Log Service

Product Introduction

Product Introduction

The Log Service (or Log for short) is an all-in-one service for log-type data that allows you to quickly complete log data collection, consumption, shipping, query, and analysis without the need for development. Log Service can help you increase O&M efficiency, and build processing capabilities, that help you easily handle massive log volumes.

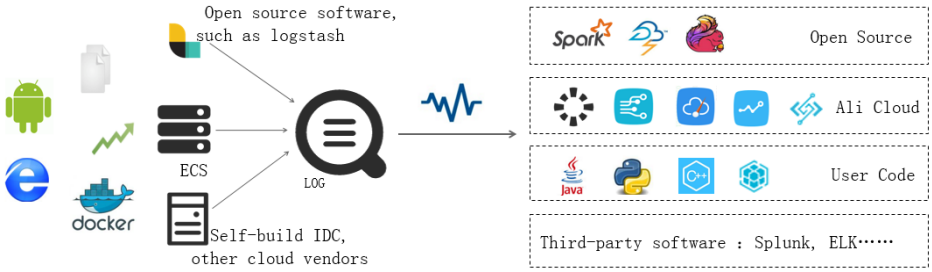
Core functions include:

Real-time log collection and consumption (LogHub)

Functions:

- The LogHub function of Log Service enables access to real-time log data (such as Metric, Event, BinLog, TextLog, and Click data) through ECS, containers, mobile terminals, open source software, and JS.
- A real-time consumption interface is provided for interconnection with real-time computing and service systems.

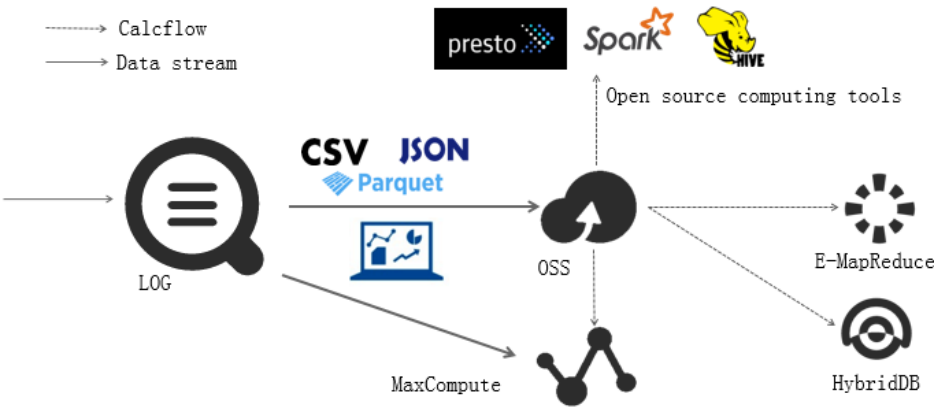
Purposes: ETL, stream computing, monitoring and alarm, machine learning, and iterative computing



LogShipper

Stable and reliable log shipping. Ships LogHub data to storage services for storage and big data analysis. Supports compression, user-defined partitions, and various storage methods such as row and column storage.

Purposes: data warehouse + data analysis, audit, recommendation system, and user profiling

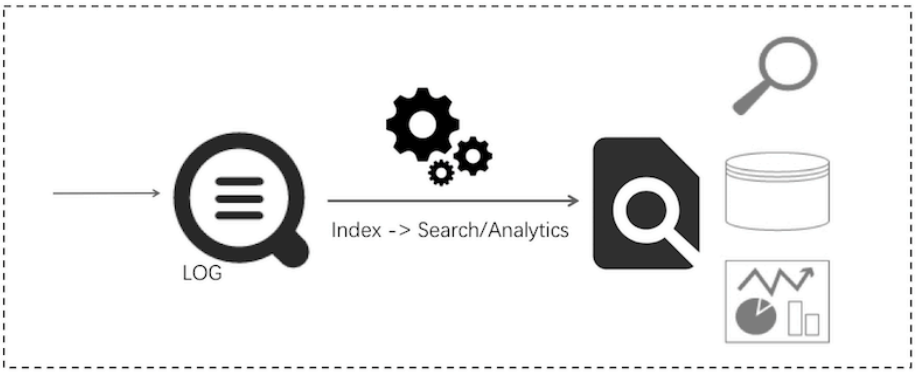


Query and real-time analysis (Search/Analytics)

