJTYPE 'table' &
INCLUDE MAPPED OBJTYPE 'index' &
INCLUDE MAPPED OBJTYPE 'SEQUENCE' &
EXCLUDE OPTYPE COMMENT
DDLOPTIONS  NOCROSSRENAME  REPORT
TABLE     OGG_TEST. *;
SEQUENCE  OGG_TEST. *;
GETUPDATEBEFORES
```

ii. Run the following command to add an Extract process. Replace extract in the following command with your actual group name.

```
add ext extract,tranlog, begin now
```

iii. Run the following command to delete an Extract process. In the following example, the process name is DP_TEST.

```
delete ext DP_TEST
```

iv. Run the following command to create a trail, associate the trail with the Extract group named extract, and set the maximum file size in the trail to 200 megabytes:

```
add exttrail ./dirdat/st,ext extract, megabytes 200
```

v. Run the following command to start the Extract process. The logs are stored in ggate/dirrpt.

```
start extract extract
```

> **Note** After the Extract process configuration is complete, you can view the changes to the database in the files stored in the *ggate/dirdat* directory.

5. Create a DEFGEN parameter file.

i. Start the GGSCI in the source database. In GGSCI, run the following command to create a DEFGEN parameter file and copy the file to the dirdef directory in the destination database:

```
edit params defgen
DEFSFILE ./dirdef/ogg_test.def
USERID ogg_test, PASSWORD ogg_test
table OGG_TEST. *;
```

ii. Run the following command from the shell to create a DEFGEN parameter file named ogg_test.def:

```
./defgen paramfile ./dirprm/defgen.prm
```

6. Install and configure OGG in the destination database.

i. Extract the OGG installation package to the destination database.

ii. Copy the dirdef/ogg_test.def file in the source database to dirdef of the destination database.

iii. Start the GGSCI and run the following command to create the default directories of OGG:

```
create subdirs
```

iv. Run the following command to configure the Manager process:

```
edit params mgr
PORT 7839
DYNAMICPORTLIST  7840-7849
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 7
LAGREPORTHOURS 1
LAGINFOMINUTES 30
LAGCRITICALMINUTES 45
PURGEDDLHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
PURGEMARKERHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
```

v. Run the following command to start the Manager process:

```
start mgr
```

7. Configure a data pump in the source database.

   Start the GGSCI and perform the following steps:

   i. Run the following command to configure a data pump:

```
edit params pump
EXTRACT pump
RMTHOST xx.xx.xx.xx, MGRPORT 7839, COMPRESS
PASSTHRU
NUMFILES 5000
RMTTRAIL ./dirdat/st
DYNAMICRESOLUTION
TABLE      OGG_TEST. *;
SEQUENCE   OGG_TEST. *;
```

   ii. Run the following command to create a data-pump Extract process. The process reads from the specified trail.

```
add ext pump,exttrailsource ./dirdat/st
```

   iii. Run the following command to create a trail and set the maximum file size in the trail to 200 megabytes:

```
add rmttrail ./dirdat/st,ext pump,megabytes 200
```

   iv. Run the following command to start the data pump:

```
start pump
```

> **Note** After the data pump is started, you can view the trail files in the dirdat directory of the destination database.

8. Install and configure the DataHub agent for OGG.

   i. Run the following command to configure the *JAVA_HOME* and *LD_LIBRARY_PATH* environment variables and specify the configurations in the ~/.bash_profile:

```
export JAVA_HOME=/xxx/xxx/jrexx
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$JAVA_HOME/lib/amd64:$JAVA_HOME/lib/amd64
/server
```

ii. After the environment variables are configured, restart the Manager process in the destination database.

iii. Download the DataHub agent for OGG and extract the installation package.

iv. Modify the javaue.properties and log4j.properties files in the conf sub-directory of the installation directory and replace {YOUR_HOME} with the target path of the extracted files:

