

Alibaba Cloud Log Service

Index and query

Issue: 20190918

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 <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid <i>Instance_ID</i></code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

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

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 Overview.....	1
2 Syntax description.....	4
3 Enable and set indexes.....	8
4 Query logs.....	15
5 Download logs.....	22
6 Data type of index.....	25
6.1 Overview.....	25
6.2 Text type.....	28
6.3 JSON type.....	30
6.4 Value type.....	35
7 Query.....	36
7.1 Query syntax.....	36
7.2 LiveTail.....	43
7.3 Use LogReduce to group log data.....	49
7.4 Context query.....	57
7.5 Saved search.....	60
7.6 Quick analysis.....	63
7.7 Other functions.....	68
8 Analysis grammar.....	72
8.1 General aggregate functions.....	72
8.2 Security detection functions.....	74
8.3 Mapping function.....	77
8.4 Estimating functions.....	80
8.5 Mathematical statistics functions.....	82
8.6 Mathematical calculation functions.....	83
8.7 String functions.....	85
8.8 Date and time functions.....	87
8.9 URL functions.....	94
8.10 Regular expression functions.....	96
8.11 JSON functions.....	98
8.12 Type conversion functions.....	99
8.13 IP functions.....	100
8.14 GROUP BY syntax.....	104
8.15 Window functions.....	105
8.16 HAVING syntax.....	108
8.17 ORDER BY syntax.....	109

8.18 LIMIT syntax.....	109
8.19 Case when and if branch syntax.....	110
8.20 Nested subquery.....	112
8.21 Arrays.....	112
8.22 Binary string functions.....	115
8.23 Bit operation.....	117
8.24 Interval-valued comparison and periodicity-valued comparison functions.....	117
8.25 Comparison functions and operators.....	124
8.26 Lambda functions.....	126
8.27 Logical functions.....	130
8.28 Column alias.....	131
8.29 Geospatial functions.....	131
8.30 Geo functions.....	135
8.31 Join syntax.....	136
8.32 UNNEST function.....	137
8.33 Phone number functions.....	140
9 Machine learning syntax and functions.....	142
9.1 Overview.....	142
9.2 Smooth function.....	145
9.3 Multi-period estimation function.....	150
9.4 Change point detection function.....	152
9.5 Maximum value detection function.....	156
9.6 Prediction and anomaly detection functions.....	158
9.7 Sequence decomposition function.....	169
9.8 Time series clustering functions.....	170
9.9 Frequent pattern statistical function.....	176
9.10 Differential pattern statistical function.....	178
9.11 Root cause analysis function.....	180
9.12 Correlation analysis functions.....	183
10 Advanced analysis.....	187
10.1 Case study.....	187
10.2 Optimized queries.....	188
11 Log analysis through JDBC.....	192
12 Query and visualization.....	196
12.1 Analysis graph.....	196
12.1.1 Graph description.....	196
12.1.2 Table.....	197
12.1.3 Line chart.....	201
12.1.4 Column chart.....	204
12.1.5 Bar chart.....	206
12.1.6 Pie chart.....	208
12.1.7 Area chart.....	211
12.1.8 Individual value plot.....	214

12.1.9 Progress bar.....	222
12.1.10 Map.....	224
12.1.11 Flow diagram.....	227
12.1.12 Sankey diagram.....	229
12.1.13 Word cloud.....	232
12.1.14 Tree map.....	233
12.2 Dashboard.....	234
12.2.1 Dashboard overview.....	234
12.2.2 Create and delete a dashboard.....	236
12.2.3 Set the display parameters for a dashboard.....	242
12.2.4 Edit a dashboard.....	247
12.2.5 Subscribe to dashboards.....	255
12.2.6 Drill-down analysis.....	259
12.2.7 Set and use a filter in a dashboard.....	274
12.2.8 Markdown chart.....	284
12.3 Other visualization methods.....	291
12.3.1 Console sharing embedment.....	291
12.3.2 Console embedment parameters.....	294
12.3.3 Use JDBC to count and visualize logs.....	306
12.3.4 OpenTracing implementation of Jaeger.....	314
12.3.5 Interconnect with DataV big screen.....	323
12.3.6 Interconnection with Grafana.....	333
13 FAQ.....	348
13.1 What can cause an inaccurate query result to return?.....	348

1 Overview

Log Service enables you to query and analyze massive amounts of logs in real time by using the LogSearch and Analytics functions. If the index function is disabled, raw data can be used in the order that is defined by Kafka based on Shards. If the index function is enabled, data statistics and query are also supported.

Functional advantages

- **Real-time:** Logs can be analyzed immediately after they are written.
- **Fast:**
 - **Query:** Billions of data can be processed and queried within one second (with five conditions).
 - **Analysis:** Hundreds of millions of data can be aggregated and analyzed within one second (with aggregation by five dimensions and the GroupBy condition).
- **Flexible:** Query and analysis conditions can be changed as required to obtain results in real time.
- **Extensive:** Besides functions such as reports, dashboards, and quick analysis provided in the console, Log Service seamlessly interconnects with products such as Grafana, DataV, and Jaeger, and supports protocols such as RESTful API and JDBC.

Indexes

The index function is designed to sort a specific column or multiple columns in logs. By using indexes, you can quickly access the collected logs. However, before using the LogSearch and Analytics functions, you must collect logs and [enable the index function and configure indexes](#) for the logs.

Log Service provides full text indexes and field indexes.

- **Full text indexes:** In this mode, the entire log is configured with indexes. The default index is used to query all keys in the log. The log can be queried even if only one key is matched.
- **field indexes:** In this mode, indexes are configured for specific keys. This allows you to query a specific key to narrow down the query range.

The data type of fields must be specified when you use field indexes. Log Service supports [text](#), [json](#), [long](#), and [double](#). For more information, see [Index data type overview](#).

Query methods

- Query logs in the console:

You can log on to the Log Service console and specify a query time range and enter a query statement on the query and analysis page. For more information, see [Query logs](#) and [#unique_10](#).

- Query logs through API calls:

You can use the [#unique_11](#) and [#unique_12](#) APIs to query logs.



Note:

Before querying logs, you must collect logs and [enable the index function and configure indexes](#) for the logs.

Query and analysis statement format

To query and analyze logs in real time, you need to enter a query and analysis statement. The statement consists of a query statement and an analysis statement, and the two statements are separated by a vertical bar (|). The following shows an example:

```
$ Search |$ Analytics
```

Statement type	Required?	Description
Query statement	No	<p>The query condition, which can contain keywords, blur values, numbers, ranges, and combined conditions</p> <p>If the query statement is empty or "", no filter condition is set for the current data. That is, all data will be returned. For more information, see #unique_10.</p>

Statement type	Required?	Description
Analysis statement	No	<p>The analysis statement, which is used to calculate and collect query results or full data.</p> <p>If the analysis statement is empty, only query results will be returned but no statistical analysis will be performed. For more information, see #unique_13.</p>

Other information

If you query a large amount of log data (such as a long query time span, where the data volume is over 10 billion), one request cannot query all the data. In this case, Log Service returns the existing data and notifies you that the query result is incomplete.

At the same time, the server caches the results of the query within 15 minutes. When the query result is partially cached, the server continues to scan log data that has not been cached. To reduce the workload of merging multiple query results, Log Service merges the result of the cache hit with the result of the new query and returns it to you.

Therefore, Log Service enables you to get the final result by calling the interface repeatedly with the same parameters.

2 Syntax description

Log Service provides a function similar to the SQL aggregate computing. This function integrates with the [query](#) function and the SQL computing function to compute the query results.

Syntax example:

```
status > 200 | select avg ( latency ), max ( latency ) , count ( 1  
) as c GROUP BY method ORDER BY c DESC LIMIT  
20
```

Basic syntax:

```
[ search query ] | [ sql query ]
```

The SEARCH condition and computing condition are separated by a vertical bar (|). This syntax indicates that the required logs are filtered from the log by the search query, and SQL queries are computed for these logs. The search query syntax is specific to Log Service. For details, see [Query syntax](#).

Prerequisites

To use the analysis function, you must click Enable of the SQL related fields in Search and Analysis config. For more information, see [#unique_15](#).

- If you do not enable analysis function, computing function of up to 10 thousand lines of data per shard is provided, and the delay is relatively high.
- With the Enable Analytics turned on, Log Service provides the quick analysis in seconds.
- Only works for new data when function is enabled.
- No additional charges are incurred after the Enable Analytics is turned on.

Supported SQL syntax

Log Service supports the following SQL syntaxes. For details, click the specific links.

- **SELECT aggregate computing functions:**

- #unique_16
- #unique_17
- #unique_18
- #unique_19
- #unique_20
- #unique_21
- #unique_22
- #unique_23
- #unique_24
- #unique_25
- #unique_26
- #unique_27
- #unique_28
- #unique_29
- #unique_30
- #unique_31
- #unique_32
- #unique_33
- #unique_34
- #unique_35
- #unique_36
- #unique_37
- #unique_38
- #unique_39
- #unique_40
- #unique_41
- #unique_42
- #unique_43
- #unique_44
- #unique_45
- #unique_46

Syntax structure

The SQL syntax structure is as follows:

- The FROM clause and WHERE clause are not required in the SQL statement. By default, FROM indicates to query the data of the current Logstore, and the WHERE condition is search query.
- The supported clauses include SELECT, GROUP BY, ORDER BY [ASC,DESC], LIMIT, and HAVING.



Note:

By default, only the first 10 results are returned. To return more results, add limit n.

For example, `* | select count (1) as c , ip group by ip order by c desc limit 100 .`

Built-in fields

Log Service has built-in fields for statistics. These built-in fields are automatically added when you configure any valid column.

Field name	Type	Meaning
<code>__time__</code>	bigint	The log time.
<code>__source__</code>	varchar	The source IP of the log. This field is source when you query. The underscores (__) are added before and after source only in SQL.
<code>__topic__</code>	varchar	The log topic.

Limits

1. The highest concurrency of each project is 15.
2. A single column varchar has the maximum length of 2048 and is truncated if the length exceeds 2048.
3. By default, 100 lines of data are returned, and page turning is not supported. If you want more data to be returned, use [#unique_42](#).

Examples

Count the hourly PV, UV, and maximum delay corresponding to a user request, with the highest delay of 10:

```
*| select  date_trunc (' hour ', from_unixtime ( __time__ )) as
time ,
count ( 1 ) as  pv ,
approx_distinct ( userid ) as  uv ,
max_by ( url , latency ) as  top_latency_url ,
max ( latency , 10 ) as  top_10_latency
group  by  1
order  by  time
```

3 Enable and set indexes

Before using the LogSearch/Analytics function of Log Service, you need to enable and set indexes for the logs.

Context

You can query the collected logs only after you enable and set indexes for the logs. Set indexes based on the log fields and your query requirements.



Note:

- After the LogSearch/Analytics function is enabled, data is indexed on the backend server. Therefore, index traffic is incurred and index storage space is required.
- Index settings take effect only on the data recorded after the settings are enabled or modified.
- At least one of the following indexes must be enabled for a log: full text index and key/value index.
- To use SQL statements to [analyze](#) the query result of a field, enable the Analytics function of the field.
- If you want to set an index for a [Tags](#) field, such as an Internet IP address or a Unix timestamp, set the Key Name to a value in the `__tag__ : key` format, for example, `__tag__ : __receive_ time__`. A Tags field does not support indexes of the numeric type. Instead, set the Type of all Tags fields to text. For example, to query a field with the key name `__tag__ : __receive_ time__`, you can use a fuzzy value, such as `__tag__ : __receive_ time__ : 1537928 *`, or the full value of the field, such as `__tag__ : __receive_ time__ : 1537928404` as the keyword.

When a log is collected, information about the log, such as the source and time, is automatically added to the log as key/value pairs. These fields are reserved in Log Service. When you enable and set indexes for logs, the indexes and the Analytics function are automatically enabled for these fields.



Note:

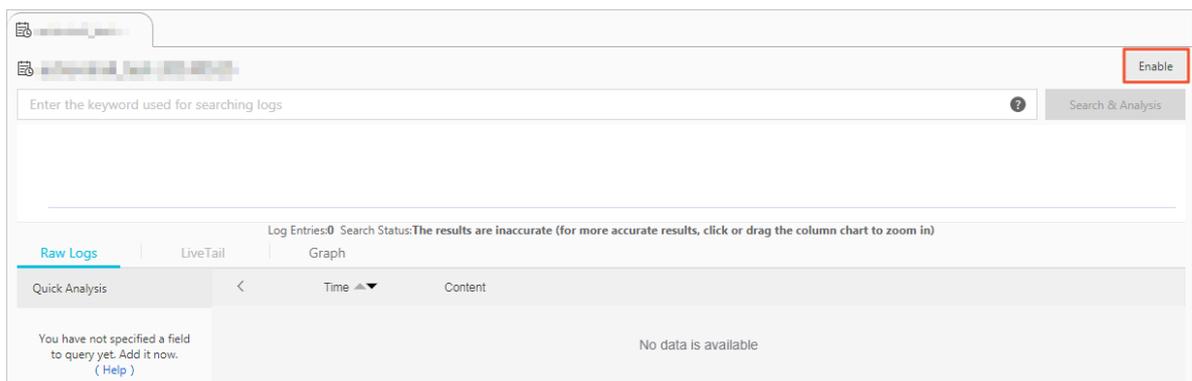
The delimiters of the `__topic__` and `__source__` fields are null. It means that the keywords used to query the two fields must match the field values.

Table 3-1: Reserved fields in Log Service

Name	Description
<code>__topic__</code>	Indicates the log topic. If you set a topic for a log, Log Service automatically adds a topic field to the log. The key of the field is <code>__topic__</code> , and the value of the field is the log topic.
<code>__source__</code>	Indicates the source equipment that generates the log.
<code>__time__</code>	Indicates the time that is specified when the log is recorded by the SDK.

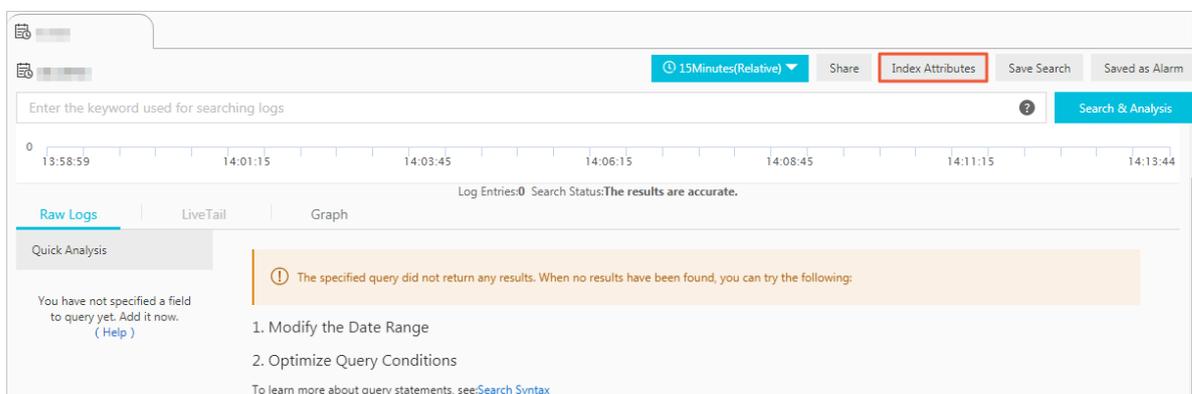
Procedure

1. Log on to the [Log Service console](#), and then click the target project name.
2. In the LogSearch column, click Search.
3. Click Enable in the upper right corner.



Note:

If you have created an index, click **Index Attributes > Modify**.



4. Set indexes for logs.

Log Service supports two indexes: full text index and key/value index. At least one of the two indexes must be set for a log.



Note:

If both a full text index and a key/value index are set for a log, the key/value index prevails.

Index type	Description
Full text index	Indicates that all fields in the log are queried as text with a key/value index. The key and value of the index are text and can both be queried. You do not need to specify the key name in queries.
Key/Value index	<p>After setting a key/value index for a field, you must specify the key name to query the field. If a full text index is set for a log and a key/value index is set for a field in the log, the full text index does not take effect on the field.</p> <p>You can set multiple data types for a field, including:</p> <ul style="list-style-type: none"> · Text · JSON · Numeric (Long and Double)

a) Set a full text index for a log.

You can set an index for the full content of a log. The values of all keys in the log are queried by default when you query the log.

Parameter	Description	Example
Full Text Index	If this option is enabled, an index is enabled for the full content of the log. The values of all keys in the log are queried by default. The log can be queried if any one of the keys matches the keyword.	-

Parameter	Description	Example
Case Sensitive	<p>Specifies whether the queries are case-sensitive.</p> <ul style="list-style-type: none"> · If this option is disabled, the queries are not case-sensitive, that is, an internal error log can be queried by both of the keywords "INTERNALERROR" and "internalerror". · If this option is enabled, the queries are case-sensitive, that is, a log that includes "internalError" can be queried only by the keyword "internalError". 	-
Chinese character	<p>Sets whether to distinguish between English and Chinese.</p> <ul style="list-style-type: none"> · After opening, if the log contains Chinese, the Chinese word segmentation is carried out according to the Chinese grammar , word Segmentation is carried out in English according to the word segmentation characters. · When closed, word all the content according to the word segmentation . 	-
Delimiter	<p>Specifies single-byte characters used to separate a log into multiple keywords. For example, if the content of a log is <code>a , b ; c ; D - F</code> , you can specify the comma (,) , semi-colon (;) , and hyphen (-) as delimiters to separate the log into five keywords: " a" , " b" , " c" , " D" , and " F" .</p>	<pre>, '";=() [] {}? @&<>/:\ n \ t</pre>

b) Set key/value indexes for a log.

You can set indexes for specified keys. After setting key/value indexes for a log, you can query specified keys to narrow down the query scope.



Note:

- Log Service automatically creates indexes for the **reserved fields** and enables the Analytics function of the fields. The reserved fields include `__topic__` , `__source__` , and `__time__` .
- The settings in the Customize tab page are described as an example in this topic. The Nginx Template and MNS Template are used only to collect Nginx logs and MNS logs and do not support customized index settings.
- If you want to set an index for a **Tags** field, such as an Internet IP address or a Unix timestamp, set the Key Name to a value in the `__tag__ : key` format, for example, `__tag__ : __receive_ time__` . A Tags field does not support indexes of the numeric type. Set the Type of all Tags fields to text. For example, to query a field with the key name `__tag__ : __receive_ time__` , you can use a fuzzy value, such as `__tag__ : __receive_ time__ : 1537928 *` , or the full value of the field, such as `__tag__ : __receive_ time__ : 1537928404` as the keyword.

Parameter	Description	Example
Key Name	Specifies the name of a field in the log.	<code>_address_</code>
Type	<p>Specifies the data type of a field in the log, including:</p> <ul style="list-style-type: none"> • text: Indicates that the content of the field is text. • long: Indicates that the content of the field is an integer. This field must be queried by a value range. • double: Indicates that the content of the field is a floating-point number. This field must be queried by a value range. • json: Indicates that the content of the field is in JSON format. <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin-top: 10px;">  Note: Numeric types (Long and Double) do not support Case Sensitive or Delimiter. </div>	-

Parameter	Description	Example
Alias	<p>Indicates the alias of a column.</p> <p>An alias is used only for SQL statistics. A field is still identified by its original name in the underlying storage. Therefore, you must use the original name of a field to query the field. For more information, see #unique_45.</p>	address
Case Sensitive	<p>Specifies whether the queries are case-sensitive. This parameter has two values:</p> <ul style="list-style-type: none"> · false: The queries are not case-sensitive, that is, the sample log can be queried by both of the keywords "INTERNALERROR" and "internalerror". · true: The queries are case-sensitive, that is, the sample log can be queried only by the keyword "internalError". 	-
Delimiter	<p>Specifies single-byte characters used to separate a log into multiple keywords.</p> <p>For example, if the content of a log is <code>a , b ; c ; D - F</code>, you can specify the comma (,), semi-colon (;), and hyphen (-) as delimiters to separate the log into five keywords: " a" , " b" , " c" , " D" , and " F" .</p>	<pre>, '";=() []{}? @&<>/:\ n \ t</pre>

Parameter	Description	Example
Enable Analytics	<p>Specifies whether the Analytics function is enabled. This function is enabled by default.</p> <p>After enabling the Analytics function , you can use query and analysis statements to analyze the query results.</p>	-

Search & Analysis ✕

Modifications (such as changing the delimiter, enabling statistics, and enabling case-sensitivity) only take effect for new data

* Logstore Name

* Full Text Index

Case Sensitive

Delimiter:

* Field Search

[Customize](#) Nginx Template MNS Template

Key Name	Enable Search				Enable Analytics	Delete
	Type	Alias	Case Sensitive	Delimiter:		
bytes_combination	text	bytes_combination	<input type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
bytes_received	long	bytes_received	<input checked="" type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
bytes_sent	long	bytes_sent	<input checked="" type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
child_process	long	child_process	<input checked="" type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
child_process_format	long	child_process_format	<input checked="" type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
client_addr	text	client_addr	<input type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
connect_addr	text	connect_addr	<input type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕
cookie_session	text	cookie_session	<input type="checkbox"/>	,\";=000?@&<>:\n\t	<input checked="" type="checkbox"/>	✕

5. Click OK.



Note:

- The index settings take effect within one minute.
- Index settings take effect only on data recorded after the settings are enabled or modified.

4 Query logs

After enabling the index function and setting indexes, you can query and analyze the collected logs in the console.

Prerequisites

- Logs have been collected.
- You have [enabled the index function and set indexes](#).

Procedure

1. Log on to the [Log Service console](#), and then click the target project name.
2. In the LogSearch column, click Search.
3. In the search box, enter a query analysis statement.

A query analysis statement consists of a query statement and an analysis statement in the format of `query statement | analysis statement`. For more information, see [#unique_15](#).

4. In the upper-right corner, click 15 Minutes (Relative) to set the time range for queries.

You can choose between a relative time period and a time frame or customize a time range.



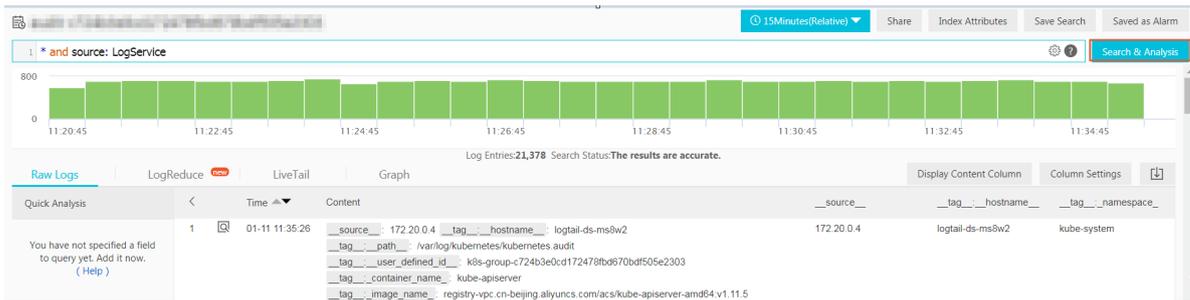
Note:

The query results may contain logs obtained one minute earlier or later than the specified time range.

The screenshot shows a 'Time' selection dialog with the following options:

- Relative**
 - 1Minute
 - 5Minutes
 - 15Minutes** (selected)
 - 1Hour
 - 4Hours
 - 1Day
 - Today
 - 1Week
 - 30Days
 - This Month
 - Custom
- Time Frame**
 - 1Minute
 - 15Minutes
 - 1Hour
 - 4Hours
 - 1Day
 - 1Week
 - 30Days
 - Today
 - Yesterday
 - The Day before Yesterday
 - This Week
 - Previous Week
 - This Month
 - This Quarter
 - This Year
 - Custom
- Custom**

5. Click Search & Analysis to view the query results.



You can view the query results through a log distribution histogram, raw logs, or various graphs.



Note:

By default, 100 query results are returned. If you need to view more results, see [#unique_42](#).

- **Log distribution histogram:**

The log distribution histogram shows the log distribution in the time dimension.

- Rest the pointer on a green data block to view the time range indicated by the data block and the number of log query results within the time range.
- Click a data block to view finer-grained log distribution. Additionally, you can view the log query results on the Raw logs tab page.



- **Raw logs:**

On the Raw logs tab page, you can view the logs that match your search conditions.

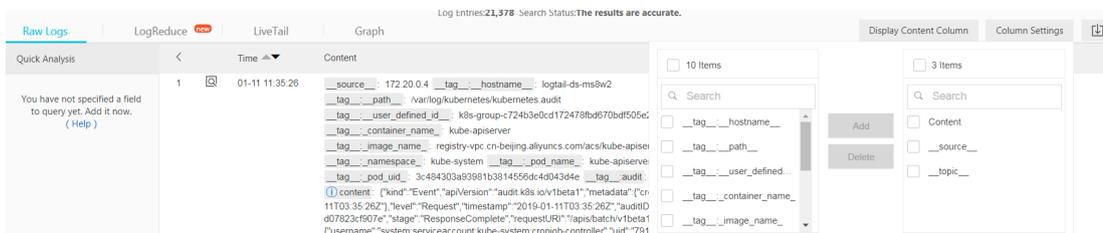
- Use the quick analysis function to receive a quick analysis of the distribution of a field over a period of time. For more information, see [#unique_52](#).
- Click the download icon in the upper-right corner to specify a download range, and then click OK.
- In the upper-right corner, click Column Settings. In the displayed dialog box, select the target fields from the left area and click Add to add the fields to the right area. Then, columns indicated by the added fields appear on the tab

page. The field names are also column names, and the columns list the field values.



Note:

To view the log content on the tab page, you must select Content.



- Set the style of the content column. If the field contains more than 3,000 characters, then some content will be hidden and a message indicating this will be displayed before the field Key. Specifically, click Display Content Column. In the displayed dialog box, set Key-Value Pair Arrangement and Truncate Character String.



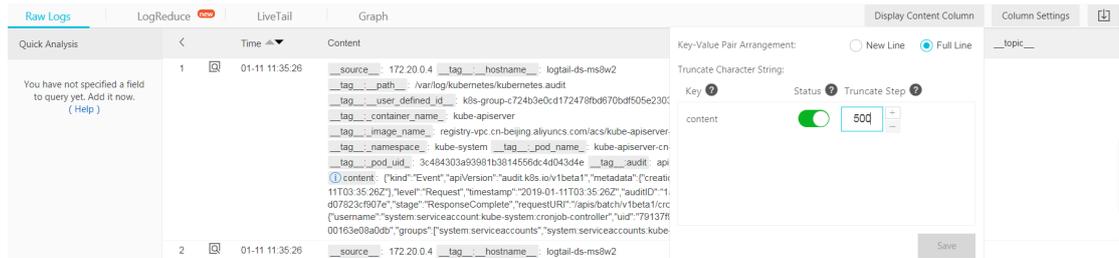
Note:

If the content limit is set to 10,000 characters, any character past this number will be downgraded. Further, none of these characters will be displayed, and no delimiter will be specified for these characters.

Parameter		Description
Key-Value Pair Arrangement		You can set this parameter to New Line or Full Line as needed.
Truncate Character String	Key	When a field value contains more than 3,000 characters, the value is truncated by default. However, this parameter remains empty if this value is not reached. The value of this parameter is the key of the truncated value.

Parameter		Description
	Status	<p>You can determine whether to enable value truncation. It is enabled by default.</p> <ul style="list-style-type: none">■ Enable: When a value length exceeds the preset value of Truncate Step, the value is automatically truncated. You can click the button at the end of a value to perform incremental expansion. The number of incremental characters is the value of Truncate Step.■ Disable: A value will not be truncated even if its length exceeds the preset value of Truncate Step.

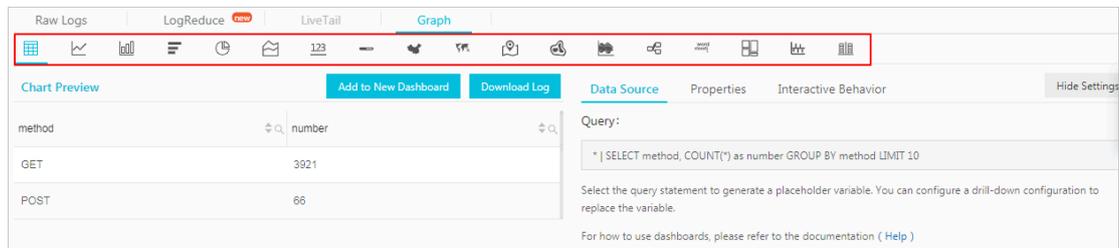
Parameter		Description
	Truncate Step	<p>This parameter indicates the maximum number of a value as well as the number of incremental characters per time.</p> <p>The parameter value ranges from 500 to 10,000 characters. The default value is 3,000.</p>



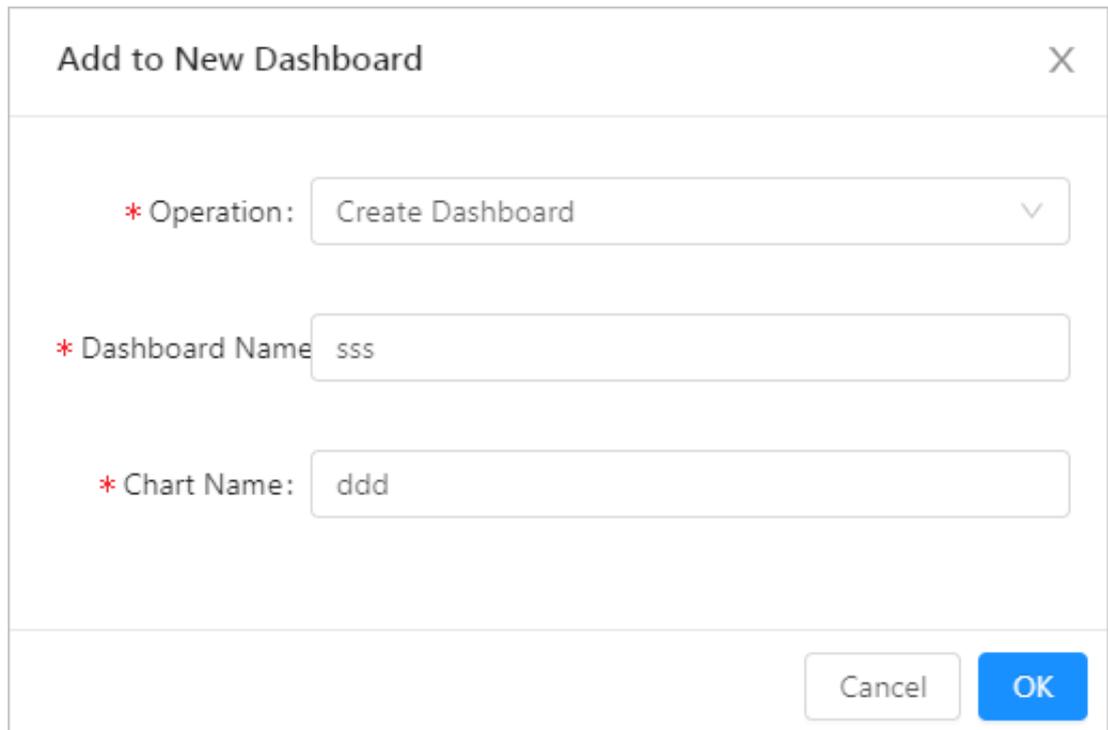
• **Graph:**

If you have enabled the statistics function and used a query analysis statement for query, you can view the analysis results on the Graph tab page.

- Select an appropriate **graph** type to show analysis results based on your requirements. Several chart types are provided in Log Serve including tables, line charts, and bar charts.



- Add the graph to the **dashboard** for real-time data analysis results to be displayed. Click Add to dashboard to save common query statements as a graph saved on the dashboard.

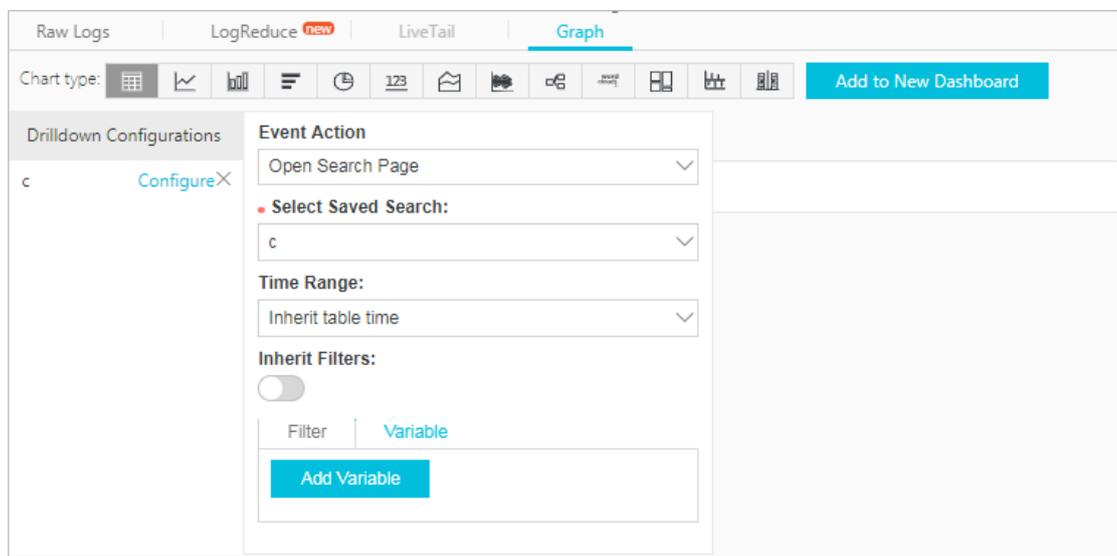


The screenshot shows a dialog box titled "Add to New Dashboard" with a close button (X) in the top right corner. It contains three required fields, each marked with a red asterisk:

- * Operation:** A dropdown menu with "Create Dashboard" selected.
- * Dashboard Name:** A text input field containing "sss".
- * Chart Name:** A text input field containing "ddd".

At the bottom right, there are two buttons: "Cancel" and "OK".

- Set drill-down configurations to gain deeper insight into the analysis results. Then, click the values in the graph to view the analysis results from more dimensions. For more information, see [#unique_55](#).



The screenshot shows the Log Service interface with the "Graph" tab selected. A "Drilldown Configurations" dialog box is open, showing the following settings:

- Event Action:** Open Search Page
- Select Saved Search:** c
- Time Range:** Inherit table time
- Inherit Filters:**
- Filter:** Variable
- Add Variable:** Button

The background interface shows a toolbar with various chart types and a "Add to New Dashboard" button.

Additionally, you can click Save Search and Save as Alarm in the upper-right corner. Then, you can use the [saved search](#) and [alarm](#) functions.

5 Download logs

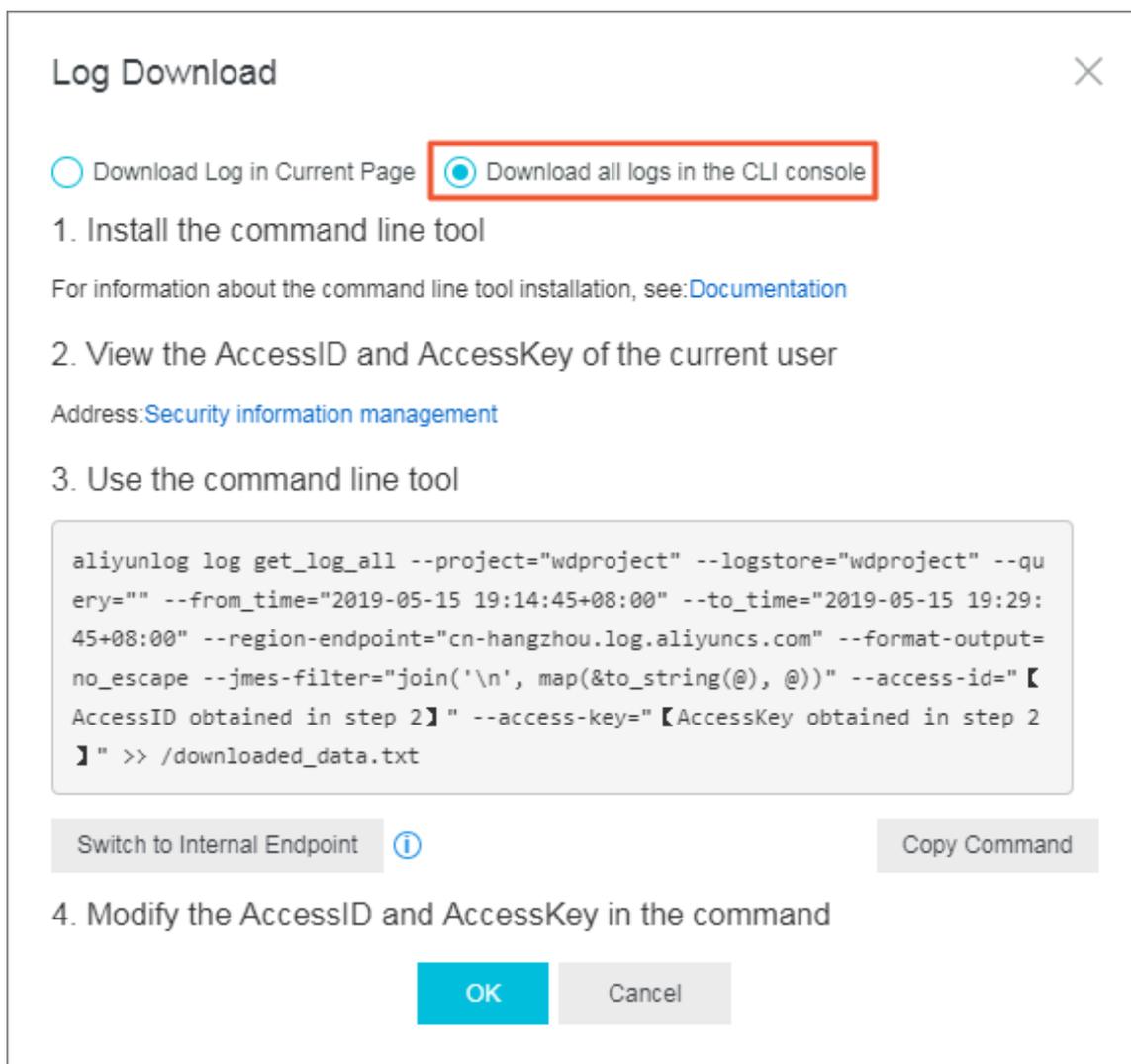
This topic describes how to download a specific page of logs in CSV format, or all logs in TXT format, to a local host.

Procedure

1. Log on to the [Log Service console](#), and then click the target project name.
2. In the LogSearch column, click Search.
3. On the Raw Logs tab page, click the icon .

4. Select the method to download logs.

- Click **Download Log in Current Page** to download logs of the current page in the CSV format to your local host.
- Click **Download all logs in the CLI console** to download all logs by using the CLI tool.



- a. Install the command line tool. For more information, see [Log Service CLI User Guide](#).
- b. Select [Security information management](#) to copy your AccessID and AccessKey.
- c. Select **Copy Command**, and use the AccessID and AccessKey copied in the preceding step to replace `【 AccessID obtained in step 2 】` and `【 AccessKey obtained in step 2 】` in the command.
- d. Run the command in the CLI tool to download logs.

**Note:**

The logs are downloaded to the `download_data.txt` file. This file is located in the directory where the command was run.

6 Data type of index

6.1 Overview

Log Service allows you to set indexes for the full text or some fields of the collected logs. If you set an index for the full text of a log, the value used to query this log is the content of the entire log. If you set indexes for some fields of a log, you can set the data type of each key used in queries.

Data type

The following table describes the supported index types.

Query type	Index type	Description	Example
Basic	text	Indicates the text type that supports keywords and fuzzy matching in queries.	<code>uri : " login *"</code> <code>method : " post "</code>
	long	Indicates the numeric type that supports interval queries.	<code>status > 200 , status</code> <code>in [200 , 500]</code>
	double	Indicates the numeric type that supports floating-point numbers.	<code>price > 28 . 95 , t</code> <code>in [20 . 0 , 37]</code>
Combination	json	Indicates that the index is a JSON field that supports nested queries. The field type is Text by default. You can set an index of the Text , Long, or Double type for element 'b' under element 'a' by using a path format such as 'a.b'. The field type is determined by the index type you set.	<code>level0 . key > 29 .</code> <code>95 level0 . key2 :</code> <code>action "</code>
	Full text	Indicates that the full content of the log is queried as text.	<code>error and " login</code> <code>fail "</code>

Example

The following log includes time and other four keys.

No.	Key	Type
0	time	-
1	class	text
2	status	long
3	latency	double
4	message	json

```

0 . time : 2018 - 01 - 01  12 : 00 : 00
1 . class : central - log
2 . status : 200
3 . latency : 68 . 75
4 . message :
  {
    " methodName " : " getProject  Info ",
    " success " : true ,
    " remoteAddr  ess " : " 1 . 1 . 1 . 1 : 11111 ",
    " usedTime " : 48 ,
    " param " : {
      " projectNam  e " : " ali - log - test - project ",
      " requestId " : " d3f0c96a - 51b0 - 4166 - a850 -
f4175dde73  23 "
    },
    " result " : {
      " message " : " successful ",
      " code " : " 200 ",
      " data " : {
        " clusterReg  ion " : " ap - southeast - 1 ",
        " ProjectNam  e " : " ali - log - test - project ",
        " CreateTime " : " 2017 - 06 - 08  20 : 22 : 41 "
      },
      " success " : true
    }
  }

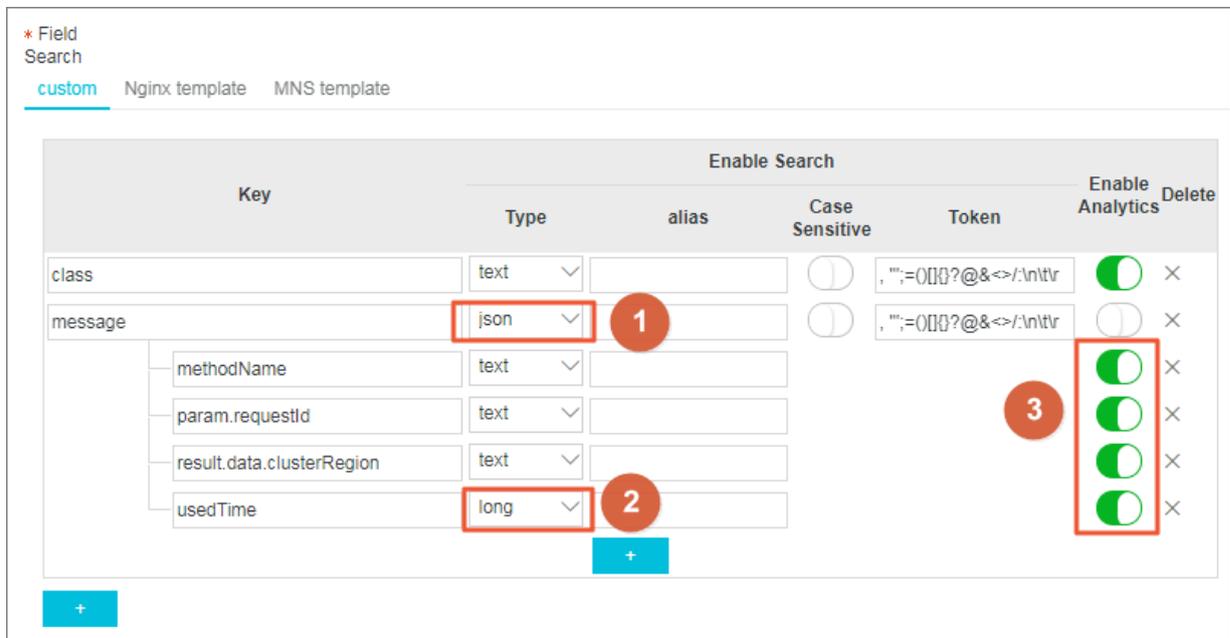
```

```
}

```

You can set indexes for a log as follows:

Figure 6-1: Index setting



In the preceding figure:

- Mark ① indicates that the index type for this field is json and all data of the string type and bool type in the field can be queried.
- Mark ② indicates that the index type for this field is long and data of the long type in the field can be queried.
- Mark ③ indicates that the fields can be analyzed by using SQL statements.

Example:

1. Query data of the string type and bool type.

- You do not need to configure the fields in the json field.
- JSON maps and arrays are automatically expanded. You can query fields that are multi-level nested with each level separated by a period (.).

```
class : cental *
message . traceInfo . requestId : 92 . 137_151813 9699935_55
99
message . param . projectNam e : ali - log - test - project
```

```
message . success : true
```

2. Query data of the Double type and Long type.

The fields in a JSON field must be configured separately and must not be contained in an array.

```
latency > 40
message . usedTime > 40
```

3. Query data with combined data types.

```
class : cental * and message . usedTime > 40 not message
. param . projectName : ali - log - test - project
```

6.2 Text type

Similar to search engines, text data is queried based on terms. Therefore, you must configure word segmentation, case sensitivity, including full text query options.

Instructions

- Case sensitivity

Determine whether to support case sensitivity when querying raw logs. For example, the raw log is `internalError`.

- After turning off the Case Sensitive switch, the sample log can be queried based on the keyword `INTERNALERROR` or `internalerror`.
- After turning on the Case Sensitive switch, the sample log can only be queried based on the keyword `internalError`.

- Token

You can separate the contents of a raw log into several keywords by using a token.

For example, the raw log is

```
/ url / pic / abc . gif
```

- If no token is set, the string is considered as an individual word `/ url / pic / abc . gif`. You can only query this log by using the complete string or fuzzy match such as `/ url / pic /*`.
- If `/` is set as the token, the raw log is separated into three words: `url`, `pic`, and `abc . gif`. You can query this log by using any of the three words or fuzzy match, for example, `url`, `abc . gif`, or `pi *`. You can also use `/`

`url / pic / abc . gif` to query this log (`url` and `pic` and `abc . gif` is separated into the following three conditions during the query: `url` , `pic` , and `abc.gif`).

- If `/.` is set as the token, the raw log is separated into four words: `url` , `pic` , `abc` , and `gif` .



Note:

You can broaden the query range by setting appropriate tokens.

• Full text index

By default, full text query (index) considers all the fields and keys of a log, except the time field, as text data, and does not need to specify keys. For example, the following log is composed of four fields (time/status/level/message):

```
[ 20180102 12 : 00 : 00 ] 200 , error , some thing is error
in this field
```

- `time:2018-01-02 12:00:00`
- `level:" error"`
- `status:200`
- `message:" some thing is error in this field"`

After enabling full text index, the following text data is assembled in the "key:value + space" mode.

```
status : 200 level : error message : " some thing is error
in this field "
```



Note:

- Prefix is not required for full text query. Enter `error` as the keyword, both `level` field and `message` field meet the query condition.
- You must set a token for the full text query. If a space is set as the token, `status:200` is considered as a phrase. If `:` is set as the token, `status` and `200` are considered as two independent phrases.
- Numbers are processed as texts. For example, you can use the keyword `200` to query this log. The time field is not processed as a text.
- You can query this log if you enter a key such as `" status"` .

6.3 JSON type

Log Service supports the query and analysis of JSON-formatted logs. You can set the data type of indexes to JSON.

JSON-formatted data is a combination of multiple data types, including text, Boolean, numeric, array, and map. JSON-formatted data, as a common type of data, is self-parsed and flexible. It can be used to record data in complex scenarios. In many logs, the content without a fixed format is recorded in JSON format. For example, the request parameters and response of an HTTP request are recorded in a log in JSON format.

Log Service allows you to set the data type of index fields to JSON so that you can query and analyze logs in JSON format.

Configuration

- Log Service can parse JSON-formatted fields and automatically generate indexes for all the fields of the text and Boolean types.

```
json_strin g . key_map . key_text : test_value
json_strin g . key_map . key_bool  : true
```

- To query the fields of the double or long type that is not in a JSON array, you can specify the JSON path.

```
Set the data type of the key_map . key_long field to long .
Query : json_strin g . key_map . key_long > 50
```

- To query the fields of the text, double, or long type that is not in a JSON array, you can enable the statistical analysis feature and use SQL statements to analyze these fields.

```
json_strin g . key_map . key_long > 10 | select count (*)
as c ,
" json_strin g . key_map . key_text " group by
" json_strin g . key_map . key_text "
```



Note:

- JSON object and JSON array types are not supported.
- Fields cannot be contained in a JSON array.
- Fields of the Boolean type can be converted into the text type.

- JSON-formatted fields must be enclosed in double quotation marks (" ") during log query and analysis.

- Log Service can parse JSON-formatted data that contains both valid and invalid content.

Log Service attempts to parse all the valid content until the invalid content appears

For example, the data after the `key_3` field is truncated and lost in the following code. Log Service can correctly parse the `json_string.key_map.key_2` field and the content before this field.

```
" json_string ":
{
  " key_1 " : " value_1 ",
  " key_map " :
  {
    " key_2 " : " value_2 ",
    " key_3 " : " valu
```

Query syntax

To query a specific key, you must add the JSON parent path as the prefix of the key in the query statement. The query syntax for the fields of the text and numeric types is the same for both JSON-formatted data and other data. For more information, see [#unique_10](#).

Query example

The following log contains the time field and four keys, among which the message field is in JSON format.

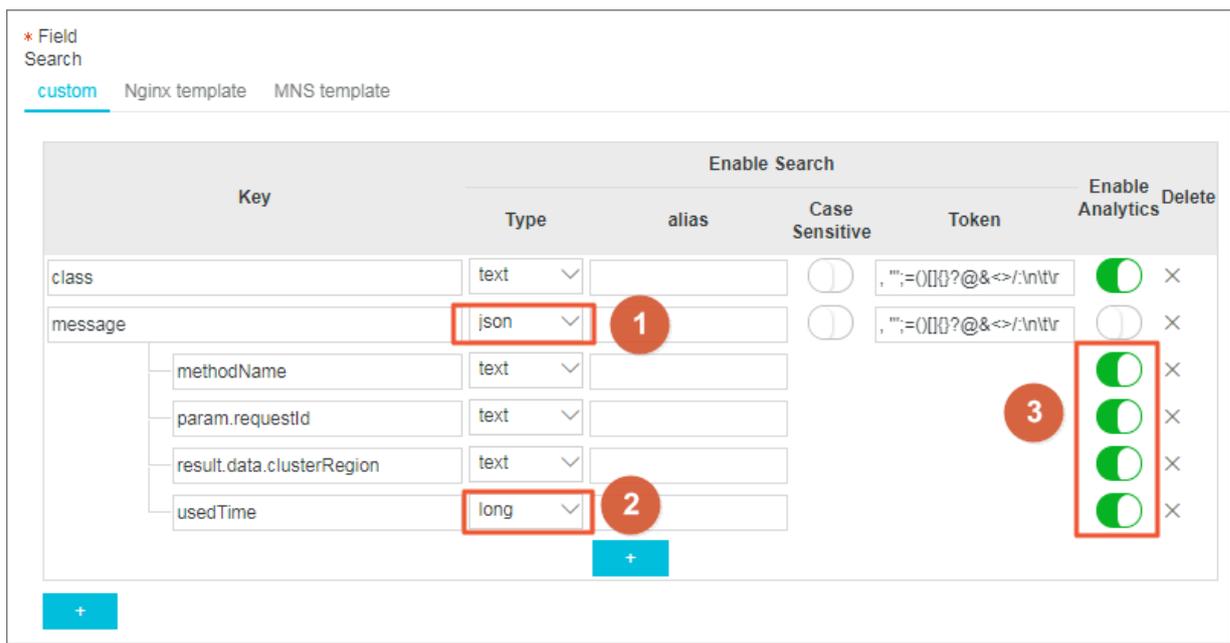
No.	Key	Type
0	time	N/A
1	class	Text
2	status	Long
3	latency	Double
4	message	JSON

```
0 . time : 2018 - 01 - 01 12 : 00 : 00
1 . class : central - log
2 . status : 200
3 . latency : 68 . 75
4 . message :
{
```

```
" methodName ": " getProject Info ",
" success ": true ,
" remoteAddress ": " 1 . 1 . 1 . 1 : 11111 ",
" usedTime ": 48 ,
" param ": {
  " projectName ": " ali - log - test - project ",
  " requestId ": " d3f0c96a - 51b0 - 4166 - a850 -
f4175dde73 23 "
},
" result ": {
  " message ": " successful ",
  " code ": " 200 ",
  " data ": {
    " clusterRegion ": " ap - southeast - 1 ",
    " projectName ": " ali - log - test - project ",
    " createTime ": " 2017 - 06 - 08 20 : 22 : 41 "
  },
  " success ": true
}
}
```

The following figure shows how to set indexes for a log.

Figure 6-2: Index settings



where:

- (1) indicates that you can query all the data of the string and Boolean types in JSON-formatted fields.
- (2) indicates that you can query the data of the long type.
- (3) indicates that you can use SQL statements to analyze the configured fields.