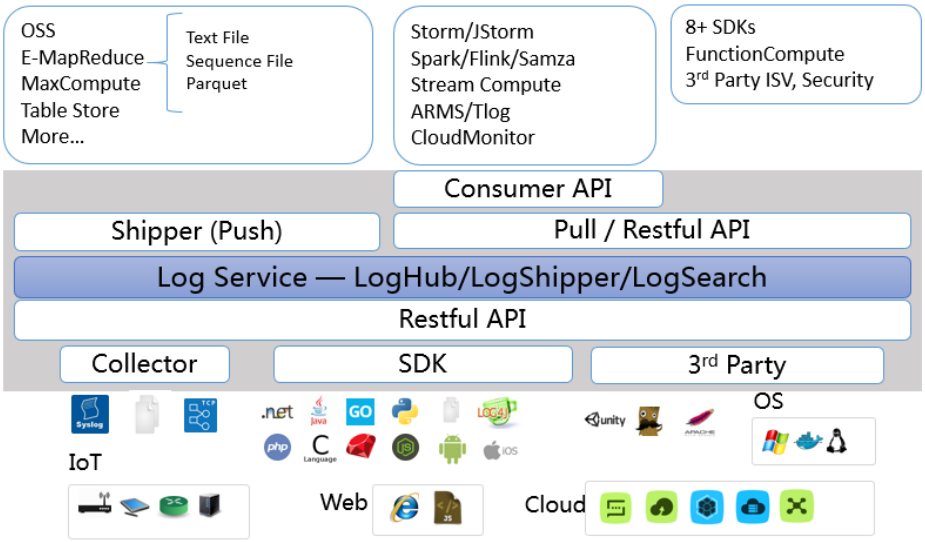
Real-time analysis data indexing and querying.

- Search: keyword, fuzzy match, context, and range
- Analytics: various means such as SQL aggregation
- Visualized: dashboard and report functions
- Interconnection: Grafana and JDBC/SQL92

Purposes: DevOps/online O&M, real-time log data analysis, security diagnosis and analysis, and operation and customer service systems



The Log Service system architecture is shown in the following diagram.



Logtail

Logtail helps you quickly collect logs through the following features :

- Non-invasive log collection based on log files
  - Only read files.
  - Unobtrusive during reading process.
- Secure and reliable
  - Supports file rotation, so data are not lost.
  - Supports local caching.
  - Provides network exception retry mechanism.
- Convenient management
  - Web client.
  - Visualization configuration.
- Comprehensive self-protection
  - Real-time monitoring of process CPU and memory.
  - Consumption and restrictions on CPU/memory usage.

Frontend servers

Frontend machines are built using LVS+Nginx. Its features are as follows:

- HTTP and REST protocols
- Horizontal scaling
  - Support horizontal scaling When traffic increases
  - Frontend machines can be quickly added to improve processing capabilities.
- High throughput, low latency
  - Pure asynchronous processing, a single request exception will not affect other requests.
  - Lz4 compression is adopted to increase the processing capabilities of individual

machines and reduce network bandwidth consumption.

Backend servers

The backend is a distributed process deployed on multiple machines. It provides real-time Logstore data persistence, indexing, query, and shipping to MaxCompute (coming soon). The features of the overall backend service are as follows:

- High data security
  - Each log you write is saved in triplicate.
  - Data are automatically recovered in case of any disk damage or machine downtime.
- Stable service
  - Logstores automatically migrate in case of a process crash or machine downtime.
  - Automatic server load balancing ensures that traffic is distributed evenly among different machines.
  - Strict quota restrictions that prevent abnormal behavior of a single user from affecting other users.
- Horizontal scaling
  - Horizontal scaling is performed using shards as the basic unit.
  - You can dynamically add shards as needed to increase throughput.

Managed security service

- Great accessibility allows you to set up and connect to the service in five minutes, and use Agents to collect data in any network environment.
- LogHub has all the functions of Kafka, and provides complete functional data, such as monitoring and alarms, and supports auto scalling (by PB/day). The use cost is less than 50% of the self-development cost.
- LogSearch/Analytics provide the query saving, dashboard, and alarm functions. The use cost is less than 20% of the self-development cost.
- More than 30 Access Methods, seamless interworking with cloud products (OSS, E-MapReduce, MaxCompute, Table Store, MNS and CDN) and open-source software (e.g. Storm and Spark).