```
gg.handlerlist=ggdatahub
gg.handler.ggdatahub.type=com.aliyun.odps.ogg.handler.datahub.DatahubHandler
gg.handler.ggdatahub.configureFileName={YOUR_HOME}/datahub-ogg-plugin/conf/configur
e.xml
goldengate.userexit.nochkpt=false
goldengate.userexit.timestamp=utc
gg.classpath={YOUR_HOME}/datahub-ogg-plugin/lib/*
gg.log.level=debug
jvm.bootoptions=-Xmx512m -Dlog4j.configuration=file:{YOUR_HOME}/datahub-ogg-plugin/
conf/log4j.properties -Djava.class.path=ggjava/ggjava.jar
```

v. Modify the configure.xml file in the conf sub-directory of the installation directory as follows:

```
<? xml version="1.0" encoding="UTF-8"? >
<configue>
    <defaultOracleConfigure>
        <! --(Required) The Oracle database system identifier (SID).-->
        <sid>100</sid>
        <! --The schema of the Oracle table, which can be overwritten by oracleSche
ma in the column mappings. At least one of them must be specified.-->
        <schema>ogg_test</schema>
    </defaultOracleConfigure>
    <defalutDatahubConfigure>
        <! --(Required) The endpoint of DataHub.-->
        <endPoint>YOUR_DATAHUB_ENDPOINT</endPoint>
        <! --The DataHub project, which can be overwritten by datahubProject in the
column mappings. At least one of them must be specified.-->
        <project>YOUR_DATAHUB_PROJECT</project>
        <! --The AccessKey ID for accessing DataHub, which can be overwritten by da
tahubAccessId in the column mappings. At least one of them must be specified.-->
        <accessId>YOUR_DATAHUB_ACCESS_ID</accessId>
        <! --The AccessKey Secret for accessing DataHub, which can be overwritten b
y datahubAccessKey in the column mappings. At least one of them must be specified.-
->
        <accessKey>YOUR_DATAHUB_ACCESS_KEY</accessKey>
        <! --The column in DataHub that indicates the data update type, which can b
e overwritten by ctypeColumn in the column mappings.-->
        <ctypeColumn>optype</ctypeColumn>
        <! -- The column in DataHub that indicates the data update time, which can
be overwritten by ctimeColumn in the column mappings.-->
        <ctimeColumn>readtime</ctimeColumn>
        <! -- The column in DataHub that indicates the sequence number of the updat
ed data, which can be overwritten by cidColumn in the column mappings. The sequence
number increases as more data are updated, but may not be consecutive.-->
```

```
        <cidColumn>record_id</cidColumn>
    </defalutDatahubConfigure>
    <! --The approach to handling errors. If an error occurs, the system either ign
ores the error and continues running or retries the operation repeatedly.-->
    <! --(Optional) The maximum number of records operated at one time. Default val
ue: 1000.-->
    <batchSize>1000</batchSize>
    <! --(Optional) The format that the timestamp is converted into. Default: yyyy-
MM-dd HH:mm:ss.-->
    <defaultDateFormat>yyyy-MM-dd HH:mm:ss</defaultDateFormat>
    <! --(Optional) Indicates whether the system needs to ignore dirty records. Def
ault value: false.-->
    <dirtyDataContinue>true</dirtyDataContinue>
    <! --(Optional) The dirty record file name. Default value: datahub_ogg_plugin.d
irty-->
    <dirtyDataFile>datahub_ogg_plugin.dirty</dirtyDataFile>
    <! --(Optional) The maximum size of the dirty record file. Unit: MB. Default va
lue: 500.-->
    <dirtyDataFileMaxSize>200</dirtyDataFileMaxSize>
    <! --(Optional) The maximum number of retries if an error occurs. -1: Unlimited
. 0: No retries. n: The number of retries. Default value: -1.-->
    <retryTimes>0</retryTimes>
    <! --(Optional) The interval between retries. Unit: milliseconds. Default value
: 3000.-->
    <retryInterval>4000</retryInterval>
    <! --(Optional) The checkpoint file name. Default value: datahub_ogg_plugin.chk
.-->
    <checkPointFileName>datahub_ogg_plugin.chk</checkPointFileName>
    <mappings>
        <mapping>
            <! --The schema of the Oracle table.-->
            <oracleSchema></oracleSchema>
            <! --(Required) The Oracle table name.-->
            <oracleTable>t_person</oracleTable>
            <! --The DataHub project name.-->
            <datahubProject></datahubProject>
            <! --The AccessKey ID for accessing DataHub.-->
            <datahubAccessId></datahubAccessId>
            <! --The AccessKey Secret for accessing DataHub.-->
            <datahubAccessKey></datahubAccessKey>
            <! --(Required) The DataHub topic name.-->
            <datahubTopic>t_person</datahubTopic>
            <ctypeColumn></ctypeColumn>
            <ctimeColumn></ctimeColumn>
            <cidColumn></cidColumn>
            <columnMapping>
                <! --
                src: (Required) The column names in the Oracle table.
                dest: (Required) The column names in the DataHub topic.
                destOld: (Optional) The DataHub topic column that records the data
before it is updated.
                isShardColumn: (Optional) Indicates whether the shard ID is generat
ed based on the hash key value, which can be overwritten by shardId. Default value:
false.
```