Examples:

1. Query the fields of the string and Boolean types

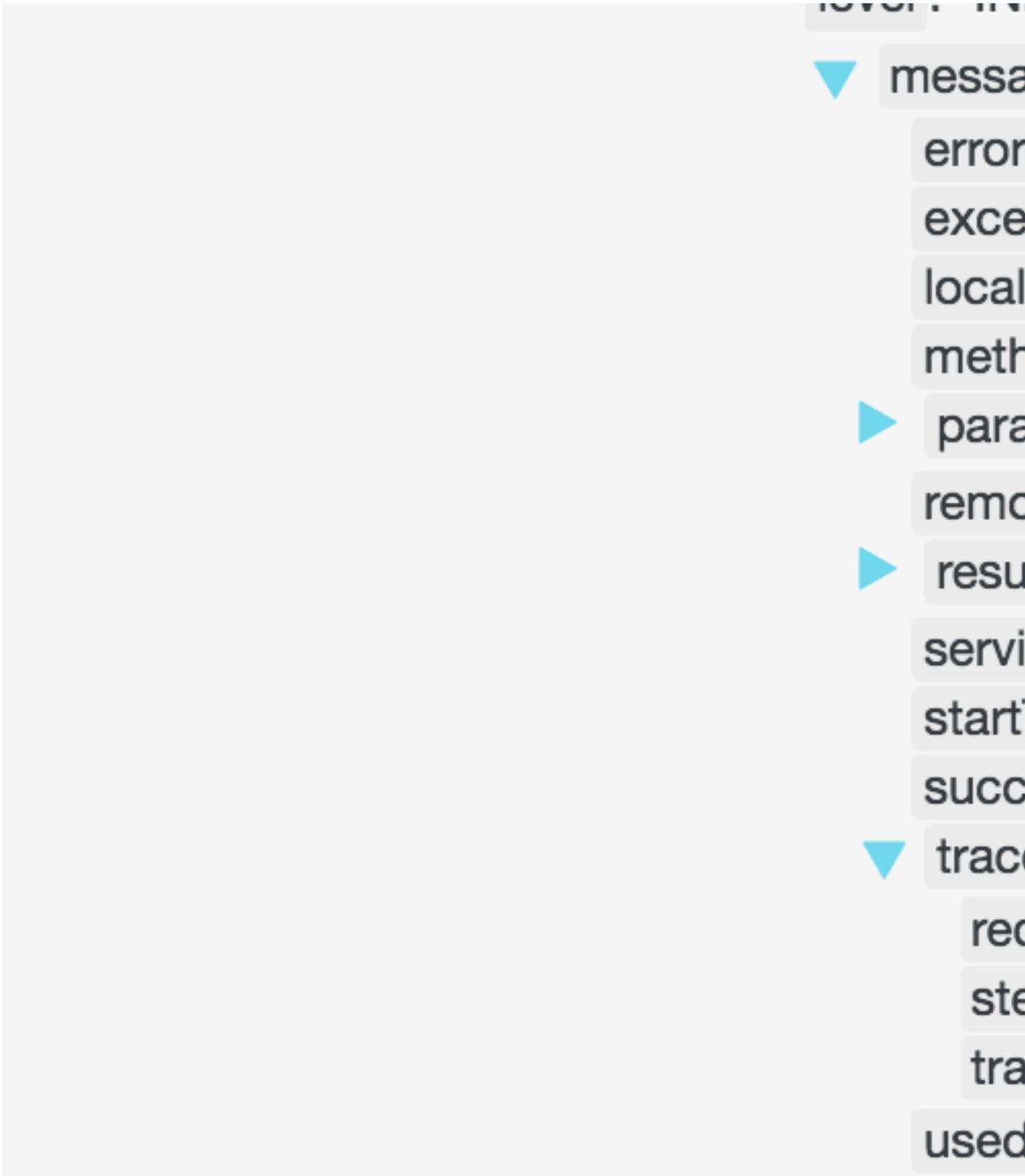


Note:

- You do not need to configure JSON-formatted fields.
- JSON maps and JSON arrays are automatically expanded. You can query fields that are multi-level nested with each level separated by a period (.).

```
message . traceInfo . requestId : 92 . 137_151813 9699935_55
99
message . param . projectName : ali - log - test - project
message . success : true
```

```
message . result . data . ProjectSta tus : Normal
```



2. Query the fields of the double and long types



Note:

You need to configure each JSON-formatted field separately. Fields cannot be contained in a JSON array.

```
message . usedTime > 40
```

3. Use SQL statements to analyze fields



Note:

- You need to configure each JSON-formatted field separately. Fields cannot be contained in a JSON array.
- Fields to be queried must be enclosed in double quotation marks (") or be configured with an alias.

```
* | select avg (" message . usedTime ") as avg_time ,  
" message . methodName " group by " message . methodName "
```

6.4 Value type

When configuring indexes, you can configure a field as the value type and query the key by using a value range.

Instructions

Supported types: `long` (long integer) and `double` (decimal). After configuring a field as the value type, you can only query the key by using a value range.

Example

To query the longkey whose key range is (1000 2000], use the following methods.

- Use values to query the longkey:

```
longKey > 1000 and longKey <= 2000
```

- Use an interval to query the longkey:

```
longKey in ( 1000 2000 ]
```

For more syntaxes, see [#unique_10](#).

7 Query

7.1 Query syntax

To help you query logs more effectively, Log Service provides a set of query syntax to express query conditions.

Query methods

After [enabling the index function and configuring indexes](#), you can enter a query and analysis statement on the log query page to query logs.

The query statement is the first part of a query analysis statement, and is used to specify rules for filtering logs and logs that conform to the query condition. Both full text query and key/value query are supported.

- Full text query

In a full text query, the entire log is regarded as a special key/value pair, in which the log content is regarded as the value. You can specify keywords for a full text query. Specifically, you can specify the keywords which must be included in or excluded from the query condition. The log that meets the specified query condition will be returned as a query result.

Log Service also supports phrase query and fuzzy query.

- Common full text query: You need to specify a keyword and rule. Logs that contain the keyword and conform to the rule will be returned as query results.

For example, `a and b` indicates that the query results must contain both the keywords `a` and `b`.

- Phrase query: If the target phrase contains a space, you can enclose the phrase with double quotation marks (""). In this case, the phrase will be regarded as a complete keyword for log query.

For example, `" http error "` indicates that the query results must contain `http error`.

- Fuzzy query: You can specify a partial word up to 64 characters in length, and add a fuzzy query keyword (* or ?) at the middle or end of the word. By doing so,

up to 100 words that meet the query condition among all logs will be queried, and the logs corresponding to the 100 words will be returned as query results.

For example, `addr ?` indicates that Log Service needs to query up to 100 words starting with `addr`, and return the corresponding logs.

- **Key/value query**

After configuring indexes for fields, you can query the name or content of a specific field. For fields of the double or long type, you can also specify the value range for query. For example, the key/value query statement `Latency > 5000 and Method : Get * and not Status : 200` indicates that the query results must meet the following conditions:

- The value of `Latency` must be greater than 5000.
- The `Method` field must start with `Get`.
- The value of the `Status` field is not 200.

You can perform various types of basic query and combined query according to the data types set for field indexes. For more information about key/value query examples, see [Index data type overview](#).

Precautions

- When both full text query and key/value query are performed, if the delimiters set for the two query methods are different, the delimiter set for key/value query is used, and the query results of full text query become invalid.
- You can query fields with a specified value range only after setting the data type of the fields to double or long. If the field data type is unspecified or the syntax for querying value ranges is incorrect, Log Service determines that the query condition is for full text query. This may return unexpected query results.
- If the data type of a field is changed from text to numeric, the data collected before the change only support `=` query.

Operators

The following operators can be used in query statements:

Operator	Description
and	Binary operator. Format: <code>query1 and query2</code> . Indicates the intersection of the query results of <code>query1</code> and <code>query2</code> . With no syntax keyword among multiple words, the relation is <code>and</code> by default.
or	Binary operator. Format: <code>query1 or query2</code> . Indicates the union of the query results of <code>query1</code> and <code>query2</code> .
not	Binary operator. Format: <code>query1 not query2</code> . Indicates a result that matches <code>query1</code> and does not match <code>query2</code> , which is equivalent to <code>query1 - query2</code> . If only <code>not query1</code> exists, it indicates to select the results excluding <code>query1</code> from all the logs.
(,)	Parentheses () are used to merge one or more sub-queries into one query condition to increase the priority of the query in the parentheses ().
:	Used to query the key-value pairs. <code>term1 : term2</code> makes up a key-value pair. If the key or value contains reserved characters such as spaces and colons (:), use quotation marks (") to enclose the entire key or value.
"	Converts a keyword to a common query character. Each term enclosed in quotation marks (") can be queried and is not be considered as a syntax keyword. Or all the terms enclosed in quotation marks (") are regarded as a whole in the key-value query.
\	Escape character. Used to escape quotation marks. The escaped quotation marks indicate the symbols themselves, and they cannot be used as escape characters, such as <code>"\"</code> .
	The pipeline operator indicates more calculations based on the previous calculation, such as <code>query1 timeslice 1h count</code> .
timeslice	The time-slice operator indicates how long the data is calculated as a whole. <code>Timeslice 1h, 1m, 1s</code> indicates 1 hour, 1 minute, and 1 second respectively. For example, <code>query1 timeslice 1h count</code> represents the query query condition, and returns to the total number of hours divided by 1 hour.
count	The count operator indicates the number of log lines.

Operator	Description
*	<p>Fuzzy query keyword. Used to replace zero or multiple characters. For example, the query results of <code>que * start with que</code> .</p> <p> Note: At most 100 query results can be returned.</p>
?	Fuzzy query keyword. Used to replace one character. For example, the query results of <code>qu ? ry</code> start with <code>qu</code> , end with <code>ry</code> , and have a character in the middle.
<code>__topic__</code>	Topic data query. You can query the data of zero or multiple topics in the query. For example, <code>__topic__ : mytopicname</code> .
<code>__tag__</code>	Query a tag value in a tag key. For example, <code>__tag__ : tagkey : tagvalue</code> .
Source	Query the data of an IP. For example, <code>source : 127 . 0 . 0 . 1</code> .
>	Query the logs with a field value greater than a specific number. For example, <code>latency > 100</code> .
>=	Query the logs with a field value greater than or equal to a specific number. For example, <code>latency >= 100</code> .
<	Query the logs with a field value less than a specific number. For example, <code>latency < 100</code> .
<=	Query the logs with a field value less than or equal to a specific number. For example, <code>latency <= 100</code> .
=	Query the logs with a field value equal to a specific number. For example, <code>latency = 100</code> .
in	<p>Query the logs with a field staying within a specific range. Braces ([]) are used to indicate closed intervals and parentheses (()) are used to indicate open intervals. Enclose two numbers in braces ([]) or parentheses (()) and separate the numbers with several spaces. For example, <code>latency in [100 200]</code> or <code>latency in (100 200)</code> .</p>



Note:

- Operators are case-insensitive.
- Priorities of operators are sorted in descending order as follows: `:>">()>` and `>` `not >` or `.`
- Log Service reserves the right to use the following operators: `sort` , `asc` , `desc` , `group by` , `avg` , `sum` , `min` , `max` , and `limit` .To use these keywords, enclose them in quotation marks ("").

Query examples

Query demand	Example
Logs that contain a and b at the same time	<code>a and b or a b</code>
Logs that contain a or b	<code>a or b</code>
Logs that contain a but do not contain b	<code>a not b</code>
All the logs that do not contain a	<code>not a</code>
Logs that contain a and b, but do not contain c	<code>a and b not c</code>
Logs that contain a or b and must contain c	<code>(a or b) and c</code>
Logs that contain a or b, but do not contain c	<code>(a or b) not c</code>
Logs that contain a and b and may contain c	<code>a and b or c</code>
Logs whose FILE field contains apsara	<code>FILE : apsara</code>
Logs whose FILE field contains apsara and shennong	<code>FILE : " apsara shennong " , FILE : apsara FILE : shennong , or FILE : apsara and FILE : shennong</code>
Logs that contain and	<code>and</code>
Logs with the FILE field containing apsara or shennong	<code>FILE : apsara or FILE : shennong</code>
Logs with the file info field containing apsara	<code>" file info " : apsara</code>
Logs that contain quotation marks ("")	<code>\"</code>
All logs starting with shen	<code>shen *</code>

Query demand	Example
All logs starting with shen in the FILE field	<code>FILE : shen *</code>
All logs with the FILE field of shen*	<code>FILE : " shen *"</code>
Logs starting with shen, ending with ong, and having a character in the middle	<code>shen ? ong</code>
Logs starting with shen and aps	<code>shen * and aps *</code>
Logs starting with shen every 20 minutes	<code>shen * timeslice 20m count</code>
All data in the topic1 and topic2	<code>__topic__ : topic1 or __topic__ : topic2</code>
All data of the tagvalue2 in the tagkey1	<code>__tag__ : tagkey1 : tagvalue2</code>
All data with a latency greater than or equal to 100 and less than 200	<code>latency >= 100 and latency < 200 or latency in [100 200)</code>
All requests with a latency greater than 100	<code>latency > 100</code>
Logs that do not contain spider and do not contain opx in http_referer	<code>not spider not bot not http_refer er : opx</code>
Logs with the empty cdnIP field	<code>not cdnIP :""</code>
Logs without cdnIP field	<code>not cdnIP :*</code>
Logs with the cdnIP field	<code>cdnIP :*</code>

Specified or cross-topic query

Each LogStore can be divided into one or more subspaces by the topic. During the `therfhfrg` query, specifying topics can limit the query range so as to increase the speed. Therefore, we recommend that you use topic to divide the LogStore if you have a secondary classification requirement for the LogStore.

With one or more topics specified, the query is only performed in the topics that meet the conditions. However, if no topic is specified, data of all the topics is queried by default.

For example, use topic to classify logs with the different domain names:

Figure 7-1: Log topic

time	ip	method	url	host	topic
1481270421	127.0.0.1	POST	/users?u=1	a.aliyun.com	siteA
1481270422	127.0.0.1	POST	/users?u=1	a.aliyun.com	siteA
1481270423	127.0.0.1	POST	/users?u=1	b.aliyun.com	siteB
1481270424	127.0.0.1	POST	/users?u=1	b.aliyun.com	siteB
1481270425	127.0.0.1	POST	/users?u=1	c.aliyun.com	siteC
1481270426	127.0.0.1	POST	/users?u=1	c.aliyun.com	siteC
1481270427	127.0.0.1	POST	/users?u=1	d.aliyun.com	siteD
1481270428	127.0.0.1	POST	/users?u=1	d.aliyun.com	siteD
1481270429	127.0.0.1	POST	/users?u=1	e.aliyun.com	siteE
1481270430	127.0.0.1	POST	/users?u=1	e.aliyun.com	siteE

Topic query syntax:

- Data of all the topics can be queried. If no topic is specified in the query syntax and parameter, data of all the topics is queried.
- Supports query by topic. The query syntax is `__topic__ : topicName` . The old mode (specify the topic in the URL parameter) is still supported.
- Multiple topics can be queried. For example, `__topic__ : topic1 or __topic__ : topic2` indicates the union query of data from Topic1 and Topic2 .

Fuzzy search

Log Service support fuzzy search. Specify a word within 64 characters, and add fuzzy search operators such as `*` and `?` in the middle or in the end of the word. 100 eligible words will be searched out, in the meantime, all the logs eligible and contain the 100 words will be returned.

Limits:

- Prefix must be specified when query logs, that is, the word can not begin with `*` and `?` .
- Precise the specified word, you will get a more accurate result.
- Fuzzy search cannot be used to search for words that exceeds 64 characters. It is recommended that you specified a word under 64 characters.

7.2 LiveTail

LiveTail is an interactive function provided by Log Service in the console to help you monitor logs in real time and extract key log information.

Scenarios

In scenarios of online Operation & Maintenance (O&M), it is often necessary to monitor inbound data of the log queue in real time, and to extract key information from the latest log data to quickly find the cause of the exception. By using the traditional O&M method, you need to run the `tail -f` command on log files on the server to monitor the log files in real time. If the log information you require is not apparent enough, you can add `grep` or `grep -v` to the command to filter keywords. Log Service provides LiveTail in the console, an interactive function that monitors and analyzes online log data in real time, making O&M easier.

Benefits

- Monitors real-time log information, and marks and filters keywords.
- Distinguishes collected logs by using indexes through the collection configuration.
- Perform word segmentation for log fields to query the context logs that contain segmented words.
- Tracks the log file for real-time monitoring according to a single log entry without the need to connect to the server.

Limits

- LiveTail is only applicable to the logs collected by Logtail.
- LiveTail is available only when logs are collected.

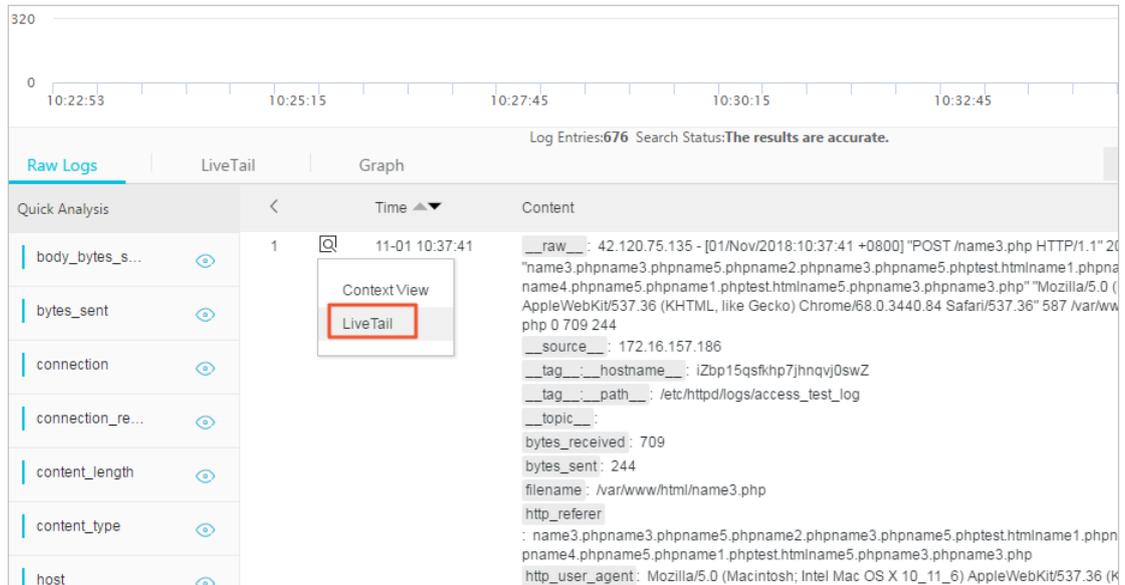
Use LiveTail to monitor logs in real time

1. Click Search in the LogSearch column.

2. You can use LiveTail in one of the following two ways:

- Quickly start LiveTail.

- a. On the Raw Logs tab, click the  icon on the right of the sequence number of the raw log, and select LiveTail.



The screenshot shows the Log Service interface with the 'Raw Logs' tab selected. A search bar at the top indicates 'Log Entries: 676' and 'Search Status: The results are accurate.' Below the search bar, there are three tabs: 'Raw Logs', 'LiveTail', and 'Graph'. The 'Raw Logs' tab is active, showing a list of log entries. The first entry is selected, and a context menu is open over it. The context menu has two options: 'Context View' and 'LiveTail'. The 'LiveTail' option is highlighted with a red box. The log entry content is displayed on the right side of the interface, showing details such as the source IP address, hostname, path, and user agent.

- b. The system automatically starts LiveTail and starts timing.

Source Type, Machine Name, and File Name are pre-configured to specify the raw logs.

After LiveTail is started, the log data collected by Logtail is displayed in order on the page. The latest log data is always displayed at the bottom of the page. The scrollbar is at the lowest position on the page by default so that you can immediately see the latest data. The page displays up to 1000 log entries

- When 1000 log entries are displayed, the page automatically refreshes to display the latest log entry at the bottom of the page.

The screenshot displays the Log Service interface. At the top, there is a search bar with the text "Enter keywords to filter log entries containing the keywords". Below the search bar is a bar chart showing the number of log entries over time, with a peak of 32 entries around 10:39:47. The interface includes a "Source Type" dropdown set to "Common ...", a "Machine Name" field containing "iZbp15qsfkhp7jhn", and a "File Name" field containing "/etc/httpd/logs/acc". There are also "Stop LiveTail" and "Enable" buttons. The log entries are displayed in a list format, with the following details for entry 50:

```

50 http_referer
   : name3.phpname3.phpname5.phpname2.phpname3.phpname5.phpptest.htmlname1.phpname4.phpname1.phpname6.phpname6.phpname4.phpname
   e3.php
   http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.84 Safari/537.36 keep_
   remote_port: 80 remote_user: - request_method: POST request_protocol: HTTP/1.1 request_query: request_time_msec: 462 request_time_sec: 0
   response_handler: application/x-httpd-php response_size_bytes: 10 status: 200 time_local: [01/Nov/2018:10:40:08 +0800]

   __raw__: 42.120.75.135 - [01/Nov/2018:10:40:08 +0800] "POST /name2.php HTTP/1.1" 200 5
   "name3.phpname3.phpname5.phpname2.phpname3.phpname5.phpptest.htmlname1.phpname4.phpname1.phpname6.phpname6.phpname4.phpname:
   e3.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.84 Safari/537.36" 492 /var/www/
   244
   __source__: 172.16.157.186 __topic__: bytes_received: 709 bytes_sent: 244 filename: /var/www/html/name2.php
51 http_referer
   : name3.phpname3.phpname5.phpname2.phpname3.phpname5.phpptest.htmlname1.phpname4.phpname1.phpname6.phpname6.phpname4.phpname
   e3.php
   http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.84 Safari/537.36 keep_
   remote_port: 80 remote_user: - request_method: POST request_protocol: HTTP/1.1 request_query: request_time_msec: 492 request_time_sec: 0
   response_handler: application/x-httpd-php response_size_bytes: 5 status: 200 time_local: [01/Nov/2018:10:40:08 +0800]

```

- c. (Optional) Enter keywords in the search box.

Only log entries that contain the keywords can be displayed in the monitoring list. By filtering logs that contain the keywords, you can monitor the content of the logs in real time.

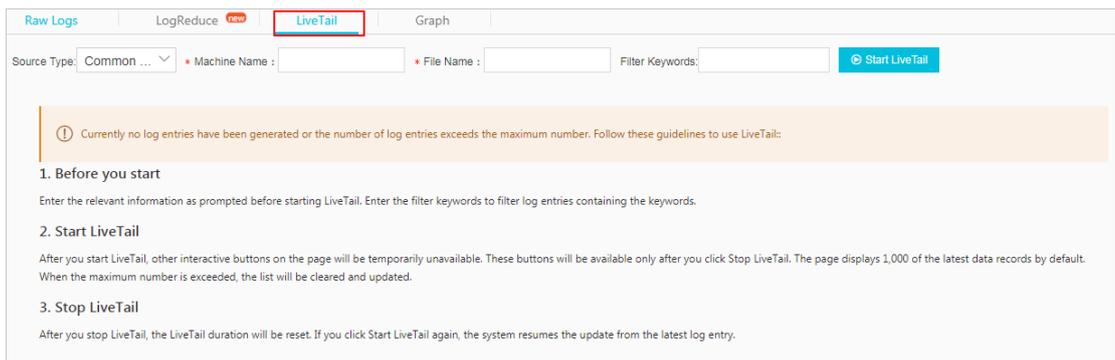
- d. To analyze logs in which exceptions may exist during the real-time log monitoring process, click Stop LiveTail.

After you stop LiveTail, the LiveTail timing and the real-time log data update also stop.

For exceptions found in the process of log monitoring, Log Service provides multiple analysis methods. For more information, see [Use LiveTail to analyze logs](#).

- Customize LiveTail settings.

- a. Click the LiveTail tab.



b. Configure LiveTail.

Configuration	Required	Description
Source type	Yes	Log source, including: - Common log - Kubernetes - Docker
Machine name	Yes	Name of the log source server.
File name	Yes	Full path and file name of the log file.
Filter keywords	No	Keywords. After you configure a keyword, only the logs that contain the keyword can be displayed in the real-time monitoring window.

c. Click Start LiveTail.

After LiveTail is started, log data collected by Logtail are displayed in orders on the page. The latest log data is always displayed at the bottom of the page . The scrollbar resides at the lowest position of the page by default so that you can see the latest data. The page displays up to 1000 log entries. When 1000 log entries are displayed, the page automatically refreshes to display the latest log entry at the bottom of the page.

d. To analyze logs in which exceptions may exist during the real-time log monitoring process, click Stop LiveTail.

After you stop LiveTail, the LiveTail timing and the real-time log data update stop as well.

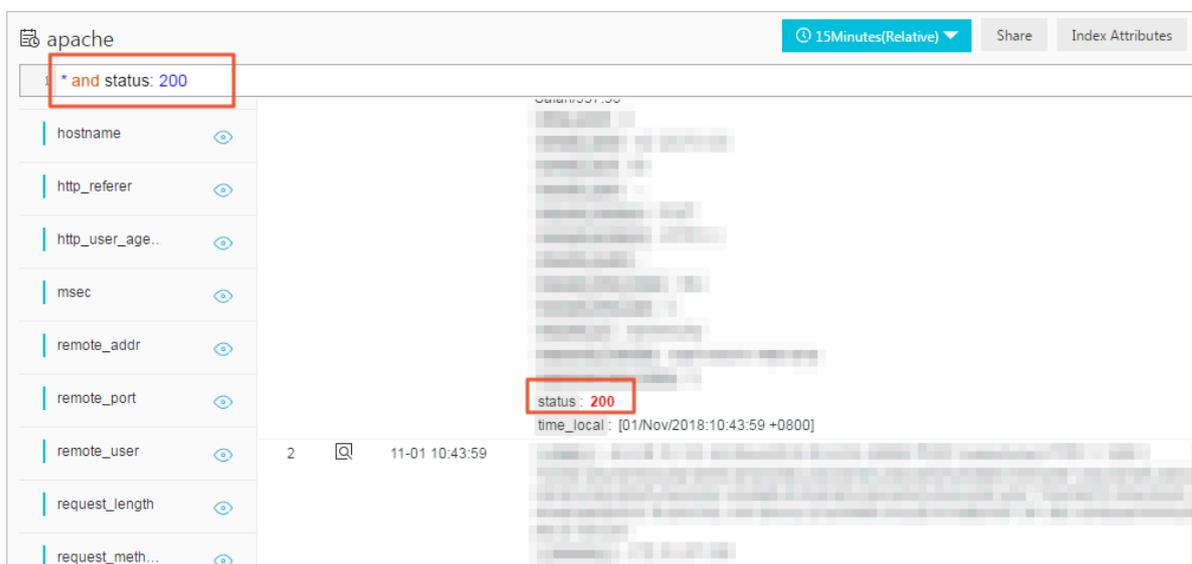
For exceptions found in the process of log monitoring, Log Service provides multiple analysis methods. For more information, see [Use LiveTail to analyze logs](#).

Use LiveTail to analyze logs

After you stop LiveTail, the real-time monitoring window stops updating logs, and you can analyze and troubleshoot the exceptions found in the monitoring process.

- View the logs that contain the specified field.

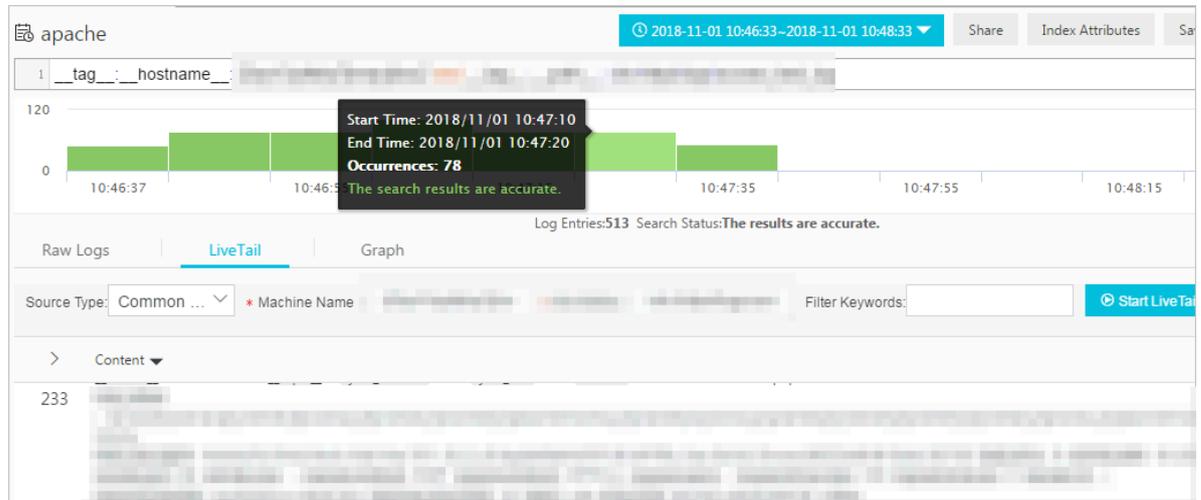
Word segmentation has been conducted to all fields. When you click the exception field content, that is, a keyword, the page automatically jumps to the Raw Logs tab, and the system filters all logs to show the logs that contain the keyword. In addition, you can also analyze the logs that contain the keyword by using context view, statistical charts, and other analysis methods.



- Narrow the time range of a query according to the log distribution histogram.

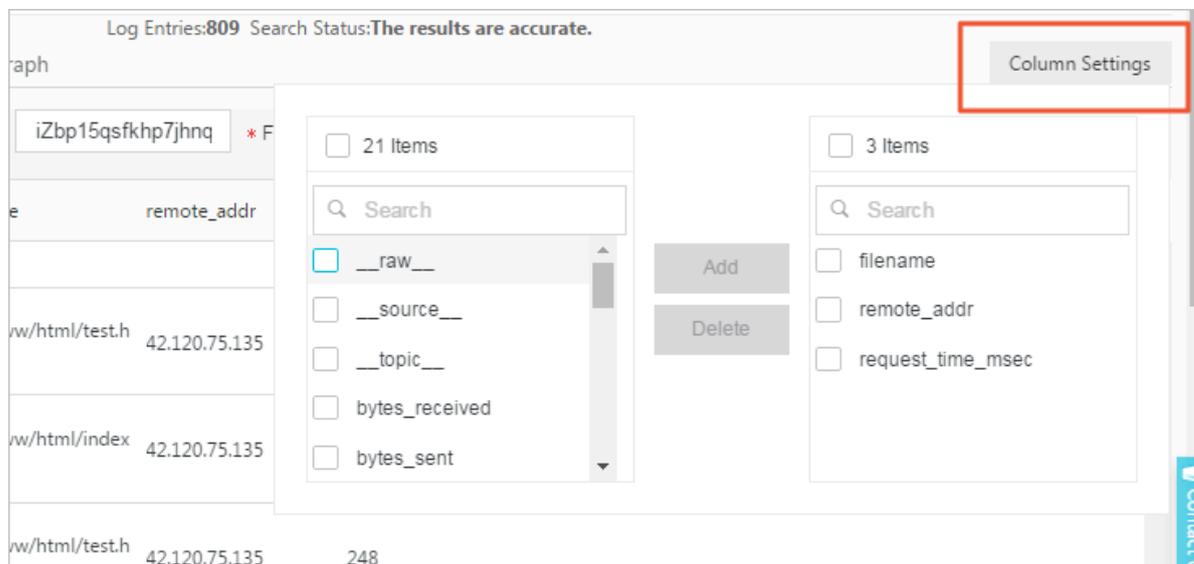
When LiveTail is started, the log distribution histogram is also updated synchronously. If you find an exception of log distribution for a time period, for example a significant increase in the number of logs, you can click the green rectangle of the time period to narrow the time range of the query. The timeline of the raw logs redirected from the LiveTail page is associated with the timeline clicked in LiveTail

. You can view all the raw logs and the detailed log distribution over time during this time period.



- Highlight key information with column settings.

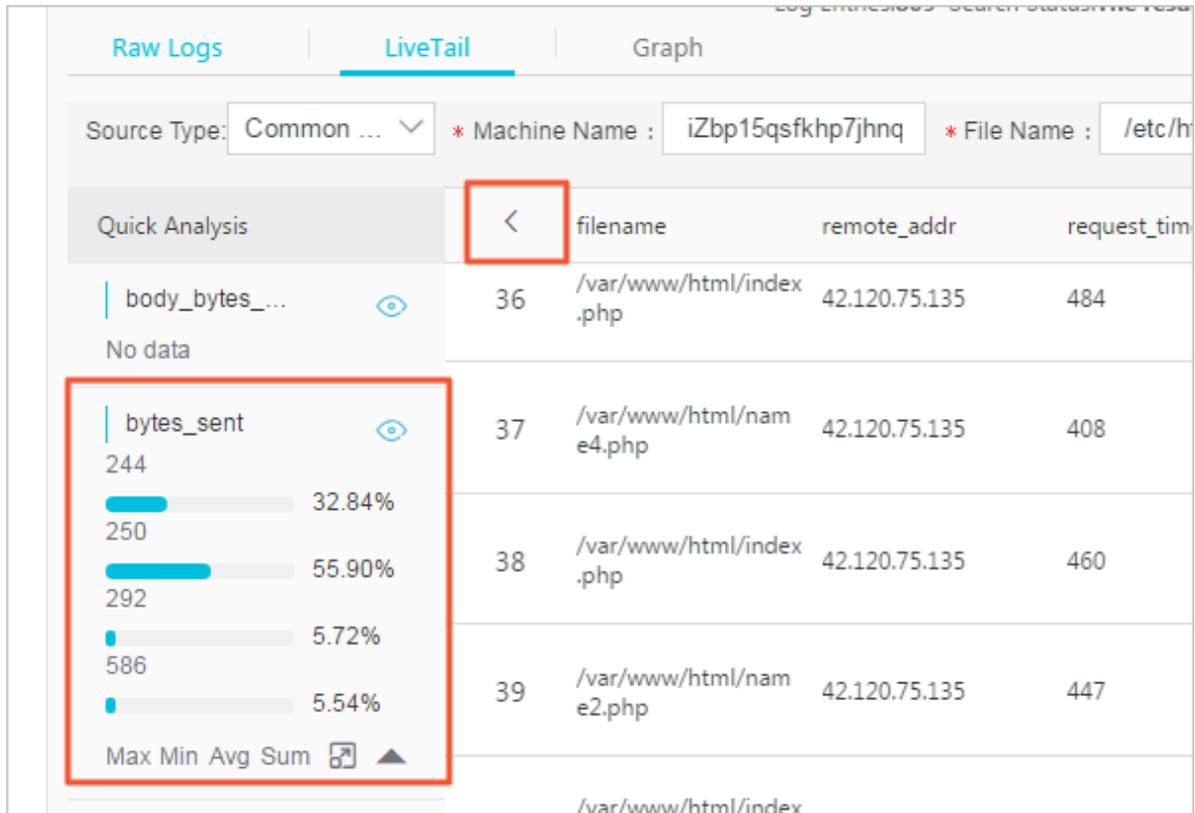
On the LiveTail tab, click Column Settings in the upper-right corner of the log list, you can set a specified field as a separate column to make the data in this column more obvious. You can configure the data that requires high attention as one column to make it easier to view and recognize exceptions.



- Quickly analyze log data.

On the LiveTail tab, by clicking the arrow in the upper-left corner of the log list, you can expand the quick analysis area. The time interval of the quick analysis is the period from when LiveTail starts to when it stops. The quick analysis provided

in LiveTail is the same as that provided in the raw logs. For more information, see [#unique_52](#).



7.3 Use LogReduce to group log data

This topic describes how to use LogReduce to group and analyze collected log entries that are extremely similar to detect frequently occurring log patterns such as conditions that trigger alarms.

Scenario

With LogReduce, you can locate problems, detect exceptions, and perform other O&M-related actions for DevOps, or detect network intrusions that may have compromised the security of your services. In addition, you can save the log grouping result as an analysis chart to a dashboard, and then view the grouped data in real time.

Benefits

- Log entries in three formats (Log4J, JSON, or Syslog) can be grouped by using the LogReduce function.
- Gigabytes of data can be grouped in seconds.

- You can view the log entries that are grouped for each log pattern, and you can display the number of grouped log entries during different time ranges.
- You can dynamically adjust the tolerance of log grouping.

Index size



Note:

After you enable the LogReduce function, the size of log indexes increases by 10%. For example, if the size of raw log data is 100 GB/day, the size of the log indexes increases by 10 GB after you enable the function.

Raw log size	Proportion of indexes in the raw log	Size of indexes generated by LogReduce	Index size
100 GB	20% (20 GB)	100 * 10%	30 GB
100 GB	40% (40 GB)	100 * 10%	50 GB
100 GB	100% (100 GB)	100 * 10%	110 GB

Enable LogReduce



Note:

By default, LogReduce is disabled.

1. Log on to the [Log Service console](#), and then click the target project name.
2. On the Logstores page, click Search on the right of the target Logstore.

3. If you have enabled the index function, choose **Index Attributes > Modify**. If you have not enabled the index function, click **Enable**.

Figure 7-2: Enable the index function

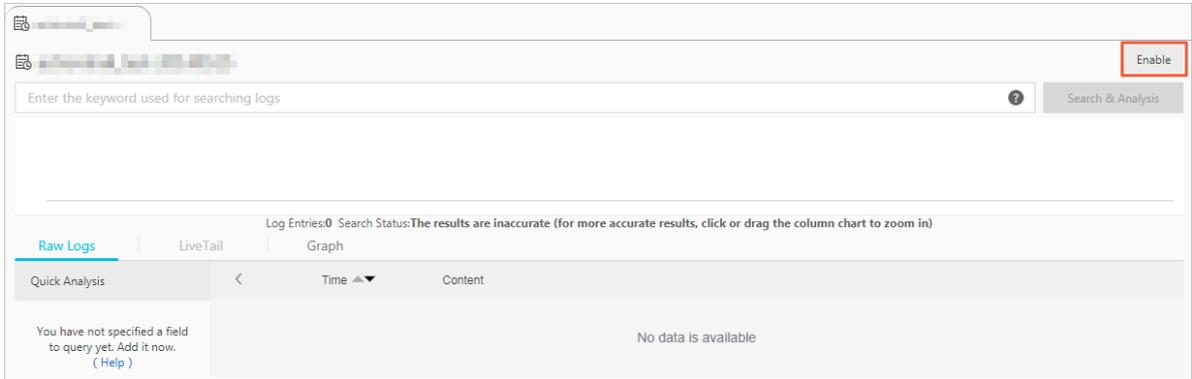
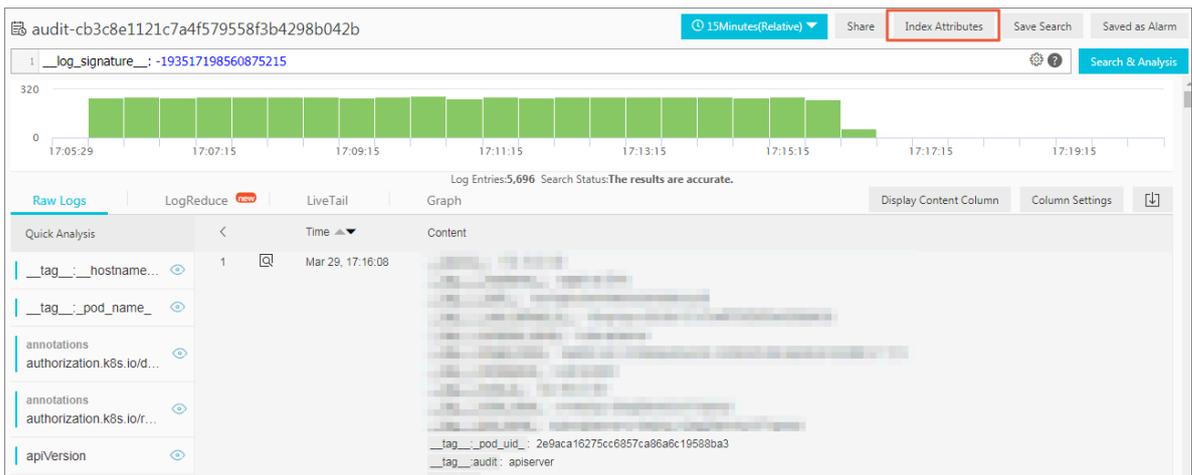
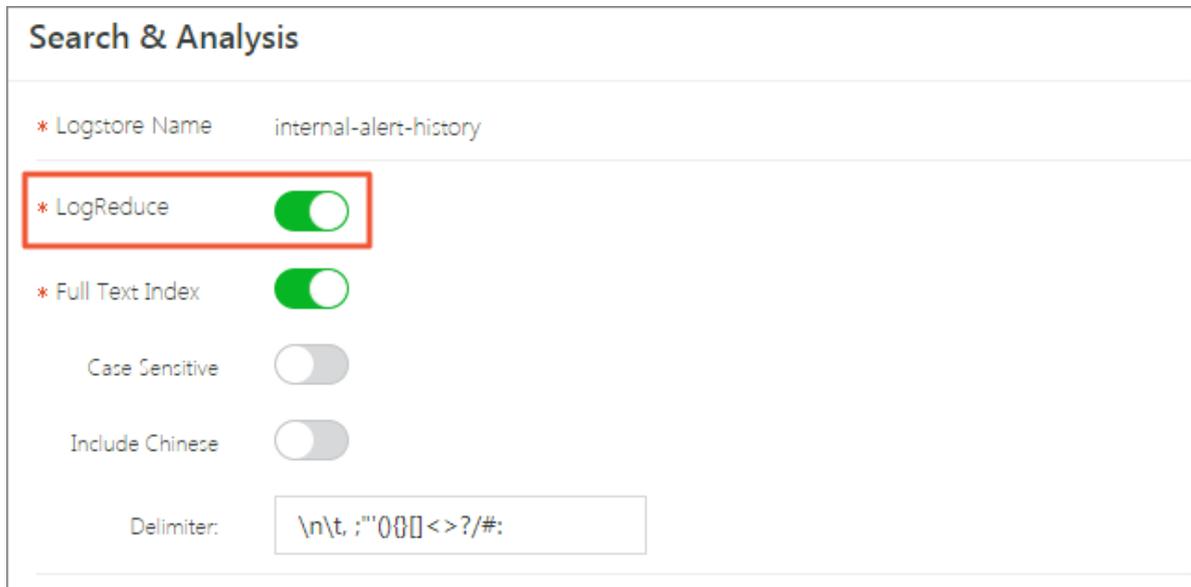


Figure 7-3: Modify the log index



4. Set index parameters, and click the switch to enable LogReduce.

Figure 7-4: Enable LogReduce



5. Click OK.



Note:

After you enable LogReduce, Log Service automatically groups collected log data.

Then, you can perform the following operations:

- [View the log grouping result and the raw log.](#)
- [Adjust the log grouping precision.](#)
- [Show the number of grouped log entries in different periods.](#)

View the log grouping result and the raw log

1. On the Search & Analysis page, enter a search and analysis statement in the search box, and click Search & Analysis.



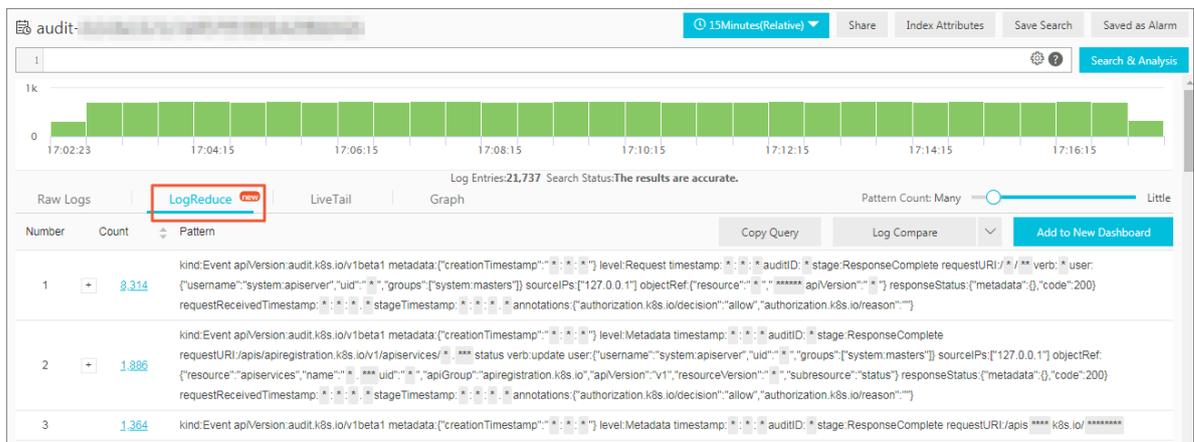
Note:

- You can also use key words to filter the grouped log entries.
- The SQL type of statements is not supported by the LogReduce function. This means that analysis results of log data cannot be grouped by this function.

2. Click the LogReduce tab to view the result.

Item	Description
Number	Indicates the sequence number of a log group.
Count	Indicates the number of log entries of a log group.
Pattern	Indicates the log patterns. Each log group has one or more than one sub-patterns.

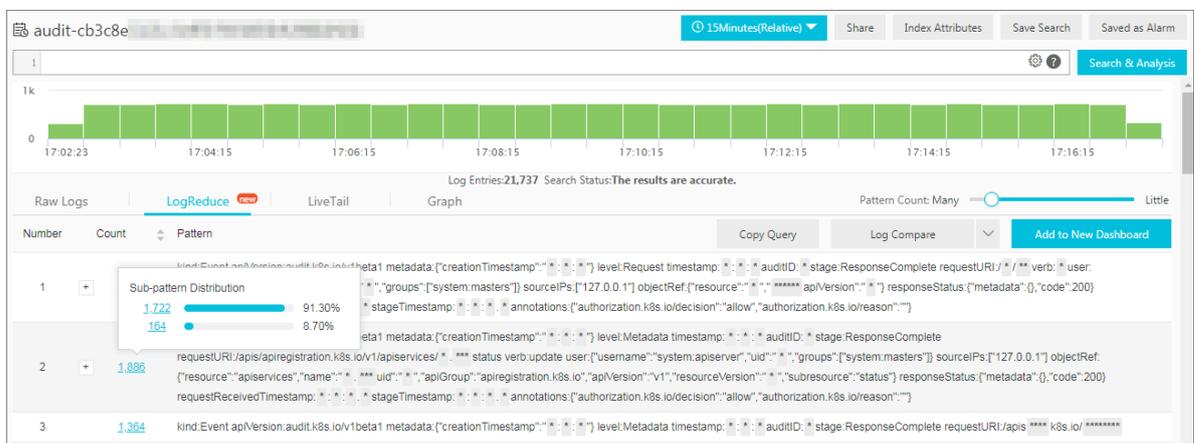
Figure 7-5: Result



3. Move your pointer over a Count value to show the sub-patterns of this log group and the proportion of each sub-pattern in the log group.

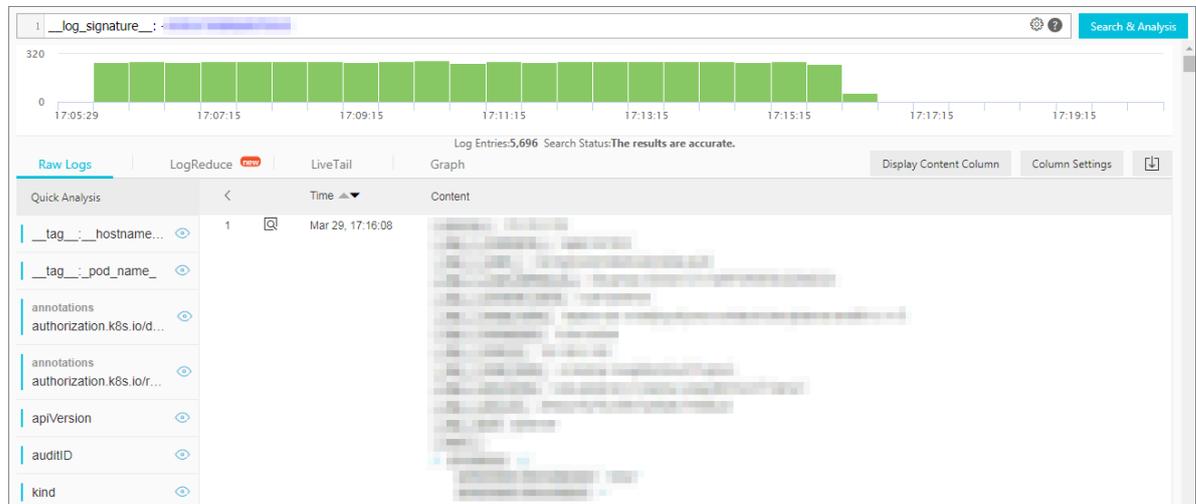
 **Note:**
 You can also click + in front of a Count value to show the pattern list of the log group.

Figure 7-6: View log grouping details



4. Click a Count value to view the raw log of the log group.

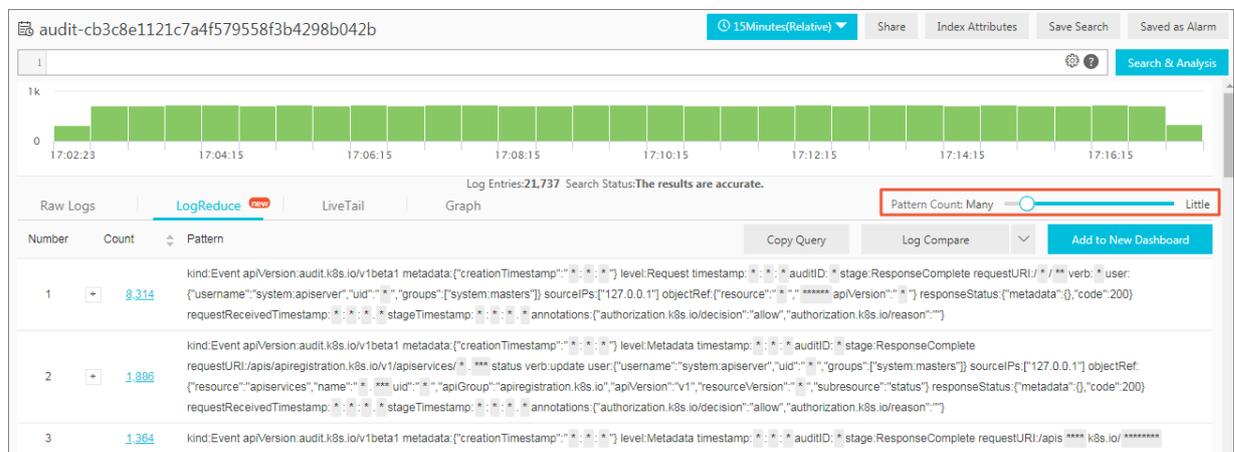
Figure 7-7: View the raw log



Adjust the log grouping tolerance

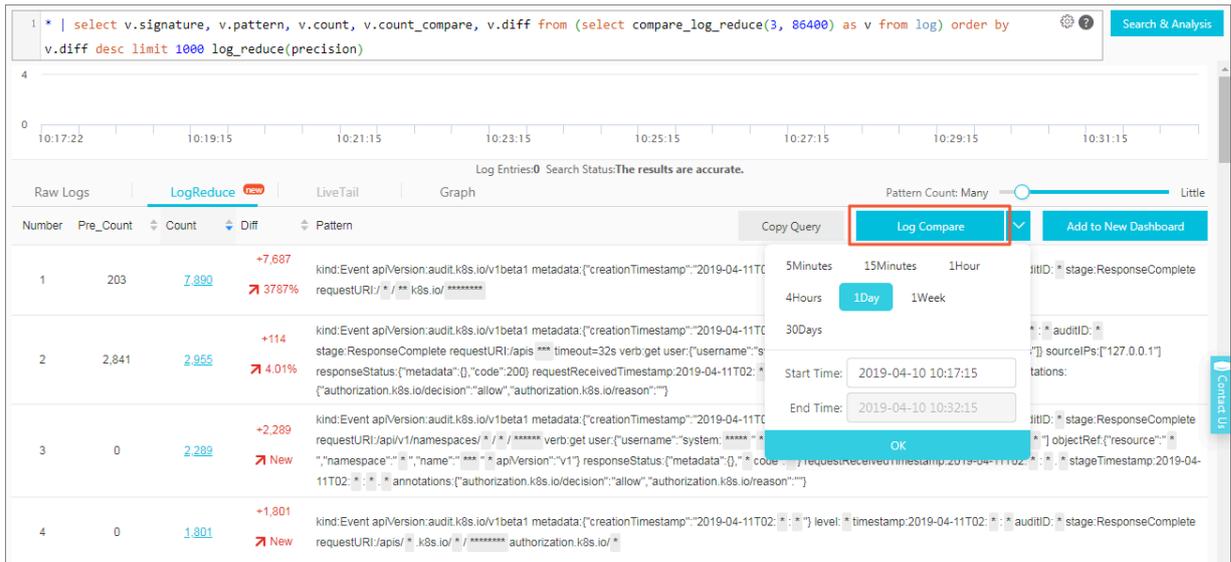
1. On the Search & Analysis page, click the LogReduce tab.
2. In the upper-right corner of the tab page, drag the Pattern slider to adjust the log grouping tolerance.
 - If you drag the slider towards Many, the system outputs a more specific log grouping result and shows patterns in greater detail.
 - If you drag the slider towards Little, the system outputs a less specific log grouping result and shows patterns in less detail.

Figure 7-8: Adjust the log grouping tolerance



Compare the number of group log entries during different periods of time

Click Log Compare, select a time length, and then click OK.



Item	Description
Number	Displays the sequence number of a log group.
Pre_Count	Displays the number of log entries during a time range.
Count	Displays the number of log entries for the log pattern for the current time range.
Diff	Displays the difference in the number of log entries for the log pattern for the current time range and a past time range.
Pattern	Displays the log pattern.

Use the LogReduce function through API

- To obtain a log grouping result, execute the following SQL statement:

```
* | select a . pattern , a . count , a . signature , a .
origin_sig natures from ( select log_reduce ( 3 ) as a
from log ) limit 1000
```

 **Note:**

If you directly view the result through the Log Service console, you can click Copy Query to get the SQL statement executed by the system in the backend.

Parameter and field description

- The parameter in the SQL statement that you need to customize is `log_reduce (precision)`.

This parameter must be set to an integer that is in the range of 1 to 16. Its default value is 3. A lower tolerance value outputs a grouping result of a higher tolerance, and more log patterns.

- The execution result of the SQL statement contains the following returned fields:
 - `pattern` : indicates the sub-patterns of log entries in a log group.
 - `count` : indicates the number of log entries in a log group.
 - `signature` : indicates the log pattern of a log group.
 - `origin_signatures` : indicates the original signature of a log group. You can use this field to search the log entries of this log group.
- To show the difference of log grouping results between different times, execute the following SQL statement:

```
* | select v . pattern , v . signature , v . count , v .
count_compare , v . diff from ( select compare_log_reduce
( 3 , 86400 ) as v from log ) order by v . diff desc
limit 1000
```



Note:

If you click Log Compare in the Log Service console to show the difference of log grouping results between different times, the system then executes an SQL statement for the log entries. You can click Copy Query to get the SQL statement.

Parameter and field description

- The parameter in the SQL statement that you need to customize is `compare_log_reduce (precision , compare_interval)`.
 - The tolerance parameter must be an integer that is in the range of 1 to 16. Its default value is 3. A lower tolerance value outputs a grouping result of a higher tolerance, and more log patterns.
 - The `compare_interval` parameter indicates the number of seconds before which the log entries to be compared with was generated. This parameter must be set as a positive integer.
- The execution result of the SQL statement contains the following returned fields:
 - `pattern` : indicates the sub-patterns of log entries in a log group.
 - `signature` : indicates the log pattern of a log group.
 - `count` : indicates the number of log entries in a log group.
 - `count_compare` : indicates the number of log entries for a log group of the same log pattern within the specified time range.
 - `diff` : indicates the difference between the count field value and the `count_compare` field value.

7.4 Context query

When you expand a log file, each log records an event. Generally, logs are not independent from each other. Several consecutive logs allow you to view the process of a whole event in sequence.

Log context query specifies the log source (machine + files) and a log in the log source. It also queries several logs before and after the log in the original log file, providing a helpful method for troubleshooting the problem in the DevOps scenario.

The Log Service console provides a query page, you can view the context information of the specified log in the original file in the console. It is similar to paging up and down in the original log file. By viewing the context information of a specified log, you can quickly locate the problem.