Rich ecosystem

- LogHub supports over 30 collectors, including LogStash and Fluent, and can be easily connected using embedded devices, web pages, servers, and programs. It can also be interconnected with consumption systems such as Spark Streaming, Storm, and CloudMonitor.
- LogShipper supports a variety of data formats (including TextFile, SequenceFile, and Parquet) and user-defined partitions. The data can be directly used by storage engines such as Presto, Hive, Spark, Hadoop, E-MapReduce, and HybridDB.

- LogSearch/Analytics have complete syntaxes and are compatible with SQL-92. The JDBC protocol for interconnection with Grafana will be provided.

Real-time processing

- LogHub: Data can be used as soon as being written. Logtail (data collection agent) can collect and transfer data to the server side within one second (in 99.9% cases).
- LogSearch/Analytics: Data can be searched and analyzed as soon as being written. When multiple search criteria are used, billions of data pieces can be searched within one second. When multiple aggregation conditions are used, hundreds of millions of data pieces can be analyzed within one second.

Complete API/SDK

- The service supports user-defined management and secondary development.
- All functions can be implemented using APIs/SDKs. SDKs for multiple languages are provided to facilitate service management.
- The search and analytic syntaxes are simple (compatible with SQL-92). The interfaces can be used to easily interconnected with the common open software environment (solution for interconnection with Grafana will be available soon).

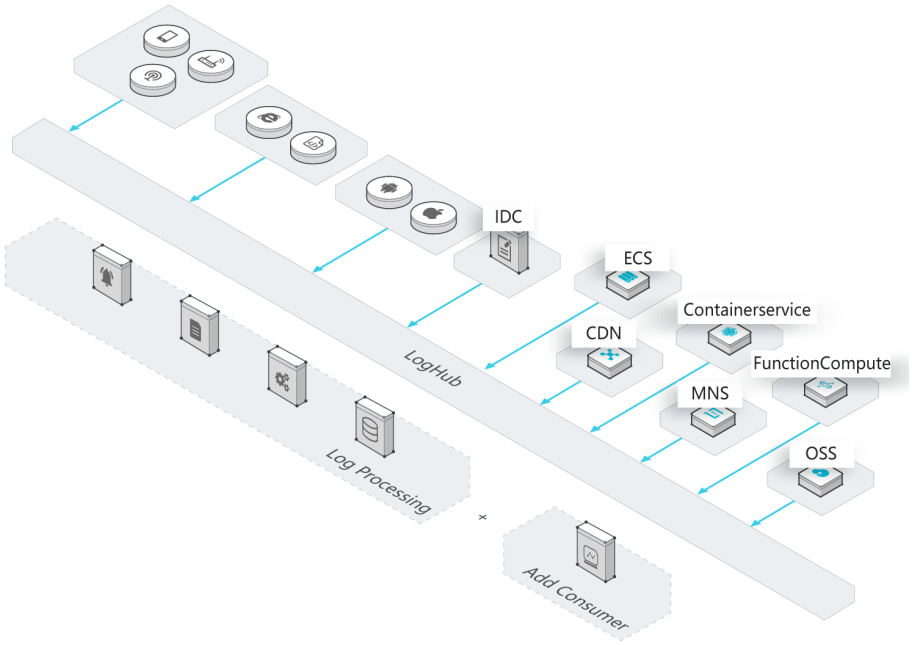
Typical Log Service application scenarios include data collection, real-time computing, data warehousing and offline analysis, product operation and analysis, and O&M and management. The following lists the typical application scenarios. For more details about referrers and their configuration, see Best Practices.

Data collection and integration

The LogHub function of Log Service enables access to massive real-time log data (including Metric, Event, BinLog, TextLog, and Click data) at low costs.

Advantages of the solution:

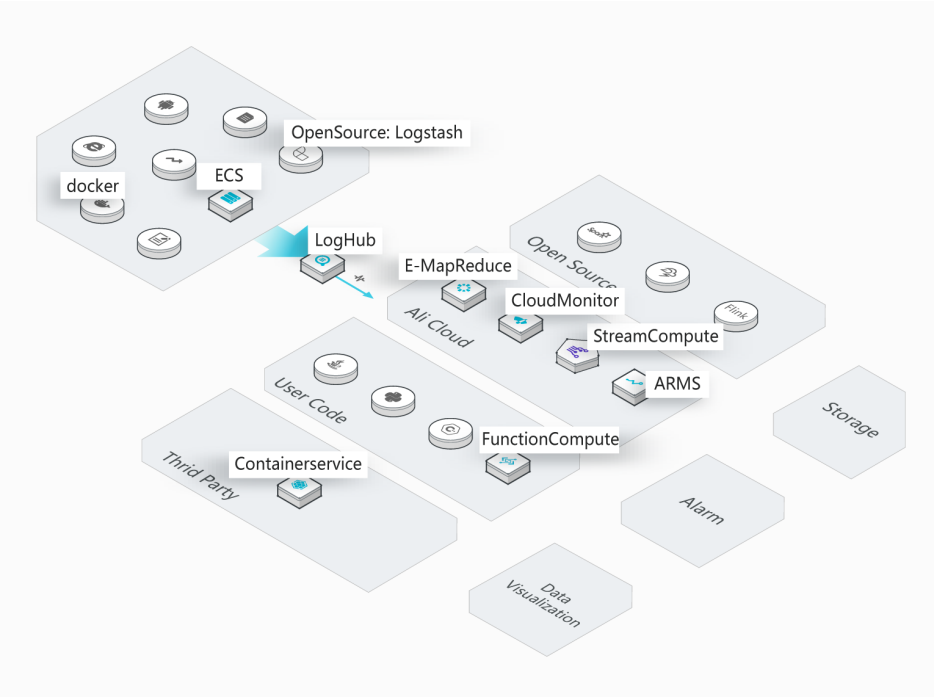
- Easy to use: Over 30 real-time data collection methods are provided for you to quickly set up your platform. The powerful configuration and management capabilities can ease O&M workload. Nodes are available across China and the rest of the world.
- Highly elastic: It helps easily address traffic peaks and service growth.



ETL/Stream Processing

LogHub can work with all types of real-time computing and services, provides complete progress monitoring and alerting functions, and supports SDK/API-based custom consumption.

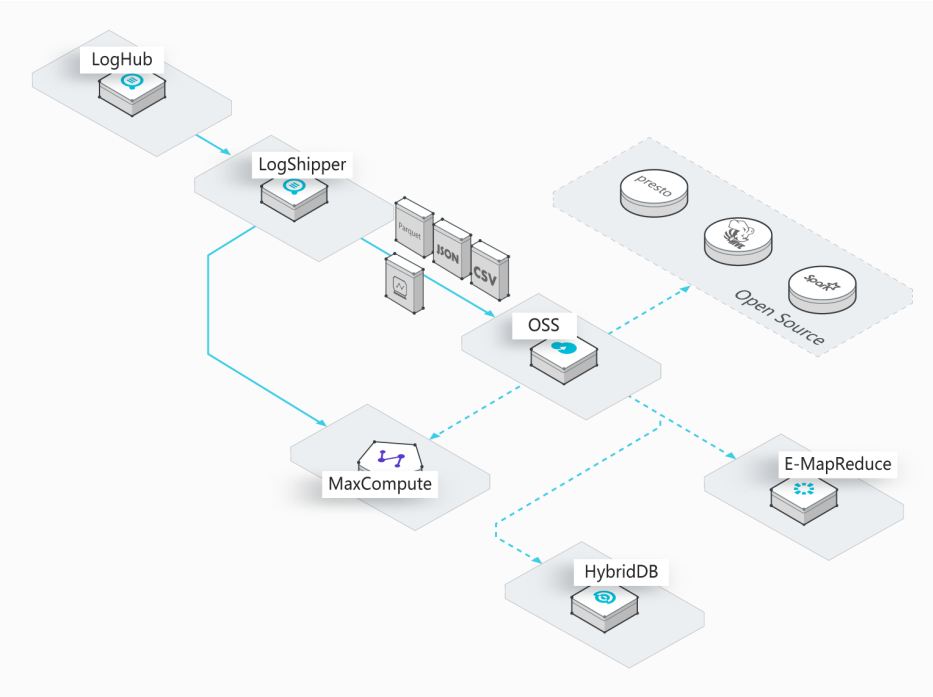
- Easy to operate: It provides various SDKs and programming frameworks and can be seamlessly interconnected with various stream computing engines.
- Rich functions: Various monitoring data and alarm postponing are provided.
- Elastic: PB-grade elasticity and zero latency.



Data Warehouse

LogShipper enables data in LogHub be shipped to storage services and stored in a range of formats such as compression, custom partition, and row/column.