```
                isDateFormat: Indicates whether the timestamp is converted into a s
tring based on dateFormat. Default value: true. If you set the value to false, the
data type in the source database must be long.
                dateFormat: The format that the timestamp is converted into. If thi
s parameter is left blank, the default format is used.
                -->
                <column src="id" dest="id" isShardColumn="true"  isDateFormat="fals
e" dateFormat="yyyy-MM-dd HH:mm:ss"/>
                <column src="name" dest="name" isShardColumn="true"/>
                <column src="age" dest="age"/>
                <column src="address" dest="address"/>
                <column src="comments" dest="comments"/>
                <column src="sex" dest="sex"/>
                <column src="temp" dest="temp" destOld="temp1"/>
            </columnMapping>
            <! --(Optional) The ID of the shard prioritized to be written into.-->
            <shardId>1</shardId>
        </mapping>
    </mappings>
</configue>
```

vi. Run the following command in GGSCI to start the DataHub writer:

```
edit params dhwriter
extract dhwriter
getEnv (JAVA_HOME)
getEnv (LD_LIBRARY_PATH)
getEnv (PATH)
CUSEREXIT ./libggjava_ue.so CUSEREXIT PASSTHRU INCLUDEUPDATEBEFORES, PARAMS "{YOUR_
HOME}/datahub-ogg-plugin/conf/javaue.properties"
sourcedefs ./dirdef/ogg_test.def
table OGG_TEST. *;
```

vii. Run the following command to add a DataHub writer:

```
add extract dhwriter, exttrailsource ./dirdat/st
```

viii. Run the following command to start the writer:

```
start dhwriter
```

Use case

For example, you have an Oracle table that stores order information. The table has three columns. The column names are oid, pid, and num, which indicate order ID, product ID, and product quantity. You can synchronize incremental data to DataHub by using the DataHub agent for OGG. The steps are as follows:

> **Note** Before performing incremental data synchronization, you must synchronize existing data from the source table to MaxCompute by using DataX.

1. Create a topic in DataHub. The schema of the topic is as follows:

```
string record_id, string optype, string readtime, bigint oid_before, bigint oid_after,
bigint pid_before, bigint pid_after, bigint num_before, bigint num_after
```

2. Make sure that you have completed the deployment of the DataHub agent for OGG. Then configure the column mappings as follows:

```
<ctypeColumn>optype</ctypeColumn>
            <ctimeColumn>readtime</ctimeColumn>
            <cidColumn>record_id</cidColumn>
            <columnMapping>
                <column src="oid" dest="oid_after" destOld="oid_before" isShardColumn="
true"/>
                <column src="pid" dest="pid_after" destOld="pid_before"/>
                <column src="num" dest="num_after" destOld="num_before"/>
            </columnMapping>
```

> **Note** The optype parameter indicates the type of the data update. Valid values of the optype parameter are **I**, **D**, and **U**, which represent an **insert**, **delete**, and **update** operation, respectively. The readtime parameter indicates the time of the data update.

3. When the agent can run properly, data updates are synchronized from the source table to DataHub.

6.Data synchronization

6.1. Overview

You can synchronize real-time data from DataHub to other data warehouses by using DataConnectors so that you can analyze and process historical data.

The following topics describe how to synchronize data from DataHub to MaxCompute.

6.2. Synchronize data to MaxCompute

6.2.1. Create a DataConnector

This topic describes how to create a DataConnector to synchronize data from DataHub to MaxCompute.

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Connector**. In the Create connector dialog box, select the data warehouse to which data is synchronized.
4. In the Create connector dialog box, set relevant parameters and click **Create**.

Note

The following table describes the parameters of the DataConnector for synchronizing data from DataHub to MaxCompute.

Parameter	Description
Project Name	The name of the MaxCompute project to which data in the topic is synchronized.
Table Name	The name of the MaxCompute table to which data in the topic is synchronized.
AccessID and AccessKey	The AccessKey pair used to access MaxCompute. The AccessKey pair must belong to a RAM user that has CreateInstance, Desc, and Alter permissions on the MaxCompute table.
Partition Mode	The method used to create partitions. Valid values: SYSTEM_TIME, EVENT_TIME, USER_DEFINE, and META_TIME. If you select SYSTEM_TIME, partitions are created based on the recording time. If you select EVENT_TIME, partitions are created based on the value of the event_time field. When you create the topic, you must define a field named event_time for the topic and set its data type to TIMESTAMP. The value of the event_time field must be accurate to microseconds. If you select USER_DEFINE, partitions are created based on the user-defined partition key.
Partition Config	The format of the time based on which partitions are created. This parameter takes effect only when you set the Partition Mode parameter to SYSTEM_TIME, EVENT_TIME, or META_TIME.
Time Range	The interval of creating partitions. This parameter takes effect only when you set the Partition Mode parameter to SYSTEM_TIME, EVENT_TIME, or META_TIME. The minimum value is 15 minutes.
Timezone	The time zone of the time based on which partitions are created. This parameter takes effect only when you set the Partition Mode parameter to SYSTEM_TIME, EVENT_TIME, or META_TIME.
Start Time	The time when data synchronization starts.

6.2.2. View data synchronization details

This section describes how to view data synchronization details after a DataConnector is created.

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click **View** in the Operate column.
3. On the project details page, find the target topic and click **View** in the Operate column.
4. On the topic details page, click the **Connector** tab. On the Connector tab, find the target DataConnector and click **View** in the Operate column.

> **Notice** You can restart or stop a DataConnector. Exercise caution when you perform the operations.

7.Metric statistics

This topic describes how to view the metric statistics of a topic in DataHub.

In the DataHub console, you can view the metric statistics of topics in quasi-real-time, such as QPS and throughput. The following metrics are available:

- Read and write QPS
- Read and write records per second (RPS)
- Read and write throughput, measured in KB per second
- Read and write latency, measured in microseconds per request

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click **View** in the Operate column.
3. On the project details page, find the target topic and click **View** in the Operate column.
4. On the topic details page, click the **Metric Statistics** tab.

You can view the metric statistics for a specified time range.

8.Data subscription

8.1. Overview

Resumable consumption is required in scenarios where you consume data in DataHub topics and want to resume the consumption from the time when your application fails. If you need to resume consumption, you must save the current consumption offset and make sure that the service for saving consumption offsets supports high availability. This increases the complexity of developing applications. The subscription feature of DataHub supports saving consumption offsets to the server to solve the preceding problem. You only need to enable this feature and add a few lines of code to your application to obtain a consumption offset maintenance service with high availability.

In addition, the subscription feature allows you to reset consumption offsets. This ensures that the data can be consumed at least once. For example, if an error occurs when your application processes the data consumed in a specific time period and you need to consume the data again, you can reset the consumption offset without restarting the application. Your application automatically consumes data from the specified consumption offset.

8.2. Create a subscription

You can create subscriptions only in the DataHub console. Make sure that your account is authorized to subscribe to topics of the specified project.

Perform the following steps to create a subscription:

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Subscription**. In the Create subscription dialog box, set relevant parameters and click **Create**.
4. After the subscription is created, click the **Subscription List** tab on the topic details page to view the subscriptions of the topic.

> **Note** You can click **Reset** or **Delete** in the Operate column of a subscription.
>
> - **Reset**: resets the consumption offset of the subscription to the required time. Specify the time in the *mm-dd-yyyy HH:MM:SS* format.
> - **Delete**: permanently deletes the subscription, including all consumption offsets that are saved for the subscription.

8.3. Use case

The subscription feature allows you to save consumption offsets. You can use the read and write capabilities of DataHub with the capability of saving consumption offsets in scenarios where you must save consumption offsets after data is read.

The following sample code is used for reference only.