Scenarios

For example, the O2O take-out website will record the transaction track of a order in the program log on the server:

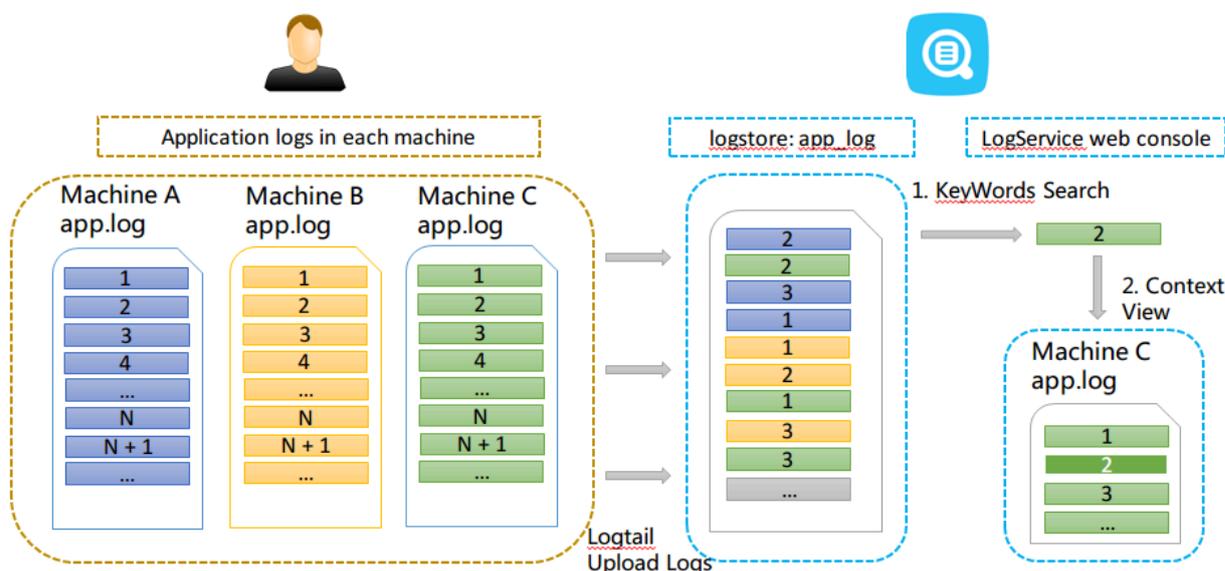
User logon > Browse products > Click items > Add to shopping cart > Place an order > Pay for the order > Deduct payment > Generate an order

If the order cannot be placed, the Operation & Maintenance (O&M) personnel must quickly locate the cause of the problem. In the conventional context query, the administrator grants the machine logon permission to related members, and then the investigator logs on to each machine where applications are deployed in turn, uses the order ID as the keyword to search application log files, and determines what causes the failure.

In Log Service, you can troubleshoot the problem by following these steps:

1. Install the log collection client Logtail on the server, and add the machine group and log collection configuration in the console. Then, Logtail starts to upload the incremental logs. You can also use producer-related SDK uploads, such as Log4J, LogBack, C-Producer
2. On the log query page in the Log Service console, specify the time range, and find the order failure log according to the order ID.
3. Based on the found error log, page up until other related logs are found (for example, the deduction failure of credit card).

Figure 7-9: Scenarios



Benefits

- No intrusion into the application. No need to modify the log file format.
- You can view the log context information of any machine or file in the Log Service console, without logging on to each machine to view the log file.
- Combined with the time when the event occurred, you can specify the time range to quickly locate the suspicious log and then query its context information in the Log Service console to improve the efficiency.
- No need to worry about the data loss caused by insufficient server storage space or log file rotation. You can view historical data in the Log Service console at any time.

Prerequisites

- [Use Logtail to collect logs](#) . Upload data to the Logstore. Create the machine groups and collection configuration. No other configurations are needed. You can also use producer-related SDK upload, such as Producer Library.
- Enable the Query logs function.



Note:

Currently, you cannot query the context information of syslog data.

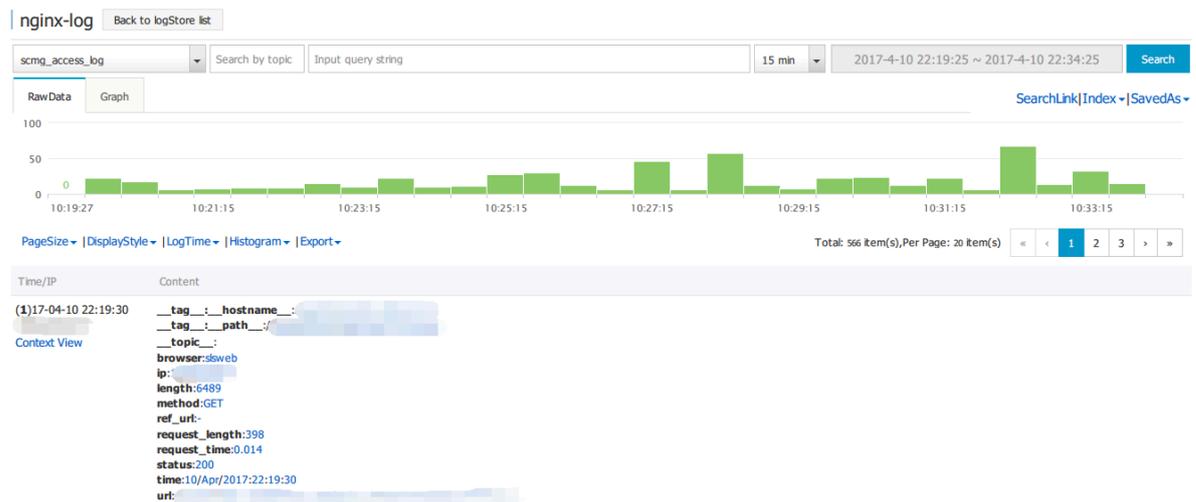
Procedure

1. Log on to the Log Service console.
2. On the Project List page, click the project name.
3. On the Logstore List page, click Query at the right of the Logstore to enter the query interface.

4. Enter your query and analysis statement and select the time range. Then, click Search.

Click Context View at the left side of the log, and the window with the context information of the target log is displayed on the right.

Figure 7-10: Query log



5. Select a log and click Context View. View the context log for the target log on the right pop-up page.
6. Scroll with the mouse on the page to view the context information of the selected log. To view more context logs, click Earlier or Later.

7.5 Saved search

Saved search is a one-click query and analysis feature provided by Log Service.

Prerequisites

Indexes are enabled and set.

Context

If you need to frequently view the results of a query and analysis statement, you can save the statement as a saved search. In next searches, you only need to click the name of the saved search on the left side of the search page, instead of entering the statement again. You can also use this saved search in alert rules. Log Service runs the statement of this saved search periodically and sends an alert notification when the search result meets the preset condition of the statement.

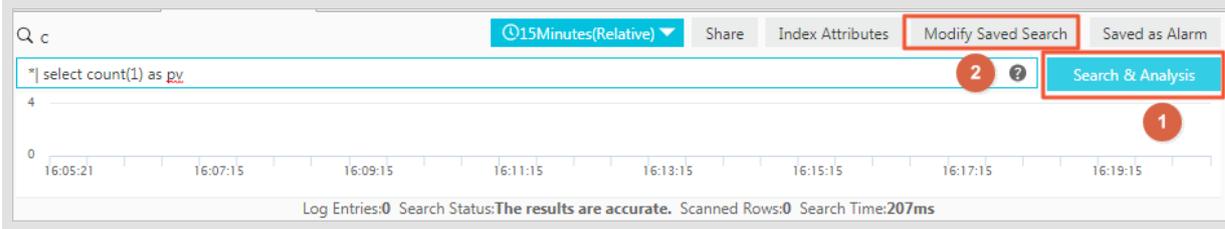
To configure a drill-down event to jump to a saved search when configuring [#unique_74](#), you must preset a saved search and set a placeholder in the query and analysis statement.



Note:

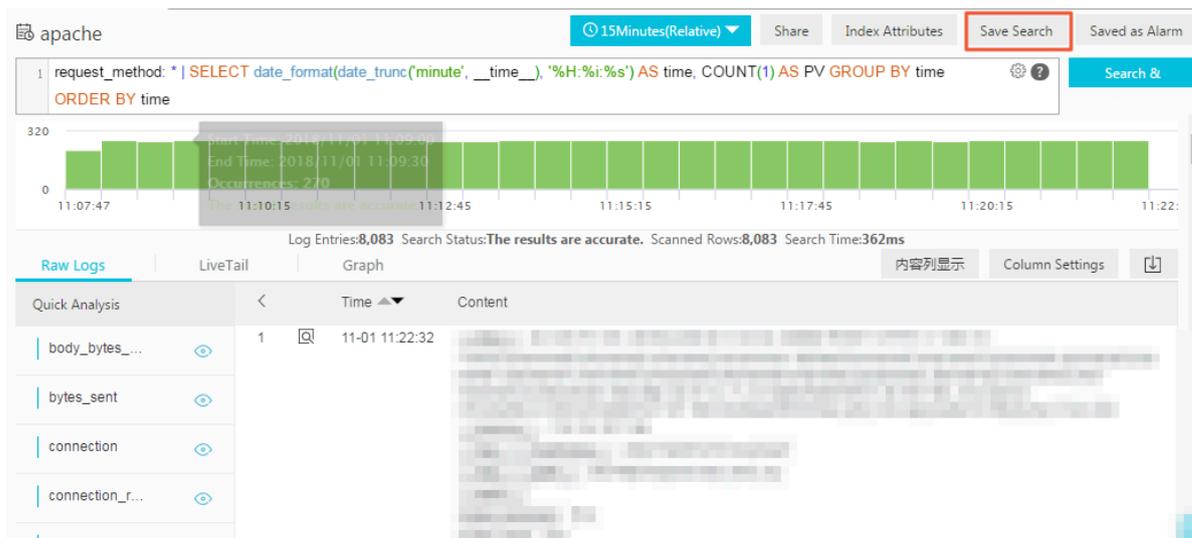
Q: How can I modify a saved search?

A: To modify a saved search, enter a new query and analysis statement, click Search & Analysis to run the statement, and then click Modify Saved Search.



Procedure

1. 登录日志服务控制台，单击Project名称。
2. On the Logstores page, find the target Logstore and click Search in the LogSearch column.
3. Enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
4. Click Save Search in the upper-right corner of the page.



5. Configure saved search attributes.

a) Set Saved Search Name.

- The name can contain only lowercase letters, digits, hyphens (-), and underscores (_).
- The name must start and end with a lowercase letter or digit.
- The name must be 3 to 63 characters in length.

b) Check the values of Logstores, Topic, and Query.

If the values of Logstores and Topic do not meet your requirements, return to the search page, go to the target Logstore, enter the query and analysis statement, and then click Save Search again.

c) Optional: Select part of the query statement and click Generate Variable.

The generated variable is a placeholder variable. Set the placeholder variable name in the Variable Name field. Default Value indicates the selected word.



Note:

If the drill-down event of a chart is configured to jump to this saved search and the chart has the same variable as the value of Variable Name of this saved search, clicking the chart triggers the jump. Additionally, the value of Default Value of the placeholder variable is replaced with the chart value that triggers

the drill-down event. The query statement with the replaced variable replaced is run. For more information, see [#unique_74](#).

Saved Search Details
✕

* Saved Search Name

Attributes

Logstores

Topic

Query

Select the query statement to generate a placeholder variable. You can configure a drill-down configuration to replace the variable.

Variable Config

Variable Name:	Default Value:	Matching Mode:	
stage	stage	Global Match ▼	✕

Result

```
* | SELECT S{stage}, COUNT(*) as number GROUP BY S{stage} LIMIT 10
```

6. Click OK.

7.6 Quick analysis

The quick analysis function of Log Service supports an interactive query with only one click, allowing you to quickly analyze the distribution of a field over a period of time and reduce the cost of indexing key data.

Functions and features

- Support grouping statistics for the first 10 of the first 100,000 pieces of data of `Text` fields.
- Support generating `approx_distinct` statements quickly for `Text` fields.
- Support histogram statistics for the approximate distribution of `long` or `double` fields.

- Support the quick search for the maximum, minimum, average, or sum of long or double fields.
- Support generating query statements based on quick analysis and query.

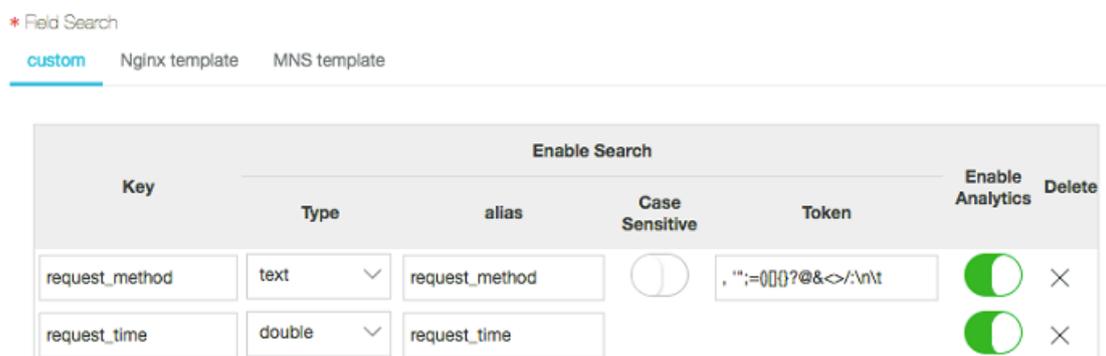
Prerequisite

You must specify the field query properties before using the quick analysis.

1. For specified field query, you must enable the index to activate the query and analysis function. For how to enable the index, see [Query and analysis](#).
2. Set the key in the log as the field name and set the type, alias, and separator.

If the access log contains the request_method and request_time, you can configure the following settings.

Figure 7-11: Prerequisites



User Guide

After setting the specified field query, you can see the fields in Quick Analysis under the Raw Data tab on the query page. By clicking the 1 button above the serial number,

you can fold the page. By clicking the eye button, you can perform quick analysis based on the Current Temporal Interval and Current \$Search conditions.

Figure 7-12: Original log

The screenshot shows the Log Service interface with the following components:

- Navigation:** Raw Logs (selected), LogReduce (new), LiveTail, Graph.
- Quick Analysis:** A list of tags with eye icons for quick analysis. The first tag, `__tag__:pod_name__`, is highlighted with a red box.
- Log Entry:**
 - Index: 1
 - Time: Jul 3, 14:42:47
 - Content: A list of tags and their values, including `__source__`, `__tag__:hostname__`, `__tag__:path__`, `__tag__:user_defined_id__`, `__tag__:container_ip__`, `__tag__:container_name__`, `__tag__:image_name__`, `__tag__:namespace__: kube-system`, `__tag__:node_ip__`, `__tag__:node_name__: cn-4nag-100.100.0.72`, `__tag__:pod_name__: kube-`, `__tag__:pod_uid__: ea382`, and `__tag__:audit__: apiserver`.

Text

- Grouping statistics for Text fields

Click the eye button at the right of the field to quickly group the first 100,000 pieces of data of this Text field and return the ratio of the first 10 pieces.

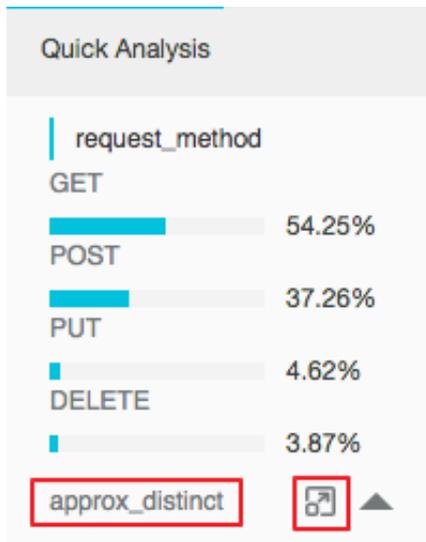
Query statement:

```
$ Search | select ${ keyName }, pv , pv * 1 . 0 / sum ( pv )
over ( ) as percentage from ( select count ( 1 ) as pv ,
"${ keyName }" from ( select "${ keyName }" from log limit
```

```
100000 ) group by "${ keyName }" order by pv desc )
order by pv desc limit 10
```

`request_method` returns the following result based on the grouping statistics, where GET requests are in the majority.

Figure 7-13: Group statistics



- Check the number of unique entries of the field

Under the target fields in Quick Analysis, click `approx_distinct` to check the number of unique entries for `${ keyName }`.

`request_method` can get the following result by grouping statistics, and GET requests account for the majority:

- Extend the query statement of grouping statistics to the search box

Click the button at the right of `approx_distinct` to extend the query statement of grouping statistics to the search box for further operations.

long/double

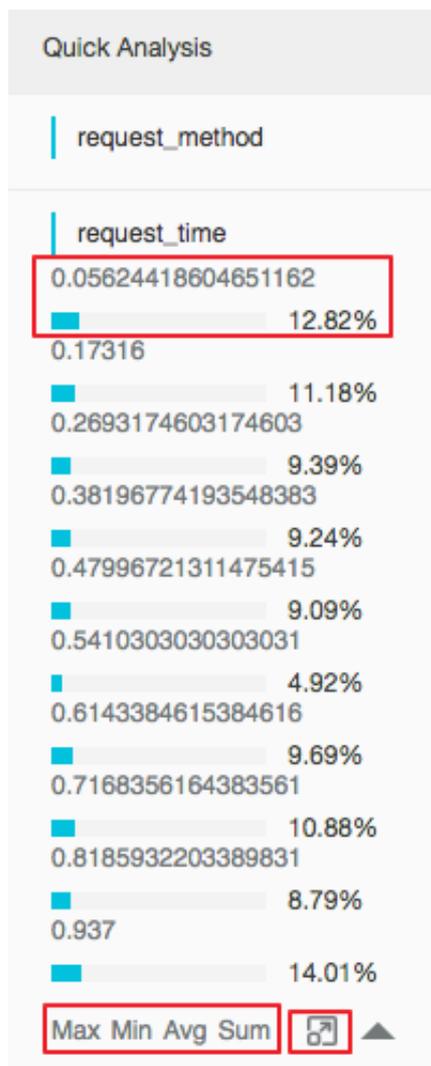
- Histogram statistics for the approximate distribution

Grouping statistics is of little significance for the `long / double` fields, which have multiple type values. Therefore, histogram statistics for the approximate distribution is adopted by using 10 buckets.

```
$ Search | select numeric_histogram ( 10 , ${ keyName })
```

`request_time` returns the following result based on the histogram statistics for the approximate distribution, from which you can see that the request time is mostly distributed around 0.056.

Figure 7-14: Request Distribution



- Quick analysis of the `Max` `Min` `Avg` `Sum` statements

Respectively click `Max` , `Min` , `Avg` , and `Sum` under the target fields to quickly search for the maximum, minimum, average, and sum of all `${keyName}`.

- Extend the query statement of grouping statistics to the search box

Click the button at the right of `Sum` to extend the query statement of the histogram statistics for the approximate distribution to the search box for further operations.

7.7 Other functions

Search and analysis functions help you to query various statements in logs, [query raw logs](#), [view graphs](#), [query context](#), [perform quick analysis](#), [perform quick queries](#), [create a dashboard](#), and [save a graph as an alarm](#).

Query raw logs

After the index is enabled, enter the keywords in the search box and select the search time range. Then, click Search to view the histogram of the log quantity, the raw logs, and the statistical graph.

The histogram of the log quantity displays the time-based distribution of log search hit counts. With the histogram, you can view the log quantity changes over a certain period of time. By clicking the rectangular area to narrow down the time range, you can view the information about the log hits within the specified time range to refine the display of the log search results.

On the Raw Data tab, you can view the hit logs in chronological order.

- By clicking the triangle symbol next to Time, you can switch between the chronological and reverse chronological orders.
- By clicking Display Content Column, you can switch between Display with Line Breaks and Display in One Line, or you can set Truncate Character String.
- By clicking the value keyword in the log content, you can view all logs containing this keyword.
- By clicking the Download button in the upper-right corner of the Raw Data tab, you can download the query results in CSV format. By clicking the Config button, you can add fields as displayed columns in the display results of raw logs so that

- You can set the [drill-down analysis](#) action for a graph. Then, after a graph is added to the dashboard, any click to a data point on the graph will trigger the drill-down analysis action, allowing you to review queries in more details.

Figure 7-16: Statistical graphs



Query context

The Log Service console provides a query page, you can view the context information of the specified log in the original file in the console. It is similar to paging up and down in the original log file. By viewing the context information of the specified log, you can quickly locate the failure information during the business troubleshooting. For more information, see [#unique_77](#).

Perform quick analysis

The quick analysis function of Log Service supports an interactive query with only one click, allowing you to quickly analyze the distribution of a field over a period of time and reduce the cost of indexing key data. For more information, see [#unique_52](#).

Perform quick queries

You can save the current query condition as a saved search. To perform this query again, you simply need to go to the saved search page. For more information, see [#unique_56](#).

You can also apply the saved search condition to alarm rules. After you set an alarm rule, Log Service will automatically run the saved search on a regular basis. If query results meet the preset threshold, Log Service will send an alarm message.

Create a dashboard

Log Service provides the dashboard function, which can visualize the query and analysis statements. For more information, see [#unique_54](#).

Figure 7-17: Dashboard



Save a graph as an alarm

Log Service can generate an alarm based on your LogSearch Results. You can configure the alarm rules so that specific alarm content can be sent to you by using in-site notifications or DingTalk messages.

For more information, see [#unique_57](#).

8 Analysis grammar

8.1 General aggregate functions

The query and analysis feature of Log Service allows you to use general aggregate functions to analyze logs. The following table describes the specific statements.

Statement	Description	Example
<code>arbitrary (x)</code>	Returns an arbitrary value in column x.	latency > 100 select arbitrary (method)
<code>avg (x)</code>	Calculates the arithmetic mean of all the values in column x.	latency > 100 select avg (latency)
<code>checksum (x)</code>	Calculates the checksum of all the values in column x and returns a Base64-encoded value.	latency > 100 select checksum (method)
<code>count (*)</code>	Calculates the number of rows.	N/A
<code>count (x)</code>	Calculates the number of non-null values in column x.	latency > 100 count (method)
<code>count (digit)</code>	Functions the same as <code>count (*)</code> to calculate the number of rows. For example, <code>count (1)</code> .	N/A
<code>count_if (x)</code>	Calculates the number of true values.	latency > 100 count_if (url like '% abc ')
<code>geometric_ mean (x)</code>	Calculates the geometric mean of all the values in column x.	latency > 100 select geometric_ mean (latency)

Statement	Description	Example
<code>max_by (x , y)</code>	Returns the value of x associated with the maximum value of y .	To query the method for the maximum latency: <pre>latency > 100 select max_by (method , latency)</pre>
<code>max_by (x , y , n)</code>	Returns the values of x associated with the n largest values of y in descending order of y .	To query the method for the top three rows with the maximum latency: <pre>latency > 100 select max_by (method , latency , 3)</pre>
<code>min_by (x , y)</code>	Returns the value of x associated with the minimum value of y .	To query the method for the minimum latency: * <pre> select min_by (method , latency)</pre>
<code>min_by (x , y , n)</code>	Returns the values of x associated with the n smallest values of y in ascending order of y .	To query the method for the top three rows with the minimum latency: * <pre> select min_by (method , latency , 3)</pre>
<code>max (x)</code>	Returns the maximum value of all the values in column x .	<pre>latency > 100 select max (inflow)</pre>
<code>min (x)</code>	Returns the minimum value of all the values in column x .	<pre>latency > 100 select min (inflow)</pre>
<code>sum (x)</code>	Returns the sum of all the values in column x .	<pre>latency > 10 select sum (inflow)</pre>
<code>bitwise_and_agg (x)</code>	Returns the bitwise AND of all the values in column x in 2's complement representation.	N/A
<code>bitwise_or_agg (x)</code>	Returns the bitwise OR of all the values in column x in 2's complement representation.	N/A

8.2 Security detection functions

Based on the global white hat shared security asset library, Log Service provides security detection functions. All you need to do is to pass any IP address, domain name, or URL in the log to security detection functions, you can detect whether it is secure or not.

Scenarios

1. Enterprises and institutions that have a strong demand for service operation and maintenance, such as enterprises of Internet, games, information, and more. The IT and security Operation and Maintenance (O&M) personnel of these industries can use security detection functions to timely filter for suspicious accesses, attacks, and intrusions. The security detection function also supports further in-depth analysis and measures to defend against them.
2. Enterprises and institutions that have strong demand for internal asset protection, such as banks, securities, e-commerce, and more. Their IT and security O&M personnel can instantly discover internal access to dangerous websites, download the trojan horse, and more, and take immediate action.

Features

- **Reliable:** Relies on the global shared white hat security asset library with timely update.
- **Fast:** Takes only a few seconds to detect millions of IP address, domain names, or URLs.
- **Simple:** Seamlessly supports any network log. The result can be obtained by calling three SQL functions: `security_check_ip`, `security_check_domain`, and `security_check_url`.
- **Flexible:** Supports both interactive queries and building report views. You can configure alarms and take further action.

Function list

Function name	Description	Example
security_check_ip	Check if the IP address is secure, where: <ul style="list-style-type: none"> Return 1: Hit, indicating insecure Return 0: Missing 	<pre>select security_c heck_ip (real_clien t_ip)</pre>
security_check_domain	Check if the domain is secure, where: <ul style="list-style-type: none"> Return 1: Hit, indicating insecure Return 0: Missing 	<pre>select security_c heck_domai n (site)</pre>
security_check_url	Check if the URL is secure, where: <ul style="list-style-type: none"> Return 1: Hit, indicating insecure Return 0: Missing 	<pre>select security_c heck_domai n (concat (host , url)</pre>

Example

- Check external suspicious access behavior and generate reports

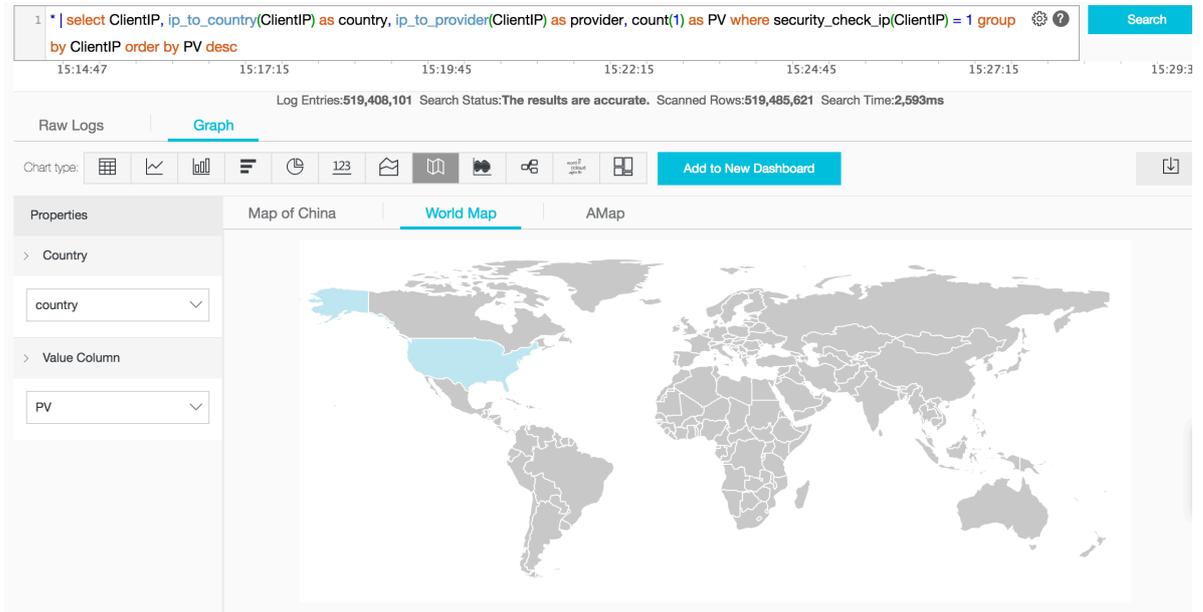
An e-commerce collects logs of the Nginx server that it operates, and intends to scan clients that access the server to check if insecure client IP addresses exist. In this case, pass the ClientIP field in the Nginx log to the `security_c heck_ip` function, display IP addresses whose return value is 1, and show the country, the network operator and other related information of the IP addresses.

The query analysis statement is:

```
* | select ClientIP , ip_to_coun try ( ClientIP ) as country
, ip_to_prov ider ( ClientIP ) as provider , count ( 1 ) as
```

```
PV where security_check_ip ( ClientIP ) = 1 group by
ClientIP order by PV desc
```

Set to map view display:



- Check internal suspicious access behavior and configure alarms

For example, a securities operator collects network traffic logs recorded when its internal devices access the external network through a gateway proxy. To check if someone has accessed websites with problems, perform the following query:

```
* | select client_ip , count ( 1 ) as PV where
security_check_ip ( remote_address ) = 1 or security_check_site ( site ) = 1 or security_check_url ( concat ( site , url ) ) = 1 group by client_ip order by PV desc
```

You can also save this statement as a quick query and configure a security alarm.

When a client access dangerous websites frequently, the alarm is triggered.

Configure 5-minute intervals for checking if someone has accessed dangerous

websites frequently (more than 5 times) during the past one hour. Change parameters based on your needs. The configuration is as follows:

Create Alert
✕

Alert Configuration

Notifications

* Alert Name 22/64

* Add to New Dashboard ? 16/64

* Chart Name 22/64

Query

* | select client_ip, count(1) as PV where security_check_ip(remote_addr) = 1 or security_check_site(site) = 1 or security_check_url(concat(site, url)) = 1 group by client_ip order by PV desc

* Search Period ? 🕒 1Hour(Relative) ▼

* Search Interval + -

* Trigger Condition ?

Support the addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) operations and comparison operations including >, >=, <, <=, ==, !=, =~, and !~. [Documentation](#)

Advanced ▼

* Notification Trigger Threshold ? + -

* Notification Interval ?

8.3 Mapping function

The Log Service query analysis function supports log analysis by using mapping functions with detailed statements and implications described in the following table:

Function	Description	Example
Subscript operator []	Gets the result of a key in the map.	-
histogram(x)	Performs GROUP BY according to each value of column x and calculates the count. The syntax is equivalent to <code>select count group by x</code> .  Note: Returned information must be in JSON format.	<code>latency > 10 select histogram (status), which is equivalent to latency > 10 select count (1) group by status</code>
histogram_u(x)	Performs GROUP BY according to each value of column x and calculates the count.  Note: Returned information must be in multi-row multi-column format.	<code>latency > 10 select histogram (status), which is equivalent to latency > 10 select count (1) group by status</code>
map_agg(Key,Value)	Returns a map of key, value, and shows the random latency of each method.	<code>latency > 100 select map_agg (method , latency)</code>
multimap_agg(Key,Value)	Returns a multi-value map of key, value, and returns all the latency for each method.	<code>latency > 100 select multimap_agg (method , latency)</code>
cardinality(x) → bigint	Gets the size of the map.	-
element_at(map< K , V >, key) → V	Gets the value corresponding to the key.	-
map() → map< unknown , unknown >	Returns an empty map.	-

Function	Description	Example
<code>map(array< K >, array< V >) → map< K , V ></code>	Converts two arrays into 1-to-1 maps.	<pre>SELECT map (ARRAY [1 , 3], ARRAY [2 , 4]); - { 1 -> 2 , 3 -> 4 }</pre>
<code>map_from_entries(array< row < K , V >>) → map< K , V ></code>	Converts a multidimensional array into a map.	<pre>SELECT map_from_entries (ARRAY [(1 , ' x '), (2 , ' y ')]); - { 1 -> ' x ', 2 -> ' y ' }</pre>
<code>map_entries(map< K , V >) → array< row < K , V >></code>	Converts an element in a map into an array.	<pre>SELECT map_entries (MAP (ARRAY [1 , 2], ARRAY [' x ', ' y '])); - [ROW (1 , ' x '), ROW (2 , ' y ')]</pre>
<code>map_concat(map1< K , V >, map2< K , V >, ..., mapN< K , V >) → map< K , V ></code>	The Union of multiple maps is required, if a key exists in multiple maps, take the first one.	-
<code>map_filter(map< K , V >, function) → map< K , V ></code>	Refer to the lambda map_filter function.	-
<code>transform_keys(map< K1 , V >, function) → MAP< K2 , V ></code>	Refer to the lambda transform_keys function.	-
<code>transform_values(map< K , V1 >, function) → MAP< K , V2 ></code>	Refer to the lambda transform_values function.	-
<code>map_keys(x< K , V >) → array< K ></code>	Gets all the keys in the map and returns an array.	-
<code>map_values(x< K , V >) → array< V ></code>	Gets all values in the map and returns an array.	-

Function	Description	Example
<code>map_zip_with(map< K , V1 >, map< K , V2 >, function< K , V1 , V2 , V3 >) → map< K , V3 ></code>	Refer to power functions in Lambda.	-

8.4 Estimating functions

The query and analysis function of Log Service supports analyzing logs by using estimating functions. The specific statements and meanings are as follows.

Function	Description	Examples
<code>approx_distinct (x)</code>	Estimates the number of unique values in column x.	-
<code>approx_percentile (x , percentage)</code>	Sorts the column x and returns the value approximately at the given percentage position.	Returns the value at the half position: <code>approx_percentile (x , 0 . 5)</code>
<code>approx_percentile (x , percentage s)</code>	Similar to the preceding statement, but you can specify multiple percentages to return the values at each specified percentage position.	<code>approx_percentile (x , array [0 . 1 , 0 . 2])</code>

Function	Description	Examples
<pre>numeric_histogram (buckets , Value)</pre>	<p>Collects values in the numeric column by bucket. That is, you need to enter the <i>Value</i> column into buckets. The number of buckets is determined by <i>buckets</i>.</p> <p>The returned information is the key of each bucket and the corresponding count. This works similarly to <code>select count group by for numbers</code>.</p> <div data-bbox="644 981 1023 1137" style="background-color: #f0f0f0; padding: 5px;">  Note: Returned results must be in JSON format. </div>	<p>For POST requests, the delay is divided into 10 buckets. You can run <code>method : POST select numeric_histogram (10 , latency)</code> to check the size of each bucket.</p>

Function	Description	Examples
<code>numeric_histogram (buckets , Value)</code>	<p>Collects values in the numeric column by bucket. That is, you need to enter the <i>Value</i> column into buckets. The number of buckets is determined by <i>buckets</i>.</p> <p>The returned information is the key of each bucket and the corresponding count. This works similarly to <code>select count group by for numbers</code>.</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  Note: Returned results must be in multi-row multi-column format. </div>	<p>For POST requests, the delay is divided into 10 buckets. You can run <code>method : POST select numeric_histogram (10 , latency)</code> to check the size of each bucket.</p>

8.5 Mathematical statistics functions

The query and analysis function of Log Service supports analyzing logs by using mathematical statistics functions. The specific statements and meanings are as follows.

Statements	Meaning	Example
<code>corr (y , x)</code>	Returns the correlation coefficient of two columns . The result is from 0 to 1.	<code>latency > 100 select corr (latency , request_size)</code>
<code>covar_pop (y , x)</code>	Calculates the population covariance.	<code>latency > 100 select covar_pop (request_size , latency)</code>

Statements	Meaning	Example
<code>covar_samp (y , x)</code>	Calculates the sample covariance.	latency > 100 select covar_samp (request_si ze , latency)
<code>regr_inter cept (y , x)</code>	Returns the linear regression intercept of input values. y is the dependent value. x is the independent value.	latency > 100 select regr_inter cept (request_si ze , latency)
<code>regr_slope (y , x)</code>	Returns the linear regression slope of input values. y is the dependent value. x is the independent value.	latency > 100 select regr_slope (request_si ze , latency)
<code>stddev (x)</code> or <code>stddev_sam p (x)</code>	Returns the sample standard deviation of column x.	latency > 100 select stddev (latency)
<code>stddev_pop (x)</code>	Returns the population standard deviation of column x.	latency > 100 select stddev_pop (latency)
<code>variance (x)</code> or <code>Var_samp (X)</code>	Calculates the sample variance of column x.	latency > 100 select variance (latency)
<code>var_pop (x)</code>	Calculates the population variance of column x.	latency > 100 select variance (latency)

8.6 Mathematical calculation functions

The query and analysis function of Log Service supports analyzing logs by using mathematical calculation functions. By combining query statements with mathematical calculation functions, you can perform mathematical calculation to the log query results.

Mathematical operators

Mathematical operators support plus sign (+), minus sign (-), multiplication sign (*), division sign (/), and percent sign (%), which can be used in the SELECT clause.

Example:

```
*| select avg ( latency )/ 100 , sum ( latency )/ count ( 1 )
```

Description of mathematical calculation function

Log Service supports the following operating functions.

Function name	Meaning
<code>abs (x)</code>	Returns the absolute value of column x.
<code>Cbrt (X)</code>	Returns the cube root of column x.
<code>ceiling (x)</code>	Returns the number rounded up to the nearest integer of column x.
<code>cosine_similarity (x , y)</code>	Returns the cosine similarity between the sparse vectors x and y.
<code>degrees</code>	Converts radians to degrees.
<code>e ()</code>	Returns the natural constant.
<code>exp (x)</code>	Returns the exponent of the natural constant.
<code>floor (x)</code>	Returns the number rounded down to the nearest integer of column x.
<code>from_base (string , radix)</code>	Returns the string interpreted in the base-radix notation.
<code>ln (x)</code>	Returns the natural logarithm. Returns the natural log.
<code>log2 (x)</code>	Returns the base-2 logarithm of x.
<code>log10 (x)</code>	Returns the base-10 logarithm of x.
<code>log (x , b)</code>	Returns the base-b logarithm of x.
<code>pi ()</code>	Returns π .
<code>pow (x , b)</code>	Returns x to the power of b.
<code>radians (x)</code>	Converts degrees to radians.
<code>rand ()</code>	Returns a random number.
<code>random (0 , n)</code>	Returns a random number in the range of [0,n).
<code>round (x)</code>	Returns x rounded to the nearest integer.

Function name	Meaning
<code>round (x , y)</code>	Returns x rounded to the nearest integer.
<code>sqrt (x)</code>	Returns the square root of x.
<code>to_base (x , radix)</code>	Returns the base-radix representation of x.
<code>truncate (x)</code>	Returns x rounded to integer by dropping digits after decimal point.
<code>acos (x)</code>	Returns the arc cosine.
<code>Asin (X)</code>	Returns the arc sine.
<code>atan (x)</code>	Returns the arc tangent.
<code>atan2 (y , x)</code>	Returns the arc tangent of y/x.
<code>cos (x)</code>	Returns the cosine.
<code>sin (x)</code>	Returns the sine.
<code>cosh (x)</code>	Returns the hyperbolic cosine.
<code>tan (x)</code>	Returns the tangent.
<code>tanh (x)</code>	Returns the hyperbolic tangent.
<code>Infinity ()</code>	Returns the double maximum value.
<code>is_infinite (x)</code>	Determines whether it is the maximum value or not.
<code>is_finite (x)</code>	Determines whether it is the maximum value or not.
<code>is_nan (x)</code>	Determines whether it is a number or not.

8.7 String functions

The query and analysis function of Log Service supports analyzing logs by using string functions. The specific statements and description are as follows.

Function name	Description
<code>chr (x)</code>	Converts the int type to the corresponding ASCII string, for example <code>chr (65)=' A ' .</code>

Function name	Description
<code>codepoint (x)</code>	Converts the ASCII type to the corresponding int string, for example <code>codepoint (' a ')=97</code> .
<code>length (x)</code>	Returns the length of a field.
<code>levenshtein_distance (string1 , string2)</code>	Returns the minimum edit distance between two strings.
<code>lower (string)</code>	Converts the string to lowercase characters.
<code>lpad (string , size , padstring)</code>	Aligns the string to the size. If it is smaller than the size, uses padstring to fill the size from the left side. If it is larger than size, it is truncated to size.
<code>rpad (string , size , padstring)</code>	The same as the lpad, complement the string from the right.
<code>ltrim (string)</code>	Deletes the white-space characters on the left.
<code>replace (string , search)</code>	Deletes search from the string.
<code>replace (string , search , rep)</code>	Replaces search with rep in the string.
<code>reverse (string)</code>	Returns a string with the characters in the reverse order.
<code>rtrim (string)</code>	Deletes the white-space characters at the end of the string.
<code>split (string , delimiter , limit)</code>	Split the string into array and get a maximum of limit values. The generated result is an array with subscripts starting at 1.
<code>split_part (string , delimiter , offset)</code>	Splits the string into an array and obtains the offset string. The generated result is an array with subscripts starting at 1.
<code>split_to_map (string , entryDelimiter , keyValueDelimiter) → map < varchar , varchar ></code>	The string is divided into multiple entries according to entryDelimiter, and each entry is divided into key values according to keyValueDelimiter. Eventually returns a map.
<code>position (substring IN string)</code>	Get the position in the string where the substring starts.

Function name	Description
<code>strpos (string , substring)</code>	Finds the starting position of the substring in the string. The returned result starts at 1. If not found, 0 is returned.
<code>substr (string , start)</code>	Returns a substring of a string with a subscript starting at 1.
<code>substr (string , start , length)</code>	Returns a substring of a string with a subscript starting at 1 and length.
<code>trim (string)</code>	Deletes the white-space characters at the beginning and end of the string.
<code>upper (string)</code>	Converts the string to uppercase characters.
<code>concat (string , string)</code>	Splices two or more strings into a single string.
<code>hamming_distance (string1 , string2)</code>	Returns the hamming distance between two strings.

**Note:**

Strings must be enclosed in single quotation marks, and double quotation marks indicate column names. For example, `a=' abc'` indicates column `a = string abc`, and `a = "abc"` means column `a = column abc`.

8.8 Date and time functions

Log Service supports time functions, date functions, interval functions, and a time series padding function. You can use the functions described in this topic in analysis statements.

Date and time data types

1. **Unix timestamp:** the number of seconds that have elapsed since 00:00:00 (UTC) on January 1, 1970. The value is of the integer type. For example, `1512374067` indicates the time `Mon Dec 4 15 : 54 : 27 CST 2017`. The built-in time `__time__` in each log of Log Service is of this type.
2. **timestamp:** the time in the format of string, for example, `2017 - 11 - 01 13 : 30 : 00`.

Date functions

The following table describes the common date functions supported by Log Service.

Function	Description	Example
<code>current_date</code>	Returns the current date.	latency > 100 select current_date
<code>current_time</code>	Returns the current time.	latency > 100 select current_time
<code>current_timestamp</code>	Returns the current timestamp by combining the results of <code>current_date</code> and <code>current_time</code> .	latency > 100 select current_timestamp
<code>current_timezone ()</code>	Returns the current time zone.	latency > 100 select current_timezone ()
<code>from_iso8601_timestamp (string)</code>	Parses an ISO 8601-formatted time into a timestamp that contains the time zone.	latency > 100 select from_iso8601_timestamp (iso8601)
<code>from_iso8601_date (string)</code>	Parses an ISO 8601-formatted time into a date.	latency > 100 select from_iso8601_date (iso8601)
<code>from_unixtime (unixtime)</code>	Returns a Unix timestamp as a timestamp.	latency > 100 select from_unixtime (1494985275)
<code>from_unixtime (unixtime , string)</code>	Returns a Unix timestamp as a timestamp that uses the specified string as the time zone.	latency > 100 select from_unixtime (1494985275 , 'Asia / Shanghai ')
<code>localtime</code>	Returns the current time.	latency > 100 select localtime
<code>localtimestamp</code>	Returns the current timestamp.	latency > 100 select localtimestamp

Function	Description	Example
<code>now ()</code>	Functions the same as <code>current_timestamp</code> .	N/A
<code>to_unixtimestamp (timestamp)</code>	Returns a timestamp as a Unix timestamp.	<pre>* select to_unixtimestamp ('2017-05-17 09:45:00.848 Asia/Shanghai')</pre>

Time functions

MySQL time formats

Log Service supports the MySQL time formats, such as %a, %b, and %y.

Function	Description	Example
<code>date_format (timestamp, format)</code>	Formats a timestamp as a string by using the specified format.	<pre>latency > 100 select date_format (date_parse ('2017-05-17 09:45:00', '%Y-%m-%d %H:%i:%S'), '%Y-%m-%d')</pre>
<code>date_parse (string, format)</code>	Parses a string into a timestamp by using the specified format.	<pre>latency > 100 select date_format (date_parse (time, '%Y-%m-%d %H:%i:%S'), '%Y-%m-%d')</pre>

Table 8-1: Format description

Format	Description
%a	The abbreviation of a day in a week, such as Sun and Sat.
%b	The abbreviation of a month, such as Jan and Dec.
%c	The month, of the numeric type. Valid values: [1, 12].
%D	The day in a month with a suffix, such as 0th, 1st, 2nd, and 3rd.

Format	Description
%d	The day in a month, in decimal format. Valid values: [01, 31].
%e	The day in a month, in decimal format. Valid values: [1, 31].
%H	The hour in 24-hour format.
%h	The hour in 12-hour format.
%I	The hour in 12-hour format.
%i	The minutes, of the numeric type. Valid values: [00, 59].
%j	The day in a year. Valid values: [001, 366].
%k	The hour. Valid values: [0, 23].
%l	The hour. Valid values: [1, 12].
%M	The month. Valid values: January, February, March, April, May, June, July, August, September, October, November, and December.
%m	The month, of the numeric type. Valid values: [01, 12].
%p	The abbreviation of a period in 12-hour format. Valid values: AM and PM.
%r	The time in 12-hour format: hh : mm : ss AM / PM .
%S	The seconds. Valid values: [00, 59].
%s	The seconds. Valid values: [00, 59].
%T	The time in 24-hour format: hh : mm : ss .
%U	The week in a year, where Sunday is the first day of each week. Valid values: [00, 53].
%u	The week in a year, where Monday is the first day of each week. Valid values: [00, 53].
%V	The week in a year, where Sunday is the first day of each week. Valid values: [01, 53]. %V is used with %X.
%v	The week in a year, where Monday is the first day of each week. Valid values: [01, 53]. %v is used with %x.
%W	The day in a week. Valid values: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday.

Format	Description
%w	The day in a week, where Sunday is the first day of each week. Valid values: [0, 6].
%Y	The year in four-digit format.
%y	The year in two-digit format.
%%	The escape character %.

Truncation function

Log Service supports a truncation function, which can return a time truncated by the second, minute, hour, day, month, and year. The truncation function is applicable to time-based statistics.

- Function syntax:

```
date_trunc ( unit , x )
```

- Parameters:

The value of the x parameter can be a timestamp or Unix timestamp.

The following table lists the values of the unit parameter and the output results when the x parameter is set to `2001 - 08 - 22 03 : 04 : 05 . 000`.

Value	Output result
second	2001-08-22 03:04:05.000
minute	2001-08-22 03:04:00.000
hour	2001-08-22 03:00:00.000
day	2001-08-22 00:00:00.000
week	2001-08-20 00:00:00.000
month	2001-08-01 00:00:00.000
quarter	2001-07-01 00:00:00.000
year	2001-01-01 00:00:00.000

• Example:

The `date_trunc` function is applicable only to statistics at a fixed time interval. For statistics based on flexible time dimensions, for example, every 5 minutes, you need to use a `GROUP BY` clause according to the mathematical modulus method.

```
* | SELECT count ( 1 ) as pv , __time__ - __time__ % 300
    as minute5 group by minute5 limit 100
```

In the preceding statement, `% 300` indicates that modulus and truncation are performed every 5 minutes.

The following example shows how to use the `date_trunc` function:

```
*| select date_trunc ( ' minute ' , __time__ ) as t ,
        truncate ( avg ( latency ) ) ,
        current_date
    group by t
    order by t desc
    limit 60
```

Interval functions

Interval functions are used to perform interval-related calculation. For example, add or subtract an interval based on a date, or calculate the interval between two dates.

Function	Description	Example
<code>date_add (unit, value, timestamp)</code>	Adds an interval value of the unit type to a timestamp. To subtract an interval, use a negative value.	<code>date_add (' day ' , - 7 , ' 2018 - 08 - 09 00 : 00 : 00 ')</code> : indicates seven days before August 9.
<code>date_diff (unit, timestamp1, timestamp2)</code>	Returns the time difference between timestamp1 and timestamp2 expressed in terms of unit.	<code>date_diff (' day ' , ' 2018 - 08 - 02 00 : 00 : 00 ' , ' 2018 - 08 - 09 00 : 00 : 00 ')</code> = 7

The following table lists the values of the unit parameter supported by the interval functions.

Value	Description
millisecond	The millisecond.
second	The second.

Value	Description
minute	The minute.
hour	The hour.
day	The day.
week	The week.
month	The month.
quarter	The quarter, namely, three months.
year	The year.

Time series padding function

The time series padding function `time_series` is used to pad time series data with the missing time.

- Function format:

```
time_series(time_column, window, format, padding_data)
```

- The following table describes the parameters.

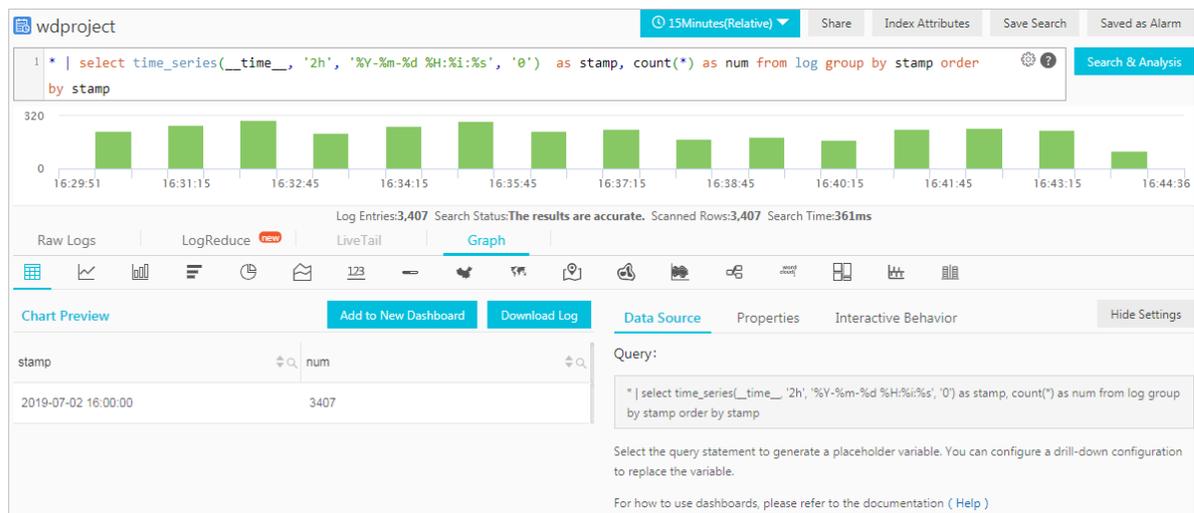
Parameter	Description
<i>time_column</i>	The sequence of time, such as the default time field <code>__time__</code> provided by Log Service. The value is of the long or timestamp type.
<i>window</i>	The size of the time window, which is composed of a number and a unit. Unit: s (seconds), m (minutes), H (hours), or d (days). For example, 2h, 5m, or 3d.
<i>format</i>	The MySQL time format, which is the final output format.
<i>padding_data</i>	The content to be added. Valid values: <ul style="list-style-type: none"> - 0: adds 0. - null: adds null. - last: adds the last value. - next: adds the next value. - avg: adds the average value of the last and next values.

• Example:

The following statement is used to format data every 2 hours:

```
* | select time_series ( __time__ , ' 2h ' , '% Y -% m -% d %
H :% i :% s ' , ' 0 ' ) as stamp , count ( * ) as num from
log group by stamp order by stamp
```

The following figure shows the output result.



8.9 URL functions

URL functions allow you to extract fields from standard URLs. A standard URL is as follows:

```
[ protocol :][// host [: port ]][ path ][? query ][# fragment ]
```

Common URL functions

Function	Description	Example	
		Input	Output result
url_extract_fragment (url)	Extracts the fragment identifier from a URL and returns the result of the varchar type.	* select url_extract_fragment (' https :// sls . console . aliyun . com /# / project / dashboard - demo / categoryList ')	/ project / dashboard - demo / categoryList

Function	Description	Example	
		Input	Output result
<code>url_extract_host (url)</code>	Extracts the host from a URL and returns the result of the varchar type.	<code>* select url_extract_host (' http :// www . aliyun . com / product / sls ')</code>	<code>www . aliyun . com</code>
<code>url_extract_parameter (url , name)</code>	Extracts the value of the name parameter in the query from a URL and returns the result of the varchar type.	<code>* select url_extract_parameter (' http :// www . aliyun . com / product / sls ? userid = testuser ', ' userid ')</code>	<code>testuser</code>
<code>url_extract_path (url)</code>	Extracts the path from a URL and returns the result of the varchar type.	<code>* select url_extract_path (' http :// www . aliyun . com / product / sls ? userid = testuser ')</code>	<code>/ product / sls</code>
<code>url_extract_port (url)</code>	Extracts the port number from a URL and returns the result of the bigint type.	<code>* select url_extract_port (' http :// www . aliyun . com : 80 / product / sls ? userid = testuser ')</code>	<code>80</code>
<code>url_extract_protocol (url)</code>	Extracts the protocol from a URL and returns the result of the varchar type.	<code>* select url_extract_protocol (' http :// www . aliyun . com : 80 / product / sls ? userid = testuser ')</code>	<code>http</code>
<code>url_extract_query (url)</code>	Extracts the query string from a URL and returns the result of the varchar type.	<code>* select url_extract_query (' http :// www . aliyun . com : 80 / product / sls ? userid = testuser ')</code>	<code>userid = testuser</code>

Function	Description	Example	
		Input	Output result
<code>url_encode (value)</code>	Encodes a URL.	<pre>* select url_encode (' http :// www . aliyun . com : 80 / product / sls ? userid = testuser ')</pre>	<pre>http % 3a % 2f % 2fwww . aliyun . com % 3a80 % 2fproduct % 2fsls % 3fuserid % 3dtestuser</pre>
<code>url_decode (value)</code>	Decodes a URL.	<pre>* select url_decode (' http % 3a % 2f % 2fwww . aliyun . com % 3a80 % 2fproduct % 2fsls % 3fuserid % 3dtestuser ')</pre>	<pre>http :// www . aliyun . com : 80 / product / sls ? userid = testuser</pre>

8.10 Regular expression functions

You can use a regular expression function to parse a string and return the needed substrings. .

The following table describes the common regular expression functions.

Function	Description	Example
<code>regexp_ext_ract_all (string , pattern)</code>	Returns all the substrings that match the regular expression in the specified string as a string array.	The returned result of <code>* SELECT regexp_ext_ract_all (' 5a 67b 890m ', '\ d +')</code> is <code>[' 5 ', ' 67 ', ' 890 ']</code> . The returned result of <code>* SELECT regexp_ext_ract_all (' 5a 67a 890m ', '(\ d +) a ')</code> is <code>[' 5a ', ' 67a ']</code> .
<code>regexp_ext_ract_all (string , pattern , group)</code>	Returns the part of the capturing group number that matches the regular expression in the specified string as a string array.	The returned result of <code>* SELECT regexp_ext_ract_all (' 5a 67a 890m ', '(\ d +) a ', 1)</code> is <code>[' 5 ', ' 67 ']</code> .
<code>regexp_ext_ract (string , pattern)</code>	Returns the first substring that matches the regular expression in the specified string.	The returned result of <code>* SELECT regexp_ext_ract (' 5a 67b 890m ', '\ d +')</code> is <code>' 5 '</code> .
<code>regexp_ext_ract (string , pattern , group)</code>	Returns the first substring in the part of the capturing group number that matches the regular expression in the specified string.	The returned result of <code>* SELECT regexp_ext_ract (' 5a 67b 890m ', '(\ d +)([a - z]+)', 2)</code> is <code>' a '</code> .
<code>regexp_lik_e (string , pattern)</code>	Determines whether the string matches the regular expression and returns a Boolean result. The regular expression is allowed to match a part of the string.	The returned result of <code>* SELECT regexp_lik_e (' 5a 67b 890m ', '\ d + m ')</code> is <code>true</code> .
<code>regexp_rep_lace (string , pattern , replacemen t)</code>	Replaces the part that matches the regular expression in the specified string with the replacement part.	The returned result of <code>* SELECT regexp_rep_lace (' 5a 67b 890m ', '\ d +', ' a ')</code> is <code>' aa ab am '</code> .

