Alibaba Cloud
Log Service

Real-time subscription and consumption

Issue: 20190805

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 due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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 contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Overview

After LogHub collects logs, Log Service consumes these logs in the following ways.

Method	Scenario	Real-time performance	Storage duration
Real-time consumption (LogHub)	StreamCompute and Realtime Compute	Real time	Custom
Index and query (LogSearch)	Applicable to online query of recent hot data	Real time (delayed for one second in 99 .99% cases, and up to three seconds)	Custom
Shipping and storage (LogShipper)	Applicable to full log storage for offline analysis	Delayed for 5-30 minutes	Depends on the storage system

Real-time consumption

LogHub provides the API operation to pull logs and support real-time log consumption. Log Service consumes logs in a shard in the following steps:

1. Obtain a cursor based on conditions such as time, Begin, and End.
2. Read logs by using the cursor and step, and return the next cursor.
3. Move the cursor continuously to consume logs.

> Note:
> To consume or query logs, Log Service needs to read logs. For more information about the differences between consuming logs and querying logs, see #unique_4.

Consume logs by using SDKs

Log Service provides SDKs in multiple programming languages such as Java, Python, and Go. These SDKs support log consumption based on API operations. For more information about the SDKs, see Overview.

Consume logs by using consumer groups

Consumer groups are the advanced method Log Service provides for LogHub consumers to consume logs. Consumer groups provide a lightweight computing

framework that allows multiple consumers to concurrently consume data in a Logstore. Consumer groups can also automatically assign shards, maintain the order of log processing, and resume transmission from a breakpoint. Go, Python, and Java SDKs support consumer groups.

Consume logs by using StreamCompute

- Use the Spark Streaming client to consume logs.
- Use Storm Spout to consume logs.
- Use the Flink connector to consume logs. The Flink connector consists of the consumer and producer.

Consume logs by using cloud services

- Use CloudMonitor to consume logs in monitoring scenarios.
- Use Function Compute to consume logs.
- Use E-MapReduce to consume logs. For more information, see Use Storm to consume LogHub logs and Use Spark Streaming to consume LogHub logs.

Consume logs by using open-source services

Use Flume to consume LogHub logs: you can use Flume to consume logs and import logs to Hadoop file system (HDFS) instances.

Query and analysis

For more information, see Overview. You can query and analyze logs in the following ways:

- Query logs in the Log Service console. For more information, see Overview.
- Query logs by using the SDKs or API operations of Log Service. Log Service provides HTTP-based RESTful API operations. The API operations support full-featured log queries. For more information, see Overview.

Shipping and storage

- Ship logs to Object Storage Service (OSS): stores logs for a long period or analyzes logs based on E-MapReduce.
- Use Function Compute to customize shipping.
- Ship data to MaxCompute by using DataWorks: ships logs to MaxCompute by using Data Integration of DataWorks to perform big data analysis.

Other method for log consumption

Security log service: Log Service connects to cloud security services and uses an independent software vendor (ISV) to consume logs of cloud services.

2 Consume logs

Log Service provides the SDK in various languages, such as Java, Python, and Go. You can use the SDK to call Log Service operations and consume logs.

Use the SDK to consume logs

The following example shows how to use the Java SDK to consume data in ShardId. For more information, see SDK Reference.

```
Client   client  =  new   Client ( host ,  accessId ,  accessKey );

    String   cursor  =  client . GetCursor ( project ,  logStore ,
shardId ,  CursorMode . END ). GetCursor ();
    System . out . println (" cursor  = " + cursor );
    try  {
      while  ( true ) {
        PullLogsRe  quest   request  =  new   PullLogsRe  quest (
project ,  logStore ,  shardId ,  1000 ,  cursor );
        PullLogsRe  sponse   response  =  client . pullLogs ( request
);
        System . out . println ( response . getCount ());
        System . out . println (" cursor  = " +  cursor  + "
next_curso  r  = " +  response . getNextCur  sor ());
        if  ( cursor . equals ( response . getNextCur  sor ())) {
            break ;
               }
        cursor  =  response . getNextCur  sor ();
        Thread . sleep ( 200 );
     }
   }
    catch ( LogExcepti  on   e ) {
      System . out . println ( e . GetRequest  Id () +  e .
GetErrorMe  ssage ());
   }
```

Preview logs in the console

Log preview also consumes logs. You can use a browser to log on to the Log Service console and preview some logs in a Logstore on the dedicated preview page.

1. Log on to the Log Service console, and then click the target project name.
2. On the Logstores page, find the target Logstore and click Preview in the Log Consumption column.

3. On the log preview page, select the shard and the log time range, and then click Preview.

   The log preview page displays the log data of the first 10 packets in the specified time range.

3 Consumption by consumer groups

3.1 Use a consumer group to consume logs

The consumer library is an advanced mode of log consumption in Log Service, and provides the consumer group concept to abstract and manage the consumption end. Compared with using SDKs directly to read data, you can only focus on the business logic by using the consumer library, without caring about the implementation details of Log Service, or the load balancing or failover between consumers.

Spark Streaming, Storm, and Flink connector use consumer library as the base implementation.

Basic concepts

You must understand two concepts before using the consumer library: consumer group and consumer.

- Consumer group

  A consumer group is composed of multiple consumers. Consumers in the same consumer group consume the data in the same Logstore and the data consumed by each consumer is different.

- Consumer

  Consumers, as a unit that composes the consumer group, must consume data. The names of consumers in the same consumer group must be unique.

In Log Service, a Logstore can have multiple shards. The consumer library is used to allocate a shard to the consumers in a consumer group. The allocation rules are as follows:

- Each shard can only be allocated to one consumer.
- One consumer can have multiple shards at the same time.

After a new consumer is added to a consumer group, the affiliations of the shards for this consumer group is adjusted to achieve the load balancing of consumption. However, the preceding allocation rules are not changed. The allocation process is transparent to users.

The consumer library can also save the checkpoint, which allows consumers to consume data starting from the breakpoint after the program fault is resolved and makes sure that the data is consumed only once.

Usage

Add maven dependency

```
< dependency >
  < groupId > com . google . protobuf </ groupId >
  < artifactId > protobuf - java </ artifactId >
  < version > 2 . 5 . 0 </ version >
</ dependency >
< dependency >
  < groupId > com . aliyun . openservic  es </ groupId >
  < artifactId > loghub - client - lib </ artifactId >
  < version > 0 . 6 . 16 </ version >
</ dependency >
```

main .java file

```
public   class   Main  {
   //  Enter   the   domain   name   of   Log   Service   according
to   your   actual   situation .
  private   static   String   sEndpoint  = " cn - hangzhou . log .
aliyuncs . com ";
   //  Enter   the   project   name   of   Log   Service   according
  to   your   actual   situation .
  private   static   String   sProject  = " ali - cn - hangzhou - sls
- admin ";
   //  Enter   the   Logstore   name   of   Log   Service   according
  to   your   actual   situation .
  private   static   String   sLogstore  = " sls_operat  ion_log ";
   //  Enter   the   consumer   group   name   according   to   your
  actual   situation .
  private   static   String   sConsumerG  roup  = " consumerGr  oupX
";
   //  Enter   the   AccessKey   of   data   consumptio  n
according   to   your   actual   situation .
  private   static   String   sAccessKey  Id  = "";
  private   static   String   sAccessKey  = "";
  public   static   void   main ( String  [] args )  throws
LogHubClie  ntWorkerEx  ception ,  Interrupte  dException
 {
            //  The   second   parameter   is   the   consumer
name .  The   consumer   names   in   the   same   consumer   group
must   be   unique .  However ,  the   consumer   group   names   can
  be   duplicate .  Different   consumer   names   start   multiple
processes   on   multiple   machines   to   consume   a   Logstore
in   a   load   balancing   way .  In   this   case ,  the   consumer
  names   can   be   classified   by   machine   IP   address .   The
  ninth   parameter   maxFetchLo  gGroupSize   is   the   number   of
  Logstores   each   time   obtained   from   Log   Service .  Use
  the   default   value .  If   you   must   adjust   the   value ,
make   sure   the   value   range   is  ( 0 , 1000 ].
      LogHubConf  ig   config  =  new   LogHubConf  ig ( sConsumerG
roup , " consumer_1 ",  sEndpoint ,  sProject ,  sLogstore ,
sAccessKey  Id ,  sAccessKey ,  LogHubConf  ig . ConsumePos  ition .
BEGIN_CURS  OR );
```

```
      ClientWork er worker = new ClientWork er ( new
SampleLogH ubProcesso rFactory (), config );
       Thread thread = new Thread ( worker );
      // The ClientWork er automatica lly runs after the
  thread is running and extends the Runnable API .
      thread . start ();
      Thread . sleep ( 60 * 60 * 1000 );
           // Call the Shutdown function of worker to
exit the consumptio n instance . The associated thread
is automatica lly stopped .
       worker . shutdown ();
      // Multiple asynchrono us tasks are generated when
the ClientWork er is running . We recommend that you
wait 30 seconds until the running tasks exit after
the shutdown .
       Thread . sleep ( 30 * 1000 );
 }
}
```

SampleLogHubProcessor.java files

```
public class SampleLogH ubProcesso r implements ILogHubPro
cessor
{
  private int mShardId ;
 // Records the last persistent checkpoint time .
  private long mLastCheck Time = 0 ;
  public void initialize ( int shardId )
 {
     mShardId = shardId ;
 }
 // The main logic of data consumptio n . Catch all
  the exceptions but the caught exceptions cannot be
thrown .
  public String process ( List < LogGroupDa ta > logGroups ,
         ILogHubChe ckPointTra cker checkPoint Tracker )
 {
        // Write checkpoint to Log Service every 30
  seconds . If worker crashes within 30 seconds , the
newly started worker consumes data starting from the
last checkpoint . Slight duplicate data may exist .
      for ( LogGroupDa ta logGroup : logGroups ){
          FastLogGro up flg = logGroup . GetFastLog Group ();
          System . out . println ( String . format ("\ tcategory \ t
:\ t % s \ n \ tsource \ t :\ t % s \ n \ ttopic \ t :\ t % s \ n \
tmachineUU ID \ t :\ t % s ",
                flg . getCategor y (), flg . getSource (), flg .
getTopic (), flg . getMachine UUID ()));
          System . out . println (" Tags ");
          for ( int tagIdx = 0 ; tagIdx < flg . getLogTags
Count (); ++ tagIdx ) {
              FastLogTag logtag = flg . getLogTags ( tagIdx );
              System . out . println ( String . format ("\ t % s \ t
:\ t % s ", logtag . getKey (), logtag . getValue ()));
         }
          for ( int lIdx = 0 ; lIdx < flg . getLogsCou nt
(); ++ lIdx ) {
              FastLog log = flg . getLogs ( lIdx );
              System . out . println ("--------\ nLog : " + lIdx +
", time : " + log . getTime () + ", GetContent Count : " + log .
getContent sCount ());
              for ( int cIdx = 0 ; cIdx < log . getContent
sCount (); ++ cIdx ) {
```

```
                    FastLogCon  tent   content  =  log . getContent  s
( cIdx );
                    System . out . println ( content . getKey () + "\ t
:\ t " +  content . getValue ());
                }
            }
        }
         long   curTime  =  System . currentTim  eMillis ();
        //  Write   checkpoint   to   Log   Service   every   30
seconds .  If   worker   crashes   within   30   seconds ,
        //  the   newly   started   worker   consumes   data   starting
  from   the   last   checkpoint .  Slight   duplicate   data   may
exist .
         if  ( curTime  -  mLastCheck  Time  >  30  *  1000 )
        {
             try
            {
                        // The   parameter   true   indicates   to
update   the   checkpoint   to   Log   Service   immediatel  y .  The
  parameter   false   indicates   to   cache   the   checkpoint   to
  your   local   machine   and   refresh   the   cached   checkpoint
  to   Log   Service   every   60   seconds   by   default .
                 checkPoint  Tracker . saveCheckP  oint ( true );
            }
             catch  ( LogHubChec  kPointExce  ption   e )
            {
                e . printStack  Trace ();
            }
             mLastCheck  Time  =  curTime ;
        }
         return   null ;
    }
    //  The   worker   calls   this   function   upon   exit .  You
can   perform   cleanup   here .
     public   void   shutdown ( ILogHubChe  ckPointTra  cker
checkPoint  Tracker )
    {
        // Saves   the   consumptio  n   breakpoint   to   the   Log
Service .
         try  {
             checkPoint  Tracker . saveCheckP  oint ( true );
        }  catch  ( LogHubChec  kPointExce  ption   e ) {
             e . printStack  Trace ();
        }
    }
}
class   SampleLogH  ubProcesso  rFactory   implements   ILogHubPro
cessorFact  ory
{
   public   ILogHubPro  cessor   generatorP  rocessor ()
    {
        //  Generates   a   consumptio  n   instance .
         return   new   SampleLogH  ubProcesso  r ();
    }
}
```