```
// The following sample code consumes data from a saved consumption offset and submit consu
mption offsets during consumption.
public void offset_consumption(int maxRetry) {
```

```
    String endpoint = "<YourEndPoint>";
    String accessId = "<YourAccessId>";
    String accessKey = "<YourAccessKey>";
    String projectName = "<YourProjectName>";
    String topicName = "<YourTopicName>";
    String subId = "<YourSubId>";
    String shardId = "0";
    List<String> shardIds = Arrays.asList(shardId);
    // Create a DataHub client.
    DatahubClient datahubClient = DatahubClientBuilder.newBuilder()
            .setDatahubConfig(
                    new DatahubConfig(endpoint,
                            // Specify whether to enable binary data transmission. The serv
er of V2.12 or later supports binary data transmission.
                            new AliyunAccount(accessId, accessKey), true))
            .build();
    RecordSchema schema = datahubClient.getTopic(projectName, topicName).getRecordSchema();
    OpenSubscriptionSessionResult openSubscriptionSessionResult = datahubClient.openSubscri
ptionSession(projectName, topicName, subId, shardIds);
    SubscriptionOffset subscriptionOffset = openSubscriptionSessionResult.getOffsets().get(
shardId);
    // 1. Obtain the cursor of the record at the current consumption offset. If the record
expired or the record is not consumed, obtain the cursor of the first record within the TTL
of the topic.
    String cursor = "";
    // If the sequence number is smaller than 0, the record is not consumed.
    if (subscriptionOffset.getSequence() < 0) {
        // Obtain the cursor of the first record within the TTL of the topic.
        cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.OLDEST
).getCursor();
    } else {
        // Obtain the cursor of the next record.
        long nextSequence = subscriptionOffset.getSequence() + 1;
        try {
            // If the SeekOutOfRange error is returned after you obtain the cursor based on
the sequence number, the record expired.
            cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SE
QUENCE, nextSequence).getCursor();
        } catch (SeekOutOfRangeException e) {
            // Obtain the cursor of the first record within the TTL of the topic.
            cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.OL
DEST).getCursor();
        }
    }
    // 2. Read records and save consumption offsets. In this example, you read tuple record
s and save consumption offsets each time 1,000 records are read.
    long recordCount = 0L;
    // Read 1,000 records each time.
    int fetchNum = 1000;
    int retryNum = 0;
    int commitNum = 1000;
    while (retryNum < maxRetry) {
        try {
            GetRecordsResult getRecordsResult = datahubClient.getRecords(projectName, topic
```