Function	Description	Example
<code>regexp_replace (string , pattern)</code>	Removes the part that matches the regular expression in the specified string. This function is equivalent to <code>regexp_replace (string , pattern , '')</code> .	The returned result of <code>* SELECT regexp_replace (' 5a 67b 890m ', '\ d +') is ' a b m '.</code>
<code>regexp_split (string , pattern)</code>	Splits the specified string into arrays by using the regular expression.	The returned result of <code>* SELECT regexp_split (' 5a 67b 890m ', '\ d +') is [' a ', ' b ', ' m '].</code>

8.11 JSON functions

JSON functions can parse a string as the JSON type and extract the fields in JSON. JSON mainly has the following two structures: map and array. If a string fails to be parsed as the JSON type, the returned value is null.

To split JSON into multiple lines, see [#unique_44](#).

Log Service supports the following common JSON functions.

Function	Description	Example
<code>json_parse (string)</code>	Converts a string to the JSON type.	<code>SELECT json_parse ('[1 , 2 , 3]')</code> returns a JSON array
<code>json_format (json)</code>	Converts the JSON type to a string.	<code>SELECT json_format (json_parse ('[1 , 2 , 3]'))</code> returns a string
<code>json_array_contains (json , value)</code>	Determines whether a JSON type value or string (whose content is a JSON array) contains a value or not.	<code>SELECT json_array_contains (json_parse ('[1 , 2 , 3]'), 2)</code> or <code>SELECT json_array_contains ('[1 , 2 , 3]', 2)</code>

Function	Description	Example
<code>json_array_get (json_array , index)</code>	The same as <code>json_array_contains</code> , which is used to obtain the element of a subscript of a JSON array.	<code>SELECT json_array_get ('[" a ", " b ", " c "]', 0)</code> returns <code>' a '</code>
<code>json_array_length (json)</code>	Returns the size of the JSON array.	<code>SELECT json_array_length ('[1 , 2 , 3]')</code> Returns <code>3</code>
<code>json_extract (json , json_path)</code>	Extracts the value from a JSON object. The JSON path syntax is similar to <code>\$. store . book [0]. title</code> . The returned result is a JSON object.	<code>SELECT json_extract (json , '\$. store . book ');</code>
<code>json_extract_scalar (json , json_path)</code>	Similar to <code>json_extract</code> , but returns a string.	-
<code>json_size (json , json_path)</code>	Obtains the size of the JSON object or array.	<code>Select json_size ('[1 , 2 , 3]')</code> returns <code>3</code>

8.12 Type conversion functions

Type conversion functions are used to convert the data type of a specified value or column in a query.

Fields in index attributes of Log Service can be configured as the long, double, text, or JSON type. Log Service also supports querying fields of various data types, including bigint, double, varchar, and timestamp. To query fields of a specific data type, you can use type conversion functions to convert the data type configured in index attributes into the data type used in the query.

Function format



Note:

We recommend that you use the `try_cast()` function if the logs contain dirty data. Otherwise, the entire query may fail due to the dirty data.

- Convert a column (field) or a constant value into the specified type in a query. If the value fails to be converted, the entire query is terminated.

```
cast([key|value] AS type)
```

- Convert a column (field) or a constant value into the specified type in a query. If the value fails to be converted, NULL is returned for the value, and the query continues.

```
try_cast([key|value] AS type)
```

Parameter	Description
key	The key of the log. If you set this parameter, all values of this parameter are converted into the specified type.
value	The constant value. If you set this parameter, the specified value is converted into the specified type.

Example

- To convert number 123 into a string in varchar format, run the following statement:

```
cast ( 123 AS varchar )
```

- To convert all values of the uid field into a string in varchar format, run the following statement:

```
cast ( uid AS varchar )
```

8.13 IP functions

IP recognition function can recognize whether the IP is an intranet IP or an Internet IP, and can determine the country, province, and city to which the IP belongs.

Function name	Meaning	Example
ip_to_doma in (ip)	Determines the domain in which the IP resides and whether the IP is an intranet IP or an Internet IP. The returned value is intranet or Internet.	SELECT ip_to_doma in (ip)
ip_to_coun try (ip)	Determines the country in which the IP resides.	SELECT ip_to_coun try (ip)

Function name	Meaning	Example
<code>ip_to_province (ip)</code>	Determines the province in which the IP resides.	<code>SELECT ip_to_province (ip)</code>
<code>ip_to_city (ip)</code>	Determines the city in which the IP resides.	<code>SELECT ip_to_city (ip)</code>
<code>ip_to_geo (ip)</code>	Determines the longitude and latitude of the city where IP is located, the result of the range is in the form of <code>latitude</code> and <code>longitude</code> .	<code>SELECT ip_to_geo (ip)</code>
<code>ip_to_city_geo (ip)</code>	Determines the longitude and latitude of the city where IP is located. Returns the latitude and longitude of the city, each city has only one latitude and longitude. The result of the range is in the form of <code>latitude</code> and <code>longitude</code> .	<code>SELECT ip_to_city_geo (ip)</code>
<code>ip_to_provider (ip)</code>	Obtains the network operator of the IP.	<code>SELECT ip_to_provider (ip)</code>
<code>ip_to_country (ip , ' en ')</code>	Determines the country where the IP is located and return the international code.	<code>SELECT ip_to_country (ip , ' en ')</code>
<code>ip_to_country_code (ip)</code>	Determine the country where the IP is located and return the international code.	<code>SELECT ip_to_country_code (ip)</code>
<code>ip_to_province (ip , ' en ')</code>	Determines the province where the IP is located, and returns the English province name or the Chinese alphabet.	<code>SELECT ip_to_province (ip , ' en ')</code>

Function name	Meaning	Example
<code>ip_to_city (ip , ' en ')</code>	Determines the city where IP is located, and returns the English city name or the Chinese alphabet.	<code>SELECT ip_to_city (ip , ' en ')</code>

Example

- Filter out the intranet access requests in the query and view the total number of requests

```
* | select count ( 1 ) where ip_to_domain ( ip ) != ' intranet '
```

- View the top 10 access provinces

```
* | SELECT count ( 1 ) as pv , ip_to_province ( ip ) as province GROUP BY province order by pv desc limit 10
```

Response result example:

```
[
  {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " Zhejiang province ",
    " pv ": " 4045 "
  }, {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " Shanghai city ",
    " pv ": " 3727 "
  }, {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " Beijing city ",
    " pv ": " 954 "
  }, {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " intranet IP ",
    " pv ": " 698 "
  }, {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " Guangdong Province ",
    " pv ": " 472 "
  }, {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " Fujian Province ",
    " pv ": " 71 "
  }, {
    " __source__ ": "",
    " __time__ ": " 1512353137 ",
    " province ": " United Arab Emirates ( UAE )",
    " pv ": " 52 "
  }
]
```

```

    }, {
      " __source__ ": "",
      " __time__ ": " 1512353137 ",
      " province ": " United States ",
      " pv ": " 43 "
    }, {
      " __source__ ": "",
      " __time__ ": " 1512353137 ",
      " province ": " Germany ",
      " pv ": " 26 "
    }, {
      " __source__ ": "",
      " __time__ ": " 1512353137 ",
      " province ": " Kuala Lumpur ",
      " pv ": " 26 "
    }
  ]

```

The preceding results include the intranet IP. Sometimes developers make tests from the intranet. To filter out these access requests, use the following analysis syntax.

- Filter out the intranet requests and view the top 10 network access provinces

```

* | SELECT count ( 1 ) as pv , ip_to_province ( ip ) as
  province WHERE ip_to_domain ( ip ) != ' intranet ' GROUP
  BY province ORDER BY pv desc limit 10

```

- Check the average response latency, the maximum response latency, and the request of the maximum latency in different countries

```

* | SELECT AVG ( latency ), MAX ( latency ), MAX_BY ( requestId
  , latency ), ip_to_country ( ip ) as country group by
  country limit 100

```

- View the average latency for different network operators

```

* | SELECT AVG ( latency ), ip_to_provider ( ip ) as
  provider group by provider limit 100

```

- View the latitude and longitude of the IP, and build a map

```

* | select count ( 1 ) as pv , ip_to_geo ( ip ) as geo
  group by geo order by pv desc

```

The returned format is:

pv	geo
100	35.3284,-80.7459

8.14 GROUP BY syntax

GROUP BY supports multiple columns and indicating the corresponding KEY by using the SELECT column alias.

Example:

```
method : PostLogsto reLogs | select avg ( latency ), projectName, date_trunc ( ' hour ', __time__ ) as hour group by projectName, hour
```

The alias `hour` represents the third SELECT column `date_trunc (' hour ', __time__)('hour',__time__)`. This kind of usage is very helpful for some very complicated queries.

GROUP BY supports GROUPING SETS, CUBE, and ROLLUP.

Example:

```
method : PostLogsto reLogs | select avg ( latency ) group by cube ( projectName, logstore )
method : PostLogsto reLogs | select avg ( latency ) group by GROUPING SETS ( ( projectName, logstore ), ( projectName, method ) )
method : PostLogsto reLogs | select avg ( latency ) group by rollup ( projectName, logstore )
```

Practical example

Perform GROUP BY according to time

Each log has a built-in time column `__time__`. When the statistical function of any column is activated, the statistics will be automatically made for the time column.

Use the `date_trunc` function to align the time column to hour, minute, day, month, and year. `date_trunc` accepts an aligned unit and a UNIX time or timestamp type column, such as `__time__`.

- Count and compute PV every hour or minute

```
* | SELECT count ( 1 ) as pv, date_trunc ( ' hour ', __time__ ) as hour group by hour order by hour limit 100
* | SELECT count ( 1 ) as pv, date_trunc ( ' minute ', __time__ ) as minute group by minute order by minute limit 100
```



Note:

limit limit 100 indicates to obtain 100 rows at most. If the LIMIT statement is not added, at most 10 rows of data can be obtained by default.

- Make statistics according to flexible time dimension. For example, make the statistics every five minutes. date_trunc can only make statistics every fixed time period. In this situation, perform GROUP BY according to the mathematical modulus method.

```
* | SELECT count ( 1 ) as pv , __time__ - __time__ % 300
   as minute5 group by minute5 limit 100
```

The %300 indicates to make the modulus and alignment every five minutes.

Extract non-agg column in GROUP BY

In the standard SQL, if you use the GROUP BY syntax, you can only select the original contents of the SELECT GROUP BY columns when you perform SELECT or you are not allowed to obtain the contents of non-GROUP BY columns when you perform aggregation calculation on any column.

For example, the following syntax is illegal. This is because b is the non-GROUP BY column and multiple rows of b are available when you perform GROUP BY according to a, the system does not know which row of output is to be selected.

```
* | select a , b , count ( c ) group by a
```

To achieve the preceding aim, use the arbitrary function to output b:

```
* | select a , arbitrary ( b ), count ( c ) group by a
```

8.15 Window functions

Window functions are used for cross-row calculation. Common SQL aggregate functions calculate the results of only one row or aggregate all rows into one row for calculation. Window functions support cross-row calculation and enter the calculation results in each row.

Syntax of window functions:

```
SELECT key1 , key2 , value ,
       rank ( ) OVER ( PARTITION BY key2
                      ORDER BY value DESC ) AS rnk
FROM orders
```

```
ORDER BY key1 , rnk
```

Core part is:

```
rank () OVER ( PARTITION BY KEY1 ORDER BY KEY2 DESC )
```

rank() is an aggregate function. You can use any function in analysis syntax or the function listed in this document. **PARTITION BY** indicates the buckets based on which values are calculated.

Special aggregate functions used in windows

Function name	Meaning
rank()	Sorts data based on a specific column in a window and returns the serial numbers in the window.
row_number()	Returns the row numbers in the window.
first_value(x)	Returns the first value in the window. Generally used to obtain the maximum value after values are sorted in the window.
last_value(x)	Opposite to first_value.
nth_value(x, offset)	Value of the No. offset row in xth column in the window.
lead(x,offset,default_value)	Value of the No. offset row after a certain row in xth column in the window. If that row does not exist, use the default_value.
lag(x,offset,default_value)	Value of the No. offset row before a certain row in xth column in the window . If that row does not exist, use the default_value.

Example

- Rank the salaries of employees in their respective departments

```
* | select department, personId, salary, rank() over
  ( PARTITION BY department order by salary desc ) as
  salary_rank order by department, salary_rank
```

Response results:

department	personId	salary	salary_rank
dev	john	9000	1
dev	Smith	8000	2
dev	Snow	7000	3
dev	Achilles	6000	4
Marketing	Blan Stark	9000	1
Marketing	Rob Stark	8000	2
Marketing	Sansa Stark	7000	3

- Calculate the salaries of employees as percentages in their respective departments

```
* | select department, personId, salary * 1.0 / sum (
  salary ) over ( PARTITION BY department ) as salary_pe
  rcentage
```

Response results:

department	personId	salary	salary_percentage
dev	john	9000	0.3
dev	Smith	8000	0.26
dev	Snow	7000	0.23
dev	Achilles	6000	0.2
Marketing	Blan Stark	9000	0.375
Marketing	Rob Stark	8000	0.333
Marketing	Sansa Stark	7000	0.29

- Calculate the daily UV increase over the previous day

```
* | select day, uv, uv * 1.0 / ( lag ( uv, 1, 0 ) over
  ( ) ) as diff_percentage from
```

```
select approx_distinct ( ip ) as uv , date_trunc ( ' day ',
__time__ ) as day from log group by day order by
day asc
```

Response results:

day	uv	diff_percentage
2017-12-01 00:00:00	100	null
2017-12-02 00:00:00	125	1.25
2017-12-03 00:00:00	150	1.2
2017-12-04 00:00:00	175	1.16
2017-12-05 00:00:00	200	1.14
2017-12-06 00:00:00	225	1.125
2017-12-07 00:00:00	250	1.11

8.16 HAVING syntax

The query and analysis function of Log Service supports the Having syntax of standard SQL, which is used together with the GROUP BY syntax to filter the GROUP BY results.

Format:

```
method : PostLogsto reLogs | select avg ( latency ), projectName
group by projectName HAVING avg ( latency ) > 100
```

Difference between HAVING and WHERE

HAVING is used to filter the aggregation and calculation results after performing GROUP BY. WHERE is used to filter the original data during the aggregation calculation.

Example

Calculate the average rainfall of each province whose temperature is greater than 10 °C and only display the provinces whose average rainfall is greater than 100 mL in the final result:

```
* | select avg ( rain ) , province where temperatur e > 10
group by province having avg ( rain ) > 100
```

8.17 ORDER BY syntax

ORDER BY is used to sort the output results. Currently, you can only sort the results by one column.

Syntax format:

```
order by Column name [ desc | asc ]
```

Example:

```
method : PostLogsto reLogs | select avg ( latency ) as
avg_latenc y , projectNam e group by projectNam e
HAVING avg ( latency ) > 5700000
order by avg_latenc y desc
```

8.18 LIMIT syntax

The LIMIT syntax is used to limit the number of rows returned in an SQL statement.

Syntax format

Log Service supports the following LIMIT syntax formats:

- Reads only the first N rows:

```
limit N
```

- Reads N rows starting from the Sth row:

```
limit S , N
```



Note:

- When you use the LIMIT syntax to paginate results, you get only the final results but not the intermediate results of the SQL query.

- The LIMIT syntax cannot be used in subqueries. For example, the following statement is not supported:

```
* | select count ( 1 ) from ( select distinct ( url ) from
  limit 0 , 1000 )
```

- When you use the LIMIT syntax for pagination, the offset value cannot exceed 1,000,000. That is, in the `limit S , N` statement, the sum of S and N cannot exceed 1,000,000, and the value of N cannot exceed 10,000.

Example

- To obtain only the first 100 rows of results, run the following statement:

```
* | select distinct ( url ) from log limit 100
```

- To obtain results from row 0 to row 999, totally 1,000 rows, run the following statement:

```
* | select distinct ( url ) from log limit 0 , 1000
```

- To obtain results from row 1,000 to row 1,999, totally 1,000 rows, run the following statement:

```
* | select distinct ( url ) from log limit 1000 , 1000
```

8.19 Case when and if branch syntax

Log Service supports CASE WHEN syntax to classify the continuous data. For example, extract the information from `http_user_agent` and classify the information into two types: Android and iOS.

```
SELECT
CASE
WHEN http_user_ agent like '% android %' then ' android '
WHEN http_user_ agent like '% ios %' then ' ios '
ELSE ' unknown ' END
as http_user_ agent ,
count ( 1 ) as pv
group by http_user_ agent
```

Example

- The ratio of requests with 200 as the computing status code to the total number of requests:

```
* | SELECT
sum (
CASE
```

```

WHEN status = 200 then 1
ELSE 0 end
) * 1.0 / count ( 1 ) as status_200 _percentag e

```

- **Make statistics of the distribution of different latency intervals**

```

* | SELECT `
CASE
WHEN latency < 10 then ' s10 '
WHEN latency < 100 then ' s100 '
WHEN latency < 1000 then ' s1000 '
WHEN latency < 10000 then ' s10000 '
else ' s_large ' end
as latency_sl ot ,
count ( 1 ) as pv
group by latency_sl ot

```

IF syntax

The if syntax is logically equivalent to the CASE WHEN syntax.

```

Case
  WHEN condition THEN true_value
  [ ELSE false_valu e ]
END

```

- **if(condition, true_value)**

If condition is true, the column true_value is returned, otherwise null.

- **if(condition, true_value, false_value)**

If condition is true, the column true_value is returned, otherwise the column false_value is returned.

Coalesce syntax

Coalesce returns the first non-null value for multiple columns.

```
Coalesce ( value1 , value2 [,...])
```

NULLIF syntax

If value1 and value2 are equal, null is returned, otherwise value1 is returned.

```
nullif ( value1 , value2 )
```

TRY syntax

The try syntax can catch some of the underlying exceptions, such as the 0 error, to return a null value.

```
try ( expression )
```

8.20 Nested subquery

For some complicated query scenarios, you can use the SQL nested query to meet the complicated requirements when the one-level SQL cannot meet the requirements.

The difference between nested subquery and non-nested query is that you need to specify the from condition in the SQL statement. Specifying the keyword `from log` in the query indicates to read original data from the logs.

Example:

```
* | select sum ( pv ) from
  (
    select count ( 1 ) as pv from log group by method
  )
```

8.21 Arrays

Statement	Meaning	Example
Subscript operator []	[] is used to obtain a certain element in the array.	-

Statement	Meaning	Example
Connection operator	is used to connect two arrays into one.	<pre>SELECT ARRAY [1] ARRAY [2]; -- [1 , 2] SELECT ARRAY [1] 2 ; -- [1 , 2] SELECT 2 ARRAY [1]; -- [2 , 1]</pre>
array_distinct	Obtain the distinct elements in the array by means of array deduplication.	-
array_intersect(x, y)	Obtain the intersection of arrays x and y.	-
array_union(x, y) → array	Obtain the union of arrays x and y.	-
array_except(x, y) → array	Obtain the subtraction of arrays x and y.	-
array_join(x, delimiter, null_replacement) → varchar	Join string arrays with the delimiter into a string and replace null values with null_replacement. <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin-top: 10px;">  Note: The maximum size of the returned result of this array_join is 1 KB. If the returned result exceeds 1 KB, the excessive part will be removed. </div>	-
array_max(x) → x	Obtain the maximum value in array x.	
array_min(x) → x	Obtain the minimum value in array x.	-

Statement	Meaning	Example
<code>array_position(x, element)</code> → bigint	Obtain the subscript of the element in array x. The subscript starts from 1. 0 is returned if no subscript is found.	-
<code>Array_remove(x, element)</code> -array	Remove the element from the array.	-
<code>array_sort(x)</code> → array	Sort the array and move null values to the end.	-
<code>cardinality(x)</code> → bigint	Obtain the array size.	-
<code>concat(array1, array2, ..., arrayN)</code> → array	Concatenate arrays.	-
<code>contains(x, element)</code> → boolean	Returns TRUE if array x contains the element.	-
This is a Lambda function. See <code>filter()</code> in Lambda.	Concatenate a two-dimensional array into a one-dimensional array.	-
<code>flatten(x)</code> → array	Concatenate a two-dimensional array into a one-dimensional array.	-
<code>reduce(array, initialState, inputFunction, outputFunction)</code> → x	See function <code>reduce()</code> in #unique_34 .	-
<code>reverse(x)</code> → array	Sort array x in reverse order.	-
<code>sequence(start, stop)</code> → array	Generate a sequence from start to stop and increment each step by 1.	-
<code>sequence(start, stop, step)</code> → array	Generate a sequence from start to stop and increment each step by the specified step value.	-

Statement	Meaning	Example
<code>sequence(start, stop, step)</code> → array	Generate a timestamp array from start to stop. Start and stop are of the timestamp type. Step is of the interval type, which can be from DAY to SECOND, and can also be YEAR or MONTH.	-
<code>shuffle(x)</code> → array	Shuffle the array.	-
<code>slice(x, start, length)</code> → array	Create a new array with length elements from start in array x.	-
<code>transform(array, function)</code> → array	See <code>transform()</code> in #unique_34 .	-
<code>zip(array1, array2[, ...])</code> → array	Merge multiple arrays. In the result, the Nth parameter in the Mth element is the Mth element in the Nth original array, which is equivalent to transposing multiple arrays.	<pre>SELECT zip (ARRAY [1 , 2], ARRAY [' 1b ', null , ' 3b ']); - [ROW (1 , ' 1b '), ROW (2 , null), ROW (null , ' 3b ')]</pre>
<code>zip_with(array1, array2, function)</code> → array	For more information, see #unique_34 <code>zip_with</code> .	-
<code>array_agg (key)</code>	<code>array_agg (key)</code> is an aggregation function that is used to aggregate all the content of the key column as an array to return.	* <code>select array_agg(key)</code>

8.22 Binary string functions

The binary string type `varbinary` is different from the string type `varchar`.

Statement	Description
Connection function <code> </code>	The result of <code>a b</code> is <code>ab</code> .
<code>length(binary)</code> → bigint	Returns the length in binary.

Statement	Description
<code>concat(binary1, ..., binaryN) → varbinary</code>	Connect the binary strings, which is equivalent to <code> </code> .
<code>to_base64(binary) → varchar</code>	Convert a binary string to a Base64 string.
<code>from_base64(string) → varbinary</code>	Convert a Base64 string to a binary string.
<code>to_base64url(binary) → varchar</code>	Convert a string to a URL-safe Base64 string.
<code>from_base64url(string) → varbinary</code>	Convert a URL-safe Base64 string to a binary string.
<code>to_hex(binary) → varchar</code>	Convert a binary string to a hexadecimal string.
<code>from_hex(string) → varbinary</code>	Convert a hexadecimal string to a binary string.
<code>to_big_endian_64(bigint) → varbinary</code>	Convert a number to a binary string in big endian mode.
<code>from_big_endian_64(binary) → bigint</code>	Convert a binary string in big endian mode to a number.
<code>md5(binary) → varbinary</code>	Calculate the MD5 value of a binary string.
<code>sha1(binary) → varbinary</code>	Calculate the SHA1 value of a binary string.
<code>sha256(binary) → varbinary</code>	Calculate the SHA256 hash value of a binary string.
<code>sha512(binary) → varbinary</code>	Calculate the SHA512 value of a binary string.
<code>xxhash64(binary) → varbinary</code>	Calculate the xxhash64 value of a binary string.

8.23 Bit operation

Statements	Description	Example
<code>bit_count(x, bits) → bigint</code>	Count the number of 1 in the binary expression of x.	<pre>SELECT bit_count (9 , 64); -- 2 SELECT bit_count (9 , 8); -- 2 SELECT bit_count (- 7 , 64); -- 62 SELECT bit_count (- 7 , 8); -- 6</pre>
<code>bitwise_and(x, y) → bigint</code>	Perform the AND operation on x and y in the binary form.	-
<code>bitwise_not(x) → bigint</code>	Calculate the opposite values of all bits of x in the binary form.	-
<code>bitwise_or(x, y) → bigint</code>	Perform the OR operation on x and y in the binary form.	-
<code>bitwise_xor(x, y) → bigint</code>	Perform the XOR operation on x and y in the binary form.	-

8.24 Interval-valued comparison and periodicity-valued comparison functions

Interval-valued comparison and periodicity-valued comparison functions are used to compare the calculation results of the current period with those of a specified previous period.

Function	Description	Example
<pre>compare (value , time_windo w)</pre>	<p>This function compares the value calculated for the current period with that calculated by time_window.</p> <p>The values are double or long type values. The unit of time_window is seconds . The return values are in array format.</p> <p>Possible return values include the current value, the value before time_window, and the ratio of the current value to the value before time_window.</p>	<pre>* select compare (pv , 86400) from (select count (1) as pv from log)</pre>
<pre>compare (value , time_windo w1 , time_windo w2)</pre>	<p>This function compares the current value with the values of periods before time_window1 and time_window2. The comparison results are in JSON array format, where the values must be placed in the following sequence : [current value, value before time_window1, value before time_windo w2, current value/value before time_window1, current value/value before time_window2].</p>	<pre>* select compare (pv , 86400 , 172800) from (select count (1) as pv from log)</pre>

Function	Description	Example
<pre>compare (value , time_windo w1 , time_windo w2 , time_windo w3)</pre>	<p>This function compares the current value with the values of periods before time_window1, time_window2 and time_window3. The comparison results are in JSON array format, where the values must be placed in the following sequence: [current value, value before time_window1, value before time_window2, value before time_windo w3, current value/Value before time_window1, current value/value before time_window2, current value/value before time_window3].</p>	<pre>* select compare (pv , 86400 , 172800 , 604800) from (select count (1) as pv from log)</pre>
<pre>compare_re sult (value , time_windo w)</pre>	<p>This function works similarly to compare (value , time_windo w). However, the return values are string type values in the format of " Current value (Increased percentage %)". The increased percentage value is rounded to two decimal places by default.</p>	<pre>* select compare_re sult (pv , 86400) from (select count (1) as pv from log)</pre>

Function	Description	Example
<pre>compare_result (value , time_window w1 , time_window w2)</pre>	<p>This function works similarly to <code>compare (value , time_window w1 , time_window w2)</code>. However, the return values are string type values in the format of " Current value (Increased percentage compared with the first period %) (Increased percentage compared with the second period)". The increased percentage values are rounded to two decimal places by default.</p>	<pre>* select compare_result (pv , 86400 , 172800) from (select count (1) as pv from log)</pre>

Function	Description	Example
<pre>ts_compare (value , time_windo w)</pre>	<p>This function compares the current value with the values of periods before <code>time_windo w1</code> and <code>time_windo w2</code>. The comparison results are in JSON array format, where the values must be placed in the following sequence: [current value, value before <code>time_window1</code>, current value/value before <code>time_window1</code>, unix timestamp at the start time of the previous period].</p> <p>This function is used to compare time series functions. This requires the GROUP BY operation to be included in SQL statements on the time column.</p>	<p>For example, * select</p> <pre>t , ts_compare (pv , 86400) as d from (select date_trunc (' minute ', __time__) as t , count (1) as pv from log group by t order by t) group by t indicates that the function compares the calculation result of every minute in the current period with that of every minute in the last period. <p>The comparison result is</p> <pre>d : [1251 . 0 , 1264 . 0 , 0 . 9897151898 734177 , 1539843780 . 0 , 1539757380 . 0] t : 2018 - 10 - 19 14 : 23 : 00 . 000 .</pre> </pre>

Examples

- Calculate the ratio of the PV in the current hour to that in the same time period as yesterday.

The start time is 2018-07-25 14:00:00, and the end time is 2018-07-25 15:00:00.

Statement for query and analysis:

```
* | select compare ( pv , 86400 ) from ( select count ( 1
  ) as pv from log )
```

where 86400 indicates that 86400 seconds are subtracted from the current period.

Return result:

```
[ 9 . 0 , 19 . 0 , 0 . 4736842105 2631579 ]
```

where:

- 9.0 is the PV value from 2018-07-25 14:00:00 to 2018-07-25 15:00:00.
- 19.0 is the PV value from 2018-07-24 14:00:00 to 2018-07-24 15:00:00.
- 0.47368421052631579 is the ratio of the PV value of the current period to that of a previous period.

If you want to expand the array into three columns of numbers, the analysis statement is:

```
* | select diff [ 1 ], diff [ 2 ], diff [ 3 ] from ( select
  compare ( pv , 86400 ) as diff from ( select count ( 1
  ) as pv from log )
```

- Calculate the ratio of the PV in every minute of the current hour to that in the same time period as yesterday, and display the results in a line chart.

1. Calculate the ratio of the PV in every minute of the current hour to that in the same time period as yesterday. The start time is 2018-07-25 14:00:00, and the end time is 2018-07-25 15:00:00.

Statement for query and analysis:

```
*| select t , compare ( pv , 86400 ) as diff from (
  select count ( 1 ) as pv , date_format ( from_unixt ime
```

```
( __time__ ), '% H :% i ' ) as t from log group by t
) group by t order by t
```

Return results:

t	diff
14:00	[9520.0,7606.0,1.2516434393899554]
14:01	[8596.0,8553.0,1.0050274757395066]
14:02	[8722.0,8435.0,1.0340248962655603]
14:03	[7499.0,5912.0,1.2684370771312586]

where t indicates the time in the format of Hour : Minute . The content of the diff column is an array containing the following:

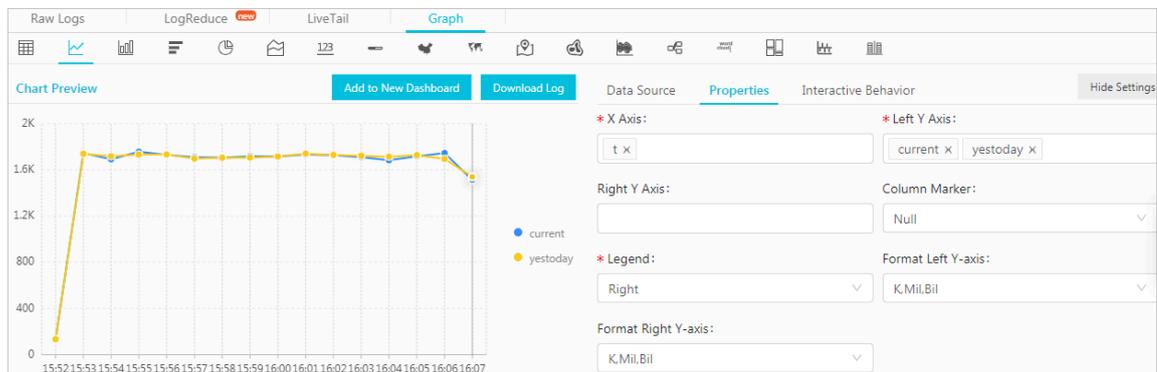
- The PV value of the current period.
- The PV value of the previous period.
- The ratio of the PV value in the current period to that in the previous period.

2. To show the query results in a line chart, use the following statement:

```
*| select t , diff [ 1 ] as current , diff [ 2 ] as
yestoday , diff [ 3 ] as percentage from ( select t ,
compare ( pv , 86400 ) as diff from ( select count ( 1
) as pv , date_format t ( from_unixtime ( __time__ ), '% H
:% i ' ) as t from log group by t ) group by t
order by t )
```

The two lines indicate the PV values of today and yesterday.

Figure 8-1: Line chart



8.25 Comparison functions and operators

Comparison functions and operators

A comparison operation compares the values of two parameters, which can be used for any comparable types, such as int, bigint, double, and text.

Comparison operators

A comparison operator is used to compare two parameter values. During the comparison, if the logic is true, TRUE is returned. Otherwise, FALSE is returned.

Operator	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
<>	Not equal to
!=	Not equal to

Range operator BETWEEN

BETWEEN is used to determine whether a parameter value is between the values of two other parameters. The range is a closed interval.

- If the logic is true, TRUE is returned. Otherwise, FALSE is returned.

Example: `SELECT 3 BETWEEN 2 AND 6 ;`. The logic is true, and TRUE is returned.

The preceding example is equivalent to `SELECT 3 >= 2 AND 3 <= 6 ;`.

- BETWEEN can follow NOT to determine the opposite logic.

Example: `SELECT 3 NOT BETWEEN 2 AND 6 ;`. The logic is false, and FALSE is returned.

The preceding example is equivalent to `SELECT 3 < 2 OR 3 > 6 ;`.

- If the value of any parameter is NULL, NULL is returned.

IS NULL and IS NOT NULL

These operators are used to determine whether a parameter value is NULL.

IS DISTINCT FROM and IS NOT DISTINCT FROM

Similar to determining whether two values are equal or not, but these operators can determine whether a NULL value exists.

Example:

```
SELECT NULL IS DISTINCT FROM NULL ; -- false
SELECT NULL IS NOT DISTINCT FROM NULL ; -- true
```

As described in the following table, the DISTINCT operator can be used to compare parameter values in most cases.

a	b	a = b	a <> b	a DISTINCT b	a NOT DISTINCT b
1	1	TRUE	FALSE	FALSE	TRUE
1	2	FALSE	TRUE	TRUE	FALSE
1	NULL	NULL	NULL	TRUE	FALSE
NULL	NULL	NULL	NULL	FALSE	TRUE

GREATEST and LEAST

These operators are used to obtain the maximum or minimum values among multiple columns.

Example:

```
select greatest ( 1 , 2 , 3 ) ; -- 3 is returned .
```

Comparison conditions: ALL, ANY, and SOME

Comparison conditions are used to determine whether a parameter meets the specified conditions.

- ALL is used to determine whether a parameter meets all the conditions. If the logic is true, TRUE is returned. Otherwise, FALSE is returned.
- ANY is used to determine whether a parameter meets any of the conditions. If the logic is true, TRUE is returned. Otherwise, FALSE is returned.
- Same as ANY, SOME is used to determine whether a parameter meets any of the conditions.

- ALL, ANY, and SOME must immediately follow the comparison operators.

comparison and determination in many cases.

Expression	Meaning
A = ALL (…)	TRUE is returned when A is equal to all values.
A <> ALL (…)	TRUE is returned when A is not equal to all values.
A < ALL (…)	TRUE is returned when A is less than all values.
A = ANY (…)	TRUE is returned when A is equal to any value, which is equivalent to A IN (…).
A <> ANY (…)	TRUE is returned when A is not equal to any value.
A < ANY (…)	TRUE is returned when A is less than the maximum value.

Example:

```
SELECT 'hello' = ANY ( VALUES 'hello', 'world '); -- true
SELECT 21 < ALL ( VALUES 19, 20, 21 ); -- false
SELECT 42 >= SOME ( SELECT 41 UNION ALL SELECT 42
UNION ALL SELECT 43 ); -- true
```

8.26 Lambda functions

Lambda expressions

Lambda expressions are written with `->`.

Example:

```
x -> x + 1
(x, y) -> x + y
x -> regexp_like ( x, 'a+')
x -> x [ 1 ] / x [ 2 ]
x -> IF ( x > 0, x, - x )
x -> COALESCE ( x, 0 )
x -> CAST ( x AS JSON )
x -> x + TRY ( 1 / 0 )
```

Most MySQL expressions can be used in Lambda.

`filter(array<T>, function<T, boolean>) → ARRAY<T>`

Filter data from an array and obtain only elements that the function returns TRUE.

Example:

```

SELECT filter ( ARRAY [], x -> true ); -- []
SELECT filter ( ARRAY [ 5 , - 6 , NULL , 7 ], x -> x > 0 );
-- [ 5 , 7 ]
SELECT filter ( ARRAY [ 5 , NULL , 7 , NULL ], x -> x IS
NOT NULL ); -- [ 5 , 7 ]

```

map_filter(map<K, V>, function<K, V, boolean>) → MAP<K,V>

Filter data from a map and obtain only element pairs that the function returns TRUE.

Example:

```

SELECT map_filter ( MAP ( ARRAY [], ARRAY []), ( k , v ) -> true
); -- {}
SELECT map_filter ( MAP ( ARRAY [ 10 , 20 , 30 ], ARRAY [' a ',
NULL , ' c ']), ( k , v ) -> v IS NOT NULL ); -- { 10 -> a
, 30 -> c }
SELECT map_filter ( MAP ( ARRAY [' k1 ', ' k2 ', ' k3 '], ARRAY [
20 , 3 , 15 ]), ( k , v ) -> v > 10 ); -- { k1 -> 20 , k3 -
> 15 }

```

reduce(array<T>, initialState S, inputFunction<S, T, S>, outputFunction<S, R>) → R

The reduce() function traverses each element in the array in turn from the initial state, calculates inputFunction(S,T) based on the state S, and generates a new state. It finally applies outputFunction to convert the final state S to the output result R.

1. Initial state S.
2. Traverse each element T.
3. Calculate inputFunction(S,T) and generate the new state S.
4. Repeat steps 2 and 3 until the last element is traversed and has the new state generated.
5. Uses the final state S to obtain the final output result R.

Example:

```

SELECT reduce ( ARRAY [], 0 , ( s , x ) -> s + x , s -> s
); -- 0
SELECT reduce ( ARRAY [ 5 , 20 , 50 ], 0 , ( s , x ) -> s +
x , s -> s ); -- 75
SELECT reduce ( ARRAY [ 5 , 20 , NULL , 50 ], 0 , ( s , x ) -
> s + x , s -> s ); -- NULL
SELECT reduce ( ARRAY [ 5 , 20 , NULL , 50 ], 0 , ( s , x ) -
> s + COALESCE ( x , 0 ), s -> s ); -- 75
SELECT reduce ( ARRAY [ 5 , 20 , NULL , 50 ], 0 , ( s , x ) -
> IF ( x IS NULL , s , s + x ), s -> s ); -- 75
SELECT reduce ( ARRAY [ 2147483647 , 1 ], CAST ( 0 AS
BIGINT ), ( s , x ) -> s + x , s -> s ); -- 2147483648
SELECT reduce ( ARRAY [ 5 , 6 , 10 , 20 ], -- calculates
arithmetic average : 10 . 25

```

```

CAST ( ROW ( 0 . 0 , 0 ) AS ROW ( sum DOUBLE ,
count INTEGER )),
(s , x ) -> CAST ( ROW ( x + s . sum , s . count
+ 1 ) AS ROW ( sum DOUBLE , count INTEGER )),
s -> IF ( s . count = 0 , NULL , s . sum / s .
count ));

```

transform(array<T>, function<T, U>) → ARRAY<U>

Calls function for each element in the array to generate the new result U.

Example:

```

SELECT transform ( ARRAY [], x -> x + 1 ); -- []
SELECT transform ( ARRAY [ 5 , 6 ], x -> x + 1 ); -- [ 6 ,
7 ] Increment each element by 1 .
SELECT transform ( ARRAY [ 5 , NULL , 6 ], x -> COALESCE ( x
, 0 ) + 1 ); -- [ 6 , 1 , 7 ]
SELECT transform ( ARRAY [ ' x ' , ' abc ' , ' z ' ], x -> x || '
0 ' ); -- [ ' x0 ' , ' abc0 ' , ' z0 ' ]
SELECT transform ( ARRAY [ ARRAY [ 1 , NULL , 2 ], ARRAY [ 3
, NULL ] ], a -> filter ( a , x -> x IS NOT NULL )); --
[[ 1 , 2 ], [ 3 ]]

```

transform_keys(map<K1, V>, function<K1, V, K2>) → MAP<K2, V>

Apply the function for each key in the map in turn to generate a new key.

Example:

```

SELECT transform_ keys ( MAP ( ARRAY [], ARRAY []), ( k , v ) -
> k + 1 ); -- {}
SELECT transform_ keys ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY [ '
a ' , ' b ' , ' c ' ]), ( k , v ) -> k + 1 ); -- { 2 -> a , 3 -
> b , 4 -> c } Increment each key by 1 .
SELECT transform_ keys ( MAP ( ARRAY [ ' a ' , ' b ' , ' c ' ],
ARRAY [ 1 , 2 , 3 ]), ( k , v ) -> v * v ); -- { 1 -> 1 , 4
-> 2 , 9 -> 3 }
SELECT transform_ keys ( MAP ( ARRAY [ ' a ' , ' b ' ], ARRAY [ 1
, 2 ]), ( k , v ) -> k || CAST ( v as VARCHAR )); -- { a1 -
> 1 , b2 -> 2 }
SELECT transform_ keys ( MAP ( ARRAY [ 1 , 2 ], ARRAY [ 1 . 0
, 1 . 4 ]), -- { one -> 1 . 0 , two -> 1 . 4 }
(k , v ) -> MAP ( ARRAY [ 1 , 2 ], ARRAY [ '
one ' , ' two ' ])[ k ]);

```

transform_values(map<K, V1>, function<K, V1, V2>) → MAP<K, V2>

Apply the function for all values in the map, convert V1 to V2, and generate a new map < K , V2 >.

```

SELECT transform_ values ( MAP ( ARRAY [], ARRAY []), ( k , v )
-> v + 1 ); -- {}
SELECT transform_ values ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY [
10 , 20 , 30 ]), ( k , v ) -> v + 1 ); -- { 1 -> 11 , 2 ->
22 , 3 -> 33 }

```

```

SELECT transform_values ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY
[' a ', ' b ', ' c ']), ( k , v ) -> k * k ); -- { 1 -> 1 , 2
-> 4 , 3 -> 9 }
SELECT transform_values ( MAP ( ARRAY [ ' a ', ' b '], ARRAY [
1 , 2 ]), ( k , v ) -> k || CAST ( v AS VARCHAR )); -- { a
-> a1 , b -> b2 }
SELECT transform_values ( MAP ( ARRAY [ 1 , 2 ], ARRAY [ 1 .
0 , 1 . 4 ]), -- { 1 -> one_1 . 0 , 2 -> two_1 . 4 }
(k , v ) -> MAP ( ARRAY [ 1 , 2 ], ARRAY
[' one ', ' two '])[ k ] || ' _ ' || CAST ( v AS VARCHAR ));

```

zip_with(array<T>, array<U>, function<T, U, R>) → array<R>

Merge two arrays, and specify the elements of the newly generated array by using the function. Element T in the first array and element U in the second array are used to generate the new result R.

Example:

```

SELECT zip_with ( ARRAY [ 1 , 3 , 5 ], ARRAY [ ' a ', ' b ', ' c
' ], ( x , y ) -> ( y , x )); -- Transpose the elements of
the two arrays to generate a new array . Result : [
ROW ( ' a ', 1 ), ROW ( ' b ', 3 ), ROW ( ' c ', 5 )]
SELECT zip_with ( ARRAY [ 1 , 2 ], ARRAY [ 3 , 4 ], ( x , y ) -
> x + y ); -- Result : [ 4 , 6 ]
SELECT zip_with ( ARRAY [ ' a ', ' b ', ' c '], ARRAY [ ' d ', ' e
', ' f '], ( x , y ) -> concat ( x , y )); Concatenate the
elements of the two arrays to generate a new string
. Result : [ ' ad ', ' be ', ' cf ' ]

```

map_zip_with(map<K, V1>, map<K, V2>, function<K, V1, V2, V3>) → map<K, V3>

Merge two maps, use values V1 and V2 to generate V3 based on each key, and generate a new map< K , V3 >.

```

SELECT map_zip_wi th ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY [ ' a ',
' b ', ' c ']),
MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY [ ' d ', ' e
', ' f ']),
(k , v1 , v2 ) -> concat (
v1 , v2 )); Merge values which have the same map keys
. -- { 1 -> ad , 2 -> be , 3 -> cf }
SELECT map_zip_wi th ( MAP ( ARRAY [ ' k1 ', ' k2 '], ARRAY [ 1 ,
2 ]),
MAP ( ARRAY [ ' k2 ', ' k3 '], ARRAY [ 4 , 9 ]),
(k , v1 , v2 ) -> ( v1 , v2 )); Generate an
array by using the two values . -- { k1 -> ROW ( 1 ,
null ), k2 -> ROW ( 2 , 4 ), k3 -> ROW ( null , 9 )}
SELECT map_zip_wi th ( MAP ( ARRAY [ ' a ', ' b ', ' c '], ARRAY [
1 , 8 , 27 ]),
MAP ( ARRAY [ ' a ', ' b ', ' c '], ARRAY [ 1 ,
2 , 3 ]),
(k , v1 , v2 ) -> k || CAST ( v1 / v2 AS
VARCHAR )); -- Concatenate the key values and division

```

```
results of the two values -- { a -> a1 , b -> b4 , c
-> c9 }
```

8.27 Logical functions

Logical operators

Table 8-2: Logical operators

Operator	Description	Example
AND	Returns TRUE only when both the left and right operands are TRUE.	a AND b
OR	Returns TRUE if either the left or right operand is TRUE.	a OR b
NOT	Returns TRUE only when the right operand is FALSE.	NOT a

NULL involved in logical operation

The following table lists the true values when the values of a and b are TRUE, FALSE, and NULL respectively.

Table 8-3: Truth Table 1

a	b	a AND b	A or B
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

Table 8-4: Truth Table 2

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

8.28 Column alias

In the SQL standard, the column name must be consisted of English letters, numbers, and underlines (_) and start with an English letter.

If a column name (for example, User-Agent) that does not conform to the SQL standard is configured in the log collection configuration, give the column an alias used for query on the page of configuring statistical properties. The alias is only used for the SQL statistics. In the underlying storage, the column name is the original name. Use the original column name to query.

Besides, you can give the column an alias to replace the original column name for query when the column name is very long.

Table 8-5: Alias Example:

Original column name	Alias
User-Agent	ua
User.Agent	ua
123	col
abceefghijklmnopqrstuvw	a

8.29 Geospatial functions

Geospatial concept

Geospatial functions support the geometries in the Well-Known Text (WKT) format.

Table 8-6: Geometry format

Geometry	Well-maid text (WKT) Format
Point	POINT (0 0)
Line string	LINESTRING (0 0 , 1 1 , 1 2)
Polygon	Polygon
Multi-point	MULTIPOINT (0 0 , 1 2)
Multi-line string	MULTILINESTRING ((0 0 , 1 1 , 1 2), (2 3 , 3 2 , 5 4))
Multi-polygon	MULTIPOLYGON (((0 0 , 4 0 , 4 4 , 0 4 , 0 0)), (1 1 , 2 1 , 2 2 , 1 2 , 1 1)), ((- 1 - 1 , - 1 - 2 , - 2 - 2 , - 2 - 1 , - 1 - 1)))
Geometry collection	GEOMETRYCOLLECTION (POINT (2 3), LINESTRING (2 3 , 3 4))

Constructors

Table 8-7: Constructors Description

Function	Description
ST_Point(double, double) → Point	Returns a geometry type point with the given coordinate values.
ST_LineFromText(varchar) → LineString	Returns a geometry type line string from WKT representation.
ST_Polygon(varchar) → Polygon	Returns a geometry type polygon from WKT representation.
ST_GeometryFromText(varchar) → Geometry	Returns a geometry type object from WKT representation.
ST_AsText(Geometry) → varchar	Returns the WKT representation of the geometry.

Operations

Function	Description
ST_Boundary(Geometry) → Geometry	Returns the closure of the combinatorial boundary of this geometry.
ST_Buffer(Geometry, distance) → Geometry	Returns the geometry that represents all points whose distance from the specified geometry is less than or equal to the specified distance.
ST_Difference(Geometry, Geometry) → Geometry	Returns the geometry value that represents the point set difference of the given geometries.
ST_Envelope(Geometry) → Geometry	Returns the bounding rectangular polygon of a geometry.
ST_ExteriorRing(Geometry) → Geometry	Returns a line string representing the exterior ring of the input polygon.
ST_Intersection(Geometry, Geometry) → Geometry	Returns the geometry value that represents the point set intersection of two geometries.
ST_SymDifference(Geometry, Geometry) → Geometry	Returns the geometry value that represents the point set symmetric difference of two geometries.

Relationship tests

Function	Description
ST_Contains(Geometry, Geometry) → boolean	Returns true if and only if no points of the second geometry lie in the exterior of the first geometry, and at least one point of the interior of the first geometry lies in the interior of the second geometry . Returns false if the two geometries at least share an interior point.
ST_Crosses(Geometry, Geometry) → boolean	Returns true if the supplied geometries have some, but not all, interior points in common.
ST_Disjoint(Geometry, Geometry) → boolean	Returns true if the given geometries do not spatially intersect.
ST_Equals(Geometry, Geometry) → boolean	Returns true if the given geometries represent the same geometry.

Function	Description
ST_Intersects(Geometry, Geometry) → boolean	Returns true if the given geometries spatially intersect in two dimensions (share any portion of space) and false if they do not (they are disjoint).
ST_Overlaps(Geometry, Geometry) → boolean	Returns true if the given geometries share space, are of the same dimension, but are not completely contained by each other.
ST_Relate(Geometry, Geometry) → boolean	Returns true if the first geometry is spatially related to the second geometry.
ST_Touches(Geometry, Geometry) → boolean	Geometry) → boolean Returns true if the given geometries have at least one point in common, but their interiors do not intersect.
ST_Within(Geometry, Geometry) → boolean	Returns true if the first geometry is completely inside the second geometry. Returns false if the two geometries have at least one point in common.

Accessors

Function	Description
ST_Area(Geometry) → double	Returns the area of a polygon using Euclidean measurement on a two dimensional plane in projected units.
ST_Centroid(Geometry) → Geometry	Returns the point value that is the mathematical centroid of a geometry.
ST_CoordDim(Geometry) → bigint	Returns the coordinate dimension of the geometry.
ST_Dimension(Geometry) → bigint	Returns the inherent dimension of this geometry, which must be less than or equal to the coordinate dimension.
ST_Distance(Geometry, Geometry) → double	Returns the minimum distance between two geometries.
ST_IsClosed(Geometry) → boolean	Returns true if the start and end points of the line string are coincident.
ST_IsEmpty(Geometry) → boolean	Returns true if this geometry is an empty geometry collection, polygon, or point.

Function	Description
ST_IsRing(Geometry) → boolean	Returns true if and only if the line is closed and simple.
ST_Length(Geometry) → double	Returns the length of a line string or multi-line string using Euclidean measurement on a two dimensional plane (based on spatial ref) in projected units.
ST_XMax(Geometry) → double	Returns X maxima of a bounding box of a geometry.
ST_YMax(Geometry) → double	Returns Y maxima of a bounding box of a geometry.
T_XMin(Geometry) → double	Returns X minima of a bounding box of a geometry.
ST_YMin(Geometry) → double	Returns Y minima of a bounding box of a geometry.
ST_StartPoint(Geometry) → point	Returns the first point of a line string geometry.
ST_EndPoint(Geometry) → point	Returns the last point of a line string geometry.
ST_X(Point) → double	Returns the X coordinate of the point.
ST_Y(Point) → double	Returns the Y coordinate of the point.
ST_NumPoints(Geometry) → bigint	Returns the number of points in a geometry.
ST_NumInteriorRing(Geometry) → bigint	Returns the number of interior rings of a polygon.

8.30 Geo functions

For more information about functions that determine the country, province, city, ISP, and the longitude and latitude of specified IP addresses, see [#unique_28](#).

Table 8-8: Geo functions

Function	Description	Example
geohash(string)	Returns the geohash value of the specified geographical coordinate. The geographical coordinate is represented by a string in the format of "latitude, longitude" (the values for latitude and longitude are separated by a comma).	<pre>select geohash (' 34 . 1 , 120 . 6 ')= ' wwjcbrdnzs '</pre>
geohash(lat, lon)	Returns the geohash value of the specified geographical coordinate. The geographical coordinate is represented by two separate parameters that indicate the latitude and longitude.	<pre>select geohash (34 . 1 , 120 . 6)= ' wwjcbrdnzs '</pre>

8.31 Join syntax

Join is used for combining fields from multiple tables. Besides Join for a single Logstore, Log Service also supports Join for Logstore and RDS, and for several Logstores. This document describes how to use the Join function between Logstores.

Procedure

1. [Download](#) the latest version of Python SDK.
2. Use the GetProjectLogs interface for query.

SDK sample

```
/usr/bin/env python
# encoding: utf-8
import time, sys, os
from aliyun.log.logexception import logexception
from aliyun.log.logitem import LogItem
from aliyun.log.logclient import LogClient
from aliyun.log.getlogsrequest import getlogsrequest
from aliyun.log.getprojectlogsrequest import
GetProjectLogsRequest
from aliyun.log.putlogsrequest import PutLogsRequest
from aliyun.log.listtopicsrequest import ListTopicsRequest
```

```

from aliyun . log . listlogsto resrequest import ListLogsto
resRequest
from aliyun . log . gethistogr amsrequest import GetHistogr
amsRequest
from aliyun . log . index_conf ig import *
from aliyun . log . logtail_co nfig_detai l import *
from aliyun . log . machine_gr oup_detail import *
from aliyun . log . acl_config import *
if __name__ == '__main__':
    token = None
    endpoint = " http:// cn - hangzhou . log . aliyuncs . com "
    accessKeyId = ' LTAIvKy7U '
    accessKey = ' 6gXLNTLyCf dsfwrewrfh dskfdfsuiw u '
    client = LogClient ( endpoint , accessKeyId , accessKey ,
token )
    logstore = " meta "
    # In the query statement , specify two Logstores .
For each Logstore specify its time range and the
key
    req = GetProject LogsReques t ( project ," select count
( 1 ) from sls_operat ion_log s join meta m on s .
__date__ >' 2018 - 04 - 10 00 : 00 : 00 ' and s . __date__ <
' 2018 - 04 - 11 00 : 00 : 00 ' and m . __date__ >' 2018 - 04 -
23 00 : 00 : 00 ' and m . __date__ <' 2018 - 04 - 24 00 : 00
: 00 ' and s . projectid = cast ( m . ikey as varchar )");
    Res = client . Fig ( req )
    res . log_print ();
    exit ( 0 )

```

8.32 UNNEST function

Scenario

Columns of log data are usually of a primitive data type, such as string or number. In certain scenarios with more complex data structures, the columns of log data may involve complex data types, such as arrays, maps, and JSON objects. The UNNEST function can be used to enumerate an array of complex data into rows for easier querying and analysis.

Example:

```

__source__ : 1 . 1 . 1 . 1
__tag__ : __hostname__ : vm - req - 1701032323 16569850 -
tianchi111 932 . tc
__topic__ : TestTopic_ 4
array_colu mn : [ 1 , 2 , 3 ]
double_col umn : 1 . 23
map_column : { " a " : 1 , " b " : 2 }
text_colum n : Product

```

The `array_colu mn` field is of array type. To obtain an aggregate of all elements in an `array_colu mn` array, you must enumerate all elements of the array for each row.