Run the preceding codes to print all the data in a Logstore. To allow multiple consumers to consume one Logstore, follow the program annotations to modify the program, use the same consumer group name and different consumer names, and start other consumption processes.

Limits and exception diagnosis

Each Logstore can create at most 10 consumer groups. The error `ConsumerGr oupQuotaEx ceed` is reported when the number exceeds the limit.

We recommend that you configure Log4j for the consumer program, which is used to throw the errors occurred in the consumer group and locate the exceptions. Put the log4j.properties file to the resources directory and run the program, the following exception occurs:

```
[ WARN ] 2018 - 03 - 14 12 : 01 : 52 , 747 method : com . aliyun
 . openservic es . loghub . client . LogHubCons umer . sampleLogE
 rror ( LogHubCons umer . java : 159 )
 com . aliyun . openservic es . log . exception . LogExcepti on :
 Invalid loggroup count , ( 0 , 1000 ]
```

See the following log4j.properties configuration for reference:

```
 log4j . rootLogger = info , stdout
 log4j . appender . stdout = org . apache . log4j . ConsoleApp
 ender
 log4j . appender . stdout . Target = System . out
 log4j . appender . stdout . layout = org . apache . log4j .
 PatternLay out
 log4j . appender . stdout . layout . Conversion Pattern = [%- 5p ]
 % d { yyyy - MM - dd HH : mm : ss , SSS } method :% l % n % m % n
```

Status and alarm

1. View the consumer group status on the console
2. View the consumer group delay with CloudMonitor and configure the alarm

Advanced Configuration

For ordinary users, the data can be consumed using the program above, advanced configurations will be discussed in the following.

- Want to consume data that starts at a certain time

  The loghubconfig in the code above has two constructors:

```
// The consumerst arttimeins econds parameter represents
 the number of seconds after 1970 , meaning that the
   data after this is consumed .
 public LogHubConf ig ( String consumerGr oupName ,
                        String consumerNa me ,
                        String loghubEndP oint ,
                        String project , String logStore ,
                        String accessId , String accessKey ,
                        int consumerSt artTimeInS econds );
// Position is an enumeratio n variable , loghubconf
 ig . glaseposit ion . begin_curs or indicates that
 consumptio n starts with the oldest data , loghubconf
```

```
ig . glaseposit ion . end_cursor indicates that
consumptio n starts with the latest data .
public LogHubConf ig ( String consumerGr oupName ,
                       String consumerNa me ,
                       String loghubEndP oint ,
                       String project , String logStore ,
                       String accessId , String accessKey ,
                       ConsumePos ition position );
```

You can use different construction methods according to consumer needs, but note that if the server is saved with checkpoint, then the starting consumption position is based on the checkpoint saved by the server.

- Use RAM user to access Log Service

  You need to set the ram permissions associated with the consumer group, and set the method to reference the documentation of the ram, the permissions that need to be set are as follows:

Action	Resource
log:GetCursorOrData	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:CreateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ListConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ConsumerGroupUpdateCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}
log:ConsumerGroupHeartBeat	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

Action	Resource
log:UpdateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}
log:GetConsumerGroupCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

- Reset the consumption point

  In some scenarios (fill data, repeat the calculation), we need to set a ConsumerGroup point to a certain point in time, so that the current consumer groups can start to consume from the new point. There are two ways:

  1. Delete consumer group
     - Delete consumer group on the console, and restart consumer group program.
     - consumer group program start to consume from default starting point (configured by program)
  2. Reset the current consumer group to a certain point-in-time using SDK
     - The program and Java code example are as follows
     - Restart the consumer program by using the SDK to modify the site.

```
Client client = new Client ( host , accessId , accessKey );
long time_stamp = Timestamp . valueOf (" 2017 - 11 - 15 00 :
00 : 00 "). getTime () / 1000 ;
ListShardR esponse shard_res = client . ListShard ( new
ListShardR equest ( project , logStore ));
ArrayList < Shard > all_shards = shard_res . GetShards ();
for ( Shard shard : all_shards )
{
  shardId = shard . GetShardId ();
  long cursor_tim e = time_stamp ;
  String cursor = client . GetCursor ( project , logStore ,
shardId , cursor_tim e ). GetCursor ();
  client . UpdateChec kPoint ( project , logStore , consumerGr
oup , shardId , cursor );
```

```
}
```

3.2 View consumer group status

The consumer group is an advanced mode of real-time data consumption, which provides multiple consumption instances for the automatic load balancing of Logstore consumption. Both Spark Streaming and Storm use consumer group as the basic mode.

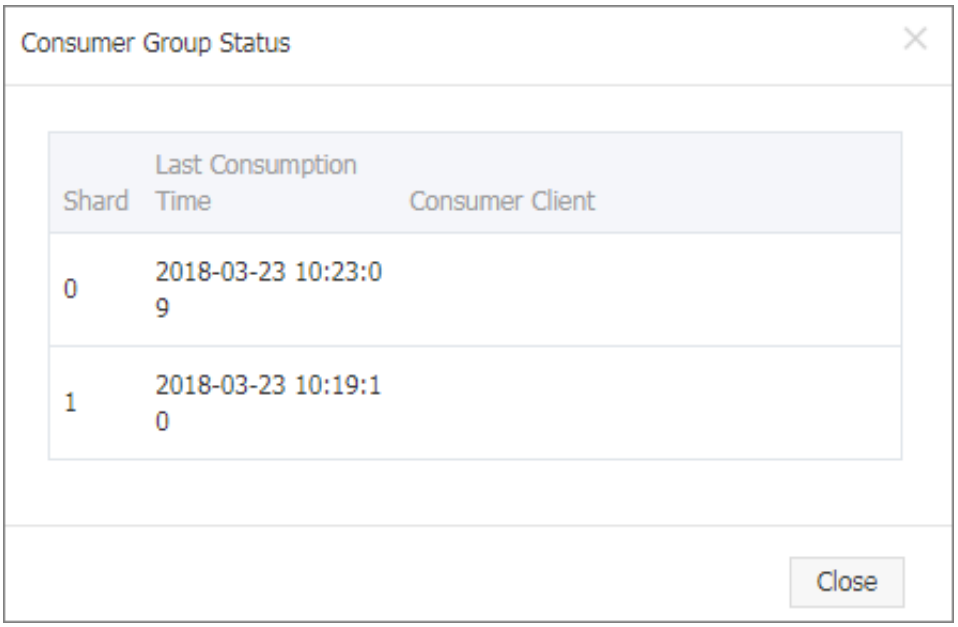
View consumption progress in the console

1. Log on to the Log Service console.
2. On the Project List page, click the project name.
3. Click LogHub - Consume > Consumerin the left-side navigation pane.
4. On the Consumer Groups page, select a Logstore to view whether or not the consumer group function is enabled or not.

Figure 3-1: Consumer

5. Click Status at the right of the consumer group to view the data consumption progress for each shard.

Figure 3-2: Consumption status



As shown in the preceding figure, the Logstore has six shards and corresponds to three consumers. The latest data consumption time for each consumer is shown under the second column. You can use the data consumption time to determine if the current data processing can keep up with data generation. If data processing severely lags behind (that is, data consumption is slower than data generation), we recommend that you increase the number of consumers.

Use APIs/SDKs to view consumption progress

The following commands use Java SDK as an example, which shows how to use APIs to obtain the consumption status:

```
package   test ;
import   java . util . ArrayList ;
import   com . aliyun . openservic  es . log . Client ;
import   com . aliyun . openservic  es . log . common . Consts .
CursorMode ;
import   com . aliyun . openservic  es . log . common . ConsumerGr
oup ;
import   com . aliyun . openservic  es . log . common . ConsumerGr
oupShardCh  eckPoint ;
import   com . aliyun . openservic  es . log . exception .
LogExcepti  on ;
public   class   ConsumerGr  oupTest  {
    static   String   endpoint  = "";
    static   String   project  = "";
    static   String   logstore  = "";
    static   String   accesskeyI  d  = "";
```

```
    static   String   accesskey  = "";
    public   static   void   main ( String []  args )  throws
LogExcepti  on  {
        Client   client  =  new   Client ( endpoint ,  accesskeyI  d
,  accesskey );
              // Retrieve   all   consumer   groups   in   this
Logstore .  If   no   consumer   group   exists ,  the   consumerGr
oups   length   is   0 .
        ArrayList < ConsumerGr  oup >  consumerGr  oups ;
        try {
            consumerGr  oups  =  client . ListConsum  erGroup (
project ,  logstore ).  GetConsume  rGroups ();

        catch ( LogExcepti  on   e ){
            if ( e . GetErrorCo  de () == " LogStoreNo  tExist ")
                System . out . println (" this   logstore   does
not   have   any   consumer   group ");
            else {
               // internal   server   error   branch

            return ;

        for ( ConsumerGr  oup   c :  consumerGr  oups ){
                        // Print   consumer   group   properties ,
including   names ,  heartbeat   timeout ,  and   whether   or   not
  the   consumptio  n   is   in   order .
                       System . out . println (" Name :" +  c .
getConsume  rGroupName ());
                        System . out . println (" Heartbeat   timeout
:" +  c . getTimeout ());
                        System . out . println (" Consumptio  n   in
  order " +  c . isInOrder ());
            for ( ConsumerGr  oupShardCh  eckPoint   cp :  client .
GetCheckPo  int ( project ,  logstore ,  c . getConsume  rGroupName
()).  GetCheckPo  ints ()){
                System . out . println (" shard : " +  cp . getShard
());
              //  Please   format ,  this   time   returns   the
  exact   time   to   millisecon  ds ,  the   length   of   the
integer
                                // Format   the   returned   time   to
  be   precise   to   millisecon  ds   in   the   long   integer .
                                System . out . println (" Last   data
  consumptio  n   time :" +  cp . getUpdateT  ime ());
                String   consumerPr  g  = "";
                if ( cp . getCheckPo  int (). isEmpty ())
                                        consumerPr  g  = "
Consumptio  n   not   started ";
                else {
                                       // UNIX   timestamp .
Measured   in   seconds .  Format   the   value   upon   output .
                    try {
                        int   prg  =  client . GetPrevCur  sorTime (
project ,  logstore ,  cp . getShard (),  cp . getCheckPo  int ()).
GetCursorT  ime ();
                        consumerPr  g  = "" +  prg ;

                    catch ( LogExcepti  on   e ){
                        if ( e . GetErrorCo  de () == " InvalidCur
sor ")
                                                            consumerPr
g  = " Invalid .  The   previous   consumptio  n   time   has
exceeded   the   data   lifecycle   in   the   Logstore .";
                        else {
```