```
Name, shardId, schema, cursor, fetchNum);
            if (getRecordsResult.getRecordCount() <= 0) {
                // If no records can be read, pause the thread for 1s and continue to read
records.
                System.out.println("no data, sleep 1 second");
                Thread.sleep(1000);
                continue;
            }
            for (RecordEntry recordEntry : getRecordsResult.getRecords()) {
                // Consume data.
                TupleRecordData data = (TupleRecordData) recordEntry.getRecordData();
                System.out.println("field1:" + data.getField("field1") + "\t"
                        + "field2:" + data.getField("field2"));
                // Save the consumption offset after the data is consumed.
                recordCount++;
                subscriptionOffset.setSequence(recordEntry.getSequence());
                subscriptionOffset.setTimestamp(recordEntry.getSystemTime());
                // commit offset every 1000 records
                if (recordCount % commitNum == 0) {
                    // Submit the consumption offset.
                    Map<String, SubscriptionOffset> offsetMap = new HashMap<>();
                    offsetMap.put(shardId, subscriptionOffset);
                    datahubClient.commitSubscriptionOffset(projectName, topicName, subId, o
ffsetMap);
                    System.out.println("commit offset successful");
                }
            }
            cursor = getRecordsResult.getNextCursor();
        } catch (SubscriptionOfflineException | SubscriptionSessionInvalidException e) {
            // The subscription session is exited. The Offline exception indicates that the
subscription is offline. The SessionChange exception indicates that the subscription is con
sumed by other clients.
            e.printStackTrace();
            throw e;
        } catch (SubscriptionOffsetResetException e) {
            // The consumption offset is reset. You must obtain the version information of
the consumption offset again.
            SubscriptionOffset offset = datahubClient.getSubscriptionOffset(projectName, to
picName, subId, shardIds).getOffsets().get(shardId);
            subscriptionOffset.setVersionId(offset.getVersionId());
            // After the consumption offset is reset, you must obtain the cursor of the rec
ord at the consumption offset again. The method for obtaining the cursor depends on the met
hod of resetting the consumption offset.
            // If both the sequence number and timestamp are specified to reset the consump
tion offset, you can obtain the cursor based on the sequence number or the timestamp.
            // If only the sequence number is specified to reset the consumption offset, yo
u can obtain the cursor only based on the sequence number.
            // If only the timestamp is specified to reset the consumption offset, you can
obtain the cursor only based on the timestamp.
            // Generally, preferentially obtain the cursor based on the sequence number. If
the cursor failed to be obtained based on the sequence number or the timestamp, obtain the
cursor of the earliest record.
            cursor = null;
            if (cursor == null) {
```