UNNEST function

Syntax	Description
<code>unnest (array) as table_alias (column_name)</code>	Splits the specified array into multiple rows. Each row has a name specified in the <code>column_name</code> column.
<code>unnest (map) as table (key_name , value_name)</code>	Splits the specified map into multiple rows. Each key has a key name specified in the <code>key_name</code> column, and a value specified in the <code>value_name</code> column.

**Note:**

Note: The UNNEST function takes only arrays or maps. If you specify a string, the string must represent a JSON object. Then you can parse the string into an array or map by using the `cast (json_parse (array_column) as array (bigint))` syntax.

Enumerate the elements of an array

Split an array into multiple rows using SQL:

```
* | select array_column, a from log, unnest ( cast (
  json_parse ( array_column ) as array ( bigint ) ) ) as t ( a
)
```

The UNNEST function splits the array into multiple rows and stores the rows in a new table referenced as `t`, with each column referenced as `a`.

- Calculate the sum of the elements in the array:

```
* | select sum ( a ) from log , unnest ( cast ( json_parse
( array_colu mn ) as array ( bigint ) ) ) as t ( a )
```

- Perform a GROUP BY operation on the array by a (each element of the array):

```
* | select a , count ( 1 ) from log , unnest ( cast (
json_parse ( array_colu mn ) as array ( bigint ) ) ) as t (
a ) group by a
```

Enumerate the elements of the map

- Enumerate the elements of the map:

```
* | select map_column , a , b from log , unnest ( cast (
json_parse ( map_column ) as map ( varchar , bigint ) ) ) as
t ( a , b )
```

- Perform a GROUP BY operation on the map by key:

```
* | select key , sum ( value ) from log , unnest ( cast (
json_parse ( map_column ) as map ( varchar , bigint ) ) ) as
t ( key , value ) GROUP BY key
```

Visualize histogram, numeric_histogram query results

- histogram

The histogram function is similar to the COUNT GROUP BY syntax. For more information about the syntax, see [#unique_18](#).

The histogram function usually returns a JSON string that cannot be directly visualized. The following is an example:

```
* | select histogram ( method )
```

You can use the UNNEST function to split JSON data into multiple rows. The following is an example:

```
* | select key , value from ( select histogram ( method )
as his from log ) , unnest ( his ) as t ( key , value )
```

- Numeric_histogram

The `numeric_histogram` syntax sorts values in a numeric column into multiple bins. This operation is similar to performing a `GROUP BY` operation on the numeric column. For more information about the syntax, see [#unique_19](#).

```
* | select numeric_histogram ( 10 , Latency )
```

Use the following query statement to visualize the result:

```
* | select key , value from ( select numeric_histogram ( 10
, Latency ) as his from log ) , unnest ( his ) as t (
key , value )
```

8.33 Phone number functions

Phone number functions are used to query the attributes of phone numbers that are registered in Mainland China.

Functions

Function name	Description	Example
<code>mobile_province</code>	This function is used to query the provincial attribute of a phone number. The phone number must be of the numeric type. If the phone number is of the string type, you can use <code>try_cast</code> to convert the type to numeric.	<pre>* select mobile_province (12345678)</pre> <pre>* select mobile_province (try_cast (' 12345678 ' as bigint))</pre>
<code>mobile_city</code>	This function is used to query the urban attribute of a phone number. The phone number must be of the numeric type. If the phone number is of the string type, you can use <code>try_cast</code> to convert the type to numeric.	<pre>* select mobile_city (12345678)</pre> <pre>* select mobile_city (try_cast (' 12345678 ' as bigint))</pre>

Function name	Description	Example
mobile_carrier	This function is used to query the telecom operator to which a phone number belongs. The phone number must be of the numeric type. If the phone number is of the string type, you can use <code>try_cast</code> to convert the type to numeric.	<pre>* select mobile_carrier (12345678)</pre> <pre>* select mobile_carrier (try_cast ('12345678 ' as bigint))</pre>

Scenarios

- Query phone number attributes and generate a report.

Assume that an e-commerce company collects logs about the activities of its customers. The company can extract the fields involving phone numbers, and then collects the attributes of the phone numbers by using the following query statement:

```
SELECT mobile_city ( try_cast ( " mobile " as bigint )) as
" city ", mobile_province ( try_cast ( " mobile " as bigint
)) as " province ", count ( 1 ) as " number of requests
" group by " province ", " city " order by " number of
request " desc limit 100
```

In the statement, `mobile` is used as the input field of the `mobile_city` and `mobile_province` functions to show the provinces and cities of the phone numbers. The returned information from the preceding query is shown in the following figure.

You can also choose to show the phone number attributes on a map, as shown in the following figure.

- Check phone number attributes and report abnormal logon information.

If a telecom operator wants to filter its customers whose daily locations are different from their phone number attributes (according to additional attributes and frequently accessed IP addresses), the following statement can be used:

```
* | select mobile , client_ip , count ( 1 ) as PV where
mobile_city ( try_cast ( " mobile " as bigint )) !=
ip_to_city ( client_ip ) and ip_to_city ( client_ip ) != ''
group by client_ip , mobile order by PV desc
```

Furthermore, you can create alarm rules that use phone number attributes.

9 Machine learning syntax and functions

9.1 Overview

Log Service provides a machine learning feature that supports multiple algorithms and calling methods. During log query and analysis, you can use SELECT statements and machine learning functions to call machine learning algorithms and analyze the characteristics of a field or fields within a period of time.

In particular, Log Service offers diversified time series analysis algorithms to help you quickly solve problems related to time series prediction, time series anomaly detection, time series decomposition, and multi-time series clustering. In addition, the algorithms are compatible with standard SQL functions. This greatly simplifies the use of the algorithms and improves troubleshooting efficiency.

Features

- Supports various smooth operations on single-time series sequences.
- Supports algorithms related to the prediction, anomaly detection, change point detection, inflexion point detection, and time series forecasting of single-time series sequences.
- Supports decomposition operations on single-time series sequences.
- Supports various clustering algorithms of multi-time series sequences.
- Supports multi-field pattern mining (based on the sequence of numeric data or text).

Limits

- The input time series data must be sampled from the same interval.
- The input time series data cannot contain data repeatedly sampled from the same time point.

Item	Limit
Valid capacity of time-series data processing	Data collected from a maximum of 150,000 consecutive time points If the limit is exceeded, you need to aggregate the data or reduce the sampling amount.

Item	Limit
Clustering capacity of the density-based clustering algorithm	A maximum of 5,000 time series curves, each of which cannot contain more than 1,440 time points
Clustering capacity of the hierarchical clustering algorithm	A maximum of 2,000 time series curves, each of which cannot contain more than 1,440 time points

Machine learning functions

	Category	Function	Description
Time series	Smooth function	ts_smooth_simple	Uses the Holt Winters algorithm to smooth time series data.
		ts_smooth_fir	Uses the finite impulse response (FIR) filter to smooth time series data.
		ts_smooth_iir	Uses the infinite impulse response (IIR) filter to smooth time series data.
	Time series forecasting function	ts_period_detect	Forecasts time series data by period.
	Change point detection function	ts_cp_detect	Finds intervals with different statistical characteristics from time series data. The interval endpoints are change points.
		ts_breakout_detect	Finds the time point when statistics steeply increase or decrease from time series data.
	Maximum value detection function	ts_find_peaks	Finds the locally maximum value of time series data in a specified window.

	Category	Function	Description
	Prediction and anomaly detection function	ts_predicate_simple	Uses default parameters to model time series data and performs simple time series prediction and anomaly detection.
		ts_predicate_ar	Uses an autoregressive model to model time series data and performs simple time series prediction and anomaly detection.
		ts_predicate_arma	Uses an autoregressive moving model to model time series data and performs simple time series prediction and anomaly detection.
		ts_predicate_arima	Uses an autoregressive moving model with differences to model time series data and performs simple time series prediction and anomaly detection.
		ts_regression_predict	Accurately predicts the long-run trend for a single periodic time series.
	Time series decomposition function	ts_decompose	Uses the Seasonal and Trend decomposition using Loess (STL) algorithm to decompose time series data.
	Time series clustering function	ts_density_cluster	Uses a density-based clustering method to cluster multiple pieces of time series data.
		ts_hierarchical_cluster	Uses a hierarchical clustering method to cluster multiple pieces of time series data.

	Category	Function	Description
		ts_similar_instance	Queries curves that are similar to a specified curve .
Pattern mining	Frequent pattern statistics	pattern_stat	Mines representative combinations of attributes among the given multi-attribute field samples to obtain the frequent pattern in statistical patterns.
	Differential pattern statistics	pattern_diff	Finds the pattern that causes differences between two collections under specified conditions .

9.2 Smooth function

The smooth function is used to smooth and filter input time series curves. Filtering is the first step to discover the shape of a time series curve.

Function list

Function	Description
ts_smooth_simple	This function is the default smooth function, which uses the Holt Winters algorithm to smooth time series data.
ts_smooth_fir	This function uses the FIR filter to smooth time series data.
ts_smooth_iir	This function uses the IIR filter to smooth time series data.

ts_smooth_simple

Function format:

```
select ts_smooth_simple(x, y)
```

The following table describes the parameters.

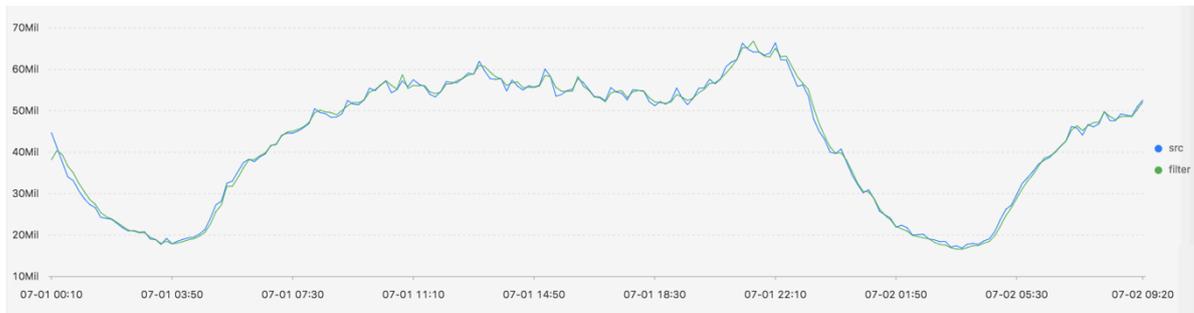
Parameter	Description	Value
x	Time column in ascending order	Unixtime timestamp in seconds
y	Numeric column corresponding to the data at a specified time point	--

Example:

- Statement for query and analysis:

```
* | select ts_smooth_simple ( stamp , value ) from ( select
  __time__ - __time__ % 120 as stamp , avg ( v ) as
  value from log GROUP BY stamp order by stamp )
```

- Result:



The following table describes the display items.

Display item	Description	
Horizontal axis	unixtime	Data timestamp in seconds
Longitudinal axis	src	Data before filtering
	filter	Data after filtering

ts_smooth_fir

Function format:

- When filter parameters are undetermined, you can use the parameters in the built-in window to filter time series data:

```
select ts_smooth_fir(x, y,winType,winSize,samplePeriod,sampleMethod)
```

- When filter parameters are specified, you can set them as needed:

```
select ts_smooth_fir(x, y,array[],samplePeriod,sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
<i>x</i>	Time column in ascending order	Unixtime timestamp in seconds
<i>y</i>	Numeric column corresponding to the data at a specified time point	-
<i>winType</i>	Filter window type	Value range: <ul style="list-style-type: none"> • rectangle: rectangle window • hanning: hanning window • hamming: hamming window • blackman: blackman window <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;">  Note: We recommend that you set this parameter to rectangle for better display. </div>
<i>winSize</i>	Length of the filter window	Long type values ranging form 2 to 15
<i>array[]</i>	Specific FIR filter parameters	Values in array format, sum of Array is 1.0. For example, [0.2, 0.4, 0.3, 0.1].
<i>samplePeriod</i>	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399 seconds

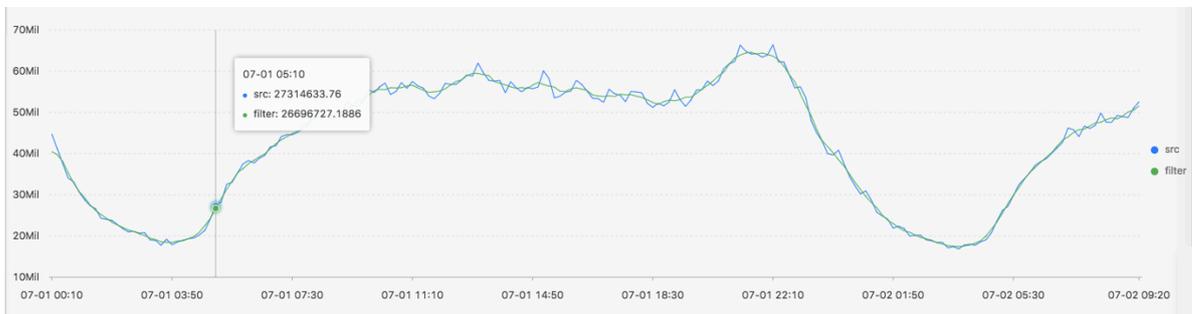
Parameter	Description	Value
<i>sampleMethod</i>	Method for sampling the data in the sampling window	<p>Value range:</p> <ul style="list-style-type: none"> • avg: average value of the data in the window • max: maximum value of the data in the window • min: minimum value of the data in the window • sum: sum of the data in the window

Example:

- Statement for query and analysis:

```
* | select ts_smooth_fir ( stamp , value , ' rectangle ' , 4
, 1 , ' avg ' ) from ( select __time__ - __time__ % 120
as stamp , avg ( v ) as value from log GROUP BY
stamp order by stamp )
```

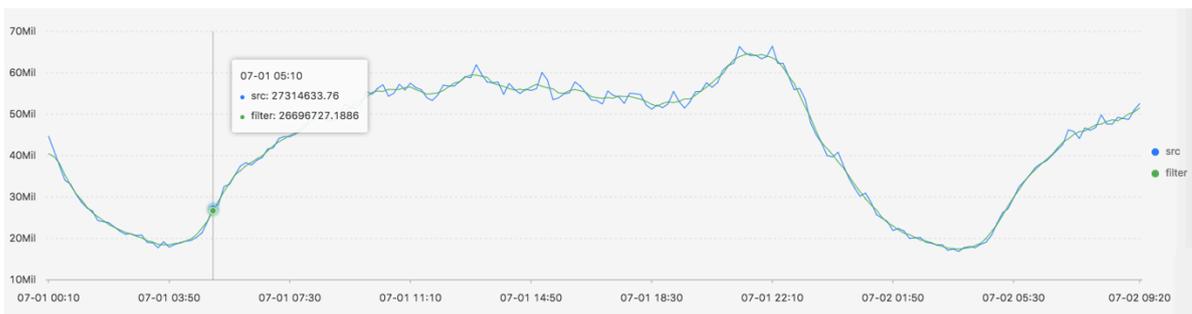
Result:



- Statement for query and analysis:

```
* | select ts_smooth_fir ( stamp , value , array [ 0 . 2 , 0
. 4 , 0 . 3 , 0 . 1 ] , 1 , ' avg ' ) from ( select __time__
- __time__ % 120 as stamp , avg ( v ) as value from
log GROUP BY stamp order by stamp )
```

Result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	Unixtime timestamp in seconds
Longitudinal axis	src	Data before filtering
	filter	Data after filtering

ts_smooth_iir

Function format:

```
select
  ts_smooth_iir(x, y, array[], array[], samplePeriod, sampleMethod)
```

The following table describes the parameters.

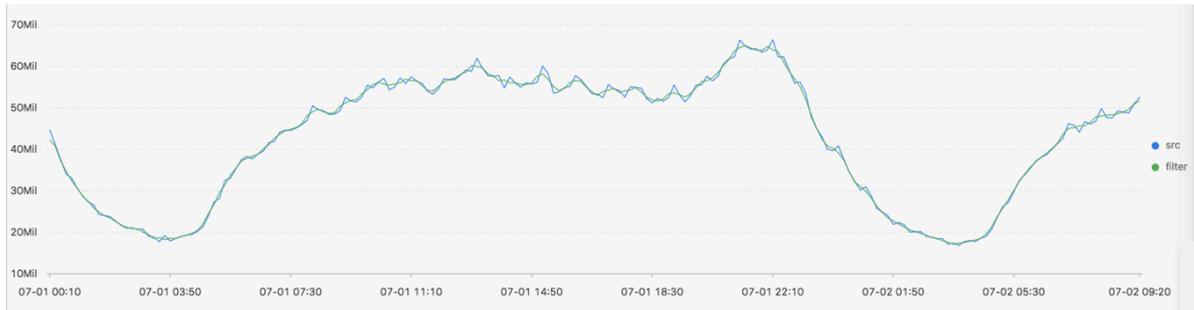
Parameter	Description	Value
<i>x</i>	Time column in ascending order	Unixtime timestamp in seconds
<i>y</i>	Numeric column corresponding to the data at a specified time point	-
<i>array[]</i>	Specific parameter of x_i in the IIR filter algorithm	Array values with a length ranging from 2 to 15. Sum of Array is 1.0. For example, array[0.2, 0.4, 0.3, 0.1]
<i>array[]</i>	Specific parameter of y_{i-1} in the IIR filter algorithm	Array values with a length ranging from 2 to 15. Sum of Array is 1.0. For example, array[0.2, 0.4, 0.3, 0.1]
<i>samplePeriod</i>	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399 seconds
<i>sampleMethod</i>	Method for sampling the data in the sampling window	Value range: <ul style="list-style-type: none"> • avg: average value of the data in the sampling window • max: maximum value of the data in the sampling window • min: minimum value of the data in the sampling window • sum: sum of the data in the sampling window

Example:

- Statement for query and analysis:

```
* | select  ts_smooth_ iir ( stamp , value , array [ 0 . 2 ,
0 . 4 , 0 . 3 , 0 . 1 ] , array [ 0 . 4 , 0 . 3 , 0 . 3 ] , 1
, ' avg ' ) from ( select  __time__ - __time__ % 120 as
stamp , avg ( v ) as value from log GROUP BY stamp
order by stamp )
```

- Result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	Unixtime timestamp in seconds
Longitudinal axis	src	Data before filtering
	filter	Data after filtering

9.3 Multi-period estimation function

The multi-period estimation function can estimate time series in different time periods and perform a series of operations, such as Fourier transform, to extract period data.

ts_period_detect

The function estimates time series data on a period basis.

Function format:

```
select
  ts_period_detect(x, y, minPeriod, maxPeriod, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
x	Time column in ascending order	Unixtime timestamp in seconds

Parameter	Description	Value
<i>y</i>	Numeric column corresponding to the data at a specified time point	-
<i>minPeriod</i>	Ratio of the minimum length of the pre-estimation period to the total length of the time series	Value range: (0, 1]
<i>maxPeriod</i>	Ratio of the maximum length of the pre-estimation period to the total length of the time series  Note: The value of <i>maxPeriod</i> must be greater than that of <i>minPeriod</i> .	Value range: (0, 1]
<i>samplePeriod</i>	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399 seconds
<i>sampleMethod</i>	Method for sampling the data in the sampling window	Value range: <ul style="list-style-type: none"> • avg: average value of the data in the window • max: maximum value of the data in the window • min: minimum value of the data in the window • sum: sum of the data in the window

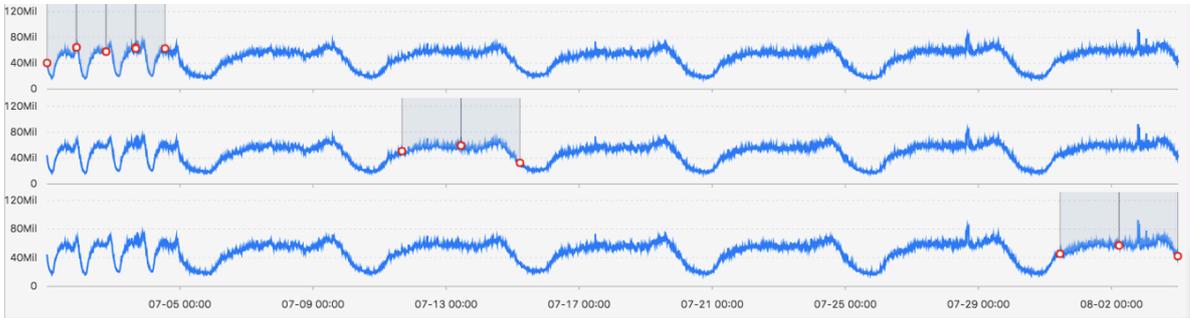
Example:

- Statement for query and analysis:

```
* | select ts_period_ detect ( stamp , value , 0 . 2 , 1 . 0
, 1 , ' avg ' ) from ( select __time__ - __time__ % 120
```

```
as stamp , avg ( v ) as value from log GROUP BY
stamp order by stamp )
```

• **Result:**



The following table describes the display items.

Display item	Description
period_id	Array composed of period IDs with an array length of 1. The value 0.0 indicates the original series.
time_series	Timestamp sequence
data_series	Result for each timestamp <ul style="list-style-type: none"> • It indicates the original sequence when period_id is 0.0. • It indicates the period estimation result after filtering when period_id is not 0.0.

9.4 Change point detection function

The change point detection function detects change points in time series data.

The change point detection function supports two types of change points:

- Changes of statistical characteristics within a specified time period
- Obvious faulting in a sequence

Function list

Function	Description
ts_cp_detect	This function finds intervals with different statistical characteristics within a time series. The interval endpoints are change points.

Function	Description
ts_breakout_detect	This function finds the time point when statistics steeply increase or decrease within a time series.

ts_cp_detect

Function format:

- If you are not sure about the window size, use the `ts_cp_detect` function in the following format. Then, the algorithm called by the function will use a window with a length of 10 to detect change points.

```
select ts_cp_detect(x, y, amplePeriod,sampleMethod)
```

- If you need to adjust the display effect of a service curve, use the `ts_cp_detect` function in the following format. Then, you can optimize the display effect by setting the `minSize` parameter.

```
select ts_cp_detect(x, y, minSize, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
<code>x</code>	Time column in ascending order	Unixtime timestamp in seconds
<code>y</code>	Numeric column corresponding to the data at a specified time point	-
<code>minSize</code>	Minimum length of consecutive intervals	The minimum value is 3, and the maximum value cannot exceed 1/10 of the length of the current input data.
<code>samplePeriod</code>	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399 seconds

Parameter	Description	Value
<i>sampleMethod</i>	Method for sampling the data in the sampling window	Value range: <ul style="list-style-type: none"> • avg: average value of the data in the window • max: maximum value of the data in the window • min: minimum value of the data in the window • sum: sum of the data in the window

Example:

- Statement for query and analysis:

```
* | select ts_cp_dete ct ( stamp , value , 3 , 1 , ' avg ' )
  from ( select __time__ - __time__ % 10 as stamp , avg
        ( v ) as value from log GROUP BY stamp order by
        stamp )
```

- Result:



The following table describes the display items.

Display item	Description	
Horizontal axis	unixtime	Data timestamp in seconds, for example, 1537071480
Longitudinal axis	src	Data before filtering, for example, 1956092.7647745228
	prob	Probability that a point is a change point . Its value ranges from 0 to 1.

ts_breakout_detect

Function format:

```
select ts_breakout_detect(x, y, winSize, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
<i>x</i>	Time column in ascending order	Unixtime timestamp in seconds
<i>y</i>	Numeric column corresponding to the data at a specified time point	-
<i>winSize</i>	Minimum length of consecutive intervals	The minimum value is 3, and the maximum value cannot exceed 1/10 of the length of the current input data.
<i>samplePeriod</i>	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399 seconds
<i>sampleMethod</i>	Method for sampling the data in the sampling window	Value range: <ul style="list-style-type: none"> • avg: average value of the data in the window • max: maximum value of the data in the window • min: minimum value of the data in the window • sum: sum of the data in the window

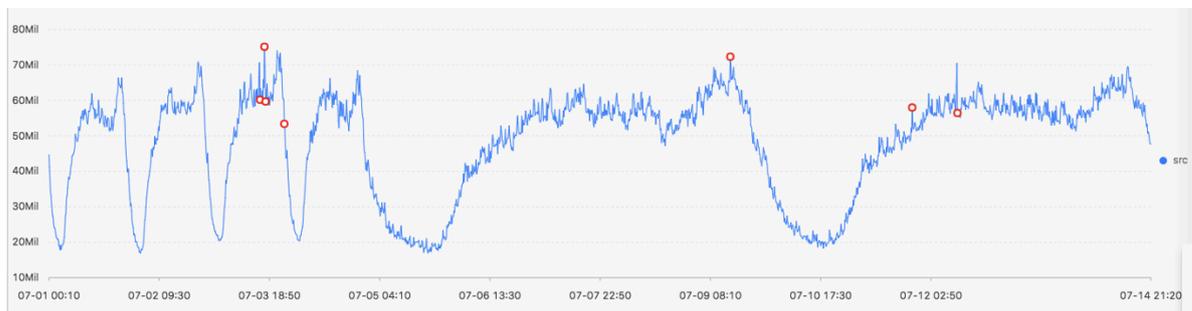
Example:

- Statement for query and analysis:

```
* | select ts_breakout t_detect ( stamp , value , 3 , 1 , '
  avg ' ) from ( select __time__ - __time__ % 10 as stamp
```

```
, avg ( v ) as value from log GROUP BY stamp order
by stamp )
```

• Result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	Data timestamp in seconds, for example, 1537071480
Longitudinal axis	src	Data before filtering, for example, 1956092.7647745228
	prob	Probability that a point is a change point . Its value ranges from 0 to 1.

9.5 Maximum value detection function

The maximum value detection function is used to identify the maximum value of a sequence in a specified window.

Function format:

```
select ts_find_peaks(x, y, winSize, samplePeriod, sampleMethod)
```

The following table describes the function parameters.

Parameter	Description	Value
x	Time column in ascending order	Unixtime timestamp in seconds
y	Numeric column corresponding to the data at a specified time point	-

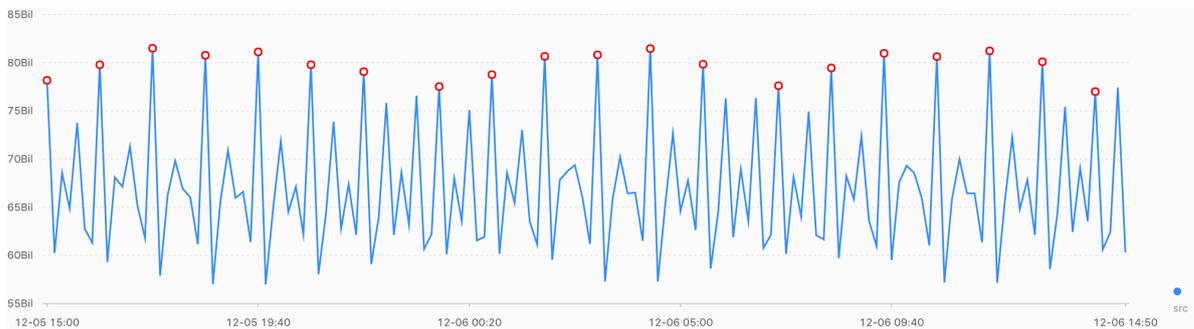
Parameter	Description	Value
<i>winSize</i>	Minimum length of a window	Long-type values range from 1 to the actual data length. We recommend that you specify one tenth of the actual data length as <i>winSize</i> .
<i>samplePeriod</i>	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399
<i>sampleMethod</i>	Method for sampling the data in the sampling window	Value range: <ul style="list-style-type: none"> • avg: average value of the data in the window • max: maximum value of the data in the window • min: minimum value of the data in the window • sum: sum of the data in the window

Examples:

- Statement for query and analysis:

```
* and h : nu2h05202 . nu8 and m : NET | select
  ts_find_peaks ( stamp , value , 30 , 1 , ' avg ' ) from (
  select __time__ - __time__ % 10 as stamp , avg ( v ) as
  value from log GROUP BY stamp order by stamp )
```

- Result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	Data timestamp in seconds, for example, 1537071480

Display item		Description
Longitudinal axis	src	Data before filtering, for example, 1956092.7647745228
	peak_flag	Whether the current point has the maximum value. Value range: <ul style="list-style-type: none"> · 1.0: The point has the maximum value. · 0.0: The point does not have the maximum value.

9.6 Prediction and anomaly detection functions

To detect anomalies, you can use a prediction and anomaly detection function to predict time series curves and identify the Ksigma and quantiles of the errors between a predicted curve and an actual curve.

- Log Service Machine Learning Introduction (01): [Time Series Statistics Modeling](#)
- Log Service Machine Learning Introduction (03): [Time Series Anomaly Detection Modeling](#)
- Log Service Machine Learning Introduction (05): [Time Series Prediction](#)
- Log Service Machine Learning Best Practices: [Time Series Anomaly Detection and Alert](#)

Functions

Function	Description
<code>ts_predicate_simple</code>	Uses default parameters to model time series data and performs simple time series prediction and anomaly detection.
<code>ts_predicate_ar</code>	Uses an autoregressive (AR) model to model time series data and performs simple time series prediction and anomaly detection.
<code>ts_predicate_arma</code>	Uses an autoregressive moving average (ARMA) model to model time series data and performs simple time series prediction and anomaly detection.
<code>ts_predicate_arima</code>	Uses an autoregressive integrated moving average (ARIMA) model to model time series data and performs simple time series prediction and anomaly detection.

Function	Description
ts_regression_predict	<p>Accurately predicts the long-run trend for a single periodic time series with a certain tendency.</p> <p>Scenario: This function can be used to predict metering data, network traffic, financial data, and different business data that follows certain rules.</p>
ts_anomaly_filter	<p>Filters the anomalies detected during time series anomaly detection on multiple curves based on the custom anomaly mode. This function helps you quickly find abnormal instance curves.</p>



Note:

The display items for all prediction and anomaly detection functions are the same. For more information about the output result and relevant description, see [the output result and display item description of the ts_predicate_simple function](#).

ts_predicate_simple

Function format:

```
select
  ts_predicate_simple(x, y, nPred, isSmooth, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
<i>x</i>	The time sequence. The time points along the x axis are sorted in ascending order.	Each time point is a Unix timestamp. Unit: second.
<i>y</i>	The sequence of the numeric values of the property under observation, corresponding to the specified time points.	N/A
<i>nPred</i>	The number of points for prediction.	The value is of the Long type. Valid values: [1, 5 × <i>p</i>].

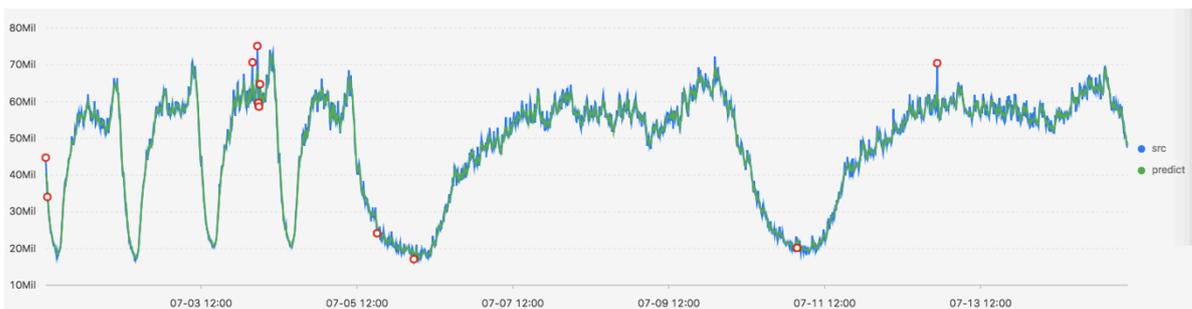
Parameter	Description	Value
<i>isSmooth</i>	Specifies whether to filter the raw data. If you do not set this parameter , the default value true is used, indicating that the raw data will be filtered.	The value is of the Boolean type. Valid values: <ul style="list-style-type: none"> · true: filters the raw data. · false: does not filter the raw data. Default value: true.
<i>samplePeriod</i>	The period during which the current time series data is sampled.	The value is of the Long type. Valid values: [1, 86399].
<i>sampleMethod</i>	The method for sampling the data in the sampling window.	Valid values: <ul style="list-style-type: none"> · avg: samples the average value of the data in the window. · max: samples the maximum value of the data in the window. · min: samples the minimum value of the data in the window. · sum: samples the sum of the data in the window.

Example:

- The statement for query and analysis is as follows:

```
* | select ts_predica te_simple ( stamp , value , 6 , 1 , '
  avg ' ) from ( select __time__ - __time__ % 60 as stamp
  , avg ( v ) as value from log GROUP BY stamp order
  by stamp )
```

- The following figure shows the output result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The Unix timestamp of the data. Unit: second.
Vertical axis	src	The raw data.
	predict	The data after filtering.
	upper	The upper limit of the prediction. By default, the confidence level is 0.85, which cannot be modified.
	lower	The lower limit of the prediction. By default, the confidence level is 0.85, which cannot be modified.
	anomaly_prob	The probability that the point is an anomaly. Valid values: [0, 1].

ts_predicate_ar

Function format:

```
select
  ts_predicate_ar(x, y, p, nPred, isSmooth, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
x	The time sequence. The time points along the x axis are sorted in ascending order.	Each time point is a Unix timestamp. Unit: second.
y	The sequence of the numeric values of the property under observation, corresponding to the specified time points.	N/A
p	The order of the AR model.	The value is of the Long type. Valid values: [2, 8].
$nPred$	The number of points for prediction.	The value is of the Long type. Valid values: [1, $5 \times p$].

Parameter	Description	Value
<i>isSmooth</i>	Specifies whether to filter the raw data. If you do not set this parameter, the default value true is used, indicating that the raw data will be filtered.	The value is of the Boolean type. Valid values: <ul style="list-style-type: none"> · true: filters the raw data. · false: does not filter the raw data. Default value: true.
<i>samplePeriod</i>	The period during which the current time series data is sampled.	The value is of the Long type. Valid values: [1, 86399].
<i>sampleMethod</i>	The method for sampling the data in the sampling window.	Valid values: <ul style="list-style-type: none"> · avg: samples the average value of the data in the window. · max: samples the maximum value of the data in the window. · min: samples the minimum value of the data in the window. · sum: samples the sum of the data in the window.

For example, the statement for query and analysis is as follows:

```
* | select ts_predicate_arma( stamp , value , 3 , 4 , 1 , '
  avg ' ) from ( select __time__ - __time__ % 60 as stamp ,
  avg ( v ) as value from log GROUP BY stamp order by
  stamp )
```

ts_predicate_arma

Function format:

```
select
  ts_predicate_arma(x, y, p, q, nPred, isSmooth, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
<i>x</i>	The time sequence. The time points along the x axis are sorted in ascending order.	Each time point is a Unix timestamp. Unit: second.
<i>y</i>	The sequence of the numeric values of the property under observation, corresponding to the specified time points.	N/A
<i>p</i>	The order of the AR model.	The value is of the Long type. Valid values: [2, 100].
<i>q</i>	The order of the ARMA model.	The value is of the Long type. Valid values: [2, 8].
<i>nPred</i>	The number of points for prediction.	The value is of the Long type. Valid values: [1, $5 \times p$].
<i>isSmooth</i>	Specifies whether to filter the raw data. If you do not set this parameter, the default value true is used, indicating that the raw data will be filtered.	The value is of the Boolean type. Valid values: <ul style="list-style-type: none"> • true: filters the raw data. • false: does not filter the raw data. Default value: true.
<i>samplePeriod</i>	The period during which the current time series data is sampled.	The value is of the Long type. Valid values: [1, 86399].
<i>sampleMethod</i>	The method for sampling the data in the sampling window.	Valid values: <ul style="list-style-type: none"> • avg: samples the average value of the data in the window. • max: samples the maximum value of the data in the window. • min: samples the minimum value of the data in the window. • sum: samples the sum of the data in the window.

For example, the statement for query and analysis is as follows:

```
* | select ts_predicate_arma ( stamp , value , 3 , 2 , 4 ,
1 , ' avg ' ) from ( select __time__ - __time__ % 60 as
stamp , avg ( v ) as value from log GROUP BY stamp
order by stamp )
```

ts_predicate_arma

Function format:

```
select
  ts_predicate_arma(x, y, p, d, qnPred, isSmooth, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
<i>x</i>	The time sequence. The time points along the x axis are sorted in ascending order.	Each time point is a Unix timestamp. Unit: second.
<i>y</i>	The sequence of the numeric values of the property under observation, corresponding to the specified time points.	N/A
<i>p</i>	The order of the AR model.	The value is of the Long type. Valid values: [2, 8].
<i>d</i>	The order of the ARIMA model.	The value is of the Long type. Valid values: [1, 3].
<i>q</i>	The order of the ARMA model.	The value is of the Long type. Valid values: [2, 8].
<i>nPred</i>	The number of points for prediction.	The value is of the Long type. Valid values: [1, $5 \times p$].
<i>isSmooth</i>	Specifies whether to filter the raw data. If you do not set this parameter, the default value true is used, indicating that the raw data will be filtered.	The value is of the Boolean type. Valid values: <ul style="list-style-type: none"> · true: filters the raw data. · false: does not filter the raw data. Default value: true.
<i>samplePeriod</i>	The period during which the current time series data is sampled.	The value is of the Long type. Valid values: [1, 86399].

Parameter	Description	Value
<i>sampleMethod</i>	The method for sampling the data in the sampling window.	<p>Valid values:</p> <ul style="list-style-type: none"> • avg: samples the average value of the data in the window. • max: samples the maximum value of the data in the window. • min: samples the minimum value of the data in the window. • sum: samples the sum of the data in the window.

For example, the statement for query and analysis is as follows:

```
* | select ts_predica te_arima ( stamp , value , 3 , 1 , 2 ,
4 , 1 , ' avg ' ) from ( select __time__ - __time__ % 60 as
stamp , avg ( v ) as value from log GROUP BY stamp
order by stamp )
```

ts_regression_predict

Function format:

```
select
ts_regression_predict(x, y, nPred, algo_type, processType, samplePeriod, sampleMe
```

The following table describes the parameters.

Parameter	Description	Value
<i>x</i>	The time sequence. The time points along the x axis are sorted in ascending order.	Each time point is a Unix timestamp. Unit: second.
<i>y</i>	The sequence of the numeric values of the property under observation, corresponding to the specified time points.	N/A
<i>nPred</i>	The number of points for prediction.	The value is of the Long type. Valid values: [1, 500].

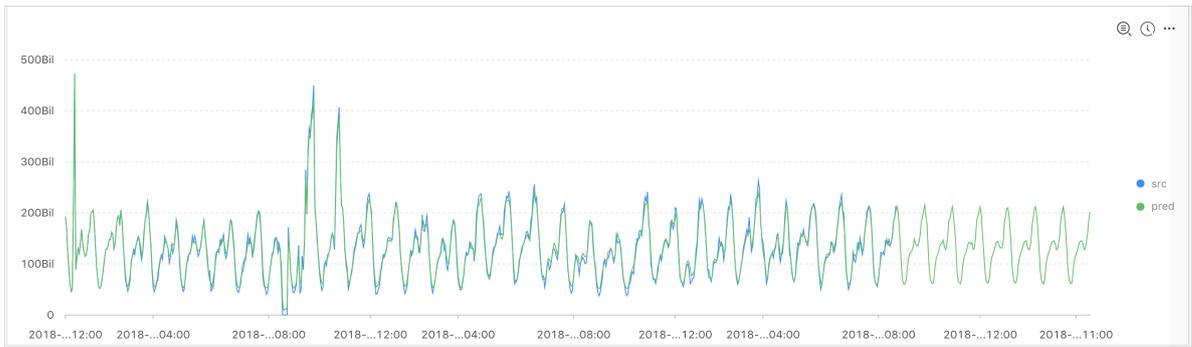
Parameter	Description	Value
<i>algo_type</i>	The algorithm type for prediction.	Valid values: <ul style="list-style-type: none"> · origin: uses the Gradient Boosted Regression Tree (GBRT) algorithm for prediction. · forest: uses the GBRT algorithm for prediction based on the trend components decomposed by Seasonal and Trend decomposition using Loess (STL), and then uses the additive model to sum up the decomposed components and obtains the predicted data. · linear: uses the Linear Regression algorithm for prediction based on the trend components decomposed by STL, and then uses the additive model to sum up the decomposed components and obtains the predicted data.
<i>processType</i>	Specifies whether to preprocess the data.	<ul style="list-style-type: none"> · 0: does not preprocess the data before the data is used for prediction. · 1: removes abnormal data before the data is used for prediction.
<i>samplePeriod</i>	The period during which the current time series data is sampled.	The value is of the Long type. Valid values: [1, 86399].
<i>sampleMethod</i>	The method for sampling the data in the sampling window	Valid values: <ul style="list-style-type: none"> · avg: samples the average value of the data in the window. · max: samples the maximum value of the data in the window. · min: samples the minimum value of the data in the window. · sum: samples the sum of the data in the window.

Example:

- The statement for query and analysis is as follows:

```
* and h : nu2h05202 . nu8 and m : NET | select
  ts_regression_predict ( stamp , value , 200 , ' origin ' ,
    1 , ' avg ' ) from ( select __time__ - __time__ % 60 as
    stamp , avg ( v ) as value from log GROUP BY stamp
    order by stamp )
```

- The following figure shows the output result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The Unix timestamp of the data. Unit: second.
Vertical axis	src	The raw data.
	predict	The predicted data.

ts_anomaly_filter

Function format:

```
select
  ts_anomaly_filter(lineName, ts, ds, preds, probs, nWatch, anomalyType)
```

The following table describes the parameters.

Parameter	Description	Value
<i>lineName</i>	The name of each curve. The value is of the Varchar type.	N/A
<i>ts</i>	The time sequence of the curve. The value is an array of the Double type. The time points are sorted in ascending order.	N/A

Parameter	Description	Value
<i>ds</i>	The actual value sequence of the curve. The value is an array of the Double type. The actual values correspond to the time points specified by the <i>ts</i> parameter in one-to-one mode.	N/A
<i>preds</i>	The predicted value sequence of the curve. The value is an array of the Double type. The predicted values correspond to the time points specified by the <i>ts</i> parameter in one-to-one mode.	N/A
<i>probs</i>	The anomaly detection result sequence of the curve. The value is an array of the Double type . The anomaly detection results correspond to the time points specified by the <i>ts</i> parameter in one-to-one mode.	N/A
<i>nWatch</i>	The number of the recently observed actual values on the curve. The value is of the Long type. The value must be smaller than the number of time points on the curve.	N/A
<i>anomalyType</i>	The type of anomaly to be filtered. The value is of the Long type.	<ul style="list-style-type: none"> • 0: all anomalies. • 1: positive anomalies. • -1: negative anomalies.

Example:

- The statement for query and analysis is as follows:

```
* | select res . name , res . ts , res . ds , res . preds ,
res . probs
```

```

from (
  select ts_anomaly_filter ( name , ts , ds , preds
, probs , cast ( 5 as bigint ), cast ( 1 as bigint )
as res
  from (
    select name , res [ 1 ] as ts , res [ 2 ] as ds
, res [ 3 ] as preds , res [ 4 ] as uppers , res [ 5 ] as
lowers , res [ 6 ] as probs
    from (
      select name , array_tran_spose ( ts_predica te_ar (
stamp , value , 10 )) as res
      from (
        select name , stamp , value from log where
name like '% asg -%') group by name ) ) );

```

• The output result is as follows:

name	ts	ds	preds	probs
asg - bp1hylzdi2 wx7civ0ivk	[1 . 5513696E9 , 1 . 5513732E9 , 1 . 5513768E9 , 1 . 5513804E9]	[1 , 2 , 3 , NaN]	[1 , 2 , 3 , 4]	[0 , 0 , 1 , NaN]

9.7 Sequence decomposition function

The sequence decomposition function can decompose service curves and highlight information about the curve trends and periods.

ts_decompose

Function format:

```
select ts_decompose(x, y, samplePeriod, sampleMethod)
```

The following table describes the parameters.

Parameter	Description	Value
x	Time column in ascending order	Unixtime timestamp in seconds
y	Numeric column corresponding to the data at a specified time point	-
samplePeriod	Period during which the current time series data is sampled	Long type values ranging from 1 to 86399 seconds

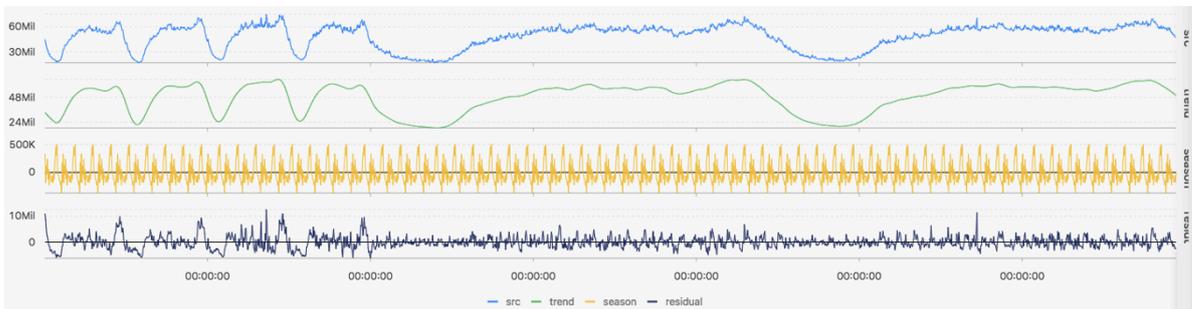
Parameter	Description	Value
<i>sampleMethod</i>	Method for sampling the data in the sampling window	Value range: <ul style="list-style-type: none"> • avg: average value of the data in the window • max: maximum value of the data in the window • min: minimum value of the data in the window • sum: sum of the data in the window

Example:

- Statement for query and analysis:

```
* | select ts_decompo se ( stamp , value , 1 , ' avg ' ) from
  ( select __time__ - __time__ % 60 as stamp , avg ( v )
    as value from log GROUP BY stamp order by stamp )
```

- Result:



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	Unixtime timestamp in seconds
Longitudinal axis	src	Raw data
	trend	Curve trend after decomposition
	season	Curve period after decomposition
	residual	Residual data after decomposition

9.8 Time series clustering functions

You can use a time series clustering function to cluster multiple pieces of time series data and obtain different curve shapes. Then, you can quickly find the corresponding

cluster center and curves with shapes that are different from the curve shapes in the cluster.

Function list

Function	Description
<code>ts_density_cluster</code>	Uses a density-based clustering method to cluster multiple pieces of time series data.
<code>ts_hierarchical_cluster</code>	Uses a hierarchical clustering method to cluster multiple pieces of time series data.
<code>ts_similar_instance</code>	Queries curves that are similar to a specified curve.

ts_density_cluster

Function format:

```
select ts_density_cluster(x, y, z)
```

The following table describes the parameters.

Parameter	Description	Value
x	The sequence of time in ascending order.	Unix timestamp. Unit: seconds.
y	The sequence of numeric data corresponding to each specified time point.	N/A
z	The metric name corresponding to the data at each specified time point.	String type, for example, machine01.cpu_usr.

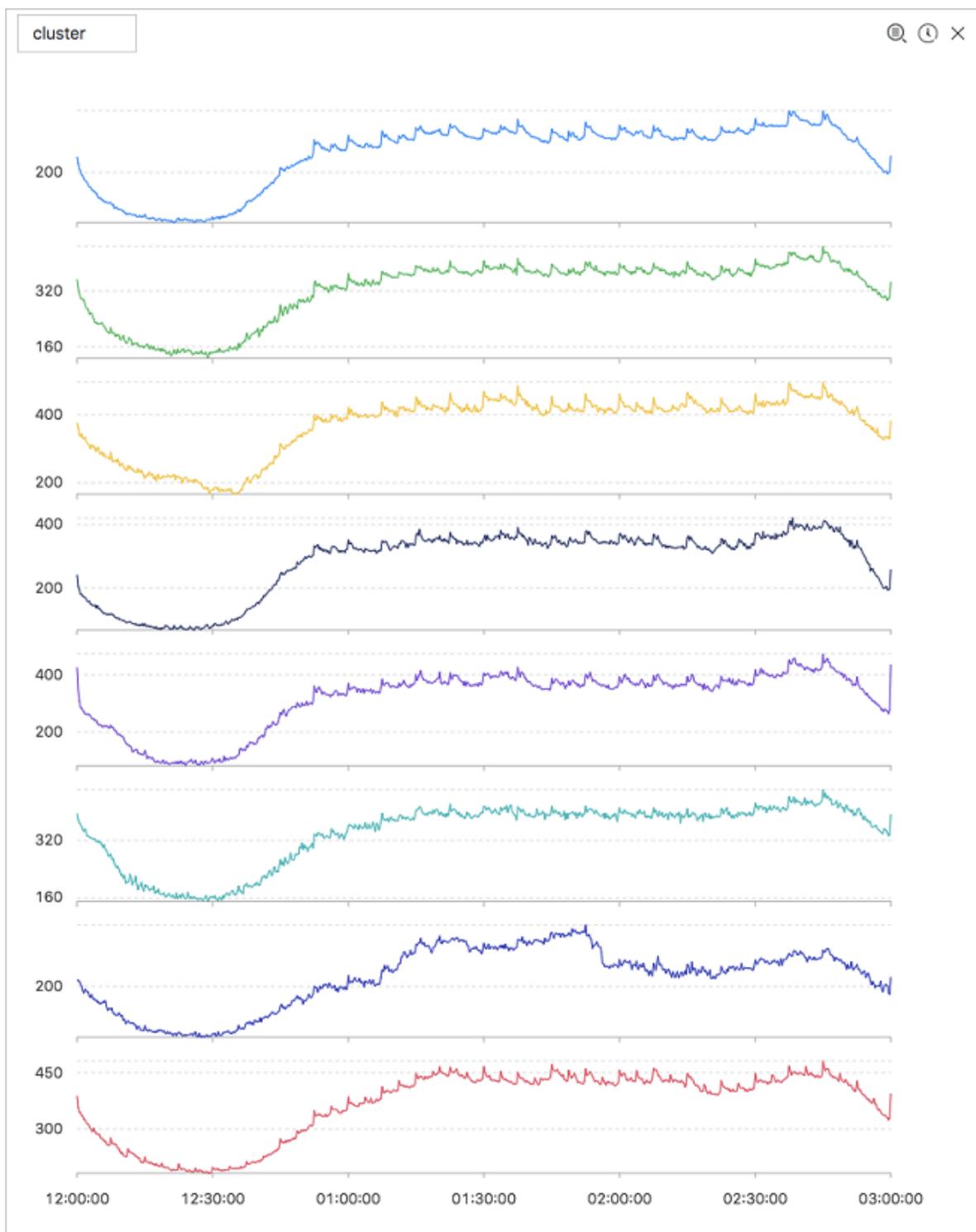
Example:

- The statement for query and analysis is as follows:

```
* and ( h : " machine_01 " OR h : " machine_02 " OR h
: " machine_03 ") | select ts_density _cluster ( stamp ,
metric_val ue , metric_nam e ) from ( select __time__ -
__time__ % 600 as stamp , avg ( v ) as metric_val ue
```

```
, h as metric_name from log GROUP BY stamp ,  
metric_name order BY metric_name , stamp )
```

- The following figure shows the output result.



The following table describes the display items.

Display item	Description
cluster_id	The category of the cluster. The value -1 indicates that the cluster is not categorized in any cluster centers.
rate	The proportion of instances in the cluster.
time_series	The timestamp sequence of the cluster center.
data_series	The data sequence of the cluster center.
instance_names	The collection of instances included in the cluster center.
sim_instance	The name of an instance in the cluster.

ts_hierarchical_cluster

Function format:

```
select ts_hierarchical_cluster(x, y, z)
```

The following table describes the parameters.

Parameter	Description	Value
x	The sequence of time in ascending order.	Unix timestamp. Unit: seconds.
y	The sequence of numeric data corresponding to each specified time point.	N/A
z	The metric name corresponding to the data at each specified time point.	String type, for example, machine01.cpu_usr.

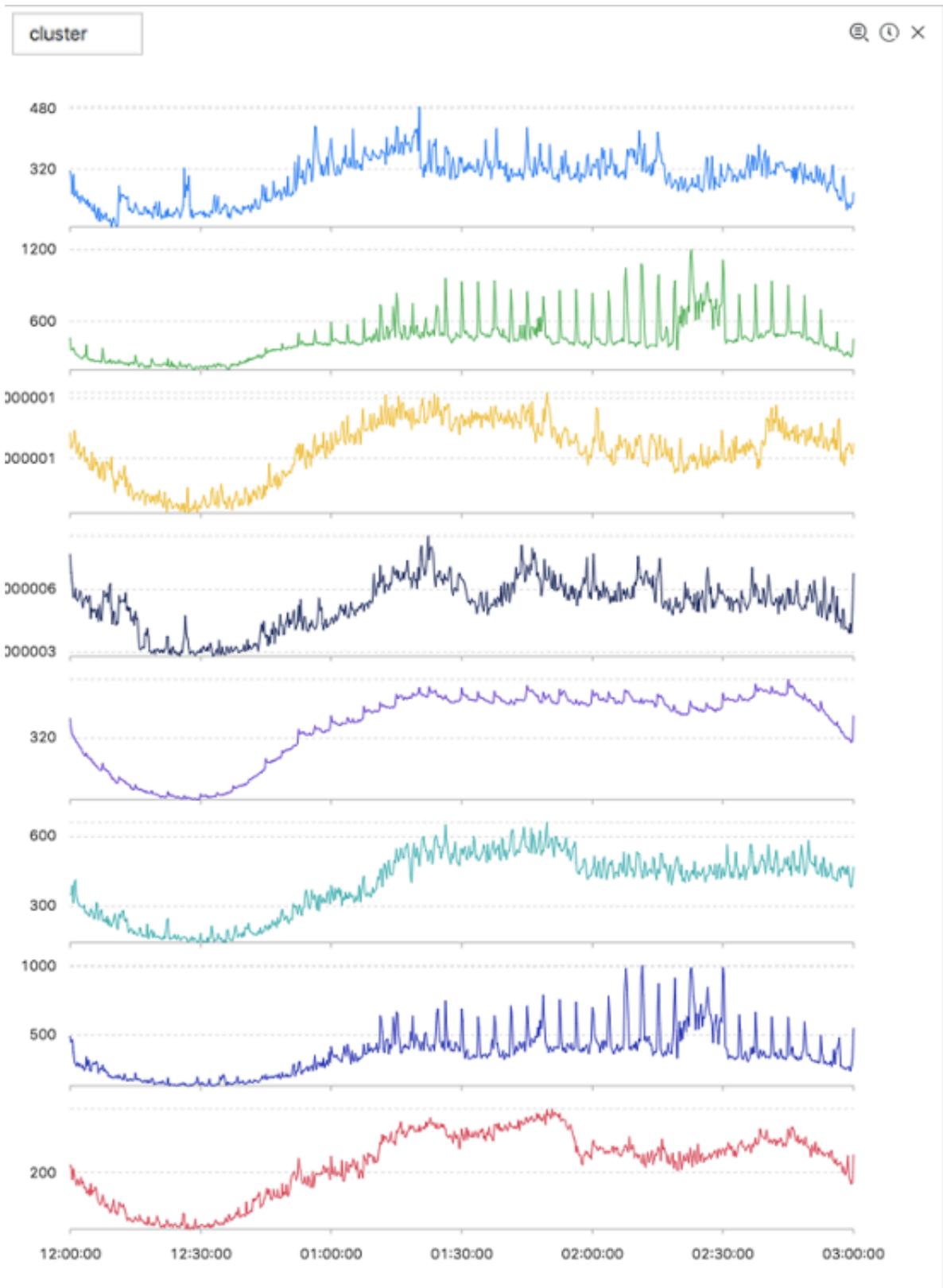
Example:

- The statement for query and analysis is as follows:

```
* and ( h : " machine_01 " OR h : " machine_02 " OR h :
" machine_03 ") | select ts_hierarc hical_clus ter ( stamp
, metric_val ue , metric_nam e ) from ( select __time__
- __time__ % 600 as stamp , avg ( v ) as metric_val
```

```
ue , h as metric_name from log GROUP BY stamp ,  
metric_name order BY metric_name , stamp )
```

- The following figure shows the output result.