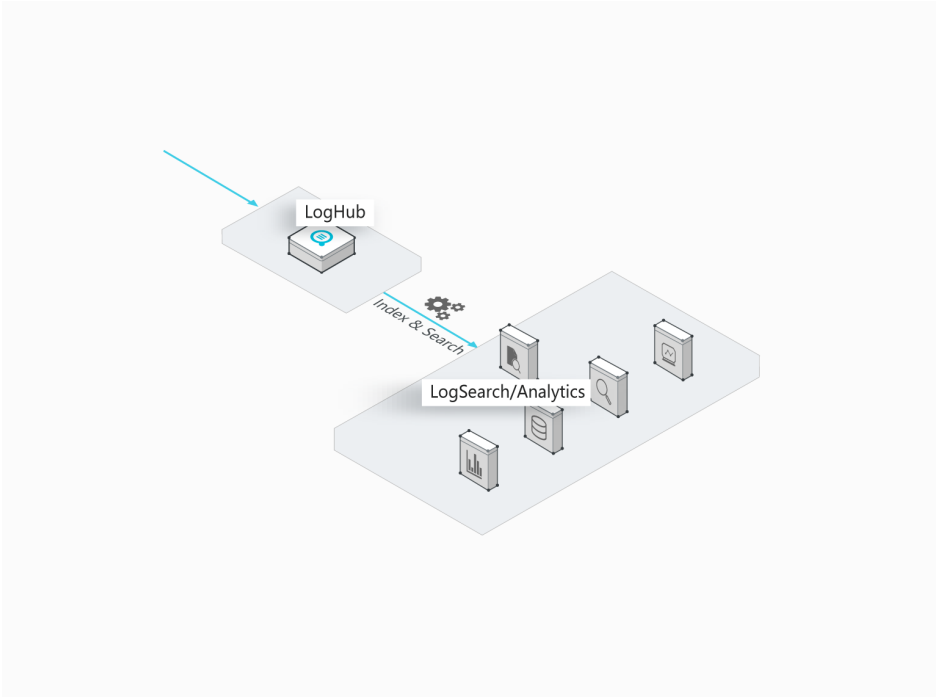
- Massive data: The amount of data is not limited.
- Rich storage formats: Various storage formats are supported, such as row storage, column storage, and TextFile.
- Flexible configuration: Configurations such as user-defined partitions are supported.



Log search and analytics

Log search and analytics support real-time searching data in LogHub, and provide rich means of inquiry: keywords, fuzzy, context, scope, SQL aggregation.

- Strong real-time performance: Write after the query.
- Massive low cost: Support PB/Day indexing capacity, and the cost is 15% of self-built program.
- Strong analytical skills: Support a variety of query means and SQL for aggregation analysis, and provide visualization and alarm function.



Basic concepts

Log

Log is an abstraction of system changes during the running process. The log content is a time-ordered collection of some operations and the corresponding operation results of specified objects. LogFile, Event, BinLog, and Metric data are different carriers of logs. In LogFile, every log file is composed of one or more logs, and every log describes a single system event. Log is the minimum data unit processed in Log Service.

Log group

A log group is a collection of logs and is the basic unit for writing and reading.

Log topic

Logs in a Logstore can be classified by log topics. You can specify the topic when writing or querying logs.

Project

Project is the resource management unit in Log Service and is used to isolate and control resources.

You can manage all the logs and the related log sources of an application by using a project. Projects manage the information of all your Logstores and the log collection machine configuration, and serve as the portal where you can access the Log Service resources.

Logstore

Logstore is a unit in Log Service for the collection, storage, and query of log data. Each Logstore belongs to a project, and each project can create multiple Logstores.