```
                try {
                    long nextSequence = offset.getSequence() + 1;
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, Curso
rType.SEQUENCE, nextSequence).getCursor();
                    System.out.println("get cursor successful");
                } catch (DatahubClientException exception) {
                    System.out.println("get cursor by SEQUENCE failed, try to get cursor by
SYSTEM_TIME");
                }
            }
            if (cursor == null) {
                try {
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, Curso
rType.SYSTEM_TIME, offset.getTimestamp()).getCursor();
                    System.out.println("get cursor successful");
                } catch (DatahubClientException exception) {
                    System.out.println("get cursor by SYSTEM_TIME failed, try to get cursor
by OLDEST");
                }
            }
            if (cursor == null) {
                try {
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, Curso
rType.OLDEST).getCursor();
                    System.out.println("get cursor successful");
                } catch (DatahubClientException exception) {
                    System.out.println("get cursor by OLDEST failed");
                    System.out.println("get cursor failed!!") ;
                    throw e;
                }
            }
        } catch (LimitExceededException e) {
            // limit exceed, retry
            e.printStackTrace();
            retryNum++;
        } catch (DatahubClientException e) {
            // other error, retry
            e.printStackTrace();
            retryNum++;
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(-1);
        }
    }
}
```

Note

- When you start the application for the first time, your application consumes data from the earliest record. During the running of the application, you can refresh the Subscription List tab in the console.
- If you reset the consumption offset by clicking Reset in the console during the consumption, your application automatically detects the change of the consumption offset and consumes data from the specified consumption offset. When the application catches OffsetResetedException, the application calls the getSubscriptionOffset method to query the latest consumption offset from the server. Then, the application can consume data from the latest consumption offset.
- Note that a shard in a subscription cannot be consumed by multiple threads or processes at the same time. Otherwise, the consumption offset submitted by a thread is overwritten by that submitted by another thread and the server cannot determine to which thread the saved consumption offset belongs. In this case, the server throws OffsetSessionChangedException. We recommend that you exit the subscription session to check whether data is repeatedly consumed if this exception is caught.

9.Collaborative consumption

9.1. Note

DataHub-client-library encapsulates the Java SDK and integrates the consumer for collaborative consumption and the producer for distributing data evenly among shards.

9.2. Overview

Offset-based data consumption

The offset-based data consumption feature allows you to save consumption offsets to the server. A consumption offset consists of the sequence number of a record and the timestamp when the record is written to DataHub.

You can create a subscription for a topic and submit the consumption offset to the server after specific data is consumed. When you application starts the next time, the application can obtain the consumption offset from the server and consume data from the next record. The consumption offsets must be saved on the server so that your application can consume data from a submitted consumption offset after shards are reallocated. This is a prerequisite for collaborative consumption.

You do not need to manually submit consumption offsets in the consumer. You only need to specify the interval of submitting consumption offsets in the configurations of the consumer. The system considers that the previous records are consumed when it reads records. If the interval of submitting consumption offsets is exceeded, the system submits a consumption offset again. If the consumption offset fails to be submitted and your application is interrupted, the consumption offset may fail to be submitted in time. In this case, your application may repeatedly consume specific data.

Collaborative consumption

The collaborative consumption feature automatically allocates shards when multiple consumers consume a topic at the same time. This feature simplifies the data processing of clients.

> **Note** Manual shard allocation is difficult because multiple consumers may reside on different machines. If multiple consumers that subscribe to the same topic are in the same consumer group, a shard can be allocated to only one consumer in the consumer group.

Example:

Assume that A, B, and C are three consumer instances and the topic has 10 shards.

1. When the consumer instance A is started at first, 10 shards are allocated to it.
2. When the other two consumer instances are started, the shards are reallocated in the following way: four to A, three to B, and three to C.
3. When one of the shards consumed by the consumer instance A is split into two and the two shards are released after consumption, the shards are reallocated in the following way: four to A, four to B, and three to C.
4. When the consumer instance C is stopped, the shards are reallocated in the following way: six to A and five to B.

Heartbeat

You must use the heartbeat feature to notify the server of the status of consumer instances. If the server has not received heartbeats from a consumer instance after the specified interval, the server considers that the consumer instance is stopped. When the status of a consumer instance changes, the server reallocates shards. The server returns the new allocation plan in heartbeat requests. Therefore, the client takes time to detect reallocation of shards.

9.3. Maven dependencies and JDK

Maven dependencies

```
<dependency>
      <groupId>com.aliyun.datahub</groupId>
      <artifactId>datahub-client-library</artifactId>
      <version>1.0.6-public</version>
</dependency>
```

JDK

```
jdk: >= 1.7
```

9.4. Use case

The following sample code is for reference only.

Initialize the producer

```
String endpoint = "http://dh-cn-hangzhou.aliyuncs.com";
String accessId = "<YourAccessKeyId>";
String accessKey = "<YourAccessKeySecret>";
String projectName = "<YourProjectName>";
String topicName = "<YourTopicName>";
ProducerConfig config = new ProducerConfig(endpoint, accessId, accessKey);
Producer producer = new Producer(projectName, topicName, config);
```

Write data to DataHub