The following table describes the display items.

Display item	Description
cluster_id	The category of the cluster. The value -1 indicates that the cluster is not categorized in any cluster centers.
rate	The proportion of instances in the cluster.
time_series	The timestamp sequence of the cluster center.
data_series	The data sequence of the cluster center.
instance_names	The collection of instances included in the cluster center.
sim_instance	The name of an instance in the cluster.

ts_similar_instance

Function format:

```
select ts_similar_instance(x, y, z, instance_name, topK, metricType)
```

The following table describes the parameters.

Parameter	Description	Value
x	The sequence of time in ascending order.	Unix timestamp. Unit: seconds.
y	The sequence of numeric data corresponding to each specified time point.	N/A
z	The metric name corresponding to the data at each specified time point.	String type, for example, machine01.cpu_usr.
instance_name	The name of the specified metric to be queried in the z collection.	String type, for example, machine01.cpu_usr.  Note: The metric must be an existing one.
topK	The curves similar to a given curve. A maximum of K curves are returned.	N/A
metricType	The metric used to measure the similarity between time series curves.	{'shape', 'manhattan', 'euclidean'}

For example, the statement for query and analysis is as follows:

```
* and m : NET and m : Tcp and ( h : " nu4e01524 . nu8
" OR h : " nu2i10267 . nu8 " OR h : " nu4q10466 . nu8
") | select ts_similar_instance ( stamp , metric_val ue ,
metric_name , ' nu4e01524 . nu8 ') from ( select __time__ -
__time__ % 600 as stamp , sum ( v ) as metric_val ue , h
as metric_name from log GROUP BY stamp , metric_name
order BY metric_name , stamp )
```

The following table describes the display items.

Display item	Description
instance_name	The list of metrics that are similar to the specified metric.
time_series	The timestamp sequence of the cluster center.
data_series	The data sequence of the cluster center.

9.9 Frequent pattern statistical function

The frequent pattern statistical function mines representative combinations of attributes from the given multi-attribute field samples to summarize the current logs.

pattern_stat

Function format:

```
select pattern_stat(array[col1, col2, col3], array['col1_name',
'col2_name', 'col3_name'], array[col5, col6], array['col5_name',
'col6_name'], supportScore, sample_ratio)
```

The following table describes the parameters.

Parameter	Description	Value
<i>array[col1, col2, col3]</i>	Input column composed of character type values	Values in array format, for example, array[clientIP, sourceIP, path, logstore]
<i>array['col1_name', 'col2_name', 'col3_name']</i>	Name corresponding to the input column composed of character type values	Values in array format, for example, array['clientIP', 'sourceIP', 'path', 'logstore']
<i>array[col5, col6]</i>	Input column composed of numeric values	Values in array format, for example, array[Inflow, OutFlow]

Parameter	Description	Value
<code>array['col5_name', 'col6_name']</code>	Name corresponding to the input column composed of numeric values	Values in array format, for example, array['Inflow', 'OutFlow']
<code>supportScore</code>	Support level of positive and negative samples for pattern mining	Double type values. Range: (0,1].
<code>sample_ratio</code>	Sampling ratio with the default value of 0.1, which indicates that only 10% of the total samples are used	Double type values. Range: (0,1].

Example:

- Statement for query and analysis:

```
* | select pattern_st at ( array [ Category , ClientIP , ProjectName , LogStore , Method , Source , UserAgent ], array [ ' Category ', ' ClientIP ', ' ProjectName ', ' LogStore ', ' Method ', ' Source ', ' UserAgent ' ], array [ InFlow , OutFlow ], array [ ' InFlow ', ' OutFlow ' ], 0 . 45 , 0 . 3 ) limit 1000
```

- Result:

count + ↓	supportscore + ↓	pattern + ↓
468235	0.9880626809484018	InFlow >= 0.0 and InFlow <= 60968.7 and OutFlow >= 0.0 and OutFlow <= 15566.4
459356	0.9693263443991458	Status = '200' and OutFlow >= 0.0 and OutFlow <= 15566.4
458757	0.9680623433187309	Status = '200' and InFlow >= 0.0 and InFlow <= 60968.7
456228	0.9627256843331392	InFlow >= 0.0 and InFlow <= 60968.7 and Status = '200' and OutFlow >= 0.0 and OutFlow <= 15566.4
417662	0.6813442725346703	InFlow >= 0.0 and InFlow <= 60968.7 and UserAgent = 'sls-cpp-sdk v0.6' and Status = '200'
417662	0.6813442725346703	UserAgent = 'sls-cpp-sdk v0.6' and InFlow >= 0.0 and InFlow <= 60968.7
415133	0.8760076135490787	OutFlow >= 0.0 and OutFlow <= 15566.4 and InFlow >= 0.0 and InFlow <= 60968.7 and UserAgent = 'sls-cpp-sdk v0.6' and Status = '200'
415133	0.8760076135490787	OutFlow >= 0.0 and OutFlow <= 15566.4 and UserAgent = 'sls-cpp-sdk v0.6' and InFlow >= 0.0 and InFlow <= 60968.7
415133	0.8760076135490787	OutFlow >= 0.0 and OutFlow <= 15566.4 and UserAgent = 'sls-cpp-sdk v0.6' and Status = '200'
415133	0.8760076135490787	UserAgent = 'sls-cpp-sdk v0.6' and OutFlow >= 0.0 and OutFlow <= 15566.4
414167	0.8739691744110473	InFlow >= 0.0 and InFlow <= 60968.7 and Method = 'PullData' and Status = '200'
414167	0.8739691744110473	Method = 'PullData' and InFlow >= 0.0 and InFlow <= 60968.7

The following table describes the display items.

Display item	Description
count	Number of samples for the current pattern
supportScore	Support level for the current pattern

Display item	Description
pattern	Pattern content, which is organized in the format of conditional queries

9.10 Differential pattern statistical function

Based on the given multi-attribute field samples and conditions, the differential pattern statistical function analyzes the set of differential patterns affecting the conditions. This helps you quickly diagnose the causes for the differences between the conditions.

pattern_diff

Function format:

```
select
  pattern_diff(array_char_value, array_char_name, array_numeric_value, array_numeric_name, condition, supportScore)
```

The following table describes the parameters.

Parameter	Description	Value
<i>array_char_value</i>	Input column composed of character type values	Values in array format, for example, array[clientIP, sourceIP, path, logstore]
<i>array_char_name</i>	Name corresponding to the input column composed of character type values	Values in array format, for example, array['clientIP', 'sourceIP', 'path', 'logstore']
<i>array_numeric_value</i>	Input column composed of numeric values	Values in array format, for example, array[Inflow, OutFlow]
<i>array_numeric_name</i>	Name corresponding to the input column composed of numeric values	Values in array format, for example, array['Inflow', 'OutFlow']
<i>condition</i>	Data filtering condition. True indicates positive samples, and False indicates negative samples.	For example: latency ≤ 300
<i>supportScore</i>	Support degree of positive and negative samples for pattern mining	Double type values. Range: (0,1].

Parameter	Description	Value
<i>posSampleRatio</i>	Sampling ratio of positive samples with a default value of 0.5, which indicates that only half of the positive samples are used	Double type values. Range: (0,1].
<i>negSampleRatio</i>	Sampling ratio of negative samples with a default value of 0.5, which indicates that only half of the negative samples are used	Double type values. Range: (0,1].

Example:

- Statement for query and analysis:

```
* | select pattern_diff ( array [ Category , ClientIP , ProjectName , LogStore , Method , Source , UserAgent ], array [ ' Category ', ' ClientIP ', ' ProjectName ', ' LogStore ', ' Method ', ' Source ', ' UserAgent ' ], array [ InFlow , OutFlow ], array [ ' InFlow ', ' OutFlow ' ], Latency > 300 , 0.2 , 0.1 , 1.0 ) limit 1000
```

- Result:

possupport +↓↑	posconfidence +↓↑	negsupport +↓↑	diffpattern +↓↑
0.11304206594120514	1.0	0.0	Category = 'sis_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and LogStore = 'sis_operation_log' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and LogStore = 'sis_operation_log' and Method = 'PostLogStoreLogs' and Source = '10.206.8.163' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sis_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and Method = 'PostLogStoreLogs' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sis_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and Method = 'PostLogStoreLogs' and Source = '10.206.8.163' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and LogStore = 'sis_operation_log' and Source = '10.206.8.163' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sis_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and LogStore = 'sis_operation_log' and Source = '10.206.8.163' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sis_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sis-admin' and Source = '10.206.8.163' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0

The following table describes the display items.

Display item	Description
possupport	Support level of positive samples for the mined pattern
posconfidence	Confidence of positive samples for the mined pattern

Display item	Description
<code>negsupport</code>	Support level of negative samples for the mined pattern
<code>diffpattern</code>	Content of the mined pattern

9.11 Root cause analysis function

Log Service provides powerful alerting and analysis capabilities that help you quickly analyze and locate the subdimensions of abnormal metrics. When a time series metric is abnormal, you can use the root cause analysis function to quickly analyze the dimension attributes that result in the abnormal metric. For example, when your page view drops at a specified time, you can use the root cause analysis function to analyze the abnormal dimensions such as the province or city, carrier, and channel. This reduces the time and cost for problem analysis.

`rca_kpi_search`

Function format:

```
select rca_kpi_search ( varchar_array , name_array , real ,
forecast , level )
```

The following table describes the parameters.

Parameter	Description	Value
<code>varchar_array</code>	The dimensions.	Array, for example, array[col1, col2, col3].
<code>name_array</code>	The dimension attributes.	Array, for example, array[' col1', 'col2', 'col3'].
<code>real</code>	The actual value of each dimension specified by the <code>varchar_array</code> parameter.	Double type. Valid values: all real numbers.
<code>forecast</code>	The predicted value of each dimension specified by the <code>varchar_array</code> parameter.	Double type. Valid values: all real numbers.

Parameter	Description	Value
<i>level</i>	The number of dimensions corresponding to the output root cause sets. A value of 0 indicates that all root cause sets that are found are returned.	Long type. Valid values: [0, number of analyzed dimensions]. The number of analyzed dimensions is the number of elements in the array specified by the <code>varchar_array</code> parameter.

Example:

- The statement for query and analysis is as follows:

Use a subquery to obtain the actual value and predicted value of each fine-grained attribute, and then call the `rca_kpi_search` function to analyze the root cause of the exception.

```
* not Status : 200 |
select rca_kpi_search (
  array [ ProjectName , LogStore , UserAgent , Method ],
  array [ ' ProjectName ', ' LogStore ', ' UserAgent ', ' Method ' ],
  real , forecast , 1 )
from (
  select ProjectName , LogStore , UserAgent , Method ,
    sum ( case when time < 1552436040 then real else 0
end ) * 1.0 / sum ( case when time < 1552436040
then 1 else 0 end ) as forecast ,
    sum ( case when time >= 1552436040 then real else 0
end ) * 1.0 / sum ( case when time >= 1552436040
then 1 else 0 end ) as real
  from (
    select __time__ - __time__ % 60 as time , ProjectName
  , LogStore , UserAgent , Method , COUNT (*) as real
  from log GROUP by time , ProjectName , LogStore ,
  UserAgent , Method )
```

```
GROUP BY ProjectName, LogStore, UserAgent, Method
limit 100000000 )
```

- The following figure shows the output result.



The following figure shows the structure of the result.

```
{
  "rcSets": [
    {
      "rcItems": [
        {
          "kpi": [{"attr": "xxx", "val": "xxx"}],
          "nleaf": 100,
          "change": 0.524543,
          "score": 0.1454543
        }
      ]
    }
  ]
}
```

The following table describes the display items.

Display item	Description
<i>rcSets</i>	The root cause sets. Each value is an array.
<i>rcItems</i>	The root cause set.
<i>kpi</i>	The KPI in the root cause set. attr indicates a dimension , and val indicates an attribute in the dimension.
<i>nleaf</i>	The number of leaves that the current KPI covers in the raw data. Each leaf is a log for the finest-grained attributes.

Display item	Description
<i>change</i>	The ratio of changes in the leaves covered by the current KPI to the total changes at the same time point.
<i>score</i>	The abnormality score of the current KPI. Valid values: [0, 1].

9.12 Correlation analysis functions

You can use a correlation analysis function to quickly find the metrics that are correlated with a specified metric or time series data among multiple observed metrics in the system.

Function list

Function	Description
<code>ts_association_analysis</code>	Quickly finds the metrics that are correlated with a specified metric among multiple observed metrics in the system.
<code>ts_similar</code>	Quickly finds the metrics that are correlated with specified time series data among multiple observed metrics in the system.

`ts_association_analysis`

Function format:

```
select
  ts_association_analysis(stamp, params, names, indexName, threshold)
```

The following table describes the parameters.

Parameter	Description	Value
<i>stamp</i>	The Unix timestamp.	Long type.
<i>params</i>	The dimensions of the metrics to be analyzed.	Array of the double type. For example, Latency, QPS, and NetFlow.
<i>names</i>	The names of the metrics to be analyzed.	Array of the varchar type. For example, Latency, QPS, and NetFlow.

Parameter	Description	Value
<i>indexName</i>	The name of the target metric.	Varchar type, for example, Latency.
<i>threshold</i>	The threshold of correlation between the metrics to be analyzed and the target metric.	Double type. Valid values: [0, 1].

Result:

- **name** : the name of the analyzed metric.
- **score** : the value of correlation between the analyzed metric and the target metric. Valid values: [0, 1].

Sample code:

```
* | select ts_association_analysis (
    time ,
    array [ inflow , outflow , latency , status ],
    array [ ' inflow ' , ' outflow ' , ' latency ' , ' status
'],
    ' latency ' ,
    0 . 1 ) from log ;
```

Sample result:

```
| results |
|-----|
| [' latency ' , ' 1 . 0 ' ] |
| [' outflow ' , ' 0 . 6265 ' ] |
| [' status ' , ' 0 . 2270 ' ] |
```

ts_similar

Function format 1:

```
select ts_similar(stamp, value, ts, ds)
select ts_similar(stamp, value, ts, ds, metricType)
```

The following table describes the parameters.

Parameter	Description	Value
<i>stamp</i>	The Unix timestamp.	Long type.
<i>value</i>	The value of the specified metric.	Double type.
<i>ts</i>	The sequence of time for the specified curve.	Array of the double type.

Parameter	Description	Value
<i>ds</i>	The sequence of numeric data for the specified curve .	Array of the double type.
<i>metricType</i>	The type of correlation between the measured curves.	Varchar type. Valid values: SHAPE , RMSE , PEARSON , SPEARMAN , R2 ,and KENDALL

Function format 2:

```
select ts_similar(stamp, value, startStamp, endStamp, step, ds)
select
  ts_similar(stamp, value, startStamp, endStamp, step, ds, metricType )
```

The following table describes the parameters.

Parameter	Description	Value
<i>stamp</i>	The Unix timestamp.	Long type.
<i>value</i>	The value of the specified metric.	Double type.
<i>startStamp</i>	The start timestamp of the specified curve.	Long type.
<i>endStamp</i>	The end timestamp of the specified curve.	Long type.
<i>step</i>	The time interval between two adjacent points in the sequence of time.	Long type.
<i>ds</i>	The sequence of numeric data for the specified curve .	Array of the double type.
<i>metricType</i>	The type of correlation between the measured curves.	Varchar type. Valid values: SHAPE , RMSE , PEARSON , SPEARMAN , R2 ,and KENDALL

Result:

score : the value of correlation between the analyzed metric and the target metric.

Valid values: [-1, 1].

Sample code:

```
* | select vhost , metric , ts_similar ( time , value ,
    1560911040 , 1560911065 , 5 , array [ 5 . 1 , 4 . 0 , 3 . 3 , 5 .
    6 , 4 . 0 , 7 . 2 ] , ' PEARSON ' ) from log group by vhost
    , metric ;
```

Sample result:

vhost	metric	score
vhost1	redolog	- 0 . 3519082537 204182
vhost1	kv_qps	- 0 . 1592216800 9772697
vhost1	file_meta_ write	NaN

10 Advanced analysis

10.1 Case study

Case list

1. [Trigger an alarm when the error 500 percentage increases rapidly](#)
2. [Trigger an alarm when traffic decreases sharply](#)
3. [Calculate the average latency of each bucket set by data interval](#)
4. [Return percentages in GROUP BY results](#)
5. [Count the number of logs that meet the query condition](#)

Trigger an alarm when the error 500 percentage increases rapidly

Count the percentage of error 500 every minute. An alarm is triggered when the percentage exceeds 40% in the last five minutes.

```
status : 500 | select  __topic__ , max_by ( error_coun t ,
window_tim e )/ 1 . 0 / sum ( error_coun t ) as  error_rati o ,
sum ( error_coun t ) as  total_erro r from (
select  __topic__ , count (*) as  error_coun t , __time__
- __time__ % 300 as  window_tim e from  log  group  by
__topic__ , window_tim e

group  by  __topic__  having  max_by ( error_coun t ,
window_tim e )/ 1 . 0 / sum ( error_coun t ) > 0 . 4 and  sum (
error_coun t ) > 500  order  by  total_erro r desc  limit
100
```

Trigger an alarm when traffic decreases sharply

Count the traffic every minute. An alarm is triggered when traffic decreases sharply recently. Data in the last one minute does not cover a full minute. Therefore, divide the statistical value by (max(time) - min(time)) for normalization to count the average traffic per minute.

```
* | SELECT  SUM ( inflow ) / ( max ( __time__ ) - min ( __time__
)) as  inflow_per _minute , date_trunc ( ' minute ' , __time__ )
as  minute  group  by  minute
```

Calculate the average latency of each bucket set by data interval

```
* | select  avg ( latency ) as  latency , case  when
originSize < 5000  then ' s1 ' when  originSize < 20000
then ' s2 ' when  originSize < 500000  then ' s3 ' when
```

```
originSize < 100000000 then 's4' else 's5' end as os
group by os
```

Return percentages in GROUP BY results

List the count results of different departments and the related percentages. This query combines subquery and window functions. `sum(c) over()` indicates to calculate the sum of values in all rows.

```
* | select department , c * 1.0 / sum ( c ) over ( ) from (
  select count ( 1 ) as c , department from log groupby
  department )
```

Count the number of logs that meet the query condition

We must count the URLs by characteristics. In this situation, use the CASE WHEN syntax. You can also use the `count_if` syntax, which is simpler.

```
* | select count_if ( uri like '% login ' ) as login_num
  , count_if ( uri like '% register ' ) as register_n um ,
  date_forma t ( date_trunc ( ' minute ' , __time__ ) , '% m -% d % H
  :% i ' ) as time group by time order by time limit
  100
```

10.2 Optimized queries

This topic describes how to optimize queries to improve query efficiency. You can use the following methods to optimize queries:

- Add shards.
- Reduce the query time range and data volume.
- Repeat queries multiple times.
- Optimize the SQL statement for query.

Add shards

More shards represent more computing resources and faster computing. You can add shards to ensure that the number of logs to be scanned in each shard does not exceed 50 million on average. You can [split shards](#) to add shards.



Note:

Splitting shards incurs more fees and only accelerates queries of new data. Old data is still stored in old shards.

Reduce the query time range and data volume

- The larger the time range, the slower the query. If you query data within a year or a month, data is computed by day. Therefore, you can reduce the time range for faster computing.
- The larger the data volume, the slower the query. Reduce the amount of data to be queried as much as possible.

Repeat queries multiple times

If you find that the result of a query is inaccurate, you can repeat the query multiple times. During each query, the underlying acceleration mechanism makes full use of the previous query result for analysis. Therefore, multiple queries make the query result more accurate.

Optimize the SQL statement for query

A time-consuming query statement has the following characteristics:

- Runs GROUP BY on string columns.
- Runs GROUP BY on more than five columns of fields.
- Includes the operation that generates strings.

You can use the following methods to optimize the query statement:

- Avoid any operation that generates strings if possible.
 - If you use the date_format function to generate a formatted timestamp, the query efficiency is low.

```
* | select date_format ( from_unixtime ( __time__ ) , '% H_% i ' ) as t , count ( 1 ) group by t
```

- If you use the substr() method, strings are generated. We recommend that you use the date_trunc or time_series function to analyze timestamps.
- Avoid running GROUP BY on string columns if possible.

Running GROUP BY on strings may result in a large number of hash calculations, which account for more than 50% of total calculations. For example:

```
* | select count ( 1 ) as pv , date_trunc ( ' hour ' , __time__ ) as time group by time
```

```
* | select count ( 1 ) as pv , from_unixt ime ( __time__ -
__time__ % 3600 ) as time group by __time__ - __time__ %
3600
```

Both query 1 and query 2 calculate the log count per hour. However, query 1 converts the time into a string, for example, 2017-12-12 00:00:00, and then runs GROUP BY on this string. Query 2 calculates the on-the-hour time value, runs GROUP BY on the result, and then converts the value into a string. Query 1 is less efficient than query 2 because the former one needs to hash strings.

- List fields alphabetically based on the initial letter when running GROUP BY on multiple columns.

For example, there are 13 provinces with 100 million users.

```
Fast : * | select province , uid , count ( 1 ) groupby
province , uid
Slow : * | select province , uid , count ( 1 ) groupby uid ,
province
```

- Use estimating functions.

Estimating functions provide much better performance than accurate calculation. Estimation achieves fast calculation by sacrificing accuracy to some acceptable extent.

```
Fast : * | select approx_dis tinct ( ip )
Slow : * | select count ( distinct ( ip ))
```

- Retrieve only required columns in the SQL statement and avoid reading all columns if possible.

Use the query syntax to retrieve all columns. To speed up calculation, retrieve only required columns in the SQL statement if possible.

```
Fast : * | select a , b c
Slow : * | select *
```

- Place non-GROUP BY columns in an aggregate function if possible.

For example, a user ID exactly matches a username. Therefore, run GROUP BY on only userid instead of on both userid and username.

```
Fast : * | select userid , arbitrary ( username ) , count ( 1 )
groupby userid
```

```
Slow : * | select  userid , username , count ( 1 ) groupby  
userid , username
```

- **Avoid using the IN operator if possible.**

Do not use the IN operator in SQL statements if possible. Instead, use the OR operator.

```
Fast : key : a  or  key : b  or  key : c | select  count  
( 1 )  
Slow : * | select  count ( 1 ) where  key  in ( ' a ', ' b ' )
```

11 Log analysis through JDBC

In addition to API, you can use Java Database Connectivity (JDBC) and standard SQL-92 to query and analyze logs.

Connection parameters

Parameter	Example	Description
host	regionid.example.com	#unique_153 The service endpoint. Currently, you can access Log Service through the intranet from a classic network or from Virtual Private Cloud (VPC).
port	10005	The port number. Default value: 10005.
user	bq2sjzesjmo86kq	The AccessKey ID .
password	4fd01fTDDuZP	The AccessKey Secret .
database	sample-project	The project under your account.
table	sample-logstore	The Logstore under the project.

For example, use a MySQL command to connect to Log Service as follows:

```
mysql -hcn - shanghai - intranet . log . aliyuncs . com -
ubq2sjzesj mo86kq - p4fd01fTDD uZP - P10005
use sample - project ; // Uses a project .
```

Prerequisites

- The AccessKey of an Alibaba Cloud account or a RAM user is obtained to access JDBC. The RAM user belongs to the project owner and has the permission to read data from the project.
- MySQL does not support pagination for queries connected through JDBC.

Syntax

Precautions

A WHERE clause must contain `__date__` or `__time__` to limit the time range of a query. The data type of `__date__` is timestamp, and the data type of `__time__` is bigint.

For example:

- `__date__ > ' 2017 - 08 - 07 00 : 00 : 00 ' and __date__ < ' 2017 - 08 - 08 00 : 00 : 00 '`
- `__time__ > 1502691923 and __time__ < 1502692923`

A WHERE clause must contain at least one of the preceding conditions.

Filter syntax

The following table describes the filter syntax in a WHERE clause.

Semantics	Example	Description
String search	<code>key = " value "</code>	Queries data after word-breaking.
String fuzzy search	<code>key = " valu *"</code>	Queries data in fuzzy match mode after word-breaking.
Value comparison	<code>num_field > 1</code>	Supports comparison operators such as <code>></code> , <code>>=</code> , <code>=</code> , <code><</code> , and <code><=</code> .
Logical operation	<code>and or not</code>	For example, <code>a = " x "</code> and <code>b = " y "</code> or <code>a = " x "</code> and <code>not b = " y "</code> .
Full-text search	<code>__line__ = " abc "</code>	Requires the special key (<code>__line__</code>).

Computation syntax

For more information about supported computation operators, see the syntax description in [Real-time analysis](#).

SQL-92 syntax

The SQL-92 syntax is a combination of filter syntax and computation syntax.

The following query is used as an example:

```
status > 200 | select avg ( latency ), max ( latency ) , count ( 1 ) as c GROUP BY method ORDER BY c DESC LIMIT 20
```

According to the standard SQL-92 syntax, the filter and the time condition in the query can be combined into a new query condition as follows:

```
select avg ( latency ), max ( latency ) , count ( 1 ) as c from sample - logstore where status > 200 and __time__ >=
```

```
1500975424 and __time__ < 1501035044 GROUP BY method
ORDER BY c DESC LIMIT 20
```

Access Log Service through JDBC

Application call

You can use the MySQL syntax to connect to Log Service in any application that supports MySQL Connector. For example, you can use JDBC or Python MySQLdb.

Example:

```
import com.mysql.jdbc.*;
import java.sql.*;
import java.sql.Connection;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
public class testjdbc {
    public static void main (String args []){
        Connection conn = null;
        Statement stmt = null;
        try {
            // Step 2 : Register the JDBC driver .
            Class.forName (" com . mysql . jdbc . Driver ");
            // Step 3 : Establish a connection .
            System.out.println (" Connecting to a selected
database ...");
            conn = DriverManager.getConnection (" jdbc :
mysql :// cn - shanghai - intranet . log . aliyuncs . com : 10005 /
sample - project ", " accessid ", " accesskey ");
            System.out.println (" Connected database
successful ly ...") ;
            // Step 4 : Start a query .
            System.out.println (" Creating statement ...");
            stmt = conn.createStatement ();
            String sql = " SELECT method , min ( latency , 10
) as c , max ( latency , 10 ) from sample - logstore where
__time__ >= 1500975424 and __time__ < 1501035044 and
latency > 0 and latency < 6142629 and not ( method = '
Postlogstorelogs ' or method = ' GetLogtail Config ') group
by method ";
            String sql_examp1 e2 = " select count ( 1 ) , max
( latency ) , avg ( latency ) , histogram ( method ) , histogram (
source ) , histogram ( status ) , histogram ( clientip ) , histogram (
__source__ ) from test10 where __date__ >' 2017 - 07 - 20
00 : 00 : 00 ' and __date__ <' 2017 - 08 - 02 00 : 00 : 00
' and __line__ = ' abc # def ' and latency < 100000 and (
method = ' getlogstorelogs ' or method = ' Get **' and method
<> ' GetCursorO rData ' )";
            String sql_examp1 e3 = " select count ( 1 ) from
sample - logstore where __date__ > ' 2017 - 08 - 07
00 : 00 : 00 ' and __date__ < ' 2017 - 08 - 08 00 : 00 :
00 ' limit 100 ";
            ResultSet rs = stmt.executeQuery ( sql );
            // Step 5 : Extract data from the result set
.
            while ( rs.next () ){
                // Retrieves data by column name .
                ResultSetMetaData data = rs.getMetaData ();
                System.out.println ( data.getColumnC ount ());
```

```
        for ( int i = 0 ; i < data . getColumnC ount
        ( ); ++ i ) {
            String name = data . getColumnN ame ( i + 1
            );
            System . out . print ( name + ":" );
            System . out . print ( rs . getObject ( name ) );
        }
        System . out . println ( );
    }
    rs . close ( );
} catch ( ClassNotFoundException e ) {
    e . printStack Trace ( );
} catch ( SQLException e ) {
    e . printStack Trace ( );
} catch ( Exception e ) {
    e . printStack Trace ( );
} finally {
    if ( stmt != null ) {
        try {
            stmt . close ( );
        } catch ( SQLException e ) {
            e . printStack Trace ( );
        }
    }
    if ( conn != null ) {
        try {
            conn . close ( );
        } catch ( SQLException e ) {
            e . printStack Trace ( );
        }
    }
}
}
```

Tool call

On a classic network or in VPC, you can use the MySQL client to connect to Log Service.



Note:

1. Enter your project name at 1.
2. Enter your Logstore name at 2.

12 Query and visualization

12.1 Analysis graph

12.1.1 Graph description

Log Service provides a function similar to the SQL aggregate computing. All the SQL aggregate computing results can be rendered by using the visualized graphs provided by Log Service.



Note:

Before using the visualized graphs, read [#unique_13](#).

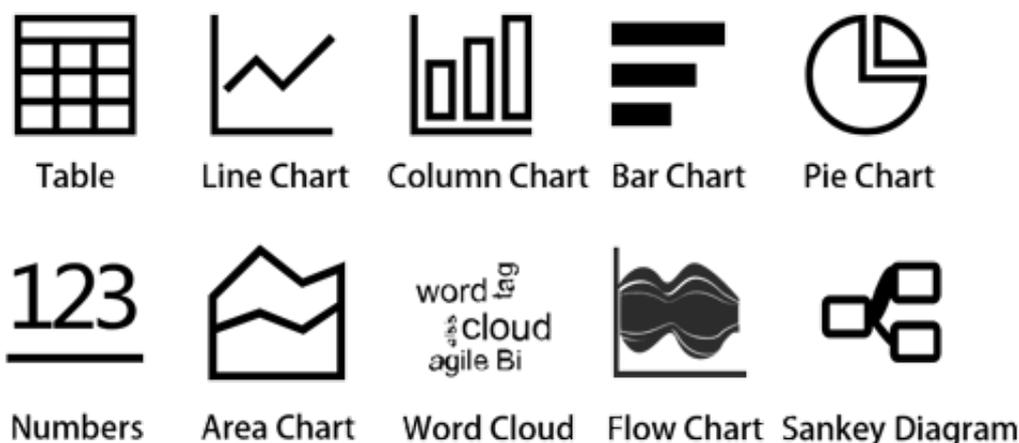
Prerequisites

1. Create an index and enable the analytics.
2. Log Service shows the graphs to you according to the statistical results only when you use the analysis statement for query.

Graph type

Currently, Log Service provides the following types of graphs.

Figure 12-1: Graph type



For how to use each type of graphs, see the following documents:

- [#unique_160](#)
- [#unique_161](#)

- [#unique_162](#)
- [#unique_163](#)
- [#unique_164](#)
- [#unique_165](#)
- [#unique_166](#)
- [#unique_167](#)
- [#unique_168](#)
- [#unique_169](#)

12.1.2 Table

The table is the most common data display form and the most basic data sorting method for quick reference and analysis. Log Service provides a feature similar to SQL aggregate computing. It allows you to use query and analysis statements to obtain results and display the data results in a table.

Components

- Table header
- Row
- Column

where:

- You can use a `SELECT` clause to specify the number of columns.
- The number of rows is computed based on the number of logs in the current time interval. The default clause is `LIMIT 100`.

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab that appears, view the data that is automatically displayed in a table. You do not need to click .
3. On the right-side Properties tab, configure the properties of the table.

Properties

Configuration item	Description
Items per Page	The number of entries to return on each page.
Zebra Striping	Specifies whether to obtain a zebra-striped table.
Transpose Rows and Columns	Click it to transpose rows and columns.
Hide Reserved Fields	Specifies whether to hide reserved fields.
Disable Sorting	Specifies whether to disable the sorting feature.
Disable Search	Specifies whether to disable the search feature.
Highlight Settings	The highlight rules for highlighting rows or columns that conform to rules.

Example

You can filter data in raw logs. The following figure shows a raw log.

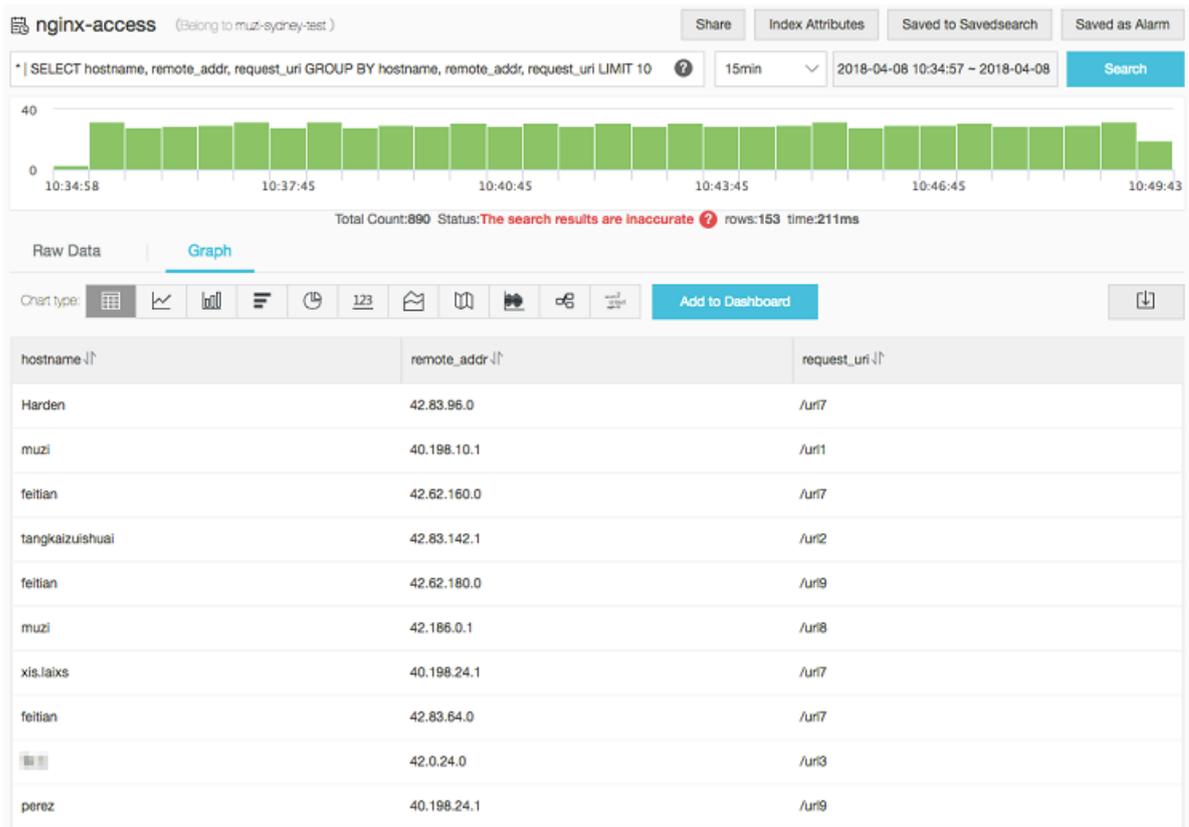
Figure 12-2: Raw log

Time	Content
04-08 10:43:24	<pre> __source__: 127.0.0.1 __topic__: body_bytes_sent: 226 hostname: xis.laixs http_referer: www.host4.com http_user_agent: Mozilla/5.0 (Linux; U; Android 5.1; zh-CN; AoleDior Build/LMY47D) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/40.0.2214.89 UCBrowser/11.5.1.944 Mobile Safari/537.36 http_x_forwarded_for: 101.101.104.0 remote_addr: 40.198.16.2 remote_user: request_method: POST request_time: 0.819 request_uri: /url9 sourceValue: slb2 status: 200 streamValue: 7.943 targetValue: host1 time_local: 08/Apr/2018:10:43:24 upstream_response_time: 1.906 </pre>

1. To filter the `method`, `request_size`, and `request_time` fields in the latest 10 logs, run the following statement:

```
* | SELECT method, request_size, request_time
  GROUP BY method, request_size, request_time
  LIMIT 10
```

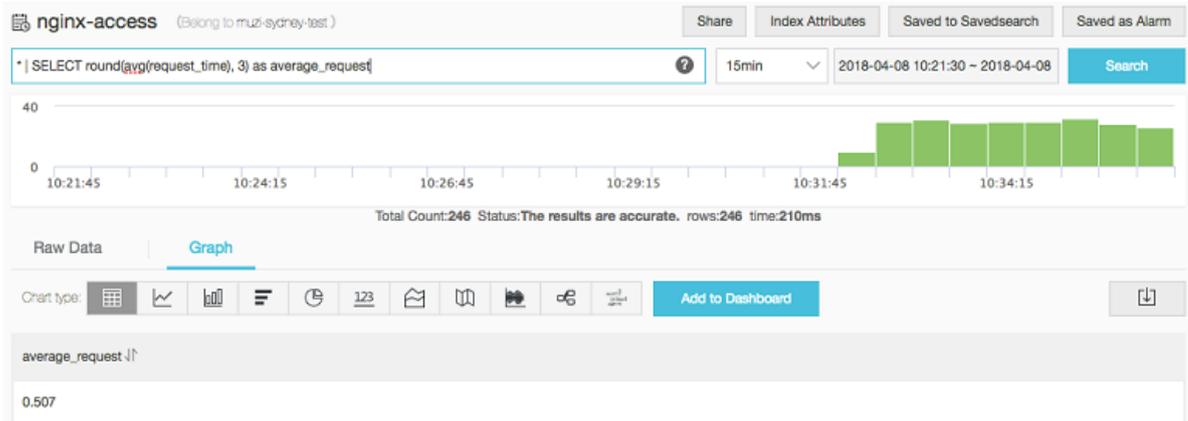
Figure 12-3: Case 1



- To compute the data of a field, for example, the average value of `request_size` (the average request time) in the current time interval, and obtain the result that is accurate to three decimal places, run the following statement:

```
* | SELECT round ( avg ( request_size ), 3 ) as average_request
```

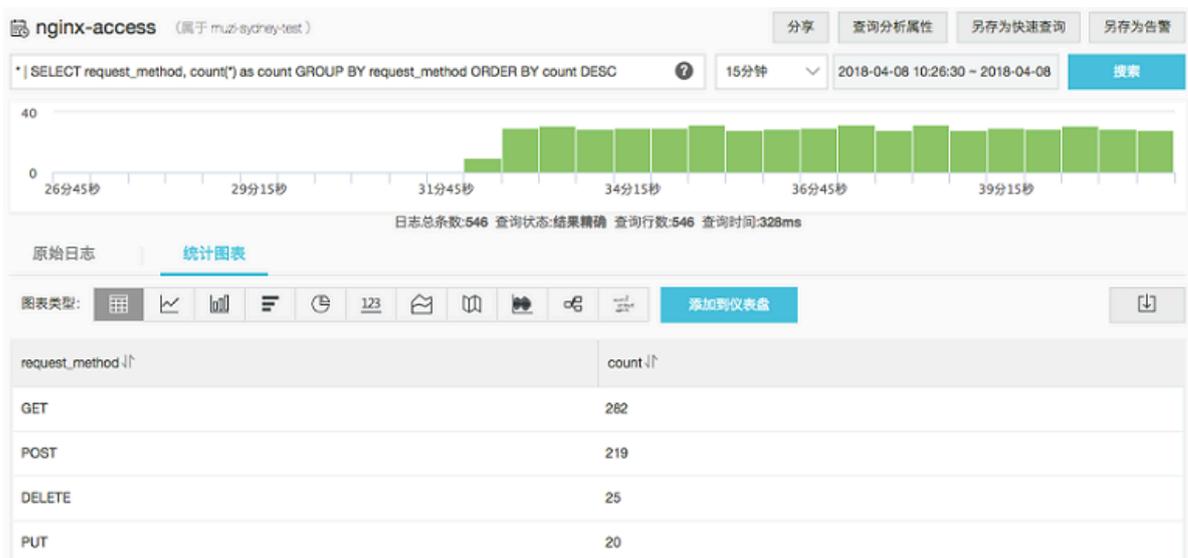
Figure 12-4: Case 2



- To compute grouped data, for example, the distribution of `client_ip` in the current time interval and sort the data in descending order, run the following statement:

```
* | SELECT client_ip , count (*) as count GROUP BY client_ip ORDER BY count DESC
```

Figure 12-5: Case 3



12.1.3 Line chart

The line chart analyzes trends. It is typically used to indicate the changes of a group of data based on an ordered data type (successive time intervals in most cases) for analyzing the trend of data changes.

A line chart clearly shows the changes of data over a period in the following aspects:

- Increment or decrement
- Increment or decrement rate
- Increment or decrement pattern, for example, periodic change
- Peak value and valley value

Therefore, the line chart is the best choice for analyzing the trend of data changes over time. You can also use multiple lines to analyze the changing trend of multiple groups of data in the same period, and then analyze the mutual effect (such as increasing or decreasing at the same time and being inversely proportional to each other) among data in different groups.

Components

- X-axis
- Left Y-axis
- (Optional) Right Y-axis
- Data point
- Line of changing trend
- Legend

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab, click  to select the line chart.
3. On the right-side Properties tab, configure the properties of the line chart.



Note:

In a line chart, a single line must contain more than two data records to guarantee that the data trend can be analyzed. We also recommend that you configure no more than five lines in a line chart.

Properties

Configuration item	Description
X Axis	The data on the X-axis, which is usually a time sequence.
Left Y Axis	The numeric data on the left Y-axis. You can map one or more columns of data to the Y-axis.
Right Y Axis	The numeric data on the right Y-axis. You can map one or more columns of data to the Y-axis. The layer of the right Y-axis is higher than that of the left Y-axis.
Column Marker	The column on the left Y-axis or right Y-axis that is selected as a histogram.
Legend	The position where the legend is located in the chart. Valid values: Top, Bottom, Left, and Right.
RightFormat Left Y-axis	The format in which data is displayed on the Y-axis.
Format Right Y-axis	
Margin	The distance of the axis to the borders of the chart, including Top Margin, Bottom Margin, Right Margin, and Left Margin.

Example

Simple line chart

To query the page views (PVs) of the IP address `10 . 0 . 192 . 0` within the last day, run the following statement:

```
remote_add r : 10 . 0 . 192 . 0 | select date_format (
date_trunc (' hour ', __time__ ), '% m -% d % H :% i ')
```

```
as time , count ( 1 ) as PV group by time order by
time limit 1000
```

Select `time` for X Axis, `PV` for Left Y Axis, and `Bottom` for Legend. Adjust the margins properly.

Figure 12-6: Simple line chart



Dual Y-axis line chart

To query the access PVs and unique visitors (UVs) within the last day, run the following statement:

```
* | select date_format ( date_trunc ( ' hour ', __time__ ), '% m
-% d % H :% i ' ) as time , count ( 1 ) as PV , approx_dis
tinct ( remote_add_r ) as UV group by time order by
time limit 1000
```

Select `time` for X Axis, `PV` for Left Y Axis, `UV` for Right Y Axis, and `PV` for Column Marker.

Figure 12-7: Dual Y-axis line chart



12.1.4 Column chart

The column chart uses vertical or horizontal columns to compare the numeric data among different types. A line chart describes the ordered data, whereas a column chart describes different types of data and counts the number in each data type.

You can also use multiple rectangular blocks to display the data of a type in grouped or stacked mode to analyze the differences of this data type in different dimensions.

Components

- X-axis (horizontal)
- Y-axis (vertical)
- Rectangular block
- Legend

Log Service uses vertical columns in a column chart by default. Each rectangular block has fixed width but varying height to indicate the value. You can use a grouped column chart to display the data if multiple columns of data are mapped to the Y-axis.

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab, click  to select the column chart.
3. On the right-side Properties tab, configure the properties of the column chart.



Note:

Use the column chart if the number of data types is no more than 20. We recommend that you use the `LIMIT` syntax to control the number of data types. Analysis results may not be clearly displayed if the chart contains excessive rectangular blocks. We also recommend that you map no more than five columns of data to the Y-axis.

Properties

Configuration item	Description
X Axis	The types of data.
Y Axis	The numeric data. You can map one or more columns of data to the Y-axis.

Configuration item	Description
Legend	The position where the legend is located in the chart. Valid values: Top, Bottom, Left, and Right.
Format Y-axis	The format in which data is displayed on the Y-axis.
Margin	The distance of the axis to the borders of the chart, including Top Margin, Bottom Margin, Right Margin, and Left Margin.

Example

Simple column chart

To query the number of visits for each `http_referer` in the current time interval, run the following statement:

```
* | select http_referer , count ( 1 ) as count group by http_referer
```

Select `http_referer` for X Axis and `count` for Y Axis.

Figure 12-8: Simple column chart



Grouped column chart

To query the number of visits and the average bytes for each `http_referer` in the current time interval, run the following statement:

```
* | select http_referer , count ( 1 ) as count , avg (
  body_bytes_sent ) as avg group by http_referer
```

Select `http_referer` for X Axis. Select `count` and `avg` for Y Axis.

Figure 12-9: Grouped column chart



12.1.5 Bar chart

The bar chart is another form of column chart, that is, a horizontal column chart. It is typically used for top N analysis and is configured in a way similar to a column chart.

Components

- X-axis (vertical)
- Y-axis (horizontal)
- Rectangular block
- Legend

Each rectangular block has fixed height but varying width to indicate the value.

You can use a grouped bar chart to display the data if multiple columns of data are mapped to the Y-axis.

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab, click  to select the bar chart.

3. On the right-side Properties tab, configure the properties of the bar chart.



Note:

- Use the bar chart if the number of data types is no more than 20. We recommend that you use the `LIMIT` syntax to control the number of data types. Analysis results may not be clearly displayed if the chart contains excessive rectangular blocks. You can use a `ORDER BY` clause for top N analysis. We also recommend that you map no more than five columns of data to the Y-axis.
- Use the grouped bar chart only if different types of data are simultaneously increased or decreased.

Properties

Table 12-1: Configuration items

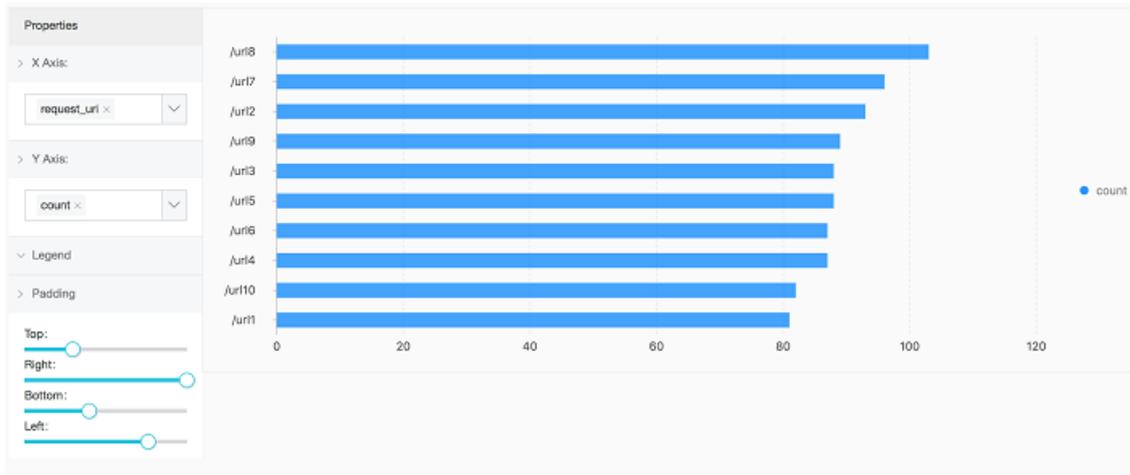
Configuration item	Description
X Axis	The types of data.
Y Axis	The numeric data. You can map one or more columns of data to the Y-axis.
Legend	The position where the legend is located in the chart. Valid values: Top, Bottom, Left, and Right.
Format X-axis	The format in which data is displayed on the X-axis.
Margin	The distance of the axis to the borders of the chart, including Top Margin, Bottom Margin, Right Margin, and Left Margin.

Example

To analyze the `request_uri` with the top 10 largest number of visits and display the analysis results in a simple bar chart, run the following statement:

```
* | select request_uri, count(1) as count group by
  request_uri order by count desc limit 10
```

Figure 12-10: Simple bar chart



12.1.6 Pie chart

The pie chart is used to indicate the ratios of different data types and compare different data types based on the arc length. A pie chart is divided into multiple sectors based on the percentages of various data types. The entire chart indicates the sum of data. Each sector (arc-shaped) indicates the ratio of a data type to the sum. The sum of percentages in all sectors is equal to 100%.

Components

- Sector
- Text percentage
- Legend

Type

Log Service provides three types of pie charts: the default pie chart, the donut chart, and the polar area chart.

Donut chart

A donut chart is a pie chart with a hollow center and has the following advantages:

- The donut chart adds the display of sum based on the original structure.
- You may find it difficult to understand the differences between two pie charts simply by comparing them. However, you can compare the ring length of two donut charts to obtain intuitive comparison results.

Polar area chart

A polar area chart is not a donut chart, but a column chart in the polar coordinate system. The data types are divided by arcs and the radius of the arc indicates the data volume. Compared with a pie chart, a polar area chart has the following advantages:

- The pie chart is suitable if the number of data types is no more than 10. The polar area chart is applicable to scenarios where the number of data types ranges from 10 to 30.
- The area is the square of radius. Therefore, the polar area chart enlarges the differences among different types of data. It is especially suitable for the comparison of similar values.
- A circle shows a periodic pattern. Therefore, the polar area chart can also be used to indicate a periodic time, such as weeks and months.

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab, click  to select the pie chart.
3. On the right-side Properties tab, configure the properties of the pie chart.



Note:

- Use the pie chart or donut chart if the number of data types is no more than 10. We recommend that you use the `LIMIT` syntax to control the number of data types. Analysis results may not be clearly displayed if the chart contains excessive sectors of different colors.
- Use the polar area chart or column chart if the number of data types is more than 10.

Properties

Configuration item	Description
Chart Types	The type of the chart. Valid values: Pie Chart, Donut Chart, and Polar Area Chart . Default value: Pie Chart.
Legend Filter	The types of data.
Value Column	The values corresponding to different types of data.
Legend	The position where the legend is located in the chart. Valid values: Top, Bottom, Left, and Right.
Format	The format in which data is displayed.
Margin	The distance of the axis to the borders of the chart, including Top Margin, Bottom Margin, Right Margin, and Left Margin.

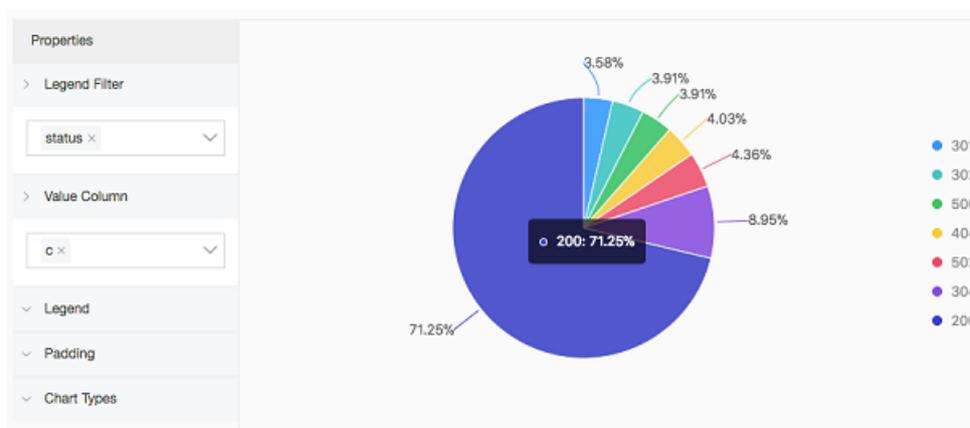
Example

Pie chart

To analyze the ratio of the access `requestURI` , run the following statement:

```
* | select requestURI as uri , count ( 1 ) as c group
  by uri limit 10
```

Figure 12-11: Pie chart

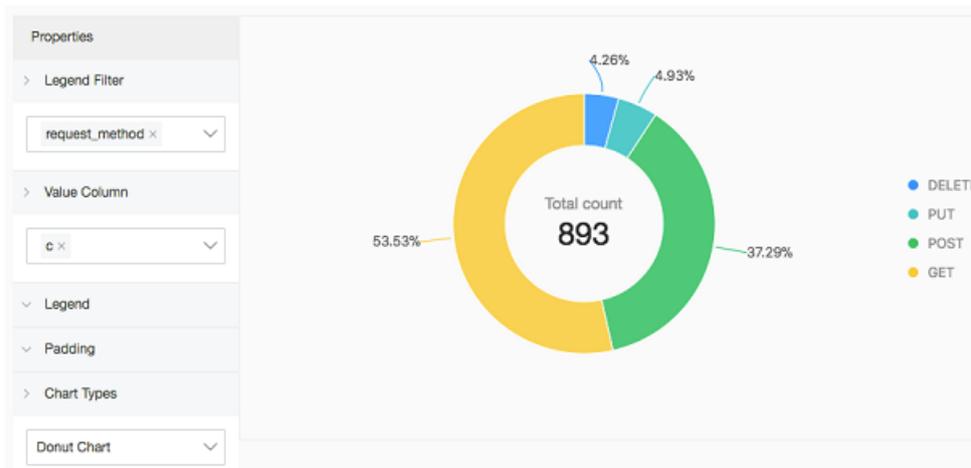


Donut chart

To analyze the ratio of the access `requestURI` , run the following statement:

```
* | select requestURI as uri , count ( 1 ) as c group
  by uri limit 10
```

Figure 12-12: Donut chart

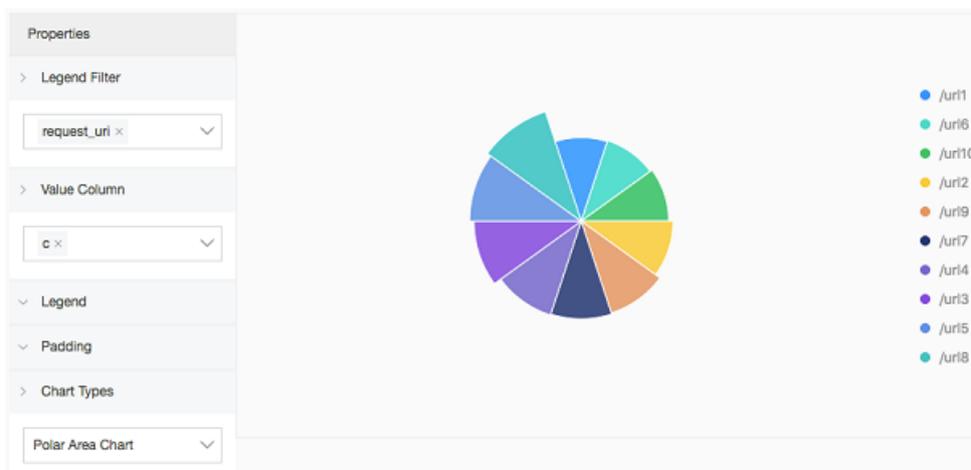


Polar area chart

To analyze the ratio of the access `requestURI` , run the following statement:

```
* | select requestURI as uri , count ( 1 ) as c group
  by uri limit 10
```

Figure 12-13: Polar area chart



12.1.7 Area chart

Based on a line chart, the area chart fills the section between a line and the axis with color. The filled section is an area block and the color highlights the trend. Similar

to a line chart, an area chart emphasizes the number changes over time, and is used to highlight the trend of the total number. Both the line chart and the area chart are mostly used to indicate the trend and relationship, but not to display specific values.