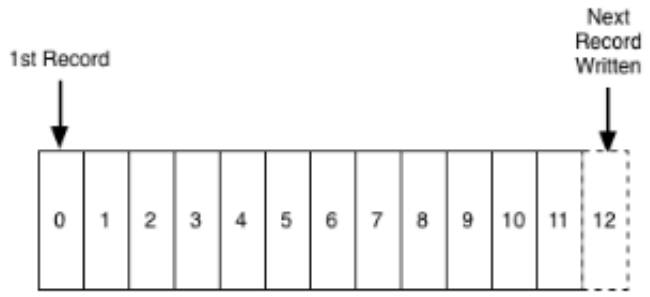
```
                            // internal   server   error
                             throw   e ;



                                 System . out . println (" Consumptio
n   progress :" +  consumerPr  g );
                 String   endCursor  =  client . GetCursor ( project
,  logstore ,  cp . getShard (),  CursorMode . END ).  GetCursor ();
                 int   endPrg  =  0 ;
                 try {
                     endPrg  =  client . GetPrevCur  sorTime ( project
,  logstore ,  cp . getShard (),  endCursor ).  GetCursorT  ime ();

                 catch ( LogExcepti  on   e ){
                    // do   nothing

                                // UNIX   timestamp .  Measured   in
seconds .  Format   the   value   upon   output .
                                 System . out . println (" The
arrival   time   of   the   last   piece   of   data :" +  endPrg );
```

3.3 Consumer group - Monitoring alarm

A consumer group is a group of consumers. Each consumer consumes some of the shards in a Logstore.

The data model of shards can be understood as a queue. The newly written data is added to the tail of the queue and each piece of data in the queue corresponds to a write time. The following shows the data model of shards.

Figure 3-3: Shard Data Model



Basic concepts in collaborative consumption latency alarm:

- Consumption process: The process that a consumer reads data from the head of the queue in sequence.
- Consumption progress: The corresponding write time of the data read by a consumer currently.

- Consumption lagging duration: The difference between the current consumption progress and the latest data write time in the queue, which is measured in seconds.

The consumption lagging duration of a ConsumerGroup takes the maximum value among the consumption lagging durations of all contained shards. When it exceeds the preset threshold (that is, data consumption lags far behind data production), an alarm is triggered.

Procedure

Procedure

1. Log on to the Log Service console. On the Project List page, click the project name.
2. On the Logstore List page, click the Monitor icon at the right of the Logstore.

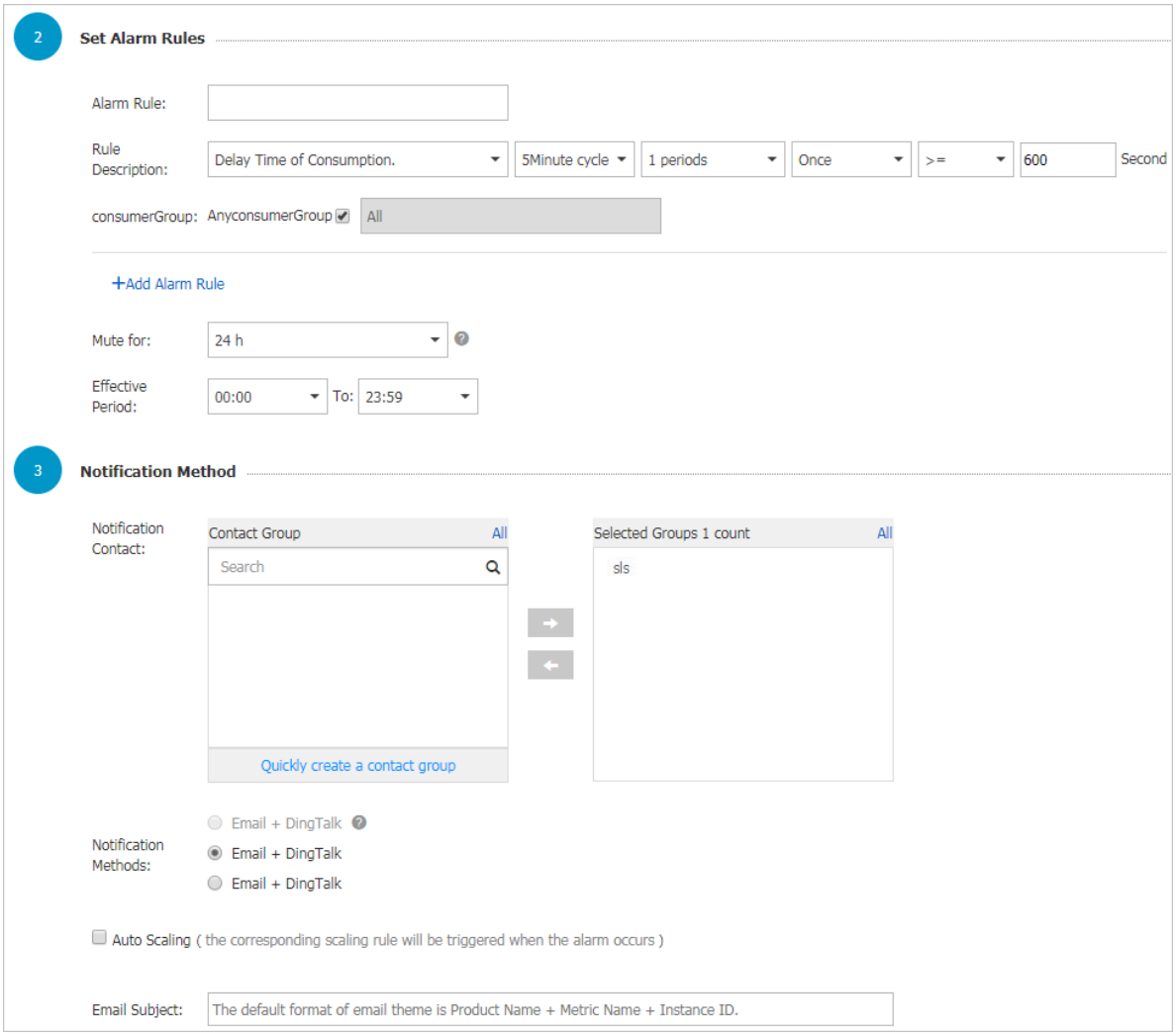
Figure 3-4: Click the Delay Time of Consumption chart name.

3. The figure shows the length, in seconds, of consumption, for all Java groups under logstore. which is measured in seconds. Click Create Alarm Rule in the upper-right corner to enter the Create Alarm Rule page.

Figure 3-5: Create an alarm rule for consumer group spamdetector-report-c.

4. The alarm is triggered if the latency within five minutes is greater than or equal to 600 seconds. Configure the Effective Period and Notification Contact, and then save the rule.

Figure 3-6: Set an alarm rule



Then, an alarm rule is created. If you have any questions about the configurations of alarm rules, open a ticket.

4 Use Fuction Compute to cosume LogHub Logs

4.1 Development guide

The data consumer terminal of Log Service custom ETL function is running on the Alibaba Cloud Function Compute service. You can use function templates provided by Log Service or user-defined functions according to different ETL purposes.

This document explains how to implement a user-defined Log Service ETL function.

Function event

The function event is a collection of input parameters used to run a function, and is in the format of a serialized JSON Object string.

Field descriptions

- jobName field

  The name of the Log Service ETL job. A Log Service trigger on the Function Compute service corresponds to a Log Service ETL job.
- taskId field

  For an ETL job, taskId is the identifier of a deterministic function call.
- cursorTime field

  The unix_timestamp when Log Service receives the last log of the data contained in this function call.
- source field

  This field is generated by Log Service. Log Service regularly triggers function This field is generated by Log Service. Log Service regularly triggers function execution based on the task interval defined in the ETL job. The source field is an important

part of the function event. This field defines the data to be consumed by this function call.

This data source range is composed of the following fields (for more information about the related field definitions, see Log Service glossary).

Field	Description
endpoint	The Service endpoint of the region where the Log Service project resides. Service endpoint
projectName	Project name
logstoreName	Logstore name
Shardid	Identifies a definite shard in the Logstore
beginCursor	The shard location from which to start consuming data
endCursor	The shard location where data consumption ends