```
RecordSchema schema = new RecordSchema();
schema.addField(new Field("field1", FieldType.STRING));
schema.addField(new Field("field2", FieldType.BIGINT));
List<RecordEntry> recordEntries = new ArrayList<>();
for (int cnt = 0; cnt < 10; ++cnt) {
    RecordEntry entry = new RecordEntry();
    entry.addAttribute("key1", "value1");
    entry.addAttribute("key2", "value2");
    TupleRecordData data = new TupleRecordData(schema);
    data.setField("field1", "testValue");
    data.setField("field2", 1);
    entry.setRecordData(data);
    recordEntries.add(entry);
}
int maxRetry = 3;
while (true) {
    try {
        producer.send(records, maxRetry);
        break;
    } catch (MalformedRecordException e) {
        // malformed RecordEntry
    } catch (InvalidParameterException e) {
        // invalid param
    } catch (ResourceNotFoundException e) {
        // project, topic or shard not found, sometimes caused by split/merge shard
    } catch (DatahubClientException e) {
        // network or other exceptions exceeded retry limit
    }
}
// close before exit
producer.close();
```

Initialize the consumer

```
String endpoint = "http://dh-cn-hangzhou.aliyuncs.com";
String accessId = "<YourAccessKeyId>";
String accessKey = "<YourAccessKeySecret>";
String projectName = "<YourProjectName>";
String topicName = "<YourTopicName>";
String SubId = "<YourSubscriptionId>";
// 1. If you need to use the collaborative consumption feature, specify the subscription ID
.
ConsumerConfig config = new ConsumerConfig(endpoint, accessId, accessKey);
Consumer consumer = new Consumer(projectName, topicName, SubId, config);
// 2. If you need to use the offset-based data consumption feature instead of the collabora
tive consumption feature, specify the subscription ID and the shards to be read by the cons
umer.
List<String> assignment = Arrays.asList("0", "1", "2");
ConsumerConfig config = new ConsumerConfig(endpoint, accessId, accessKey);
Consumer consumer = new Consumer(projectName, topicName, SubId, assignment, config);
// 3. If you do not need to use the collaborative consumption feature nor the offset-based
data consumption feature, specify the subscription ID, the shards to be read by the consume
r, and the consumption offset.
Map<String, Offset> offsetMap = new HashMap<>();
// If both the sequence number and timestamp are specified but the sequence number is inval
id, obtain the cursor based on the timestamp.
offsetMap.put("0", new Offset(100, 1548573440756L));
// If only the sequence number is specified, obtain the cursor based on the sequence number
.
offsetMap.put("1", new Offset().setSequence(1));
// If only the timestamp is specified, obtain the cursor based on the timestamp.
offsetMap.put("2", new Offset().setTimestamp(1548573440756L));
ConsumerConfig config = new ConsumerConfig(endpoint, accessId, accessKey);
Consumer consumer = new Consumer(projectName, topicName, SubId, offsetMap, config);
```

Read data from DataHub

```
int maxRetry = 3;
boolean stop = false;
while (! stop) {
    try {
        while (true) {
            RecordEntry record = consumer.read(maxRetry);
            if (record ! = null) {
                TupleRecordData data = (TupleRecordData) record.getRecordData();
                System.out.println("field1:" + data.getField(0) + ", field2:" + data.getFie
ld("field2"));
            }
        }
    } catch (SubscriptionSessionInvalidException | SubscriptionOffsetResetException e) {
        // subscription exception, will not recover
        // print some log or just use a new consumer
        consumer.close();
        consumer = new Consumer(TEST_PROJECT, TEST_TOPIC, TEST_SUB_ID, config);
    } catch (ResourceNotFoundException | InvalidParameterException e) {
        // - project, topic, shard, subscription not found
        // - seek out of range
        // - sometimes shard operation cause ResourceNotFoundException
        // should make sure if resource exists, print some log or just exit
    } catch (DatahubClientException e) {
        // - network or other exception exceed retry limit
        // can just sleep and retry
    }
}
// close before exit
consumer.close();
```

9.5. Usage notes

A consumer or producer cannot access DataHub by using multiple threads. If you need to use multiple threads, specify a different consumer or producer for each thread.