Shard

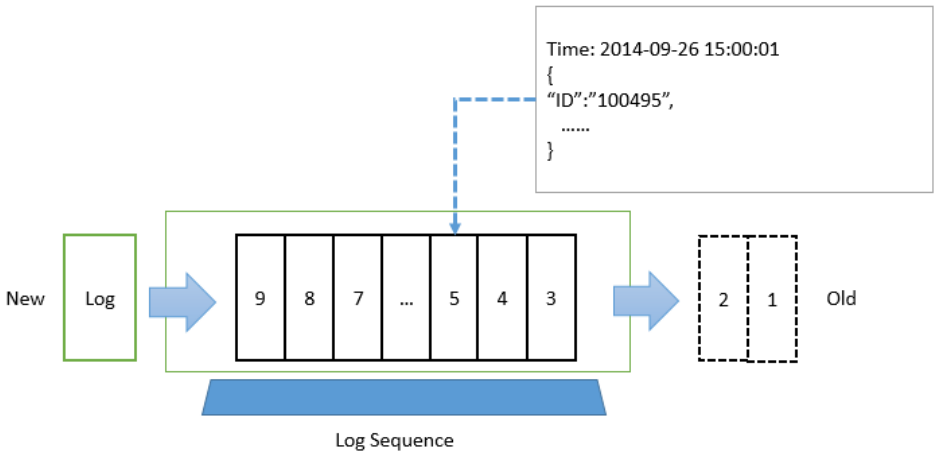
Each Logstore is divided into several shards and each shard is composed of MD5 left-closed and right-open intervals. Each interval range does not overlap with others and the range of all the intervals is the entire MD5 value range.

Half a century ago, the term “log” was associated with a thick notebook written by a ship captain or operator. Nowadays, with the advent of computers, logs are generated and used everywhere. Servers, routers, sensors, GPS devices, orders, and various IoT devices describe the world we live from different angles by generating and using logs. With the computing power, we continuously update our recognition to the whole world and system by collecting, processing, and using logs.

What is a log?

Consider the example of a ship captain’s log. In addition to a recorded timestamp, a log can contain almost all sorts of information, including a text record, an image, weather conditions, and the sailing course. After centuries passed, now the “ship captain’s log” has been expanded to various areas such as orders, payment records, user accesses, and database operations.

The reason why logs are widely used and enduring is that logs are the simplest storage abstraction. Logs are a collection of chronological records that can only be added. The subsequent image is what logs (time-series data) look like.



We can add a record to the end of a log and read the log records from left to right. Each record has a unique log record number with a sequence.

The log sequence is determined by “time”. From the preceding image, we can see that the log time sequence is from right to left. The new event is recorded, and the old event is gradually out of sight. But a log records when and what happened. This is the foundation of recognition and reasoning, no matter to computers, humans, or the whole world.

Logs in Log Service

Log is an abstraction of system changes during the running process. The log content is a time-ordered collection of some operations and the corresponding operation results of specified objects. LogFile, Event, BinLog, and Metric data are different carriers of logs. In LogFile, every log file is composed of one or more logs, and every log describes a single system event. Log is the minimum data unit processed in Log Service.

Log Service defines a log by using the semi-structured data mode. This mode is composed of four data fields: Topic, Time, Content, and Source.

Meanwhile, Log Service has different format requirements for different log fields. See the following table for details.

Data field	Meaning	Format
Topic	A custom field used to mark a batch of logs. For example, access logs can be marked according to sites.	Any string up to 128 bytes, including null strings. By default, this field is a null string.
Time	A reserved field in the log used to indicate the log generation time. Generally this filed is generated directly based on the time in the log.	An integer in the standard UNIX time format. The unit is in seconds. This field indicates the number of seconds since 1970-1-1 00:00:00 UTC.
Content	A field used to record the specific log content. The log content is composed of one or more content items, and each content item is a Key-Value pair.	The key is a UTF-8 encoded string up to 128 bytes, and can contain letters, underscores, and numbers. It cannot start with a number or use any of the following keywords: __time__, __source__, __topic__, __partition_time__, _extract_others_, and __extract_others__. The value can be any string up to 1024*1024 bytes.
Source	A field used to indicate the source of the log. For example, the IP address of the machine where the log is	Any string up to 128 bytes. By default, this field is null.

	generated.	

Various log formats are used in actual usage scenarios. For ease of understanding, the following example describes how to map an original Nginx access log to the Log Service log data model. Assume that the IP address of your Nginx server is 10.249.201.117. The following is an original log of this server.

```
10.1.168.193 - - [01/Mar/2012:16:12:07 +0800] "GET /Send?AccessKeyId=8225105404 HTTP/1.1" 200 5 "-"
"Mozilla/5.0 (X11; Linux i686 on x86_64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2"
```

Map the original log to the Log Service log data model as follows.