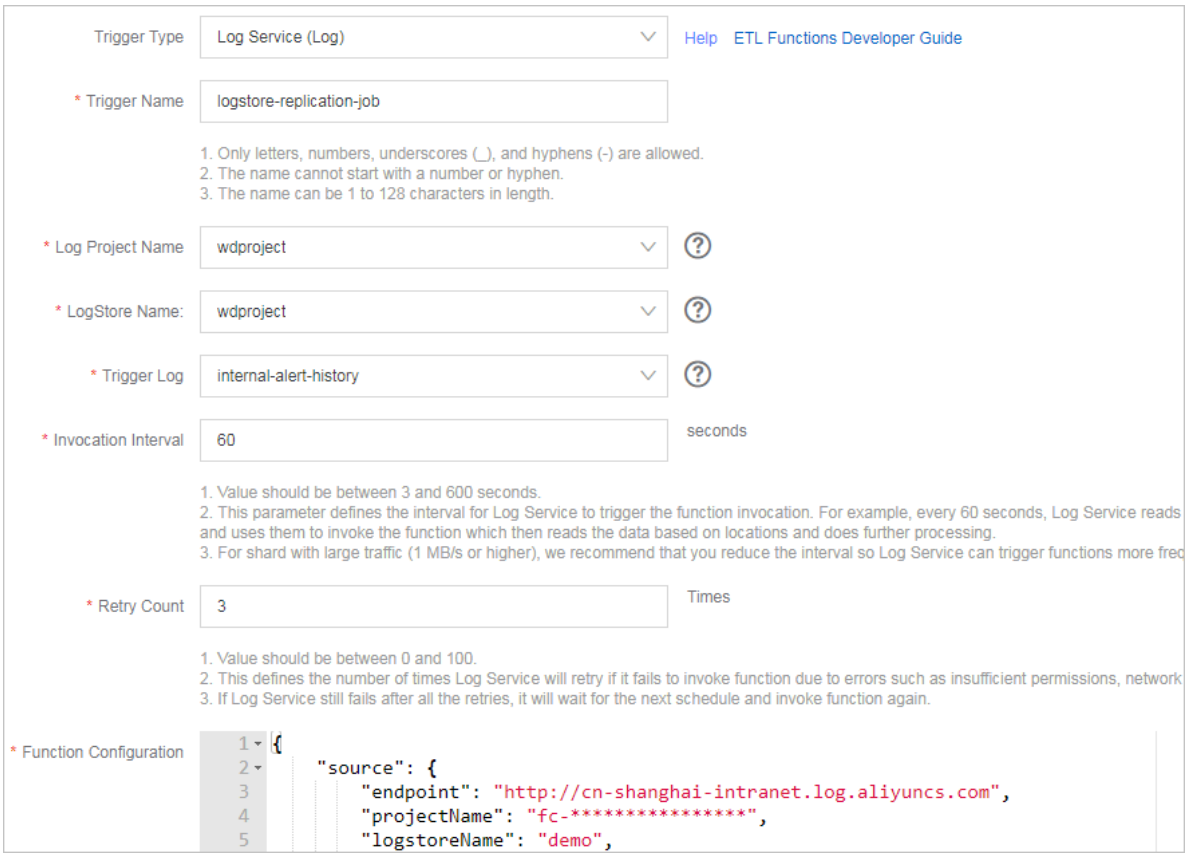
Note:
The [beginCursor, endCursor) of a shard is a left-closed and right-opened interval.

- parameter field

  This JSON Object field is set when you create the ETL job (Log Service trigger of Function Compute). This field is parsed during user-defined function operations to obtain runtime parameters required by the function.

  Set this field in the Function Configuration field when you create a Log Service trigger in the Function Compute console.

  Figure 4-1: Function configuration



Example of function event

```
{
    " source ": {
        " endpoint ": " http :// cn - shanghai - intranet . log .
 aliyuncs . com ",
        " projectNam  e ": " fc - 1584293594  28 ****",
        " logstoreNa  me ": " demo ",
        " shardId ":  0 ,
        " beginCurso  r ": " MTUwNTM5MD  I3NTY1ODcw  NzU2Ng ==",
        " endCursor ": " MTUwNTM5MD  I3NTY1ODcw  NzU2OA =="
    },
    " parameter ": {
        ...
    },
    " jobName ": " fedad35f51  a2a97b466d  a57fd71f31  5f539d2234 ",
    " taskId ": " 9bc06c96 - e364 - 4f41 - 85eb - b6e579214a  e4 ",
```

```
    " cursorTime ":  1511429883
}
```

When debugging a function, you can obtain the cursor by using the GetCursor API and manually assemble a function event for testing according to the preceding format .

Function development

You can implement functions by using many languages such as Java, Python, and Node.js. Log Service provides the corresponding runtime SDKs in various languages to facilitate function integration.

In this section, use Java 8 runtime as an example to show how to develop a Log Service ETL function. As this involves details of Java 8 function programming, read the Java programming guide for Function Compute first.

Java function Template

Currently, Log Service provides user-defined ETL function templates based on the Java 8 execution environment. You can use these templates to implement the custom requirements.

The templates have already implemented the following functions:

- Parse the source, taskId, and jobName fields in the function event.
- Use the Log Service Java SDK to pull data based on the data source defined in source and call the processData API to process each batch of data.

In the template, you must also implement the following functions:

- Use `UserDefine dFunctionP arameter . java` to parse the parameter field in the function event.
- Use the processData API of `UserDefine dFunction . java` to customize the data business logic in the function.
- Replace `UserDefine dFunction` with a name that properly describes your function.

processData method implementation

In processData, you must consume, process, and deliver the data batch according to your specific needs.

SeeLogstoreReplication, which reads data from one Logstore and writes it to another Log Service Logstore.

Notes

 Note:

1. If data is successfully processed by using processData, true is returned. If an exception occurs when data is processed and the exception persists after the retry , false is returned. However, in this case, the function continues to run and Log Service judges it as a successful ETL task, ignoring the incorrectly processed data
2. When a fatal error occurs or the business logic determines that function execution must be terminated prematurely, use the Throw Exception method to exit function execution. Log Service can detect a function operation exception and call function execution again based on the ETL job rules.

Instructions

- When shard traffic is high, configure sufficient memory for the function to prevent an abnormal termination because of function OOM.
- If time-consuming operations are performed in a function or shard traffic is high, set a short function trigger interval and long function operation timeout threshold.
- Grant sufficient permissions to function services. For example, to write Object Storage Service (OSS) data in the function, you must grant the OSS write permission to the function service.

ETL logs

- ETL scheduling logs

  Scheduling logs only record the start time and end time of the ETL task, whether or not the ETL task is successful, and the successfully returned information of the ETL task. If an ETL task encounters an error, it generates an ETL error log and sends an alert email or text message to the system administrator. When creating a trigger, set the trigger log Logstore and activate the index query function for this Logstore.

  Function execution statistics can be written out and returned by functions, such as the Java 8 function outputStream. The default template provided by Log Service writes a serialized JSON Object string. The string is recorded in the ETL task scheduling logs, which facilitates your statistics and query.

- ETL process logs

  This log records the key points and errors for each step in the ETL execution process, including step start and end times, initialization operation completion, and module error information. The ETL process log keeps you up to date on the ETL operation situation at all times. If an error occurs, you can immediately locate the cause in the process log.

  You can use context.getLogger() to record the process logs to the specific project and Logstore of Log Service. We recommend that you enable the index and query functions for this Logstore.

4.2 Configure Function Compute log consumption

Relying on the Function Compute service, Log Service provides a fully-hosted processing service for streaming data.

After configuring an ETL job, Log Service regularly retrieves updated data and triggers function execution, that is, incrementally consumes Log Service Logstore data to complete custom processing tasks in functions. Functions used to process data can be templates provided by Log Service or user-defined functions.

Applicable scenario

Data cleaning and processing

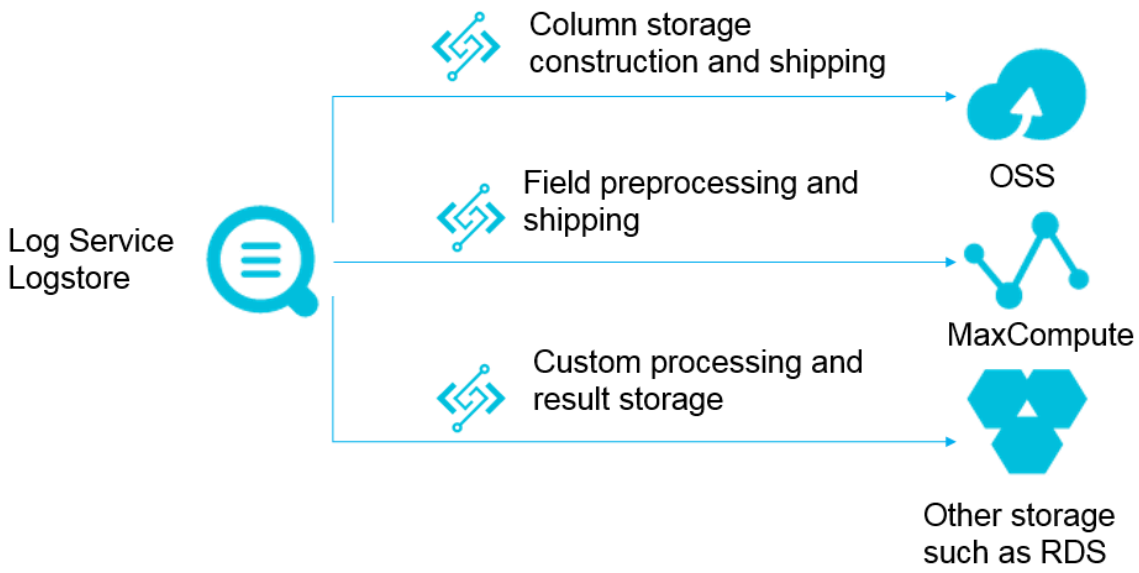
Log Service allows you to quickly collect, process, query, and analyze logs.

Figure 4-2: Data cleaning and processing

Data shipping

Log Service supports shipping data to the destination and constructs the data pipeline between cloud-based big data products.

Figure 4-3: Data shipping



Working principles

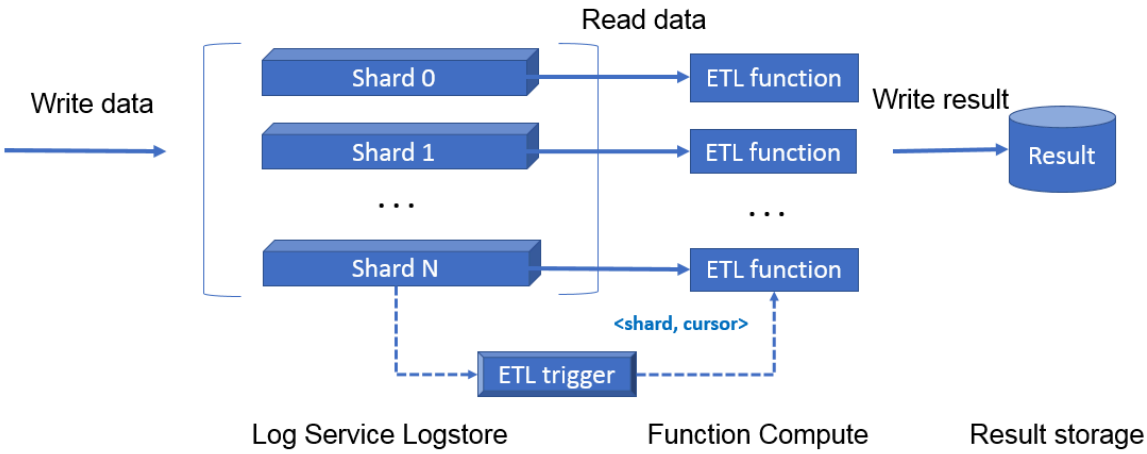
Trigger

A Log Service ETL job corresponds to a Function Compute trigger. After you create an ETL job, Log Service starts a timer based on the job configuration. The timer polls Logstore shard information. When a new log is written, the generated information which is composed of three elements < shard_id, begin_cursor, end_cursor > serves as a function event and triggers function execution.

Log Service ETL job is triggered based on time. For example, if the ETL job trigger interval is 60 seconds and data is consistently written to shard 0 of the Logstore, the function execution is triggered every minute for shard 0. If no new data is written to shard 0, the function execution is not triggered. The input for function execution is

the cursor interval for the last 60 seconds. In the function, shard 0 data is read based on the cursor and then processed.