Components

- X-axis (horizontal)
- Y-axis (vertical)
- Area block

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab, click  to select the area chart.
3. On the right-side Properties tab, configure the properties of the area chart.



Note:

In an area chart, a single area block must contain more than two data records to guarantee that the data trend can be analyzed. We also recommend that you configure no more than five area blocks in an area chart.

Properties

Configuration item	Description
X Axis	The data on the X-axis, which is usually a time sequence.
Y Axis	The numeric data. You can map one or more columns of data to the Y-axis.
Legend	The position where the legend is located in the chart. Valid values: Top, Bottom, Left, and Right.
Format	The format in which data is displayed.
Margin	The distance of the axis to the borders of the chart, including Top Margin, Bottom Margin, Right Margin, and Left Margin.

Example

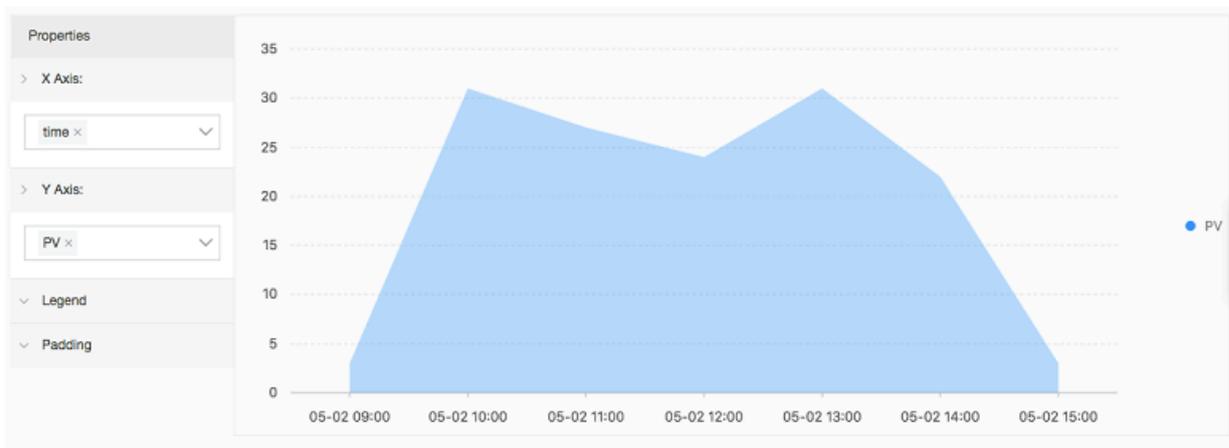
Simple area chart

To query the page views (PVs) of the IP address `10 . 0 . 192 . 0` within the last day, run the following statement:

```
remote_addr: 10 . 0 . 192 . 0 | select date_format (
date_trunc (' hour ', __time__ ), '% m -% d % H :% i ') as time
, count ( 1 ) as PV group by time order by time
limit 1000
```

Select `time` for X Axis and `PV` for Y Axis.

Figure 12-14: Simple area chart



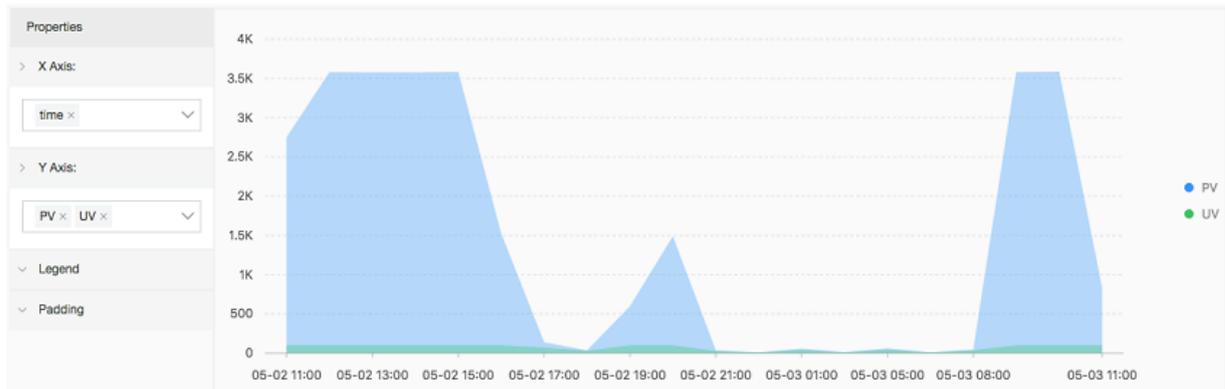
Stacked area chart

```
* | select date_format ( date_trunc (' hour ', __time__ ), '% m
-% d % H :% i ') as time , count ( 1 ) as PV , approx_dis
```

```
tinct ( remote_add r ) as UV group by time order by
time limit 1000
```

Select `time` for X Axis. Select `PV` and `UV` for Y Axis.

Figure 12-15: Stacked area chart



12.1.8 Individual value plot

The individual value plot highlights a single value. Individual value plots include the following types:

- **Rectangle Frame:** displays a general value.
- **Dial:** shows how close the current value is to the configured threshold.
- **Compare Numb Chart:** shows the SQL query results of parallel and period-on-period comparison functions. For more information about the analysis syntax, see [#unique_177](#).

Rectangle Frame is selected by default. A rectangle frame is the simplest and most direct data representation that visually and clearly displays the data at a certain point. It is generally used to show the key information at a certain time point. To display a proportional metric, you can use a dial.

Components

- Main text
- (Optional) Unit
- (Optional) Description
- Type

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. On the Graph tab, click 123 to select the individual value plot.
3. On the right-side Properties tab, configure the properties of the individual value plot.



Note:

Log Service automatically normalizes data in charts based on the value. For example, `230000` is processed as `230K`. You can use `#unique_178` to define your own numeric format in the real-time analysis phase.

Properties

- The following table describes the configuration items of a rectangle frame.

Configuration item	Description
Chart Types	The type of the chart. Select Rectangle Frame.
Value Column	The value displayed in the chart. The data in the first row of the specified column is displayed by default.
Unit	The unit of data, displayed after the value.
Unit Font Size	The font size of the unit. You can drag the slider to adjust the font size. Valid values: [10, 100]. Unit: pixels.
Description	The description of the value, displayed under the value.
Description Font Size	The font size of the value description. You can drag the slider to adjust the font size. Valid values: [10, 100]. Unit: pixels.
Format	The format in which data is displayed.
Font Size	The font size of the value. You can drag the slider to adjust the font size. Valid values: [10, 100]. Unit: pixels.
Font Color	The color of the number and text. You can select a recommended color or customize a color.
Background Color	The color of the background. You can select a recommended color or customize a color.

- The following table describes the configuration items of a dial.

Configuration item	Description
Chart Types	The type of the chart. Select Dial to display query results on a dial.
Actual Value	The actual value in the chart. The data in the first row of the specified column is displayed by default.
Unit	The unit of the value on the dial.
Font Size	The font size of the value and unit. Valid values: [10, 100]. Unit: pixels.
Description	The description of the value, displayed under the value.
Description Font Size	The font size of the value description. You can drag the slider to adjust the font size. Valid values: [10, 100]. Unit: pixels.
Dial Maximum	The maximum value of the scale on the dial. Default value: 100.
Maximum Value Column	The maximum value in the specified column. When Use Query Results is enabled, Dial Maximum is replaced by Maximum Value Column. You can select the maximum value from query results for this configuration item.
Use Query Results	Specifies whether to select a value from query results. When Use Query Results is enabled, you can select the maximum value from query results for Maximum Value Column.
Format	The format in which data is displayed.
Colored Regions	The number of value regions that the dial is divided into. Each region is displayed in a different color. Valid values: 2, 3, 4, and 5. Default value: 3.

Configuration item	Description
Region Max Value	<p>The maximum value of the scale in each colored region of the dial. By default, the maximum value in the last region is the maximum value on the dial. You do not need to specify this value.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: A dial is evenly divided into three colored regions by default. If you change the value of Colored Regions, the dial is still evenly divided based on the changed value. You can set the maximum value for each colored region based on your needs. </div>
Font Color	The color of the value on the dial.
Region	<p>The colored region that the dial is divided into. A dial is evenly divided into three regions by default, which are displayed in blue, yellow, and red, respectively.</p> <p>If you set Colored Regions to a value greater than 3, the added regions are displayed in blue by default. You can change the color of each region.</p>
Show Title	<p>Specifies whether to display the title of the dial when you add it to a dashboard as an individual value plot. You can enable or disable Show Title to show or hide the title of the individual value plot on the dashboard page. Default value: disabled, indicating that the title of the dial is not displayed.</p> <p>After you enable Show Title, the title of the dial is not displayed on the current page. You need to create or modify a dashboard and view the title on the dashboard page.</p>

- The following table describes the configuration items of a comparison chart.

Configuration item	Description
Chart Types	The type of the chart. Select Compare Numb Chart to display query results in a comparison chart.

Configuration item	Description
Show Value	The value displayed in the center of the comparison chart. This value is generally set to the statistical result calculated by the relevant comparison function in the current time range.
Compare Value	The value used to compare with the threshold. This value is generally set to the comparison result between the statistical result calculated by the relevant comparison function in the current time range and that in the previous time range.
Font Size	The font size of the show value. Valid values: [10, 100]. Unit: pixels.
Unit	The unit of the show value, displayed after the value.
Unit Font Size	The font size of the unit for the show value. Valid values: [10, 100]. Unit: pixels.
Compare Unit	The unit of the compare value, displayed after the value.
Compare Font Size	The font size of the compare value and its unit. Valid values: [10, 100]. Unit: pixels.
Description	The description of the show value and its growth trends, displayed under the value.
Description Font Size	The font size of the value description. Valid values: [10, 100]. Unit: pixels.

Configuration item	Description
Trend Comparison Threshold	<p>The value used to measure the variation trend of the compare value.</p> <p>For example, the difference between the compare value and the threshold is -1:</p> <ul style="list-style-type: none"> - If you set Trend Comparison Threshold to 0, a down arrow is displayed on the page, indicating a value decrease. - If you set Trend Comparison Threshold to -1, the system determines that the value remains unchanged and does not display the trend on the page. - If you set Trend Comparison Threshold to -2, an up arrow is displayed on the page, indicating a value increase.
Format	The format in which data is displayed.
Font Color	The color of the show value and value description.
Growth Font Color	The font color of the compare value that is greater than the threshold.
Growth Background Color	The background color displayed when the compare value is greater than the threshold.
Decrease Font Color	The font color displayed when the compare value is less than the threshold.
Decrease Background Color	The background color displayed when the compare value is less than the threshold.
Equal Background Color	The background color displayed when the compare value is equal to the threshold.

Example

Run the following query and analysis statements to view the number of visits and display the analysis results in charts:

· Rectangle frame

```
* | select count ( 1 ) as pv
```

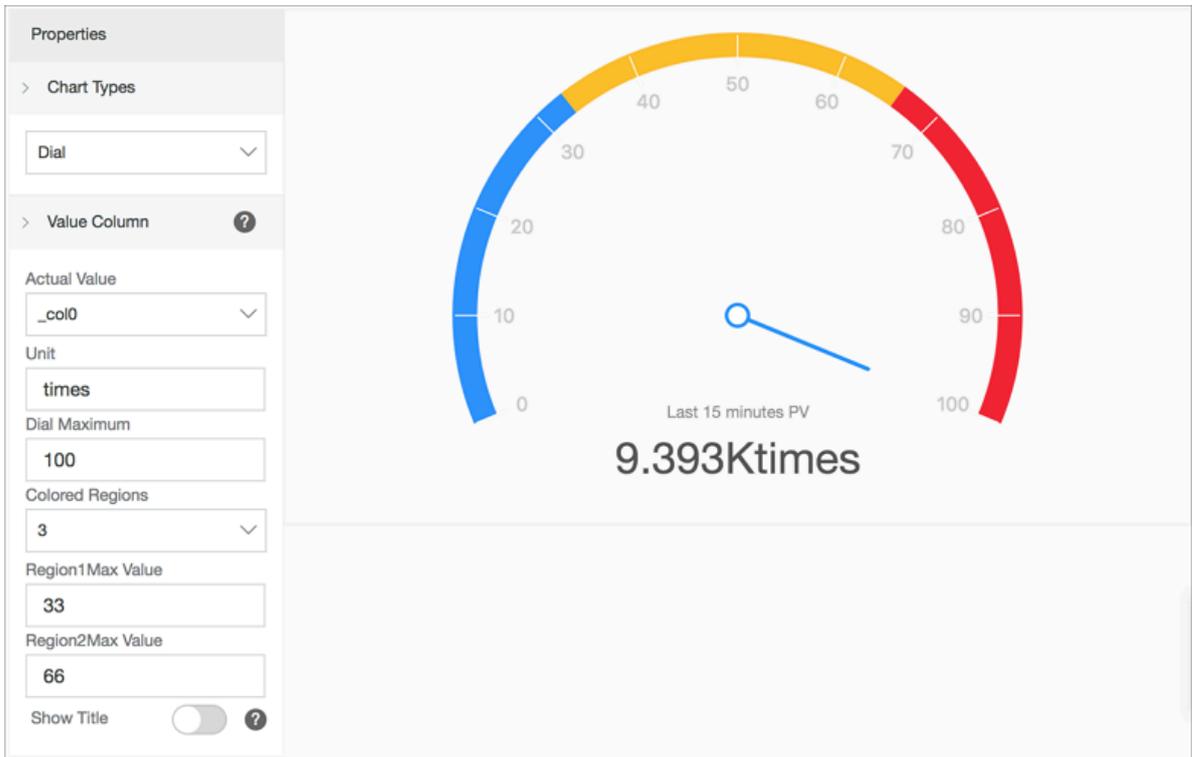
The image shows a configuration interface for a visualization tool. On the left is a 'Properties' sidebar with the following sections:

- Chart Types:** A dropdown menu set to 'Rectangle Frame'.
- Value Column:** A dropdown menu set to '_col0'.
- Color:** Includes 'Font Color' (set to black) and 'Background Color' (set to cyan).
- Text:** Includes 'Font Size' (a slider), 'Unit' (a text input set to 'Times'), 'Unit Font Size' (a slider), and 'Description' (a text input set to 'Last 15 minutes PV').

The main visualization area on the right is a cyan rectangle containing the text '8.84K Times' and 'Last 15 minutes PV' below it.

· Dial

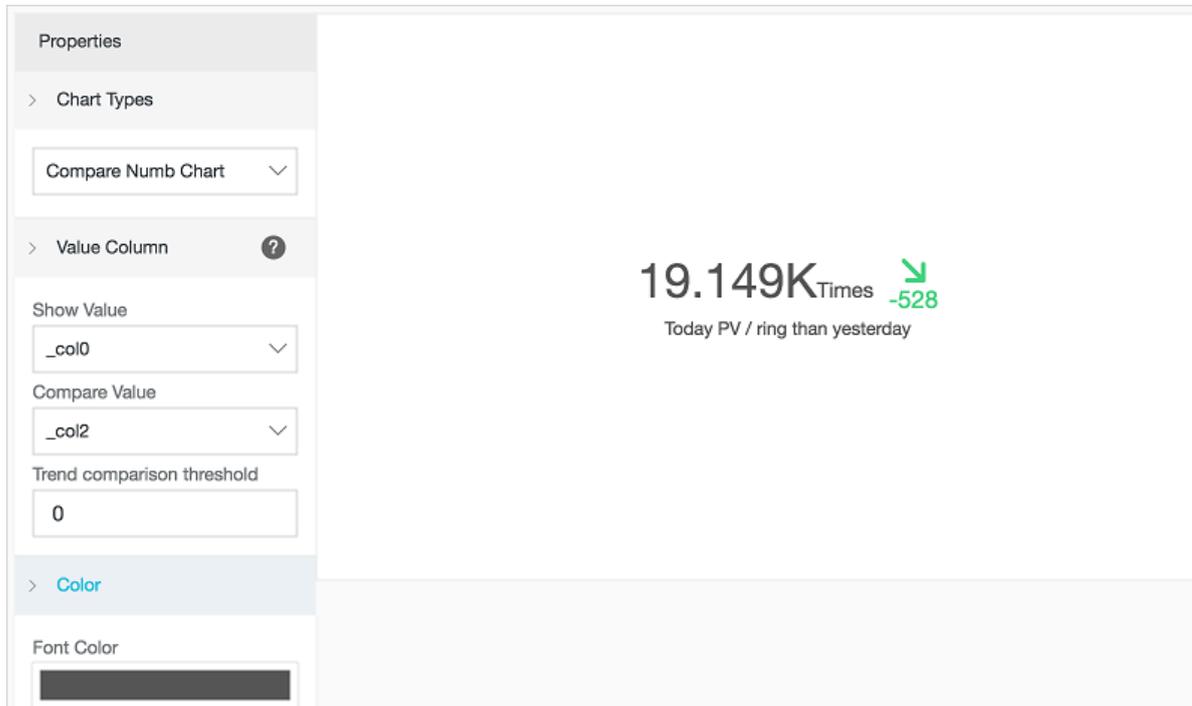
```
* | select count ( 1 ) as pv
```



- Comparison chart

View the comparison of today's visits and yesterday's visits:

```
* | select diff [ 1 ], diff [ 2 ], diff [ 1 ]- diff [ 2 ] from
( select compare ( pv , 86400 ) as diff from ( select
count ( 1 ) as pv from log ))
```



12.1.9 Progress bar

The progress bar shows the percentage. You can configure the properties of the progress bar to adjust its style and set display rules.

Components

- Actual value
- (Optional) Unit
- Total value

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
2. Click  to select the progress bar.
3. Configure the properties of the progress bar.

Properties

Configuration item	Description
Actual Value	The actual value in the chart. The data in the first row of the specified column is displayed by default.
Unit	The unit of the value in the progress bar.
Total Value	The total value indicated by the progress bar. Default value: 100.
Maximum Value Column	The maximum value in the specified column. When Use Query Results is enabled, Total Value is replaced by Maximum Value Column. You can select the maximum value from query results for this configuration item.
Use Query Results	Specifies whether to select a value from query results. When Use Query Results is enabled, you can select the maximum value from query results for Maximum Value Column.
Edge Shape	The edge shape of the progress bar.
Vertical Display	Specifies whether to display the progress bar in vertical mode.
Font Size	The font size of the progress bar.
Thickness	The thickness of the progress bar.
Background Color	The background color of the progress bar.
Font Color	The font color of the progress bar.
Default Color	The default color of the progress bar.
Color Display Mode	The color display mode of the progress bar.
Start Color	The start color of the progress bar. This configuration item is available when Color Display Mode is set to Gradient.
End Color	The end color of the progress bar. This configuration item is available when Color Display Mode is set to Gradient.

Configuration item	Description
Display Color	<p>The display color of the progress bar. This configuration item is available when Color Display Mode is set to Display by Rule.</p> <div style="background-color: #f0f0f0; padding: 5px;">  Note: The value of Actual Value is compared with that of Threshold based on the condition specified by Operator. If the actual value matches the condition specified by Operator, the progress bar is displayed in the color specified by Display Color. Otherwise, the progress bar is displayed in the default color. </div>
Operator	The condition for determining the display color of the progress bar. This configuration item is available when Color Display Mode is set to Display by Rule.
Threshold	The threshold for determining the display color of the progress bar. This configuration item is available when Color Display Mode is set to Display by Rule.

Scenarios

You can use the progress bar to display the percentage of a metric or the proportion of data.

```
* | select diff [ 1 ], diff [ 2 ] from ( select compare ( pv
, 86400 ) as diff from ( select count ( 1 ) as pv from
log ))
```

When you select Display by Rule as the color display mode, the progress bar changes its color according to the specified condition. If the specified condition is not met, the progress bar is displayed in the default color.

12.1.10 Map

You can add color blocks and marks to a map to display geographic data. Log Service provides three kinds of maps: Map of China, World Map, and AMap. Among them,

AMap offers the scatter chart and heat map. You can use specific functions in query and analysis statements to display analysis results on different maps.

Components

- Map canvas
- Color block

Properties

Configuration item	Description
Location information	The location information recorded in logs. The dimension varies with the map as follows: <ul style="list-style-type: none"> · Provinces (Map of China) · Country (World Map) · Longitude/Latitude (AMap)
Value Column	The data volume of the location information.

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.
 - Map of China: Use the `ip_to_province` function.
 - World Map: Use the `ip_to_country` function.
 - AMap: Use the `ip_to_geo` function.
2. Click  to select the map.
3. Configure the properties of the map.

Scenarios

Map of China

You can use the `ip_to_province` function to generate a map of China.

- SQL statement:

```
* | select ip_to_province ( remote_address ) as address ,
count ( 1 ) as count group by address order by count
desc limit 10
```

- Dataset:

address	count
Guangdong	163
Zhejiang	110
Fujian	107
Beijing	89
Chongqing	28
Heilongjiang	19

- Select `address` for Provinces and `count` for Value Column.

World map

You can use the `ip_to_country` function to generate a world map.

- SQL statement:

```
* | select ip_to_country ( remote_address ) as address ,
count ( 1 ) as count group by address order by count
desc limit 10
```

- Dataset:

address	count
China	8354
United States	142

- Select `address` for Country and `count` for Value Column.

AMap

You can use the `ip_to_geo` function to generate an AMap. The address column in the dataset contains the latitude and longitude in sequence, which are separated with a comma (,). If the longitude and latitude are separated in the lng column and the lat column, you can use the `concat (' lat ', ',', lng ')` function to merge the two columns.

- SQL statement:

```
* | select ip_to_geo ( remote_add r ) as address , count (
  1 ) as count group by address order by count desc
  limit 10
```

- Dataset:

address	count
39.9289,116.388	771
39.1422,117.177	724
29.5628,106.553	651
30.2936,120.161420	577
26.0614,119.306	545
34.2583,108.929	486

- Select `address` for Longitude/Latitude and `count` for Value Column.

The scatter chart is used by default. If data points are densely distributed on the map, you can switch to the heat map.

12.1.11 Flow diagram

The flow diagram, also known as ThemeRiver, is a stacked area chart around the central axis. The banded branches with different colors indicate different types of information. The band width indicates the corresponding numeric value. In addition, the centralized time attribute of the original data maps to the X-axis, which forms a three-dimensional relationship.

You can switch a flow diagram to a line chart or column chart. Note that the column chart is displayed in the stacked form by default, and the start point of each data type is at the top of the last column.

Components

- X-axis (horizontal)
- Y-axis (vertical)
- Band

Procedure

1. On the search page of a Logstore, enter a query and analysis statement in the search box, set the time range, and then click Search & Analysis.

2. On the Graph tab, click  to select the flow diagram.
3. On the right-side Properties tab, configure the properties of the flow diagram.

Properties

Configuration item	Description
Chart Types	The type of the chart. Valid values: Line Chart, Area Chart, and Column Chart. Default value: Line Chart.
X Axis	The data on the X-axis, which is usually a time sequence.
Y Axis	The numeric data. You can map one or more columns of data to the Y-axis.
Aggregate Column	The information required to be aggregated in the third dimension.
Legend	The position where the legend is located in the chart. Valid values: Top, Bottom, Left, and Right.
Format	The format in which data is displayed.
Margin	The distance of the axis to the borders of the chart, including Top Margin, Bottom Margin, Right Margin, and Left Margin.

Example

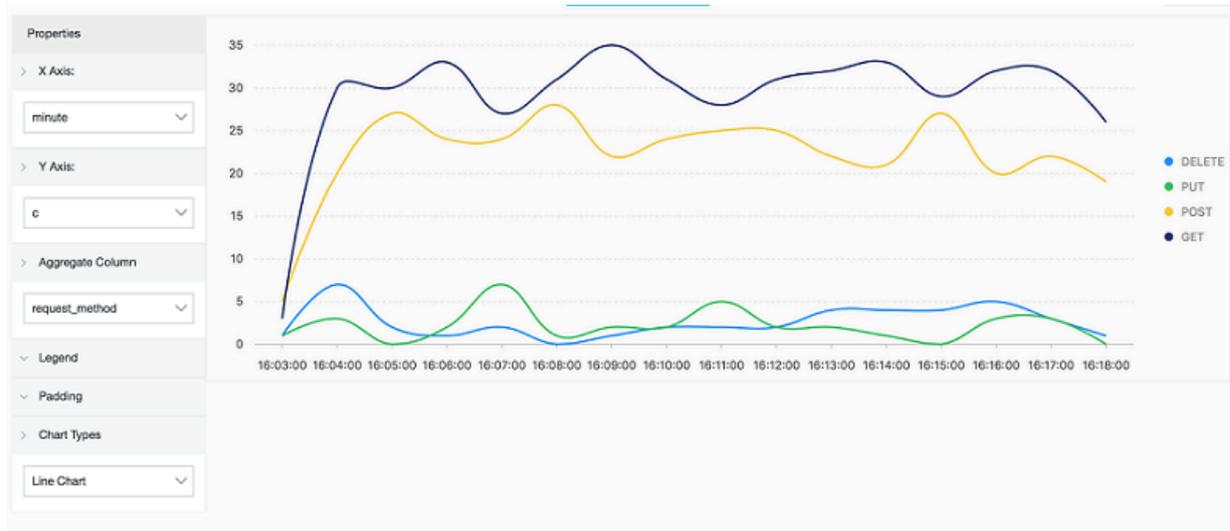
The flow diagram is suitable for the display of a three-dimensional relationship among time, type, and value.

```
* | select date_format ( from_unixtime ( __time__ - __time__
% 60 ), '% H :% i :% S ' ) as minute , count ( 1 ) as c ,
```

```
request_me thod group by minute , request_me thod order
by minute asc limit 100000
```

Select `minute` for X Axis, `c` for Y Axis, and `request_me thod` for Aggregate Column.

Figure 12-16: Flow diagram



12.1.12 Sankey diagram

Sankey diagram, a specific type of flow chart, is used to describe the flow from one set of values to another, and is applicable to scenarios such as network flow data. Generally, the Sankey diagram contains three sets of values: `source` , `target` , and `value` . `source` and `target` describes the edge relationship between nodes, and `value` describes the relationship between `source` and `target` .

Function features

The Sankey diagram has the following features:

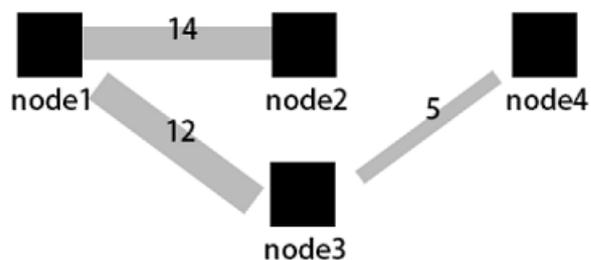
- The start flow is the same as the end flow. The overall width of all the main branches is the same as that of all the branches. Therefore, the energy balance is maintained.
- Internally, different lines indicate the distribution of different flows, and the line width indicates the flow occupied by the branch proportionally.
- Different node widths indicate the flows in the particular statuses.

For example, the following data can be displayed in a Sankey diagram.

Source	Target	Value
node1	node2	14
node1	node3	12
node3	node4	5
...

The Sankey diagram visualizes the preceding data as follows.

Figure 12-17: Sankey diagram data relation



Basic components

- Node
- Edge

Configuration items

Configuration item	Description
Start column	Describes the start node.
End column	Describes the end node.
A value column	The value that links the start node and the end node.
Spacing	The distance between the coordinate axis and the graph boundary.

Procedure

1. On the query page, enter the query statement in the search box, select the time interval, and click Search.
2. Click the Graph and select the Sankey diagram .
3. Configure the graph properties.

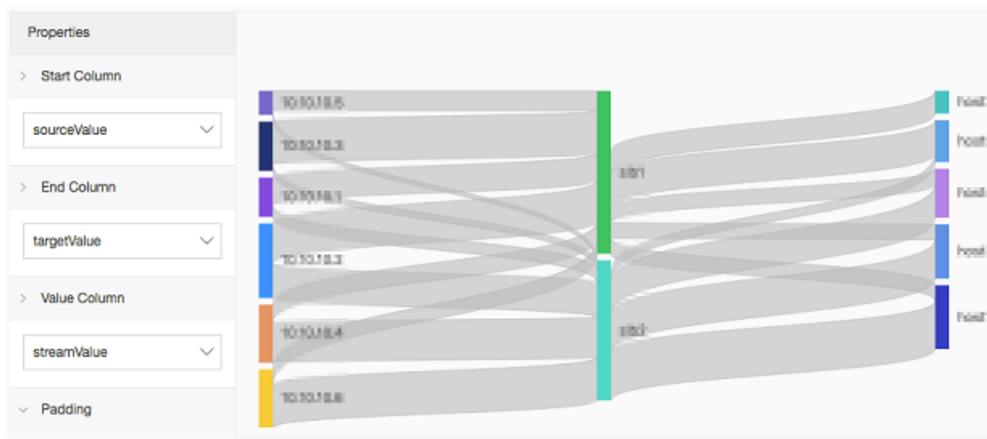
Examples

Ordinary Sankey diagram

If the log field contains `source` , `target` , and `value` , each log itself is the relationship between nodes and edges, and the sum of `streamValue` can be obtained using [#unique_46](#) Nested subquery.

```
* | select sourceValue , targetValue , sum ( streamValue )
   as streamValue from ( select sourceValue , targetValue
   , streamValue , __time__ from log group by sourceValue
   , targetValue , streamValue , __time__ order by __time__
   desc ) group by sourceValue ,
   targetValue
```

Figure 12-18: Ordinary Sankey diagram



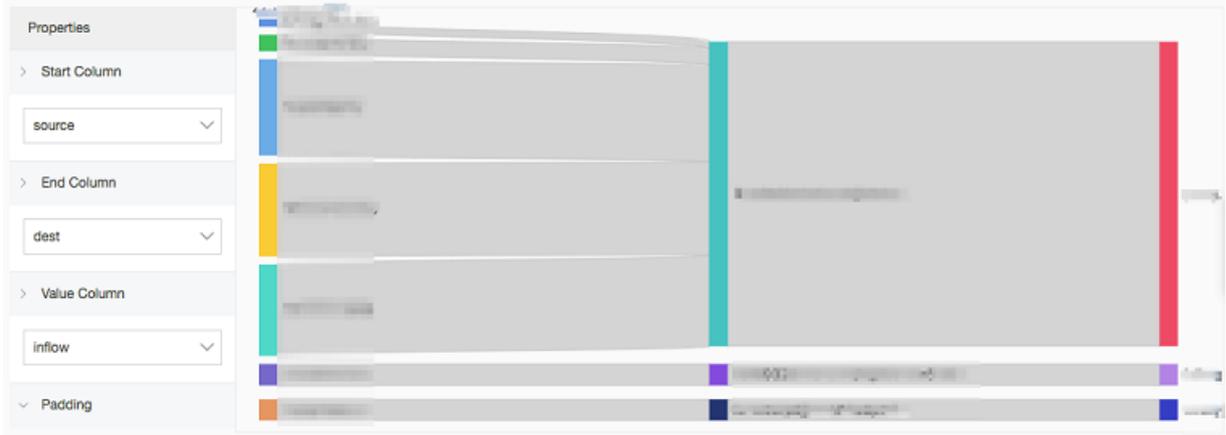
Access logs of Layer-7 Server Load Balancer scenario

Log Service supports [#unique_183](#), so you can use access logs to build a Sankey diagram.

```
* | select COALESCE ( client_ip , slbid , host ) as source ,
   COALESCE ( host , slbid , client_ip ) as dest , sum ( request_le
```

```
length ) as inflow group by grouping sets ( ( client_ip ,
slbid ), ( slbid , host ))
```

Figure 12-19: nested subquery



12.1.13 Word cloud

Word cloud is the visual representation of text data, uses words to form a colorful graph similar to a cloud, and is used to display large amounts of text data. The font size or color is determined by the significance of the word, which allows you to perceive the weight of some keywords quickly.

Basic components

The word cloud shows you the words after being computed and sorted.

Configuration items

Configuration item	Description
Word column	The words to be displayed.
Value column	The value corresponding to each word.
Font size	Adjust the font size range properly to apply to the canvas. <ul style="list-style-type: none"> • Maximum font size (50–80 px) • Minimum font size (12–24 px)
Padding	The distance between the coordinate axis and the graph boundary.

Procedure

1. On the query page, enter the query statement in the search box, select the time interval, and then click Search.
2. Click the Graph  tab and select the word cloud undefined.
3. Configure the graph properties.

Example

Analyze the distribution of the hostnames in the Nginx logs:

```
* | select hostname , count ( 1 ) as count group by
  hostname order by count desc limit 1000
```

Select `hostname` as the Word Column and `count` as the Value Column.

12.1.14 Tree map

A tree map is a rectangle chart that contains rectangle blocks in a tree structure. The area of each rectangle block in a tree map is proportional to the amount of data it represents. The larger the area is, the greater the proportion of data it represents.

Components

Rectangle blocks are generated from data calculations and distributed in the chart.

Configuration

Configuration	Description
Legend filter	Field that indicates a data type.
Value column	Value field. The greater the value of a data type, the larger the corresponding rectangle block will be.
Padding	The spacing between any two adjacent sides of different rectangle blocks. The value range of this field is 0–100 px.

Procedure

1. Enter a query statement, select a time interval, and then click Search & Analysis.
2. Select the tree map  .
3. Configure the chart properties.

Example

Analyze the distribution of the hostnames in the Nginx logs.

```
* | select hostname , count ( 1 ) as count group by
  hostname order by count desc limit 1000
```

Select `hostname` from the Legend Filter drop-down list and select `count` from the Value Column drop-down list.



12.2 Dashboard

12.2.1 Dashboard overview

This topic describes the functions of charts and dashboards provided by Log Service for real-time data analysis.

A chart provides the same functionality as a query and analysis statement. One or multiple charts can be added to and subsequently viewed from a dashboard. When you open or refresh a dashboard, each chart in the dashboard automatically runs the query and analysis statement that it represents.

Log Service also provides the [Console embed](#) function that allows you to embed a dashboard into an external web page and choose to view the dashboard from either the Log Service console, or the newly embedded web page. Additionally, Log Service also provides the [Drill-down analysis](#) function. With this function, you can set drill-down analysis configurations for a chart when you add the chart to a dashboard, and then use the function to obtain more detailed analysis results.

Limits

- You can create a maximum of 50 dashboards for a project.
- Each dashboard can contain a maximum of 50 charts.

Try out the dashboard function

Account: sls-reader1

Password: pnX-32m-MHH-xbm

[Open the trial dashboard console.](#)

Functions

A dashboard can be either in display mode or editing mode.

- **Display mode**

When you select display mode for a dashboard, you can set the parameters for displaying the analysis results as follows:

- Settings for displaying a dashboard.

You can set the global query time range for the dashboard, set alert notifications for all charts, set whether to automatically refresh the dashboard page, choose to expand the dashboard to full screen display, and set the display style of all chart titles.

- Settings for displaying a chart.

You can view analysis results of the chart, set the query time range for the chart, set alert notifications for the chart, download the chart and the chart log, and check whether **drill-down analysis** is set.

- **Editing mode**

When you select editing mode for a dashboard, you can set multiple dashboard parameters as follows:

- Set a dashboard.
 - Add elements to the dashboard. For example, you can add [markdown charts](#), customized charts, texts, icons, and other elements to the dashboard.
 - Add a line to connect two charts. Lines can automatically be adjusted according to chart location.
 - Add a [filter](#). Note that in editing mode you can add a filter, but you must switch to display mode to filter the chart data.
 - Show or hide grid lines.
- Set a chart.

You can modify the query statement represented by a chart and the chart properties, or [set a drill-down analysis](#).

12.2.2 Create and delete a dashboard

This topic describes how to create and delete a dashboard.

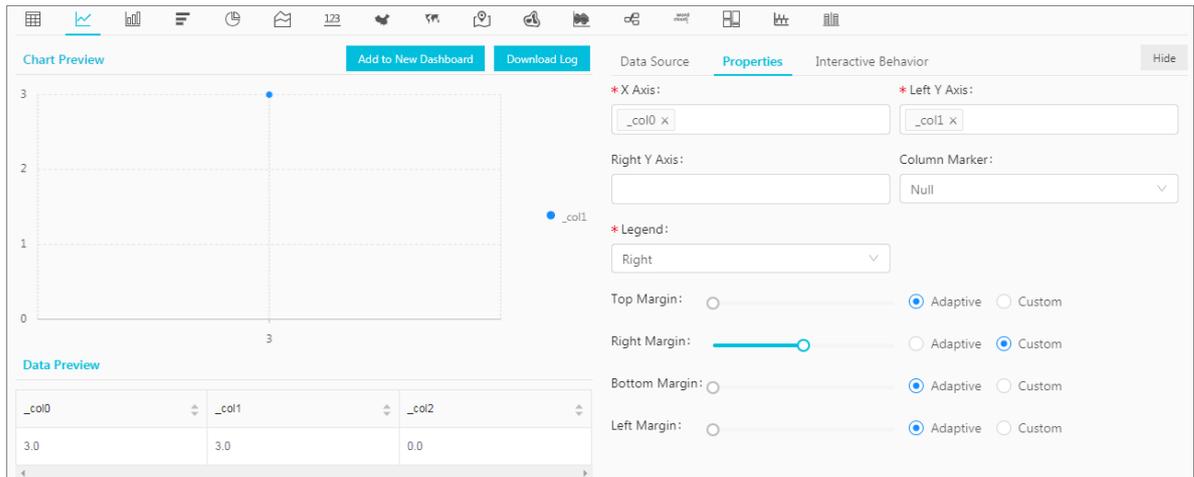
Prerequisites

- Logs are collected.
- [Indexes](#) are enabled and set.

Create a dashboard

1. Log on to the [Log Service console](#), and then click the target project name.
2. On the Logstores page, click Search in the LogSearch column.
3. In the search box, enter a query and analysis statement, and click Search & Analysis.

4. Click Properties to set the chart properties.



5. Optional. Specify a placeholder variable.



Note:

If you specify a placeholder variable for the query statement, and a drill-down event that results in a jump to the current dashboard is triggered by any other chart, the system performs the following:

- a. It replaces the placeholder variable of the query statement with the chart value that triggered the drill-down event.
- b. It refreshes the current dashboard and updates with the new query statement.

For more information, see [Set a drill-down analysis](#).

To specify a placeholder variable, follow these steps:

- a. Click the Data Source tab, and select part of the query statement.
- b. Click Generate Variable.
- c. Set Variable Config.

Configuration	Description
Variable Name	Placeholder variable name. If the placeholder variable name is the same as the variable specified in the chart that triggered the drill-down event, the placeholder variable will be replaced with the chart value.
Default Value	Default value of the placeholder variable in the current dashboard.

Configuration	Description
Result	Query statement that has the specified variable.

Properties
Data Source
Interactive Behavior

Query:

Generate Variable
| select count(1) as pv

Select the query statement to generate a placeholder variable. You can configure a drill-down configuration to replace the variable.
 For how to use dashboards, please refer to the documentation ([Help](#))

Variable Config:

*** Variable Name:**

*** Default Value:**

✕

Result

\$(method) | select count(1) as pv

6. Optional. Set a [drill-down analysis](#).

To set a drill-down analysis, follow these steps:

- a. Click the Interactive Behavior tab.
- b. Select an Event Action.
- c. Set parameters related to the selected event action.

Event Action

Open Search Page ▼

• **Select Saved Search:**

method_pv ▼

Time Range:

Default ▼

Inherit Filters:

Filter

Variable

Filter Statement

\${c}

Optional Parameter Fields

7. Click Add to New Dashboard.

8. Set the dashboard and chart names.

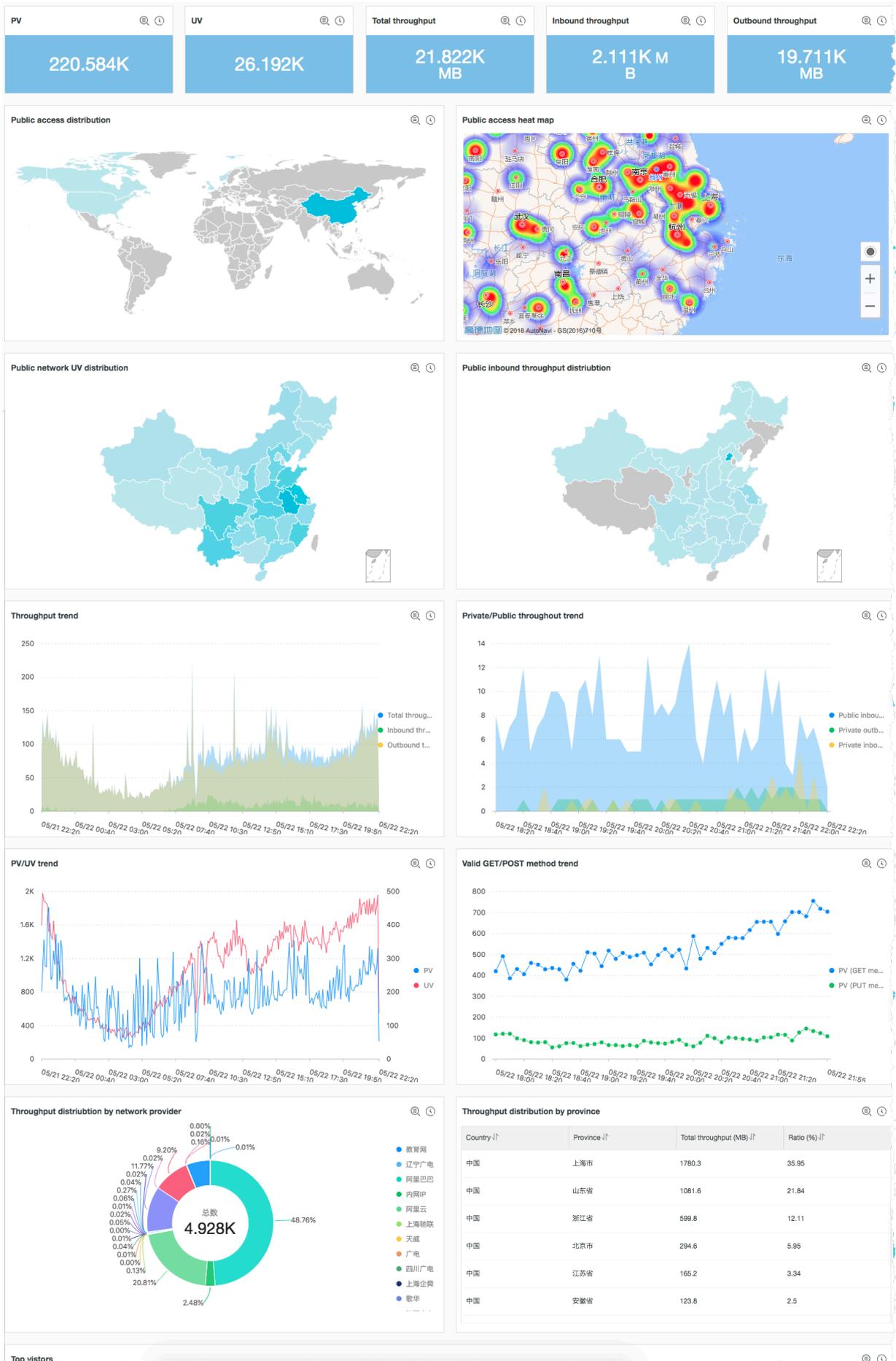
Configuration	Description
Operation	<p>Available actions are:</p> <ul style="list-style-type: none"> • Add to Existing Dashboard: add the chart to an existing dashboard. • Create Dashboard: create a new dashboard and then add the chart to the new dashboard.
Dashboards	<p>Select an existing dashboard name.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p> Note: This parameter is required only when you set the Operation parameter as Add to Existing Dashboard.</p> </div>

Configuration	Description
Dashboard Name	Enter a dashboard name.  Note: This parameter is required only when you set the Operation parameter as Create Dashboard.
Chart Name	Enter a chart name. The chart name is displayed as the chart title in the dashboard.

9. Click OK.

You can add up to fifty analysis charts to a dashboard.

The following figure shows a dashboard that contains multiple analysis charts.



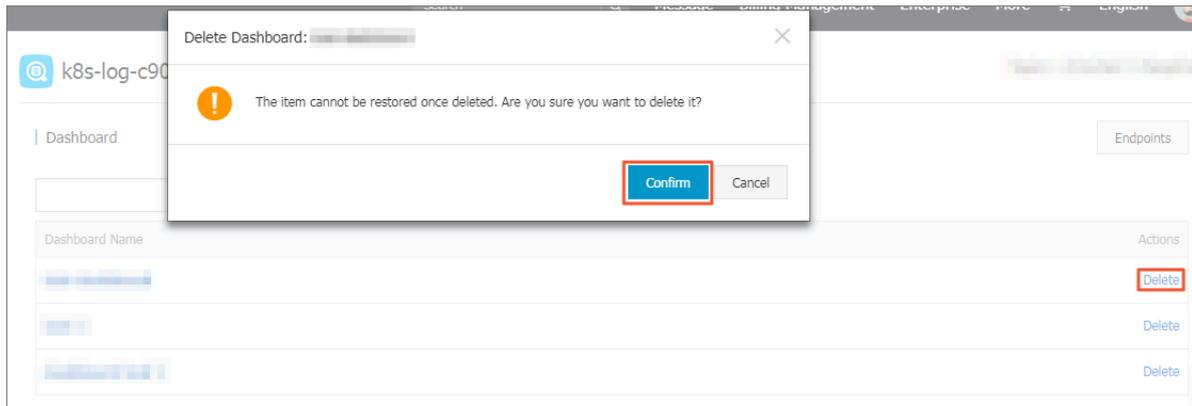
Delete a dashboard



Note:

A deleted dashboard cannot be recovered.

1. In the left-side navigation pane of the Logstores page, click Dashboard.
2. Click Delete on the right of the target dashboard.
3. In the displayed dialog box, click Confirm.



12.2.3 Set the display parameters for a dashboard

This topic describes how to set the display parameters for a dashboard in Log Service. The display mode is the default viewing mode of a dashboard.

To open your target dashboard, perform either of the following two operations:

- In the left-side navigation pane of the Logstores page, click Dashboard, and then click the target dashboard name.
- In the left-side collapsed navigation pane of the Search & Analysis page, the Saved Search page, or any other page in the Log Service console, move your pointer over the pane to show the items, and then click the target dashboard name.

Available settings

- Settings for displaying a dashboard

When display mode is selected for a dashboard, the following available function buttons are located in the upper-right corner of the dashboard (left to right): Please select, Edit, Subscribe, Alerts, Refresh, Share, Full Screen, Title Configuration, and Reset Time.

- Settings for displaying a chart

When display mode is selected for a dashboard, the list hidden in the upper-right corner of a chart provides functions for you to set parameters related to chart analysis results.



Note:

Different elements in a dashboard have different function lists.

Set the query time range for a dashboard

All charts in a dashboard use the query time range that is set for the dashboard. To set a query time range exclusive to a single chart, follow the instructions described in [Set the query time range for a chart](#).



Note:

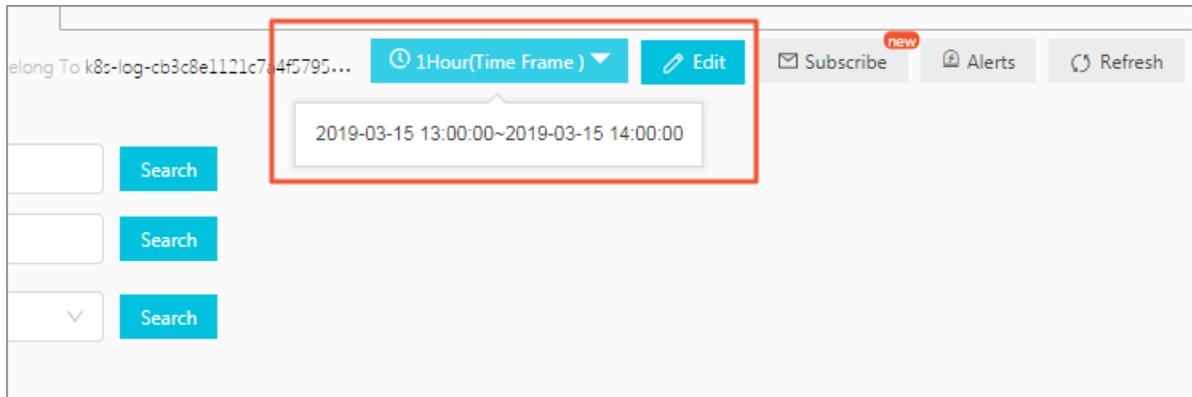
A custom set query time range for a dashboard is a temporary setting that is not saved by the system. This means that when you re-open the dashboard to view charts, the system displays the query and analysis results of the default query time range.

1. Click Please Select.
2. Select a query time range.

Available types of query time ranges for a dashboard are as follows:

- **Relative:** indicates to query the logs in an exact time range (accurate to seconds) of one minute, five minutes, fifteen minutes, or other time length that starts from the current time point. For example, if the current time point is 19:20:31 and you set this parameter to one hour, then the charts in the dashboard query logs generated from 18:30:31 to 19:20:31.
- **Time Frame:** indicates to query the logs in a whole time range at a one-minute, five-minute, fifteen-minute period, or any other time length period that starts from the current time point. For example, if the current time point is 19:20:31 and you set this parameter to one hour, then the charts in the dashboard query logs generated from 18:00:00 to 19:00:00.
- **Custom:** indicates to query the logs in a customized time range.

3. Move your cursor to Please Select to verify that the time range that you set has taken effect.



Switch to editing mode

To enter editing mode, click Edit. For more information, see [Edit a dashboard](#).

Set an alert notification

To create or modify an alert notification, choose Alerts > Create or choose Alerts > Modify in the upper-right corner. An alert must be associated with at least one chart.

For more information, see [Set an alert](#).

Set the refresh page frequency

You can manually refresh a dashboard, or set a time interval at which the dashboard automatically refreshes.

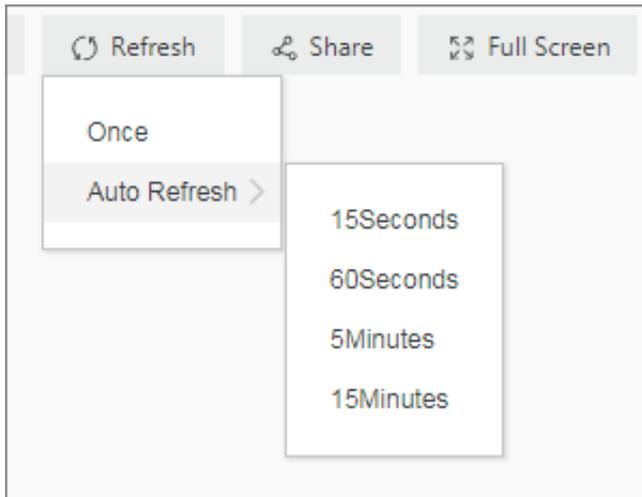
- To manually refresh the dashboard, choose Refresh > Once.
- To set the dashboard to refresh automatically at specified time intervals, choose Refresh > Auto Refresh. Then, select an interval.

A dashboard can be automatically refreshed at 15-second intervals, 60-second intervals, 5-minute intervals, or 15-minute intervals.



Note:

If no activity is detected within your browser and the screen goes to sleep, the time interval at which the dashboard refreshes may be affected.



Share a dashboard

To share a dashboard with other users, click **Share** to copy the link of the dashboard page and then send the link to users that have the permission to view the dashboard. Your current dashboard settings are saved in the shared dashboard page (such as the query time range and the style to display chart titles).



Note:

Before you share a dashboard, you must grant the target user or users the permission to view the dashboard.

Display a dashboard in full screen

Click **Full Screen**. We recommend that you perform this operation if you want to focus on data or make a presentation.

Set the chart title format

Click **Title Configuration**, and then select one of the following formats:

- Single-line Title and Time Display
- Scrolling Title and Time Display
- Alternate Title and Time Display
- Title Only
- Time Only

Reset the query time range

To restore the default query time ranges of all the charts in a dashboard, click **Reset Time**.

Select chart view

- View analysis details of a chart

To view analysis details of a chart (such as the query statement associated with a chart and the chart properties) , choose **More Option > View Analysis Details** in the upper-right corner of the chart.

- Set the query time range for a chart

To set the query time range for a chart, choose **More Option > Select Time Range** in the upper-right corner of the chart. The setting only takes effect for the current chart.

- Set an alert notification for a chart

To set an alert notification for a chart, choose **More Option > Create Alert** in the upper-right corner of the chart. For more information, see [Set an alert](#).