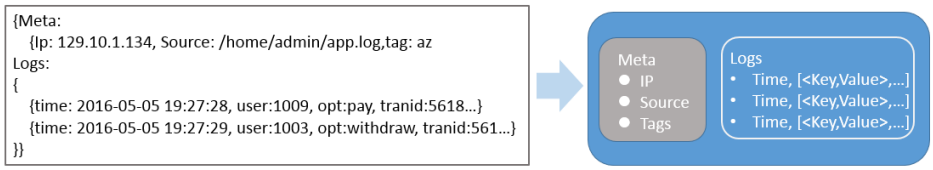
Data field	Content	Description
Topic	“”	Use the default value (null string).
Time	1330589527	The precise log generation time, indicating the number of seconds since 1970-1-1 00:00:00 UTC. The time is converted from the timestamp of the original log.
Content	Key-Value pair	Specific log content.
Source	“10.249.201.117”	Use the IP address of the server as the log source.

You can decide how to extract the original content of the log and combine them into Key-Value pairs. The following table is shown as an example.

Key	Value
ip	“10.1.168.193”
method	“GET”
status	“200”
length	“5”
ref_url	“- ”
browser	“Mozilla/5.0 (X11; Linux i686 on x86_64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2”

A log group is a collection of logs, and it is the basic unit for writing and reading.

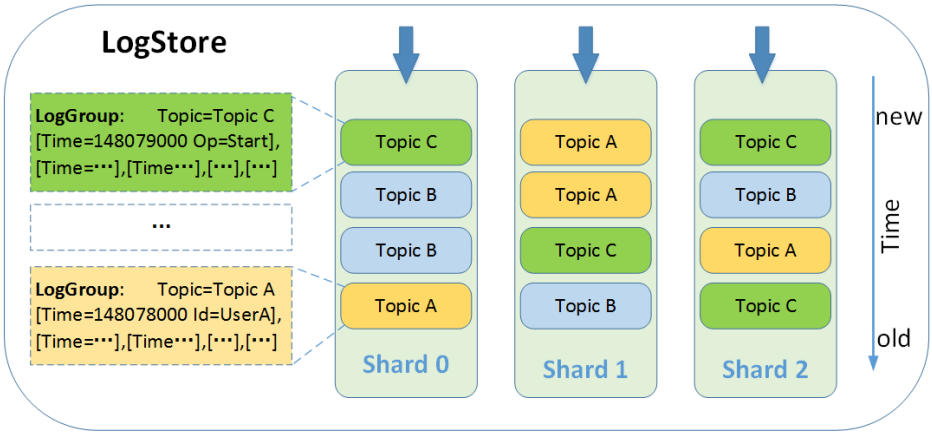
The maximum capacity of a log group is up to 4096 logs or 10 MB.



Logs in the same Logstore can be grouped by log topics. You can specify the topic when writing a log, and specify the log topic when querying logs. For example, platform users can use the user IDs as the log topics and write them into the logs. In this way, users can select to only view their own logs based on log topics. If there is no need to group the logs in one Logstore, the same log topic can be used for all logs.

> **Note:** A null string is a valid log topic, and it is the default log topic for writing and querying logs. Therefore, if there is no need to use a log topic, the easiest method is to use a null string, when writing and querying logs.

The following diagram describes the relationship between Logstore, log topic, and log.



A project is the Log Service' s resource management unit. Projects isolate and control resources.

You can use a project to manage logs and related log sources of one application. A project manages the Logstores of a user and machine configurations of a log collection. It also serves as the portal for users access the Log Service resources.

Projects provide the following functions.

Projects help you organize and manage different Logstores. In actual use, you may use Log Service to collect and store different project, product, or environment logs in a centralized manner. You can classify different logs for management in different projects to facilitate subsequent log consumption, exporting, or indexing. In addition, projects also carry the log access permission management functions.

Projects provide portals to access Log Service resources. Log Service allocates a unique access portal to each created project. This access portal supports log writing, reading, and management via the network.

You can use the Log Service console to perform the following project operations.

- Create a project
- View project list
- Manage a project
- Delete a project

Logstores are the units used in Log Service for log data collection, storage, and query. Each Logstore belongs to one project, and multiple Logstores can be created for a single project.

You can create multiple Logstores for one project according to your needs. Typically, an independent Logstore is created for each type of log in one application. For example, assume that you have a game application “big-game”, and there are three types of logs on the server: operation_log, application_log, and access_log. You can first create a project named “big-game”, and then create three Logstores for the three types of logs under this project.