Figure 4-4: Trigger



ETL functions

You can use the function templates or user-defined functions. Before you get started, we recommend that you learn about the Basic concepts of Function Compute services .

- Function templates maintained by Log Service

  Function templates are maintained on GitHub. Click aliyun-log-fc-functions to access the GitHub.
- User-defined functions

  Implement your own functions. The function configuration formats are related to the specific function implementations. For more information, see Development guide for ETL function.

User Guide

Step 1 Authorize Log Service and prepare resources

1. On the quick authorization page, click Confirm Authorization Policy to grant function trigger permission to Log Service.

2. Create a Log Service project and a Logstore for function process logs.

   If you have not created a project or a Logstore before, create one by following Preparation process.

   > Note:
   > Log Service project and Function Compute service must be in the same region.

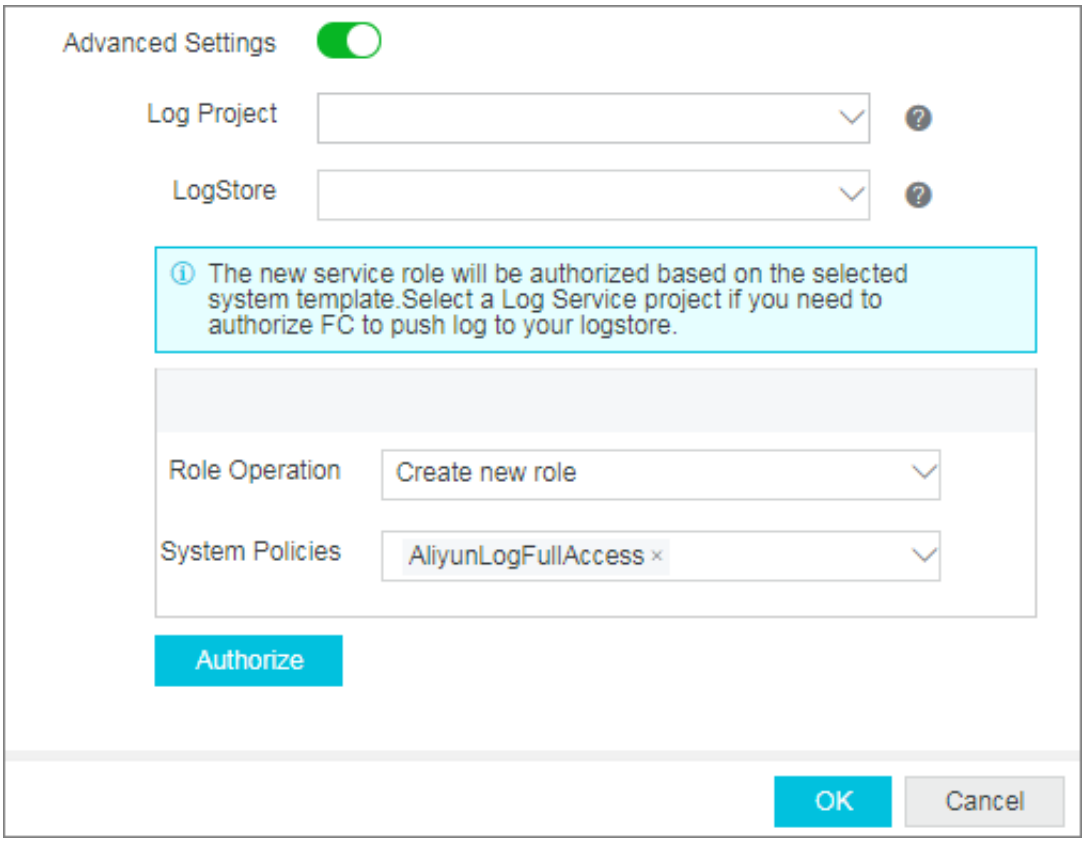
Step 2 Create a service

1. In the Function Compute console, click Create Service.
2. Enter the Service Name and Description. Turn on the Advanced Settings switch.

Configuration item	Meaning
Service name	The name of the Function Compute service to be created. Naming rules: · The name can contain uppercase letters, lowercase letters, numbers, hyphens (-), and underscores (_). · The name must begin with an uppercase letter, lowercase letter, or underscore (_). · The name is case sensitive and must contain 1–128 characters.
Feature description	The description of the new service.
Log project	The name of the Log Service project . The Logstore must be in the same region as the new Function Compute service.
Log repository	The name of the Log Service Logstore . The Logstore must be in the same region as the new Function Compute service.
Role Operation	Create a service role and create the corresponding permissions based on the selected system template. Authorize Function Compute to push logs to the specified Logstore. You can create a new role or select an existing role. To use an existing role, you must select a role that already exists.

Configuration item	Meaning
System Policies	Select a system authorization policy. Select the system authorization policies. Log Service supports two system authorization policies: `AliyunLogFullAccess` and `AliyunLogReadOnlyAccess`.

Figure 4-5: Create a service



After selecting the system authorization policy, click Authorize. The Role Templates page appears. Confirm your role information and permission information, including the Policy Name, Policy Description, and Policy Details. If you are creating a new role, you must confirm the Role Name and Role Description. In the Policy Details, you can refine the authorization policy to customize an authorization policy suitable for this role.

After the successful authorization, click OK to go to the Overview page of the service.

Step 3 Create a function and a trigger

1. On the Overview page of the service, click Create Function.

   Select a Function Template.

   You can select a business template similar to your business model and modify it to create a function, or select a blank function template to customize the function.

   - Log Service template: Log Service template: Log Service provides the business templates `logstore_r eplication` and `oss - shipper - csv`. You can create a function and a trigger based on these templates.
   - Blank template: You can use the blank function template to create a blank function. Then, on the guide page, configure the trigger, function parameters, and write the relevant code to create a function.
2. Configure the Trigger and then click Next.

   If you select a template provided by Log Service, you can configure the trigger directly. If you select the blank template, you must first select the trigger type and then configure the trigger.

   Complete the required items to configure the trigger, such as the trigger name, the project name, and the Logstore name. A Log Service type trigger of Function Compute corresponds to an ETL job of Log Service.

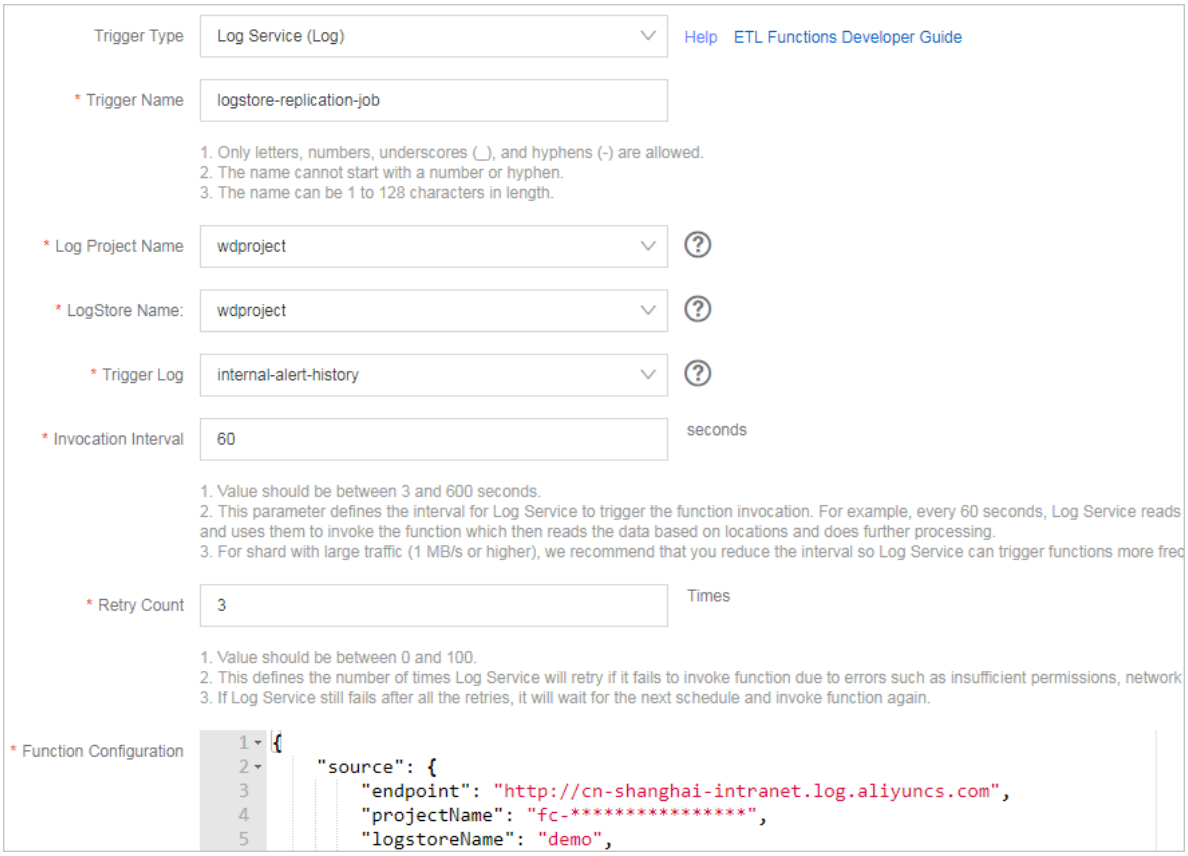
Configuration item	Meaning	Value
Trigger Name	The name of the new trigger.	The trigger name must be 1–128 bytes long and can contain English letters, numbers, underscores (_), and hyphens (-). It cannot start with a number or hyphen (-).
Log Project Name	The name of the Log Service project.	It must be the name of an existing project. This project must be in the same region as your service.

Configuration item	Meaning	Value
Logstore Name	The name of the Log Service project. This trigger regularly transmits the subscribed data of this Logstore to Function Compute for custom processing. You cannot change this parameter after the ETL job is created.	Select an existing Logstore and the Logstore must belong to the project selected in Log Project Name.
Trigger Log	Log Service regularly triggers the function execution of Function Compute. Exceptions during the trigger process and function execution statistics are recorded in this Logstore. You can create an index for the Logstore for future viewing.	It must be the name of an existing Logstore and the Logstore must belong to the project selected in Log Project Name.
Invocation Interval	The interval at which Log Service triggers function execution. For example , when set to 60 seconds , Log Service reads the data location in the last 60 seconds for each Logstore shard, using this as a function event to call function execution. In the function, the user logic reads the shard data and performs computation. If the Logstore shards have a high traffic volume (over 1 Mbit/s), we recommend you set a shorter trigger interval to ensure the data volume processed by each function operation is of a reasonable size.	The value range is 3–600 seconds.

Configuration item	Meaning	Value
Retries Count	If an error occurs when Log Service triggers function execution according to the set trigger interval (such as insufficient permissions, network failure, or function execution return exception), this parameter sets the maximum number of times the function can be re-triggered. If the function is re-triggered the maximum number of times and the operation is still unsuccessful, the trigger interval must elapse before Log Service attempts to trigger the function execution again. The impact of reties on the business varies according to the specific function code implementation logic.	The value range is 0–100 times.

Configuration item	Meaning	Value
Function Configuration	Log Service uses this configuration content as a part of the function event to pass into the function. The way in which this function is used is determined by the custom logic of the function. Different types of functions have different requirements for function configurations. For the vast majority of provided function templates, you must read the instructions when entering your parameters. When no parameters are passed in by default, enter: `{}`.	The configuration content must be a string in JSON Object format.

Figure 4-6: Trigger configuration



> Note:
> You already have the permissions to read/write Logstore data and allow Log Service to call your function.

3. Complete the basic configurations

   such as Function Name and Function Description.Then, click Next.
4. Complete the function permissions.

   Confirm the template authorization and trigger role authorization. Then, click Next.
5. Review your Function Information and Trigger Information. Then, click Create.

View trigger logs

Log on to the Log Service console and create an index for the trigger log Logstore configured in the job. This allows you to view task execution statistics.

View function operation logs

Log on to the Log Service console to view detailed information in the function execution process. For more information, see Logging.

FAQs

I created a trigger, but it does not trigger function execution

1. Make sure you have used quick authorization to authorize Log Service to trigger function execution.
2. Make sure the data in the job's Logstore is incrementally modified, as function execution is triggered when shard data changes.
3. Log on to the Log Service console and check if any exceptions exist in the trigger logs and function operation logs.

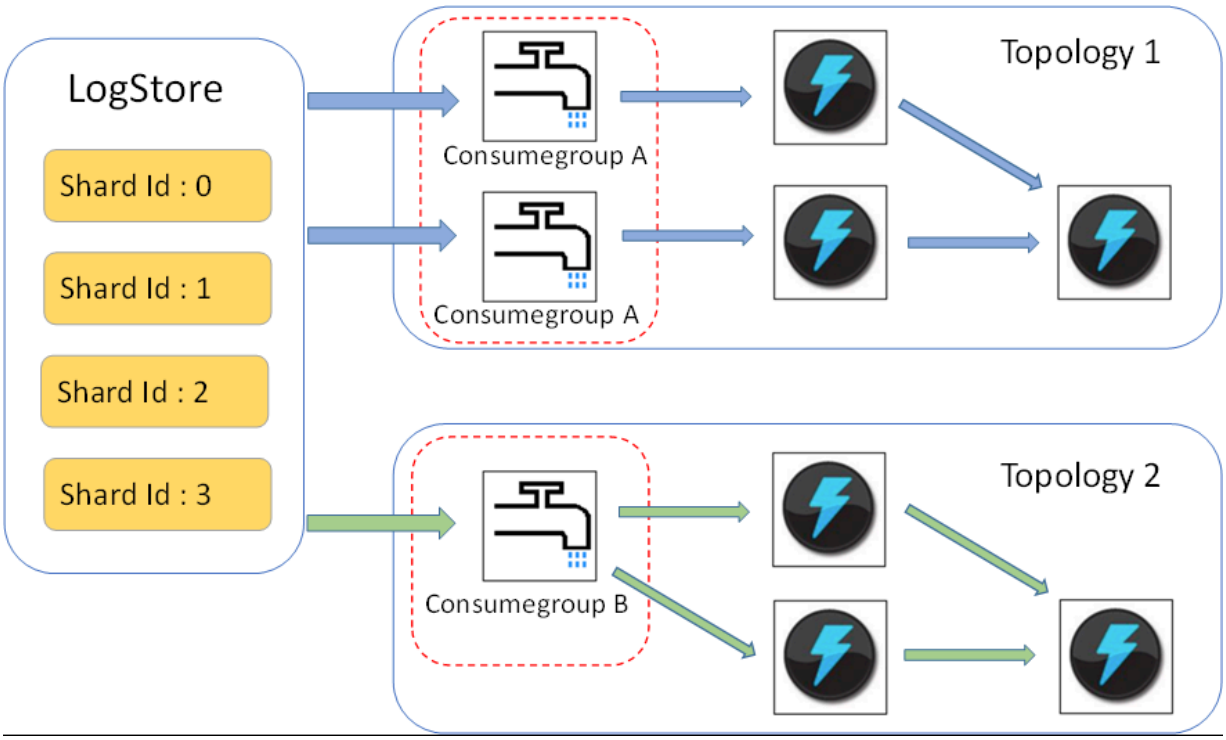
5 Use Storm to consume LogHub logs

LogHub of Log Service provides an efficient and reliable log channel. You can use various methods such as the Logtail and SDK to collect log data in real time. After logs are collected, you can consume the data that is written into LogHub in real-time systems such as Spark Streaming and Storm.

To reduce the cost of LogHub log consumption, Log Service provides LogHub Storm spouts for Storm users to read data from LogHub in real time.

Basic architecture and flowchart

Figure 5-1: Basic architecture and flowchart



- In the preceding figure, LogHub Storm spouts are enclosed in dashed-line boxes. Each Storm topology has a group of spouts that work jointly to read all data from a Logstore. Spouts in different topologies are independent of each other.
- Each topology is identified by a unique LogHub consumer group name. Spouts in the same topology use a consumer library to achieve load balancing and automatic failover.
- Spouts read data from LogHub in real time, send data to bolts in the same topology , and then save consumption checkpoints to the LogHub server on a regular basis.

Limits

- To prevent misuse, each Logstore supports up to 10 consumer groups. You can call the DeleteConsumerGroup operation of the Java SDK to delete unused consumer groups.
- We recommend that the number of spouts be equal to the number of shards. Otherwise, a single spout may be unable to process a large amount of data.
- If the data traffic in each shard exceeds the processing capacity of a single spout, you can split shards to reduce the data traffic of each shard.
- LogHub spouts are mandatorily dependent on the Storm acknowledgment (ACK) mechanism, which is used to confirm that spouts correctly send messages to bolts . Therefore, the ACK method must be called in bolts to confirm the receipt of such messages.

Example

- Create spouts to create a topology

```
    public  static  void  main (  String []  args  )
   {
        String  mode  = " Local "; //  The  local  test  mode .
           String  conumser_g  roup_name  = ""; //  The  unique
  consumer  group  name  for  each  topology .  This
parameter  is  required .  The  name  must  be  3  to
63  characters  in  length .  It  can  contain  lowercase
letters  ( a - z ),  digits  ( 0 - 9 ),  underscore  s  ( _ ),  and
  hyphens  (-).  The  name  must  start  and  end  with  a
lowercase  letter  or  digit .
        String  project  = ""; //  The  Log  Service  project
.
        String  logstore  = ""; //  The  Log  Service  Logstore
.
        String  endpoint  = ""; //  The  domain  name  used
to  access  Log  Service .
        String  access_id  = ""; //  Your  AccessKey .
        String  access_key  = "";
       //  Constructs  the  configurat  ion  of  a  LogHub
Storm  spout .
        LogHubSpou  tConfig  config  =  new  LogHubSpou  tConfig
( conumser_g  roup_name ,
                endpoint ,  project ,  logstore ,  access_id ,
                access_key ,  LogHubCurs  orPosition . END_CURSOR
);
        TopologyBu  ilder  builder  =  new  TopologyBu  ilder ();
       //  Creates  a  LogHub  Storm  spout .
        LogHubSpou  t  spout  =  new  LogHubSpou  t ( config );
       //  The  number  of  spouts  can  be  equal  to
that  of  Logstore  shards  in  actual  scenarios .
        builder . setSpout (" spout ",  spout ,  1 );
        builder . setBolt (" exclaim ",  new  SampleBolt ()).
shuffleGro  uping (" spout ");
        Config  conf  =  new  Config ();
        conf . setDebug ( false );
```

```
        conf . setMaxSpou  tPending ( 1 );
       //  Uses   the   serializat  ion   method   LogGroupDa
taSerializ  Serializer   of   LogGroupDa  ta   if   Kryo   is
used   to   serialize   and   deserializ  e   data .
        Config . registerSe  rializatio  n ( conf ,  LogGroupDa  ta
. class ,  LogGroupDa  taSerializ  Serializer . class );
        if  ( mode . equals (" Local ")) {
            logger . info (" Local   mode ...") ;
            LocalClust  er   cluster   =  new   LocalClust  er ();
            cluster . submitTopo  logy (" test - jstorm - spout ",
conf ,  builder . createTopo  logy ());
            try  {
                Thread . sleep ( 6000  *  1000 ); //  Suspends
the   thread   for   several   minutes .
            }  catch  ( Interrupte  dException   e ) {
                //  TODO   Auto - generated   catch   block
                 e . printStack  Trace ();
            }
             cluster . killTopolo  gy (" test - jstorm - spout ");
             cluster . shutdown ();
       }  else   if  ( mode . equals (" Remote ")) {
             logger . info (" Remote   mode ...");
             conf . setNumWork  ers ( 2 );
             try  {
                 StormSubmi  tter . submitTopo  logy (" stt - jstorm
- spout - 4 ",  conf ,  builder . createTopo  logy ());
             }  catch  ( AlreadyAli  veExceptio  n   e ) {
                 //  TODO   Auto - generated   catch   block
                  e . printStack  Trace ();
             }  catch  ( InvalidTop  ologyExcep  tion   e ) {
                 //  TODO   Auto - generated   catch   block
                  e . printStack  Trace ();
             }
       }  else  {
             logger . error (" invalid   mode : " +  mode );
       }
   }
}
```

- **Consume data in bolts and display only the content of each log**

```
public   class   SampleBolt   extends   BaseRichBo  lt  {
    private   static   final   long   serialVers  ionUID  =
4752656887  774402264L ;
    private   static   final   Logger   logger  =  Logger .
getLogger ( BaseBasicB  olt . class );
    private   OutputColl  ector   mCollector ;
   @ Override
    public   void   prepare (@ SuppressWa  rnings (" rawtypes ")
Map   stormConf ,  TopologyCo  ntext   context ,
            OutputColl  ector   collector ) {
        mCollector  =  collector ;
   }
   @ Override
    public   void   execute ( Tuple   tuple ) {
        String   shardId  = ( String )  tuple
                . getValueBy  Field ( LogHubSpou  t . FIELD_SHAR
D_ID );
       @ SuppressWa  rnings (" unchecked ")
        List < LogGroupDa  ta >  logGroupDa  tas  = ( ArrayList
< LogGroupDa  ta >)  tuple . getValueBy  Field ( LogHubSpou  t .
FIELD_LOGG  ROUPS );
        for  ( LogGroupDa  ta   groupData  :  logGroupDa  tas ) {
```

```
            //  Each   log   group   consists   of   one   or   more
   logs .
             LogGroup   logGroup  =  groupData . GetLogGrou  p ();
             for  ( Log   log  :  logGroup . getLogsLis  t ()) {
                 StringBuil  der   sb  =  new   StringBuil  der ();
                //  Each   log   has   a   time   field   and
 multiple   key - value   pairs .
                 int   log_time  =  log . getTime ();
                 sb . append (" LogTime :"). append ( log_time );
                 for  ( Content   content  :  log . getContent
 sList ()) {
                     sb . append ("\ t "). append ( content . getKey
 ()). append (":")
                             . append ( content . getValue ());
                }
                 logger . info ( sb . toString ());
            }
        }
        //  LogHub   spouts   are   mandatoril  y   dependent   on
 the   Storm   ACK   mechanism ,  which   is   used   to   confirm
 that   spouts   correctly   send   messages   to   bolts .
        //  Therefore ,  the   ACK   method   must   be   called
 in   bolts   to   confirm   the   receipt   of   such   messages .
         mCollector . ack ( tuple );
    }
    @ Override
     public   void   declareOut  putFields ( OutputFiel  dsDeclarer
   declarer ) {
        //  Do   nothing .
    }
}
```

Maven

Use the following code to add dependencies for versions earlier than Storm 1.0 (such as 0.9.6):

```
< dependency >
  < groupId > com . aliyun . openservic  es </ groupId >
  < artifactId > loghub - storm - spout </ artifactId >
  < version > 0 . 6 . 6 </ version >
</ dependency >
```

Use the following code to add dependencies for Storm 1.0 and later versions:

```
< dependency >
  < groupId > com . aliyun . openservic  es </ groupId >
  < artifactId > loghub - storm - 1 . 0 - spout </ artifactId >
  < version > 0 . 1 . 3 </ version >
</ dependency >
```

6 Use Flume to consume LogHub logs

You can use the aliyun-log-flume plug-in to connect Flume to LogHub of Log Service to write and consume log data.