Whether writing or querying logs, you must specify a Logstore for the operation. If you want to ship the log data to MaxCompute (coming soon) for offline analysis, the data will be shipped in Logstore units for data synchronization (that is, the data in a single Logstore are shipped to a single MaxCompute table).

Logstores provide the following functions.

- Log collection, supports real-time logging
- Log storage, supports real-time consumption
- Index creation, supports real-time log query
- A data tunnel that ships logs to MaxCompute

You can use the Log Service console to perform the following Logstore operations.

- Create a Logstore
- View Logstore list
- Modify Logstore configurations
- Delete a Logstore

Logstore read/write logs must be saved in a certain shard. Each Logstore is divided into several shards and each shard is composed of MD5 left-closed, right-open intervals. These intervals do not overlap and the interval ranges add up to the entire MD5 value range.

Range

When creating a Logstore, the entire MD5 range is automatically divided evenly based on the specified number of shards. Each shard has a certain range within the following value range: [00000000000000000000000000000000,ffffffffffffffffffffffffffffffff).

Each shard is composed of the following keys.

- BeginKey: indicates the start of the shard. This key is included in the shard range.
- EndKey: indicates the end of the shard. This key is excluded from the shard range.

With the shard range, you can write logs by specifying Hash Key, as well as split or merge shards. The corresponding shard must be specified when reading data from it. You can use load balancing mode or specified hash key mode when writing data. In load balancing mode, a data packet is written to an available shard at random. In specified Hash Key mode, data is written to the shard whose range includes the specified key.

In the following example, assume that the MD5 value range of the Logstore is [00,ff), and the Logstore contains 4 shards with the following ranges.

Shard No.	Range
Shard0	[00,40)
Shard1	[40,80)
Shard2	[80,C0)
Shard3	[C0,FF)

If you write a log and specify the MD5 key as 5F, the log data will be written into shard1 that contains the MD5 key 5F. If you specify the MD5 Key as 8C, the log data will be written into shard2 that contains the MD5 key 8C.

Shard read/write capacities

Each shard has certain service capacities.

- Write: 5 MB/s, 2000 times/s
- Read: 10 MB/s, 100 times/s

You can calculate the number of shards needed based on the traffic. If the data traffic exceeds the read and write capabilities, split the Shard in time to increase the number of Shards, so as to achieve greater read and write capabilities. If data traffic is far less than the maximum read and write capabilities of the partition, it is recommended that you merge the partitions to reduce the number of partitions, and save the zoning costs.

For example, if you have two Shards in readwrite status, and provide 10MB/s of data writting in maximum. But data-writing traffic has reached to 14MB, it is recommended to split one of the Shards so that the number of readwrite Shards reaches three. If data-writing traffic is only 3MB/s, then one Shard should meet the needs, it is recommended that you merge two Shards.

Note:

- When writing logs, if the API consistently reports 403 or 500 error, refer to Logstore CloudMonitor metrics to view the traffic and status code and determine whether you need to increase the number of shards.
- For read/write operations that exceed a shard’ s service capacities, the system will attempt to provide the needed services, but the service quality cannot be ensured.

Shard status

- readwrite: capable of reading and writing
- readonly: read-only data

When you create a partition, all partition states are in readwrite status. Split or merge operations change the partition state to readonly and generate a new readwrite partition. The Shard state does not affect the performance of its data read, while the readwrite Shard maintains normal data write performance, and the readonly state partition does not provide data-writing services.

When splitting a Shard, you need to specify a ShardId in a readwrite status and an MD5. The MD5 must be greater than the Shard’ s BeginKey and less than EndKey. Split operations can split two other Shards from one, that is, the number of Shards increase 2 after division. After the split is complete, the original Shardstate specified by split is changed from readwrite to readonly, and the data can still be consumed, but no new data can be written. The two newly generated Shard status are readwrite, behind the original Shard, and the MD5 range of the two Shards covers the range of the original Shard.

In the merge operation, you must specify a Shard in readwrite status, and the specified Shard can not be the last readwrite Shard. Log Service will automatically find the specified Shard on the right side of the Shard, and combine the two Shard. After the merge is complete, the specified Shard and its right side adjacent Shard become readable, the data can still be consumed, but new data can not be write in. At the same time a new readwrite Shard is generated, and MD5 scope of the new Shard covers the original two Shards.

Shard operations

On the Log Service console, you can perform the following shard operations.

- Split a shard
- Merge shards
- Delete a shard