After connecting Flume to LogHub, you can connect Log Service to other data systems, such as Hadoop Distributed File System (HDFS) and Kafka, through Flume. Currently, Flume supports plug-ins for data systems such as HDFS, Kafka, Hive, HBase, and Elasticsearch. You can also find plug-ins for connecting Flume to common data sources in the Flume community. The aliyun-log-flume plug-in provides the LogHub sink and source plug-ins for connecting LogHub and Flume as follows:

- Sink: uses Flume to read data from other data sources and then write data to LogHub.
- Source: uses Flume to consume LogHub data and then write data to other systems.

LogHub sink

You can use the LogHub sink to transmit data from other data sources to LogHub through Flume. Currently, the following parsing formats are supported:

- SIMPLE: writes a Flume event to LogHub as a field.
- DELIMITED: separates Flume events with a delimiter, parses an event into fields based on the configured column names, and then writes them to LogHub.

The following table lists the parameters that can be configured.

Parameter	Description	Required
type	Set this parameter to com.aliyun.loghub.flume.sink.LoghubSink.	Yes
endpoint	The service endpoint of Log Service.	Yes
project	The name of the project.	Yes
logstore	The name of the Logstore.	Yes
accessKeyI d	The AccessKey ID.	Yes
accessKey	The AccessKey Secret.	Yes

Parameter	Description	Required
batchSize	The number of data entries to be written to LogHub each time. Default value: 1000.	No
maxBufferSize	The size of the cache queue. Default value: 1000.	No
serializer	The event serialization format. Valid values: DELIMITED, SIMPLE, and custom serializer. If you specify a custom serializer, enter the complete class name. Default value: SIMPLE.	No
columns	The configured column names. You must specify this parameter if you set the serializer parameter to DELIMITED. Separate multiple columns with a comma (,) and ensure that the columns are sorted in the same order as those in actual data.	No
separatorChar	The delimiter, which is a single character. You can specify this parameter if you set the serializer parameter to DELIMITED. Default value: comma (,).	No
quoteChar	The quote character. You can specify this parameter if you set the serializer parameter to DELIMITED. Default value: double quotation mark (").	No

Parameter	Description	Required
escapeChar	The escape character. You can specify this parameter if you set the serializer parameter to DELIMITED. Default value: double quotation mark (").	No
useRecordTime	Specifies whether to use the value of the timestamp field as the log time. A value of false indicates that the current time is used. Default value: false.	No

LogHub source

You can use the LogHub source to transmit data from LogHub to other data systems through Flume. Currently, the following output formats are supported:

- DELIMITED: writes data to Flume as delimiter logs.
- JSON: writes data to Flume as JSON logs.

The following table lists the parameters that can be configured.

Parameter	Description	Required
type	Set this parameter to com.aliyun.loghub.flume.source.LoghubSource.	Yes
endpoint	The service endpoint of Log Service.	Yes
project	The name of the project.	Yes
logstore	The name of the Logstore.	Yes
accessKeyId	The AccessKey ID.	Yes
accessKey	The AccessKey Secret.	Yes
heartbeatIntervalMs	The heartbeat interval between the Flume client and LogHub, in milliseconds. Default value: 30000.	No

Parameter	Description	Required
fetchInterValMs	The interval for pulling data from LogHub, in milliseconds. Default value: 100.	No
fetchInOrder	Specifies whether to consume log data in order. Default value: false.	No
batchSize	The number of data entries to be read each time. Default value: 100.	No
consumerGroup	The name of the consumer group to be read (which is randomly generated).	No
initialPosition	The start point for reading data. Valid values: begin, end, and timestamp. Default value: begin. Note: If a checkpoint exists on the server, the checkpoint is used.	No
timestamp	The Unix timestamp. You must specify this parameter if you set the initialPosition parameter to timestamp.	No
deserializer	The event deserialization format. Valid values: DELIMITED, JSON, and custom deserializer. If you specify a custom deserializer, enter the complete class name. Default value: DELIMITED.	Yes

Parameter	Description	Required
columns	The configured column names. You must specify this parameter if you set the deserializer parameter to DELIMITED. Separate multiple columns with a comma (,) and ensure that the columns are sorted in the same order as those in actual data.	No
separatorChar	The delimiter, which is a single character. You can specify this parameter if you set the deserializer parameter to DELIMITED. Default value: comma (,).	No
quoteChar	The quote character. You can specify this parameter if you set the deserializer parameter to DELIMITED. Default value: double quotation mark (").	No
escapeChar	The escape character. You can specify this parameter if you set the deserializer parameter to DELIMITED. Default value: double quotation mark (").	No

Parameter	Description	Required
appendTimestamp	Specifies whether to automatically add the timestamp as a field to the end of each row. You can specify this parameter if you set the deserializer parameter to DELIMITED. Default value: false.	No
sourceAsField	Specifies whether to add the log source as a field with the field name __source__. You can specify this parameter if you set the deserializer parameter to JSON. Default value: false.	No
tagAsField	Specifies whether to add the log tags as a field with the field name __tag__: {tag names}. You can specify this parameter if you set the deserializer parameter to JSON. Default value: false.	No
timeAsField	Specifies whether to add the log time as a field with the field name __time__. You can specify this parameter if you set the deserializer parameter to JSON. Default value: false.	No
useRecordTime	Specifies whether to use the log time. A value of false indicates that the current time is used. Default value: false.	No

7 Use Flink to consume LogHub logs

The Flink log connector is a tool provided by Alibaba Cloud Log Service and used to connect to Flink. It consists of two parts: consumer and producer.

The consumer reads data from Log Service. It supports the exactly-once syntax and shard-based load balancing.

The producer writes data into Log Service. When using the connector, you must add the Maven dependency to the project:

```
< dependency >
            < groupId > org . apache . flink </ groupId >
            < artifactId > flink - streaming - java_2 . 11 </
 artifactId >
            < version > 1 . 3 . 2 </ version >
</ dependency >
< dependency >
            < groupId > com . aliyun . openservic  es </ groupId >
            < artifactId > flink - log - connector </ artifactId >
            < version > 0 . 1 . 7 </ version >
</ dependency >
< dependency >
            < groupId > com . google . protobuf </ groupId >
            < artifactId > protobuf - java </ artifactId >
            < version > 2 . 5 . 0 </ version >
</ dependency >
 < dependency >
            < groupId > com . aliyun . openservic  es </ groupId >
            < artifactId > aliyun - log </ artifactId >
            < version > 0 . 6 . 19 </ version >
 </ dependency >
< dependency >
            < groupId > com . aliyun . openservic  es </ groupId >
            < artifactId > log - loghub - producer </ artifactId >
            < version > 0 . 1 . 8 </ version >
</ dependency >
```

Prerequisites

1. Access key is enabled and project and logstore have been created. For detailed instructions, see Preparation.
2. To use a sub-account to access Log Service, make sure that you have properly set the Resource Access Management (RAM) policies of Logstore. For more information, see Grant RAM sub-accounts permissions to access Log Service.

Log consumer

In the connector, the Flink log consumer provides the capability of subscribing to a specific LogStore in Log Service to achieve the exactly-once syntax. During use, you do not need to concern about the change of the number of shards in the LogStore.

Each sub-task in Flink consumes some shards in the LogStore. If shards in the LogStore are split or merged, shards consumed by the sub-task change accordingly.

Associated API

The Flink log consumer uses the following Alibaba Cloud Log Service APIs:

- Getcursorordata

  This API is used to pull data from a shard. If this API is frequently called, data may exceed the shard quota of Log Service. You can use ConfigConstants.LOG_FETCH_DATA_INTERVAL_MILLIS and ConfigConstants.LOG_MAX_NUMBER_PER_FETCH to control the time interval of API calls and the number of logs pulled by each call. For more information about the shard quota, see Shard.

  ```
  configProp  s . put ( ConfigCons  tants . LOG_FETCH_  DATA_INTER
  VAL_MILLIS ,  " 100 ");
  configProp  s . put ( ConfigCons  tants . LOG_MAX_NU  MBER_PER_F
  ETCH ,  " 100 ");
  ```

- ListShards

  This API is used to obtain the list of all shards and shard status in a Logstore. If your shards are always split and merged, you can adjust the period of calling API to find shard changes in time.

  ```
  //  Call   ListShards   every   30s
   configProp  s . put ( ConfigCons  tants . LOG_SHARDS  _DISCOVERY
   _INTERVAL_  MILLIS ,  " 30000 ")
  ```

- CreateConsumerGroup

  This API is called only when consumption progress monitoring is enabled. It is used to create a consumer group to synchronize the checkpoint.

- ConsumerGroupUpdateCheckPoint

  This API is used to synchronize snapshots of Flink to a ConsumerGroup of Log Service.

User Permission

The following table lists the RAM authorization policies required for sub-users to use the Flink log consumer.

Action	Resources
log:GetCursorOrData	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:CreateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ConsumerGroupUpdateCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

Configuration steps

1. Configure the startup parameter.

```
 Properties   configProp  s  =  new   Properties ();
//  Set   the   domain   to   access   Log   Service
 configProp  s . put ( ConfigCons  tants . LOG_ENDPOI  NT ,  " cn -
 hangzhou . log . aliyuncs . com ");
//  Set   the   AccessKey
 configProp  s . put ( ConfigCons  tants . LOG_ACCESS  SKEYID ,  "");
 configProp  s . put ( ConfigCons  tants . LOG_ACCESS  KEY ,  "");
//  Set   the   Log   Service   project
 configProp  s . put ( ConfigCons  tants . LOG_PROJEC  T ,  " ali - cn
 - hangzhou - sls - admin ");
//  Set   the   Log   Service   LogStore
 configProp  s . put ( ConfigCons  tants . LOG_LOGSTO  RE ,  "
 sls_consum  ergroup_lo  g ");
//  Set   the   start   position   to   consume   Log   Service
 configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
 OSITION ,   Consts . LOG_END_CU  RSOR );
//  Set   the   message   deserializ  ation   method   for   Log
 Service
 RawLogGrou  pListDeser  ializer   deserializ  er  =  new
 RawLogGrou  pListDeser  ializer ();
 final   StreamExec  utionEnvir  onment   env  =  StreamExec
 utionEnvir  onment . getExecuti  onEnvironm  ent ();
 DataStream < RawLogGrou  pList >  logTestStr  eam  =  env . addSource
 (
```

```
        new   FlinkLogCo  nsumer < RawLogGrou  pList >( deserializ
er ,   configProp  s ));
```

The preceding is a simple consumption example. As java.util.Properties is used as the configuration tool, configurations of all consumers can be located in ConfigConstants.

Note:

The number of sub-tasks in the Flink stream is independent from that of shards in the Log Service LogStore. If the number of shards is greater than that of sub-tasks, each sub-task consumes multiple shards exactly once. If the number of shards is smaller than that of sub-tasks, some sub-tasks are idle until new shards are generated.

2 Set consumption start position

You can set the start position for consuming a shard on the Flink log consumer. By setting ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, you can set whether to consume a shard from its header or tail or at a specific time. The values are as follows : The specific values are as follows:

- Consts.LOG_BEGIN_CURSOR: Indicates that the shard is consumed from its header , that is, from the earliest data of the shard.
- Consts.LOG_END_CURSOR: Indicates that the shard is consumed from its tail, that is, from the latest data of the shard.
- Constellation S. MAID: indicates that the checkpoint that is saved from a particular Java group starts to consume through configconstants. specify a specific locergroup.
- UnixTimestamp: A string of an integer value, which is expressed in seconds from 1970-01-01. It indicates that the shard is consumed from this time point.

Examples of the preceding three values are as follows:

```
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION ,  Consts . LOG_BEGIN_  CURSOR );
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION ,  Consts . LOG_END_CU  RSOR );
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION , " 1512439000 ");
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION ,  Consts . LOG_FROM_C  HECKPOINT );
```

Note:

If you have set up recovery from the statebackend of flink itself when you start the flink task, then connector ignores the configuration above and uses checkpoint saved in statebackend.

3 set up consumer progress monitoring (optional)

The Flink log consumer supports consumption progress monitoring. The consumption progress is to obtain the real-time consumption position of each shard, which is expressed in the timestamp. For more information, see View consumer group status and Consumer group - Monitoring alarm.

```
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ERGROUP , "
your   consumer   group   name ");
```

Note:

The preceding code is optional. If set, the consumer creates a consumer group first. If the consumer group already exists, no further operation is required. Snapshots in the consumer are automatically synchronized to the consumer group of Log Service. You can view the consumption progress of the consumer in the Log Service console.

4 Support disaster tolerance and exactly once syntax

If the checkpoint function of Flink is enabled, the Flink log consumer periodically stores the consumption progress of each shard. When a job fails, Flink resumes the log consumer and starts consumption from the latest checkpoint that is stored.

The period of writing checkpoint defines the maximum amount of data to be rolled back (that is, re-consumed) if a failure occurs. The code is as follows:

```
 final   StreamExec  utionEnvir  onment   env  =  StreamExec
 utionEnvir  onment . getExecuti  onEnvironm  ent ();
//  Enable   the   exactly - once   syntax   on   Flink
 env . getCheckpo  intConfig (). setCheckpo  intingMode ( Checkpoint
 ingMode . EXACTLY_ON  CE );
//  Store   the   checkpoint   every   5s
 env . enableChec  kpointing ( 5000 );
```

For more information about the Flink checkpoint, see the Flink official document Checkpoints.

Log Producer

The Flink log producer writes data into Alibaba Cloud Log Service.

Note:

The producer supports only the Flink at-least-once syntax. It means that when a job failure occurs, data written into Log Service may be duplicated but never lost.

User Permission

The producer uses the following APIs of Log Service to write data:

- **Log: postlogstorelogs**
- **log:ListShards**

If a RAM sub-user uses the producer, the preceding two APIs must be authorized.

Action	Resources
Log: postlogstorelogs	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/alert/${alarmName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/alert/${alarmName}

Procedure

1. **Initialize the producer.**
   a. **Initialize the configuration parameter Properties for the producer,**

      which is similar to that for the consumer. The producer has some custom parameters. Generally, set these parameters to the default values. You can customize the values in special scenarios.

```
// The number of I / O threads used for sending
 data . The default value is 8 .
 ConfigCons tants . LOG_SENDER _IO_THREAD _COUNT
// The time when the log data is cached . The
 default value is 3000 .
 ConfigCons tants . LOG_PACKAG E_TIMEOUT_ MILLIS
// The number of logs in the cached package . The
   default value is 4096 .
 ConfigCons tants . LOG_LOGS_C OUNT_PER_P ACKAGE
// The size of the cached package . The default
 value is 3Mb .
 ConfigCons tants . LOG_LOGS_B YTES_PER_P ACKAGE
// The total memory size that the job can use .
 The default value is 100Mb .
```

```
ConfigCons  tants . LOG_MEM_PO  OL_BYTES
```

The preceding parameters are not mandatory. You can retain the default values.

b. Reload LogSerializationSchema to define the method for serializing data to RawLogGroup.

RawLogGroup is a collection of logs. For more information about the meaning of each field, see Data model.

To use the shardHashKey function of Log Service, specify the shard into which data is written. You can use LogPartitioner in the following way to generate the HashKey of data:

Example:

```
FlinkLogPr  oducer < String >  logProduce  r  =  new
FlinkLogPr  oducer < String >( new   SimpleLogS  erializer (),
configProp  s );
logProduce  r . setCustomP  artitioner ( new   LogPartiti  oner <
String >() {
     //  Generate   a   32 - bit   hash   value
      public   String   getHashKey ( String   element ) {
          try  {
              MessageDig  est   md  =  MessageDig  est .
getInstanc  e (" MD5 ");
              md . update ( element . getBytes ());
              String   hash  =  new   BigInteger ( 1 ,   md .
digest ()). toString ( 16 );
              while ( hash . length () <  32 )  hash  = " 0 " +
hash ;
              return   hash ;
         }  catch  ( NoSuchAlgo  rithmExcep  tion   e ) {
         }
          return  " 0000000000  0000000000  0000000000
0000000000  0000000000  0000000000  0000 ";
     }
 });
```

Note:

LogPartitioner is optional. If this parameter is not set, data is randomly written into a shard.

2. The following usage example writes a string that is generated by simulation into Log Service:

```
//  Serialize   data   to   the   data   format   of   Log
 Service
 class   SimpleLogS  erializer   implements   LogSeriali
 zationSche  ma < String > {
     public   RawLogGrou  p   serialize ( String   element ) {
         RawLogGrou  p   rlg  =  new   RawLogGrou  p ();
         RawLog   rl  =  new   RawLog ();
```

```
        rl . setTime (( int )( System . currentTim  eMillis () /
1000 ));
        rl . addContent (" message ",   element );
        rlg . addLog ( rl );
        return   rlg ;
    }
}
public   class   ProducerSa  mple  {
    public   static   String   sEndpoint  = " cn - hangzhou . log .
aliyuncs . com ";
    public   static   String   sAccessKey  Id  = "";
    public   static   String   sAccessKey  = "";
    public   static   String   sProject  = " ali - cn - hangzhou -
sls - admin ";
    public   static   String   sLogstore  = " test - flink -
producer ";
    private   static   final   Logger   LOG  =  LoggerFact  ory .
getLogger ( ConsumerSa  mple . class );
    public   static   void   main ( String []  args )  throws
Exception  {
        final   ParameterT  ool   params  =  ParameterT  ool .
fromArgs ( args );
        final   StreamExec  utionEnvir  onment   env  =
StreamExec  utionEnvir  onment . getExecuti  onEnvironm  ent ();
        env . getConfig (). setGlobalJ  obParamete  rs ( params );
        env . setParalle  lism ( 3 );
        DataStream < String >  simpleStri  ngStream  =  env .
addSource ( new   EventsGene  rator ());
        Properties   configProp  s  =  new   Properties ();
       //  Set   the   name   of   the   domain   used   to
access   Log   Service .
        configProp  s . put ( ConfigCons  tants . LOG_ENDPOI  NT
,   sEndpoint );
       //  Set   the   AccessKey   to   access   Log   Service
        configProp  s . put ( ConfigCons  tants . LOG_ACCESS
SKEYID ,   sAccessKey  Id );
        configProp  s . put ( ConfigCons  tants . LOG_ACCESS  KEY
,   sAccessKey );
       //  Set   the   Log   Service   project   into   which
logs   are   written
        configProp  s . put ( ConfigCons  tants . LOG_PROJEC  T ,
sProject );
       //  Set   the   Log   Service   LogStore   into   which
logs   are   written
        configProp  s . put ( ConfigCons  tants . LOG_LOGSTO  RE
,   sLogstore );
        FlinkLogPr  oducer < String >  logProduce  r  =  new
FlinkLogPr  oducer < String >( new   SimpleLogS  erializer (),
configProp  s );
        simpleStri  ngStream . addSink ( logProduce  r );
        env . execute (" flink   log   producer ");
    }
    //  Simulate   log   generation
    public   static   class   EventsGene  rator   implements
SourceFunc  tion < String > {
        private   boolean   running  =  true ;
       @ Override
        public   void   run ( SourceCont  ext < String >  ctx )
throws   Exception  {
            long   seq  =  0 ;
            while  ( running ) {
                Thread . sleep ( 10 );
                ctx . collect (( seq ++) + "-" +  RandomStri
ngUtils . randomAlph  abetic ( 12 ));
```

```
            }
        }
        @ Override
         public   void   cancel () {
             running  =  false ;
        }
    }
}
```

8 Use Spark Streaming to consume LogHub logs

E-MapReduce provides a set of universal interface to consume LogHub logs in real time by using Spark Streaming. For more information, see GitHub.

9 Use CloudMonitor to consume LogHub logs

CloudMonitor can directly consume Logstore data under LogHub to provide monitoring functions,

such as:

- Alarm on keywords in logs
- Statistics of QPS and RT in unit time
- Statistics of PV and UV in unit time

10 Use Go consumer groups to consume LogHub logs

Aliyun LOG Go Consumer Library is a consumer library compiled in Go. It enables multiple consumers to consume data in a Logstore at the same time. Go Consumer Library provides a high-performance mode for you to consume logs in Log Service . It allows you to control resources to be consumed and automatically reconnect to Log Service after disconnection. Using Go Consumer Library, you can focus on the business logic and do not need to worry about the shard distribution, checkpoint, or failover.

For more information about Go Consumer Library, see aliyun-log-go-sdk/consumer/ on GitHub.