- Download logs

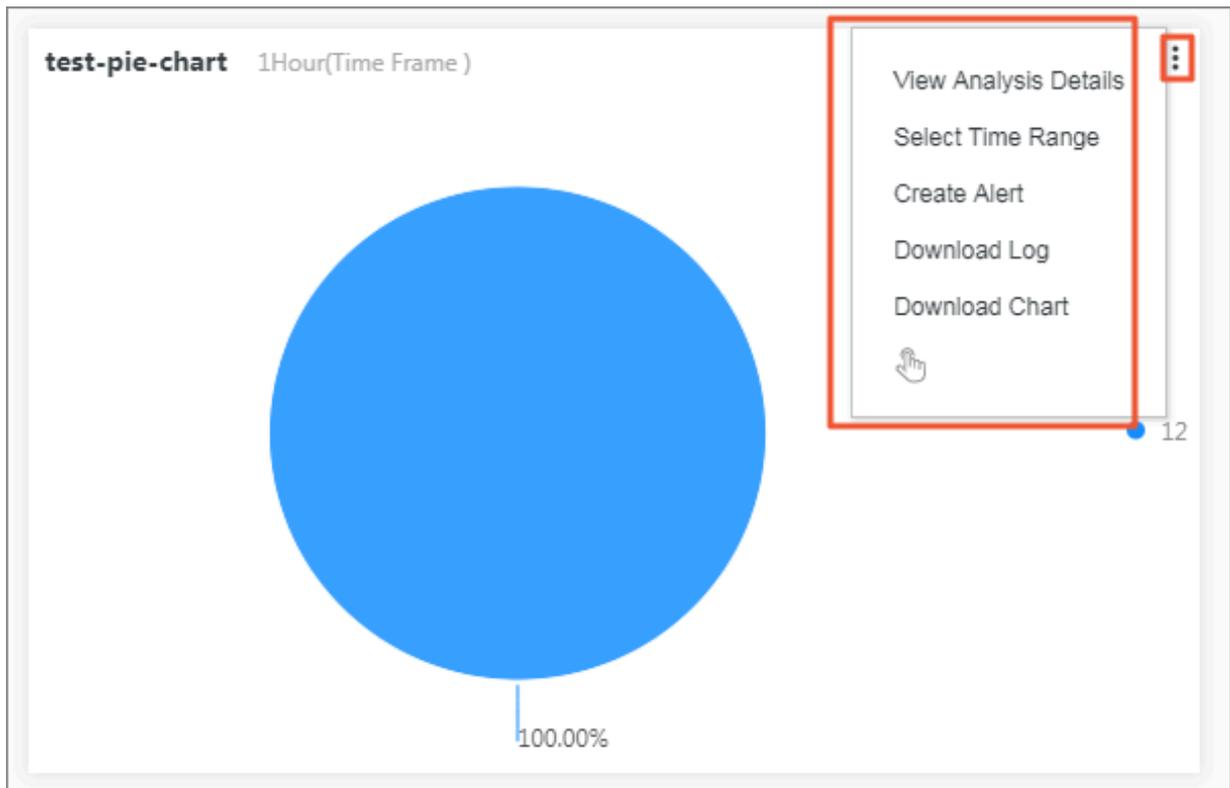
To download logs, choose **More Option > Download Log** in the upper-right corner of the chart. The raw log analysis results are downloaded as a .csv file.

- Download a chart

To download a chart, choose **More Option > Download Chart** in the upper-right corner of the chart. The chart is downloaded as a .psd file.

- Check whether a drill-down analysis is set for a chart

To check whether drill-down analysis is set for a chart, move your cursor to the **More Option** button in the upper-right corner of the chart, and check the color of the icon at the bottom of the hidden list. A red icon indicates that a drill-down analysis is set for the chart. A gray icon indicates that no drill-down analysis is set for the chart.



12.2.4 Edit a dashboard

This topic describes how to edit a dashboard.

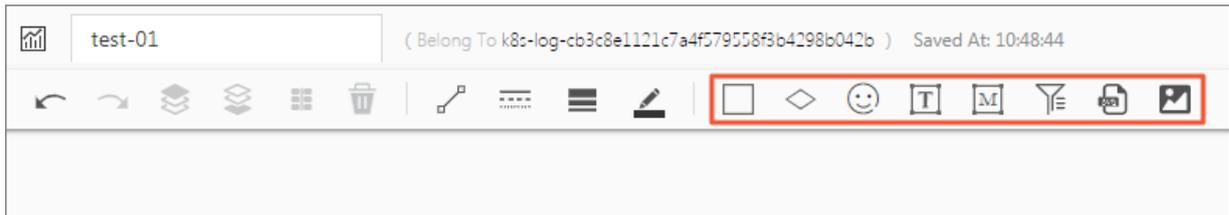
You can perform the following editing operations in a dashboard:

- Set dashboard parameters.
 - In the upper-left corner, you can click the current dashboard name to modify the dashboard name.
 - Add elements to the dashboard. For example, you can add [markdown charts](#), customized charts, texts, icons, and other elements as needed.
 - Add a line to connect two charts. Lines are automatically adjusted according to the chart location.
 - Add a [filter](#). A filter can filter specific chart data when the dashboard is in the displaying mode.
 - Set the dashboard to show grid lines.
 - Set chart properties by using the tools in the menu bar.
- Set a chart.

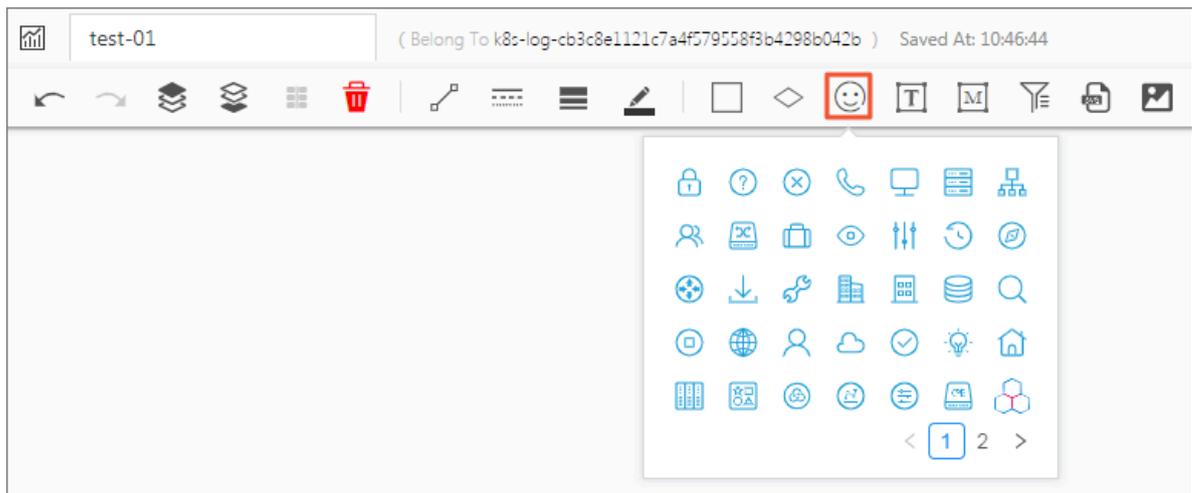
You can modify the query statement represented by a chart and the chart properties, or [set a drill-down analysis](#).

**Note:**

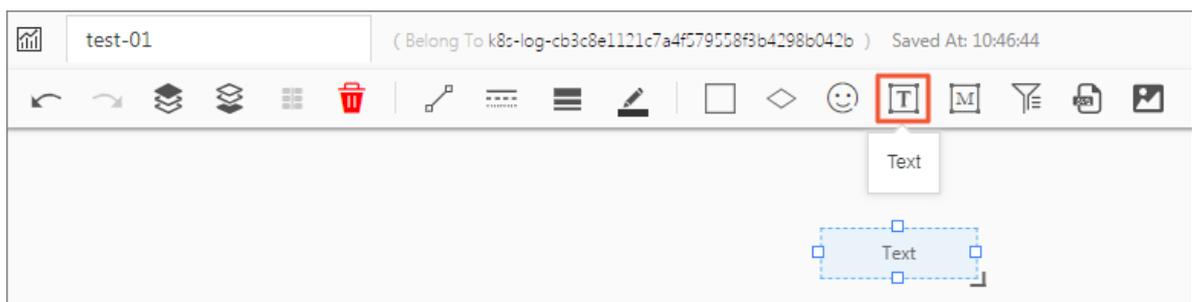
You must click Save in the upper-right corner after you make any modifications to a dashboard in editing mode. Otherwise, your changes will not be saved when you exit editing mode.

Add an element to a dashboard**• Icons**

To add an icon to the dashboard, click the icon tool in the ribbon, select the target icon, and then drag the icon to where you want to position it in the dashboard.

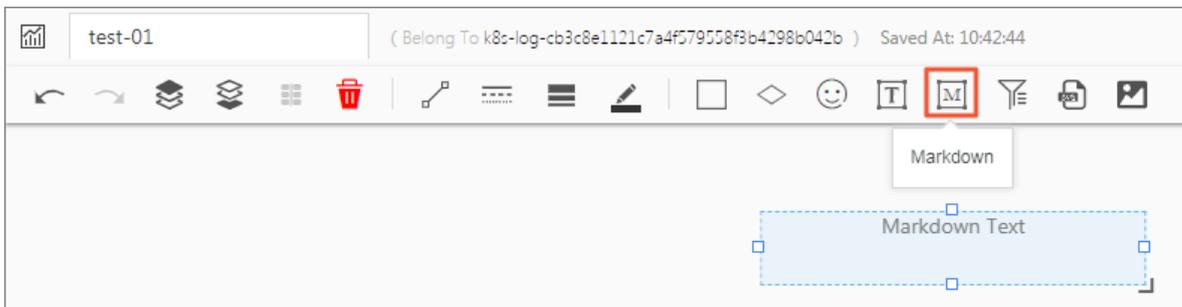
**• Text**

To add text to the dashboard, drag the text icon to the target position in the dashboard. Then, double click the text box to add text.



- **Markdown chart**

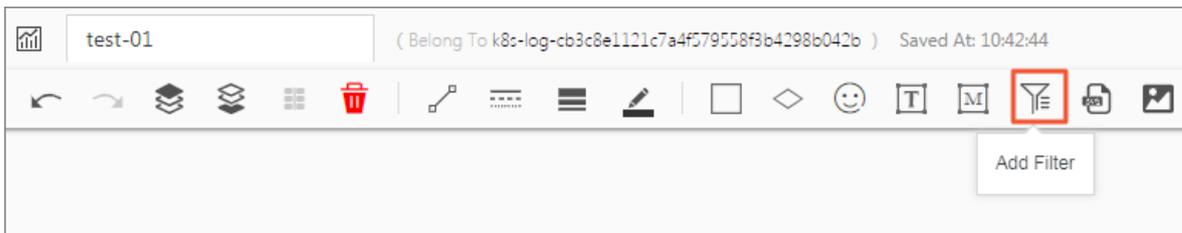
To add a markdown chart to the dashboard, first drag the markdown tool icon from the ribbon to the target position in the dashboard. To edit the markdown box, move your pointer to the upper-right corner of the markdown box, click the More Options icon and then click Edit.



- **Filter**

A filter can be used to narrow down your query scope or replace variables in the dashboard.

To add a filter to the dashboard, click the filter tool icon in the ribbon, and then set a filter in the displayed page. By default, a filter is added to the upper-left corner of the dashboard. To modify the filter settings, move your pointer to the upper-right corner of the filter box, click the More Options icon and then click Edit.



- **SVG file**

To add an SVG file to the dashboard, click the SVG tool icon in the ribbon, and then click the gray area in the displayed dialog box to select an SVG file from your local directory or drag and drop an SVG file to the gray area in the displayed dialog box.



Note:

The maximum size of an SVG file is 10 kb.

- Image from the Web

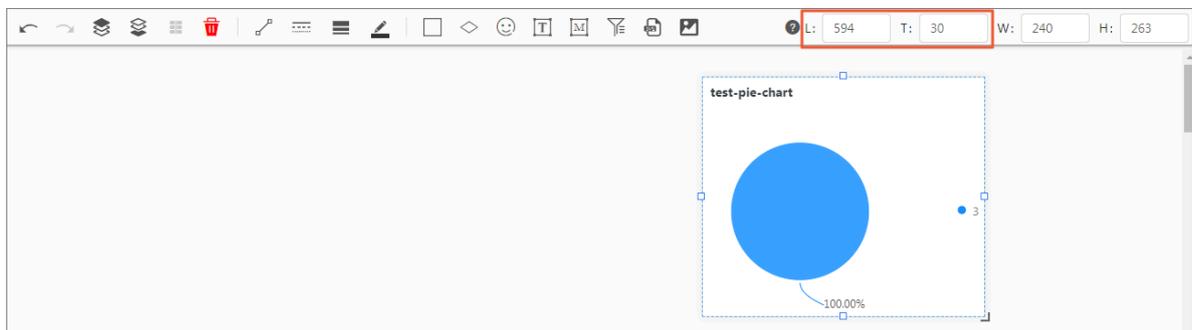
To add an image from a website to the dashboard, click the Image icon in the ribbon, enter or paste the URL of the image in the displayed dialog box, and then click OK.

Set the dashboard layout

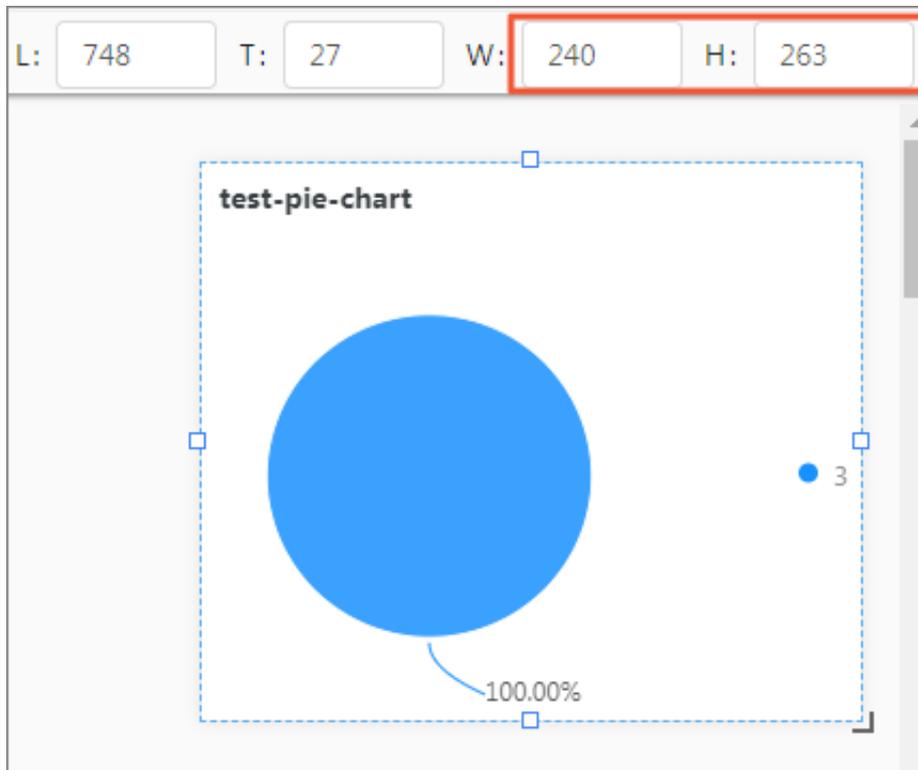
In editing mode, all the charts and elements in the dashboard can be dragged to any position or resized while maintaining proportions (except for lines used to connect charts and elements). The limit of the dashboard on the horizontal display is 1000 pixels (the vertical display has no pixel limit). To accurately set the position of a chart and the spacing between charts, we recommend that you click Show Grid Lines in the upper-right corner before setting the dashboard layout.

You can perform the following operations:

- Adjust chart position
 - Directly drag the target chart.
 - Select the target chart, and then set the L value and the T value in the ribbon.



- Adjust the width and height of a chart
 - Select the target chart, and then drag the lower-right corner of the chart to resize the chart.
 - Select the target chart, and then set the L value and the T value in the ribbon.

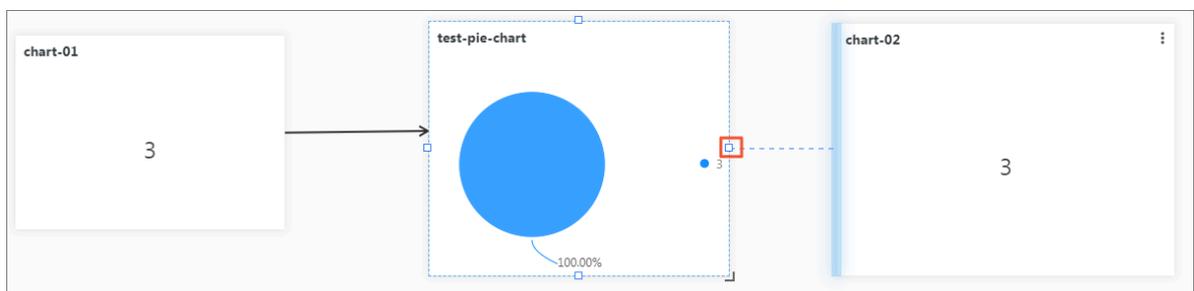


- Add a line to connect two charts

To connect a source chart with a destination chart by using a line, follow these steps:

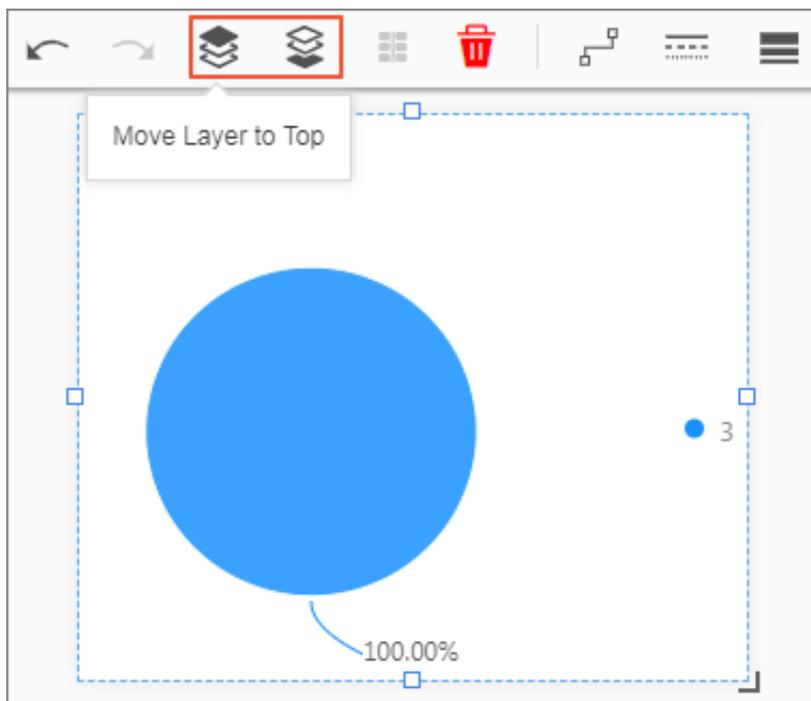
1. Select the source chart.
2. Click and hold a sizing handle on the outline of the source chart and drag it to one side of the destination chart.
3. Release your cursor when the target side of the destination chart becomes blue.

After, when you adjust the position and size of a chart that is connected with any other chart through a line, the line moves in proportion.



- Set the chart layer

To set the chart layer, select the target chart, and then click the Move Layer to Top icon or Move Layer to Bottom icon in the ribbon.



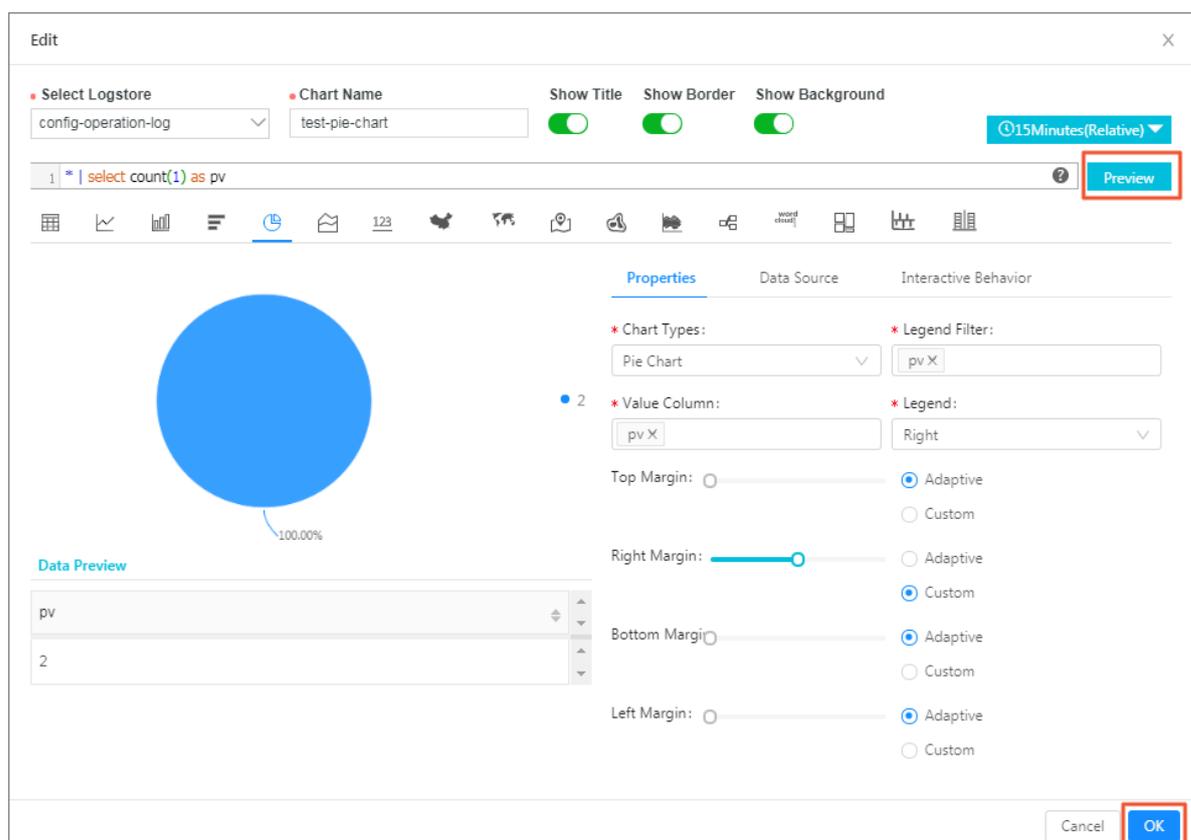
Modify a chart

In editing mode, you can modify a chart in a dashboard by performing the following operations:

- Edit a chart.

You can modify the query statement represented by a chart and the chart properties, [set a drill-down analysis](#), and perform other operations.

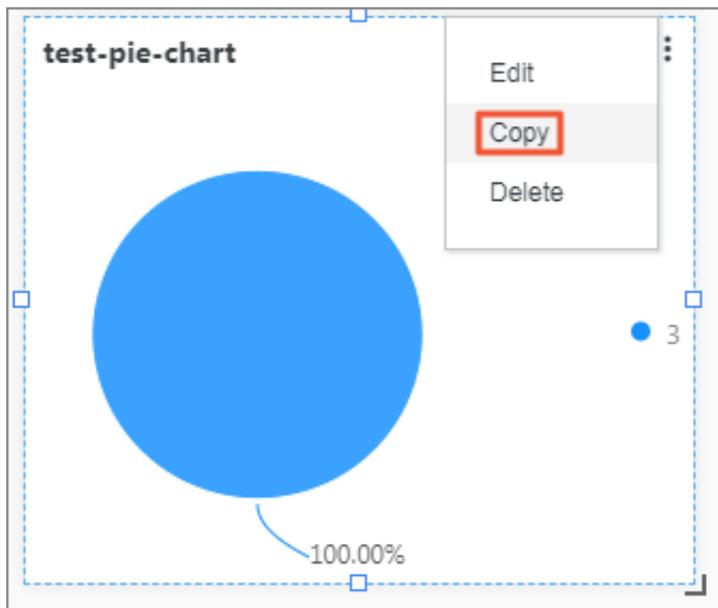
1. In the upper-right corner of the dashboard, click Edit.
2. In the upper-right corner of the target chart, click the More Options icon and then click Edit.
3. In the displayed dialog box, modify the query statement, or set Properties, Data Source or Interactive Behavior.
4. Click Preview, and then click OK.
5. In the upper-right corner of the dashboard, click Save.



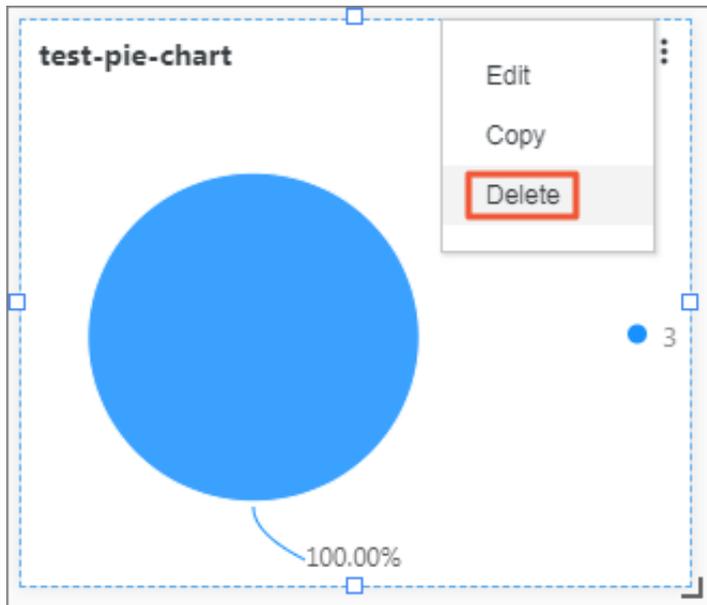
- Copy a chart

You can create a copy of a chart to preserve all current settings of the chart.

1. In the upper-right corner of the dashboard, click Edit.
2. In the upper-right corner of the target chart, click the More Options icon and then click Copy.
3. Drag the replicate chart to a new position and release.
4. In the upper-right corner of the dashboard, click Save.



- Delete a chart
 1. In the upper-right corner of the dashboard, click Edit.
 2. In the upper-right corner of the target chart, click the More Options icon and then click Delete.
 3. In the upper-right corner of the dashboard, click Save.



12.2.5 Subscribe to dashboards

Log Service dashboards provide many visual components for you to customize your own dashboard based on log query and analysis results. Now, Log Service allows you to subscribe to dashboards. Log Service can periodically take snapshots of dashboards and send the snapshots to subscribers by email or DingTalk group message.

Limits

- You can create only one subscription task for a dashboard.
- You can send up to 50 emails per day under an Alibaba Cloud account.
- CronExpression can set the minimum interval in units of minutes. We recommend that you set the interval for sending notification messages to at least 1 hour.
- The total number of subscription and alerting tasks for a project cannot exceed 100. You can open a ticket to increase the limit if you need more subscription and alerting tasks for a project.
- If a table displays data by page, Log Service sends only the snapshot of the first page to subscribers.



Note:

- You can select any time range in display mode to view the chart data of different time ranges. The chart data is not affected.
- To modify the default query time range of charts on a dashboard, you can enter the edit mode and click Edit in each chart.
- The query time range of a subscribed dashboard is the time range of queried data in charts on the dashboard.

Create a dashboard subscription task

1. Log on to the [Log Service console](#), and then click the target project name.
2. In the left-side navigation pane, click Dashboard.
3. Click a dashboard name to view the dashboard.
4. Click Subscribe in the upper-right corner of the dashboard page.
5. Configure the subscription task and click Next.

Parameter	Description	Example
Subscription Name	The name of the subscription task. The name must be 1 to 64 characters in length.	Dashboard subscription
Frequency	<p>The frequency for sending notification messages after the subscription to the dashboard.</p> <p>Valid values:</p> <ul style="list-style-type: none"> · Hourly · Daily · Weekly · Fixed interval (days) · Interval specified by CronExpression <p>CronExpression can set the minimum interval in units of minutes. We recommend that you set the interval for sending notification messages to at least 1 hour.</p>	<p>Use CronExpression to set the frequency to <code>* 0 / 1 * *</code>, which specifies that messages are sent every hour starting from 00:00.</p>

Parameter	Description	Example
Add Watermark	Specifies whether to add a watermark to the generated dashboard snapshots . The watermark content is the address used to receive notification messages , such as an email address or the access_token in the WebHook address of DingTalk Chatbot.	N/A

Create Subscription ✕

Subscription Configuration Notifications

* Subscription Name 12/64

* Frequency

Add Watermark

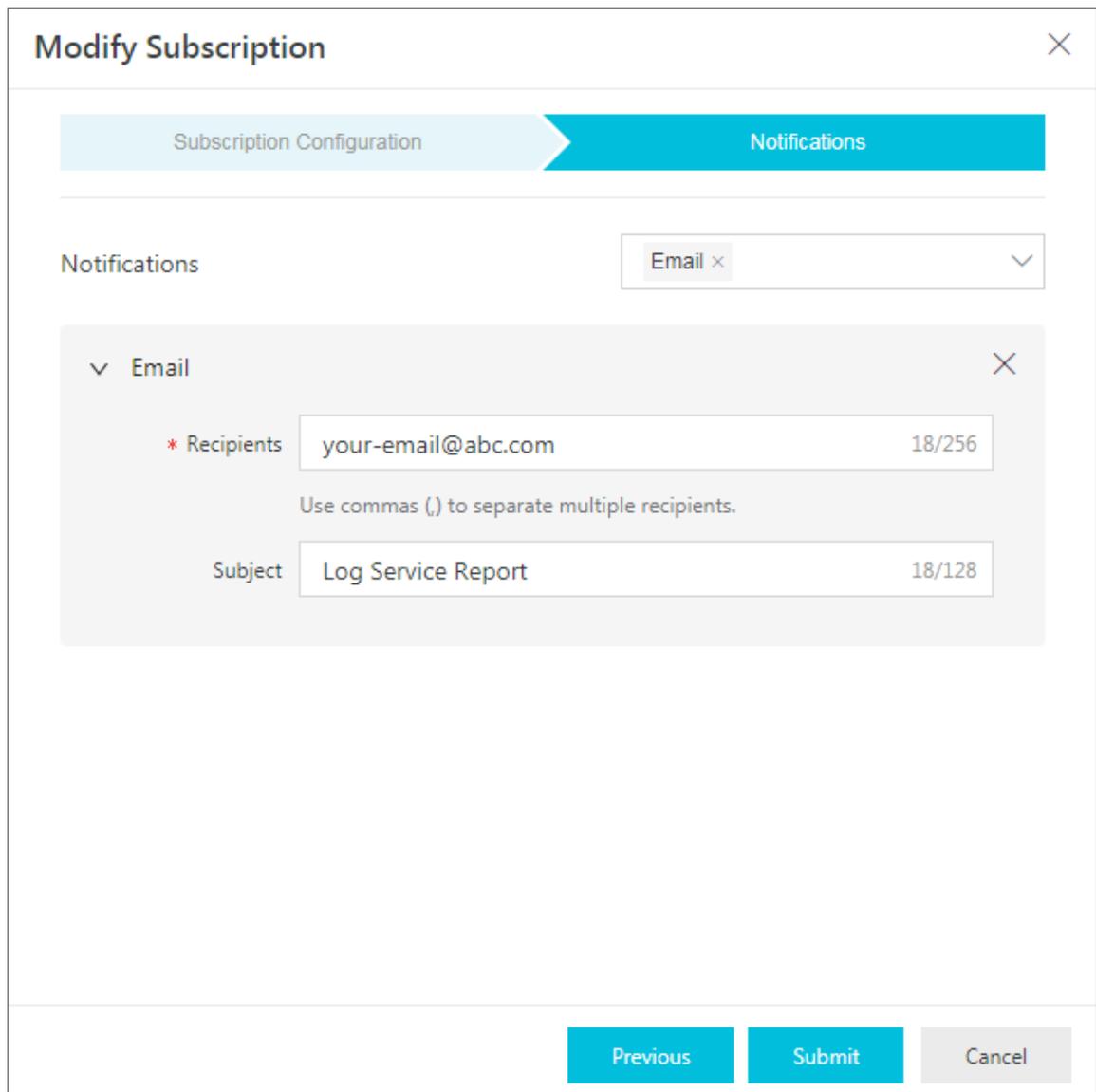
Automatically adds the email address or webhook address as a watermark to the image

6. Configure the notification method.

You can set the notification method to Email or WebHook-DingTalk Bot.

- Email

Enter email addresses in the Recipients field and set Subject. The default email subject is `Log Service Report`.



The screenshot shows a 'Modify Subscription' dialog box with a progress bar at the top. The progress bar has two segments: 'Subscription Configuration' (light blue) and 'Notifications' (dark blue). Below the progress bar, the 'Notifications' section is active, showing a dropdown menu with 'Email' selected. Below this, there is a sub-dialog box for 'Email' configuration. The 'Email' sub-dialog has a 'Recipients' field with the value 'your-email@abc.com' and a character count '18/256'. Below the 'Recipients' field is a note: 'Use commas (,) to separate multiple recipients.' The 'Subject' field has the value 'Log Service Report' and a character count '18/128'. At the bottom of the main dialog, there are three buttons: 'Previous' (blue), 'Submit' (blue), and 'Cancel' (grey).

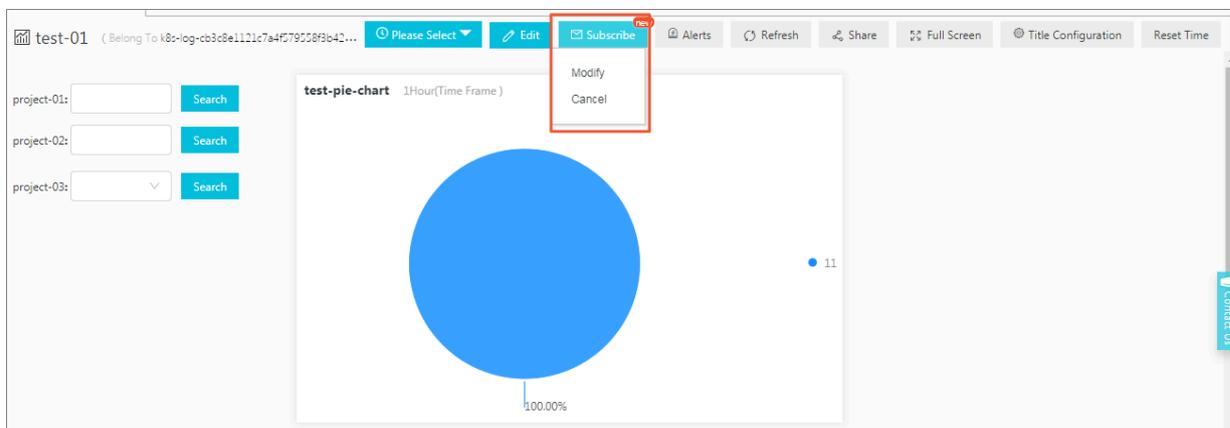
- WebHook-DingTalk Bot

Enter the WebHook address of DingTalk Chatbot in the Request URL field. For more information about how to obtain this address, see [Customize DingTalk Chatbot](#).

Modify and cancel a subscription task

If a subscription task is created for a dashboard, you can click **Subscribe** in the upper-right corner of the dashboard page to modify or cancel the subscription task.

After the subscription task is canceled, Log Service no longer sends subscription messages to subscribers.



12.2.6 Drill-down analysis

Log Service analysis charts provide the drill-down analysis in addition to the basic data visualization features. When you add a chart to the dashboard, you can modify the configurations in the drill-down list to present data in a more powerful way.

Drilling is an essential part of data analysis. This feature allows you to view more detailed information by moving to different layers of data. Drilling includes rolling up and drilling down. By rolling up, you move to higher data layers with more summarized information. By drilling down, you move to deeper data layers to reveal

more detailed information. By drilling down into data layer by layer, you can view data more clearly, extract more value out of data, and then make more accurate and efficient decisions based on the data.

Log Service supports drill-down analysis for analysis charts in the dashboard. After you set the dimension and layer of a drilldown, you can jump to the analysis page of a deeper dimension by clicking a data point in the dashboard. Analysis charts in the dashboard are actually the results of query statements. If you configure a drill-down analysis for the request status table and add the table to the dashboard, you can click a request status type in the dashboard to view logs of the request status.

Limits

In Log Service, the charts that support drill-down analysis include:

- Tables
- Line charts
- Column charts
- Bar charts
- Pie charts
- Individual value plots
- Area charts
- Tree maps

Prerequisites

1. You have enabled and configured an index.
2. You have configured a saved search, a dashboard, and a custom link that you want to jump to.
3. To add a variable, you must first configure a placeholder in the query statement for the saved search and dashboard that you want to jump to. For more information, see [Saved search](#) and [#unique_54](#).

Procedure

1. Log on to the [Log Service console](#), and then click the target project name.
2. Click Search in the LogSearch column on the Logstores page.
3. Enter your query analysis statement, set the time range, and then click Search & Analysis.

4. Click the Graph tab, select a chart type and then configure parameters on the Properties page of the chart.
5. Click the Interactive Behavior tab, and set Event Action for drill-down analysis.

A drill-down event is an event that you trigger by clicking an analysis chart on the dashboard page. This event is disabled by default. After you configure a drill-down event and click the chart data in the dashboard, your current page jumps to the corresponding page according to the event action that you have configured. Choose any of the following options:

- **Disable:** disables drill-down analysis.
- **Open Logstore:** enables drill-down analysis. The drill-down event is to open a Logstore.

When you click a value in the chart, if you have set Filter, the system automatically adds a query statement for the Logstore that you want to jump to. You cannot set Variable.

Parameter	Description
Select Logstore	The name of the Logstore that you want to jump to. For more information about how to configure a Logstore, see #unique_200 .
Open in New Tab	If you enable this option, when interaction behavior is triggered, the system opens the corresponding Logstore in a new tab.

Parameter	Description
Time Range	<p>Configure the time range for the Logstore that you want to jump to.</p> <p>Valid values:</p> <ul style="list-style-type: none">- Default: on the dashboard page, click a chart to jump to the Logstore, and use the default time range 15 minutes for a saved search. This is a relative time range.- Inherit table time: after you jump to the Logstore by clicking the chart in the dashboard, the time range of the query statement is the table time configured in the dashboard by default when the event is triggered.- Relative: set the time range of the target saved search to the specified relative time.- Time Frame: set the time range of the target saved search to the specified time frame. <p>Default value: Default.</p>
Inherit Filters	<p>If you enable Inherit Filters, the system synchronizes filtering conditions added in the dashboard to the saved search of the target Logstore, and adds the filtering conditions before the query statement by using the logical <code>AND</code> operator.</p>

Parameter	Description
Filter	<p>Click the Filter tab, and set Filter Statement. The statement can contain any options under Optional Parameter Fields.</p> <p>If you have set Filter, when you click a value in the chart, the system automatically adds a query statement for the target saved search. The query statement is the value you specify in Filter Statement.</p>

- **Open Search Page:** enables drill-down analysis. The drill-down event is to open a search page.

When you click a value in the chart, if you have set Variable, the system replaces the placeholder configured in the saved search statement with the chart value you clicked, and then performs a deeper query according to the chart value.

If you have set Filter, the system automatically adds a query statement for the

target saved search. You can specify a variable and a placeholder at the same time.

Event Action

Open Search Page ▼

• Select Saved Search:

method_pv ▼

Time Range:

Inherit table time ▼

Inherit Filters:

Variable

method| ×

Parameter	Description
Select Saved Search	The name of the saved search that you want to jump to. For more information about how to configure a saved search, see Saved search .
Open in New Tab	If you enable this option, when interaction behavior is triggered, the system opens the corresponding saved search in a new tab.

Parameter	Description
Time Range	<p>Configure the time range for the saved search that you want to jump to. Valid values:</p> <ul style="list-style-type: none"> - Default: on the dashboard page, click a chart to jump to the saved search, and use the default time range 15 minutes for a saved search. This is a relative time range. - Inherit table time: after you jump to the saved search by clicking the chart in the dashboard, the time range of the query statement is the table time configured in the dashboard by default when the event is triggered. - Relative: set the time range of the target saved search to the specified relative time. - Time Frame: set the time range of the target saved search to the specified time frame. <p>Default value: Default.</p>
Inherit Filters	<p>If you enable Inherit Filters, the system synchronizes filtering conditions added in the dashboard to the saved search, and adds the filtering conditions before the query statement by using the logical <code>AND</code> operator.</p>
Filter	<p>Click the Filter tab, and set Filter Statement. The statement can contain any options under Optional Parameter Fields.</p> <p>If you set Filter, when you click a value in the chart, the system automatically adds a query statement for the target saved search. The query statement is the value you specify in Filter Statement.</p>

Parameter	Description
Variable	<p>Click the Variable tab, click Add Variable, and then set the following parameters:</p> <ul style="list-style-type: none"> - Replace Variable Name: the variable that triggers drill-down analysis. You can click this variable to jump to the target saved search. - Replace the value in the column: the column where the value that you want to replace data with is located. To process multiple columns, you can specify the current column and other columns. The current column is the column that you want to perform drill-down analysis. This column is the column specified in Replace the value in the column. Other columns can be any other columns in the chart for drill-down analysis. <p>When the query statement variable of the target saved search matches the name of the added variable, the system replaces the query statement variable with the chart value that triggers the drill-down event. This flexibly changes the query statement of the target saved search.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> Note:</p> <ul style="list-style-type: none"> - To add a variable, you must first configure a placeholder in the query statement for the saved search that you want to jump to. - You can add up to five variables. </div>

- **Open Dashboard:** enables drill-down analysis. The drill-down event is to open a dashboard.

The chart in the dashboard is the chart-form result of the query statement. You need to pre-configure a placeholder in the query statement for the dashboard that you want to jump to. When you click a chart value in the upper-layer dashboard, if you have set Variable, the system replaces the pre-configured placeholder with the chart value. If you have set Filter, the system adds the

filtering conditions for the target dashboard, and then performs a deeper query according to the chart value.

Event Action

Open Dashboard ▼

Select Dashboard:

destination_drilldown ▼

Time Range:

Inherit table time ▼

Inherit Filters:

Variable

method

×

Parameter	Description
Select Dashboard	The name of the dashboard that you want to jump to. For more information about how to configure a dashboard, see #unique_54 .
Open in New Tab	If you enable this option, when interaction behavior is triggered, the system opens the corresponding dashboard in a new tab.

Parameter	Description
Time Range	<p>Configure the time range for the dashboard that you want to jump to. Valid values:</p> <ul style="list-style-type: none"> - Default: after you jump to the configured dashboard by clicking the chart in the current dashboard, the time range of the configured dashboard is the time configured in the current dashboard chart by default where the drill-down event is triggered. - Inherit table time: after you jump to the target dashboard, the time range of the chart in the dashboard is the chart time configured in the dashboard by default when the event is triggered. - Relative: set the time range of the target dashboard to the specified relative time. - Time Frame: set the time range of the target dashboard to the specified time frame. <p>Default value: Default.</p>
Inherit Filters	<p>If you enable Inherit Filters, the system synchronizes filtering conditions added in the dashboard to the target dashboard, and adds the filtering conditions before the query statement by using the logical <code>AND</code> operator.</p>
Filter	<p>Click the Filter tab, and set Filter Statement. The statement can contain any options under Optional Parameter Fields.</p> <p>If you set Filter, when you click a value in the chart, the system automatically adds a query statement for the target dashboard. The query statement is the value you specify in Filter Statement.</p>

Parameter	Description
Variable	<p>Click the Variable tab, click Add Variable, and then set the following parameters:</p> <ul style="list-style-type: none"> - Replace Variable Name: the variable that triggers drill-down analysis. You can click this variable to jump to the target dashboard. - Replace the value in the column: the column where the value that you want to replace data with is located. To process multiple columns, you can specify the default column and other columns. The default column is the current column that you want to perform drill-down analysis. Other columns can be any other columns in the chart for drill-down analysis. <p>When the query statement variable of the analysis chart in the target dashboard matches the name of the added variable, the system replaces the query statement variable with the chart value that triggers the drill-down event. This flexibly changes the query statement of the analysis chart in the target dashboard.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p> Note:</p> <ul style="list-style-type: none"> - To add a variable, you must first configure a placeholder in the query statement for the dashboard that you want to jump to. - You can add up to five variables. </div>

- Custom HTTP Link: enables drill-down analysis. The drill-down event is to open a custom HTTP link.

The path in the HTTP link that is the hierarchical path of the destination file . After you add optional parameter fields to the path in a custom HTTP link

and click the chart content of the dashboard, the system replaces the added parameter fields with the chart value to jump to the relocated HTTP link.

Parameter	Description
Enter Link	The destination address that you want to jump to.
Optional Parameter Fields	By clicking an optional parameter variable, you can replace part of the HTTP link with the chart value that triggers a drill-down event.

6. Click Add to New Dashboard, configure the dashboard, and then click OK.

Afterward, you can view the analysis chart on the dashboard page, and click the chart to view deeper analysis results.

Example

For example, you can store collected Nginx access logs in the Logstore named accesslog, display the common analysis scenarios of Nginx logs in the dashboard named RequestMethod, and display the trend of page view (PV) distribution over time in the dashboard named destination_drilldown. You can configure drill-down analysis for the table of request methods, add the table to the RequestMethod dashboard, and then configure the drill-down event to jump to the destination_drilldown dashboard. In the RequestMethod dashboard, click a request method to jump to the destination_drilldown dashboard to view the corresponding PV trend.

Follow these steps:

1. Configure a target dashboard named destination_drilldown.

a. Filter logs according to request types and view the PV changes over time.

The query statement is as follows:

```
request_method : * | SELECT date_format ( date_trunc ( '
minute ', __time__ ), '% H :% i :% s ' ) AS time , COUNT ( 1
) AS PV GROUP BY time ORDER BY time
```

b. Use a line chart to display the query result and save the line chart to the dashboard.

When you save the chart to the dashboard, specify the asterisk (*) as a placeholder named method. If the variable of the drill-down event that jumps to this saved search is also named method, you can replace the asterisk (*) with the chart value that you click to perform a query and analysis again.

Add to New Dashboard [Close]

Operation: Add to Existing Dashboard

* Dashboards: destination_drilldown

* Chart Name: Request method PV trend

Query: request_method: * | SELECT date_format(date_trunc('minute', __time__), '%H:%i:%s') AS time, COUNT(1) AS PV GROUP BY time ORDER BY time

Select the query statement to generate a placeholder variable. You can configure a drill-down configuration to replace the variable.

For how to use dashboards, please refer to the documentation ([Help](#))

Variable Config

Variable Name: method

Default Value: * [Close]

Result

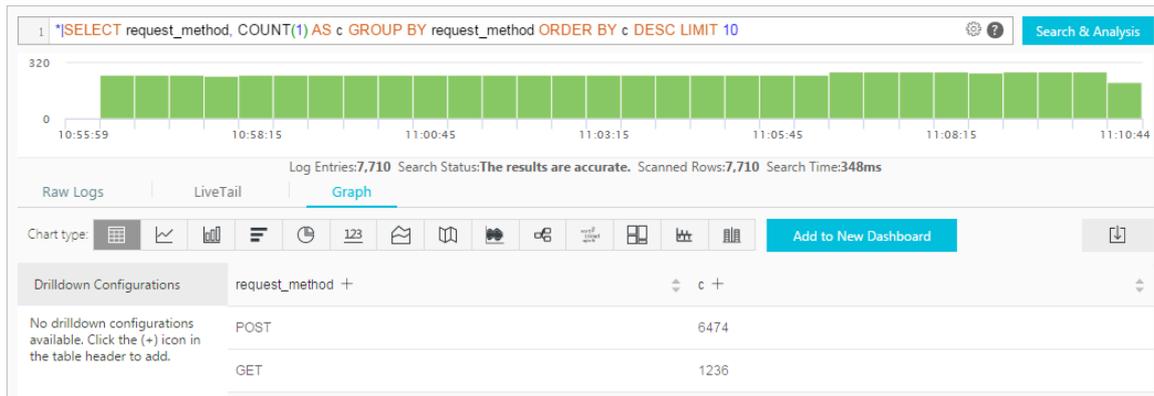
```
request_method: $method | SELECT date_format(date_trunc('minute', __time__), '%H:%i:%s') AS time, COUNT(1) AS PV GROUP BY time ORDER BY time
```

2. Configure a chart that triggers drill-down analysis, and add the chart to the dashboard named RequestMethod.

- a. On the search page, use SQL to analyze the number of logs of each request method in the Nginx access logs, and display the result in a table.

```
* | SELECT request_method, COUNT ( 1 ) AS c GROUP BY request_method ORDER BY c DESC LIMIT 10
```

The query result is as follows.



- b. Configure drill-down analysis for the request_method column in the table.

The screenshot shows the 'Drilldown Configurations' settings for the 'request_method' column. The configuration includes the following options:

- Event Action:** Open Dashboard
- Select Dashboard:** destination_drilldown
- Time Range:** Inherit table time
- Inherit Filters:** (Toggle is off)
- Variable:** method

3. Click the GET request in the RequestMethod dashboard.

The screenshot shows a dashboard titled "Access analysis" with a sub-header "(Belong To: test-apache-logs)". Below the title is a blue button labeled "Please Select". The main content is a table with the following data:

request_method	c
POST	5921
GET	1873

The "GET" link in the table is highlighted with a red rectangular box. The table has a search icon and a refresh icon in the top right corner. A horizontal scrollbar is visible at the bottom of the table area.

4. Jump to the destination_drilldown dashboard.

The page automatically jumps to the dashboard configured in step 1. The asterisk (*) in the query statement is replaced with `GET`, the chart value that you click.

Afterward, the dashboard shows changes of the GET request PV over time.



12.2.7 Set and use a filter in a dashboard

This topic describes how to set and use a filter in a Log Service dashboard. A filter in a dashboard can help you refine search results or replace placeholder variables with specified values across the whole dashboard.

Overview

Each chart in a Log Service dashboard functions as a search-and-analysis statement. By setting a filter in a dashboard, you are performing an action for each search-and-analysis statement that is indicated by a corresponding chart in the dashboard at one time. To do so, add one or more conditions for filtering the result of each search-and-analysis statement, or replace placeholder variables in each search-and-analysis

statement with a specified value. You can set the two following types of filters in a dashboard.

- **Filter type:** Add a key value as a condition before the search-and-analysis statement `[search query]` to filter the result of the search-and-analysis statement. Then, the new search-and-analysis statement can be executed as `key : value AND [search query]`. This indicates to search for logs containing `key : value` in the result of the original search-analysis.

For this type of filters, you can add multiple key values. A filter that contains multiple key values can filter out the logs that contain any of the key values from the search result of the search-and-analysis statement.

- **Replace Variable type:** Set a value to replace the placeholder variables in the search-and-analysis statements that are indicated by corresponding charts.

Components

Each filter contains the following two core components:

- The key value component, which indicates a filter operation.
- The list item component, which corresponds to the key.

Prerequisites

- An index is enabled and configured. For more information, see [#unique_4](#).
- A dashboard is created. For more information, see [Dashboard](#).
- A placeholder variable in the target search-and-analysis statement is set. This is so that you can use a filter to replace the placeholder variable in the search-and-analysis statement with a specific value.

Procedure

1. Log on to the [Log Service console](#), and then click the target project name.
2. Find the target Logstore, and then click Search in the LogSearch column.
3. In the left-side navigation pane, click the name of the target dashboard.
4. In the progress bar of the dashboard, click Edit.

5. Click the icon , and then set the filter configuration parameters:

- **Chart Name:** the filter name.
- **Title:** Turn on this switch to display the filter title in the dashboard.
- **Border:** Turn on this switch to display the filter borders.
- **Background:** Turn on this switch to add a white background for the filter.
- **Key:** There are Filter type and Replace Variable type keys. The key place holder variable must be as the one set in Prerequisites.
 - For the Filter type, set this parameter as a condition to filter the execution result of the search-and-analysis statement.
 - For the Replace Variable type, set this parameter as a placeholder variable.
- **Type:** Select Filter or Replace Variable.
 - If you select Filter, a List Item refers to the value of the condition used to filter the results of a search-and-analysis statement.
 - If you select Replace Variable, a List Item refers to the value that replaces a specific variable placeholder.

For both of these types, you can set more than one value. Furthermore, after you set a filter containing multiple values in a dashboard, you can select a value as needed when viewing the dashboard.

- **Alias:** Set the column alias.

This parameter is only required by the Filter type. After you set a column alias for a filter, the alias is shown in the filter in the dashboard.

- **Global filter:** Set whether to filter a specific value globally (that is, applied to all fields).

By default, this parameter is disabled. Furthermore, you can only set this parameter for the Filter type.

- **List Item:** There are two types Static List Item and Dynamic List Item.
 - **Static List Item:** You need to manually add list items to the filter. To add items, in the Add Static List Item box, enter list items and click Add.
 - **Dynamic List Item:** Set the system to dynamically display list items in the filter. The list items are the results of the search-and-analysis statement that you set. To enable the Add Dynamic List Item switch, enter a search-and-

analysis statement, and then click Search. Then, the dynamic list items are displayed.

Add Filter ✕

Filter Name:

Display Settings: Title Border Background

Key:

Type: Filter Replace Variable

Alias:

Global filter:

Add Static List Item:

Static List Items:

Add Dynamic List Item:

Select Logstore:

1 *|**SELECT DISTINCT** method

Dynamic List Item Preview

method
GET
POST

6. Click OK.

Examples

In the case that you have collected Nginx logs, you can analyze the Nginx logs by one of the two following methods. (For information about how to collect Nginx logs, see [#unique_202.](#))

- **Example 1: Analyze the logs by using different time granularities**

You can use a search-and-analysis statement to view the PV in terms of different time granularities, such as the second or minute granularity. To change the time granularity to be used, you must change the value of `__time__ - __time__ % 60`. Changing the search-and-analysis statement is the typical method,

which is inefficient for multiple times of operations. If you set a filter, you can more quickly replace the target placeholder variable with a specific value.

1. Use the following search-and-analysis statement to view the PV each minute:

```
* | SELECT date_format( __time__ - __time__ % 60, '%H:%i:%s') as time, count(1) as count GROUP BY time ORDER BY time
```

2. Add the search-and-analysis statement as a chart to the target dashboard, select `60` in the statement to generate a placeholder variable, and set the variable as `interval`. The default value of this variable is `60`.

time	count
15:32:00	108
15:33:00	56
15:34:00	78
15:35:00	107
15:36:00	79
15:37:00	120
15:38:00	106
15:39:00	102
15:40:00	66
15:41:00	83

Variable Config:

- * Variable Name: interval
- * Default Value: 60

Result

```
* | SELECT date_format(__time__ - __time__ % $interval, '%H:%i:%s') as time, count(1) as count GROUP BY time ORDER BY time
```

3. Set a filter in the target dashboard.

- **Type:** Select Replace Variable.
- **Key:** Enter `interval`.
- **Static List Item:** Add `1` (each second) and `120` (every two minutes) as the static list items.

Add Filter ✕

Filter Name:

Display Settings: Title Border Background

Key:

Type: Filter Replace Variable

Add Static List Item:

Static List Items:

Add Dynamic List Item:

[For more information about filter usage, see Documentation](#)

4. In the target dashboard, select **1** from the Search drop-down list of the filter to view the data recorded in seconds.

Variable: interval: 1 ✕

Time control

interval: Search

PV-01 15Minutes(Relative)

time	count
17:15:00	103
17:16:00	112
17:17:00	68
17:18:00	157



Note:

The search-and-analysis statement that has a new placeholder variable value is then changed to the following:

```
* | SELECT date_format ( __time__ - __time__ % 1 , '% H
: % i : % s ' ) as time , count ( 1 ) as count GROUP BY
time ORDER BY time
```

- **Example 2: Dynamically switch filter methods**

By adding dynamic list items to a filter, you can dynamically switch different request methods. In example 1, the search-and-analysis statement that starts with * means no condition is set to filter the results of the statement with the statement

applies to all logs. You can add a new filter to view the PV data of different request methods.

1. Set a filter in the target dashboard.

- **Type:** Select the Filter type.
- **Key:** Enter `method`.
- **Select Logstore:** Select the Logstore where the target dashboard belongs.
- **Add Dynamic List Item:** Enable this switch, and enter a search-and-analysis statement.

Add Filter ✕

Filter Name:

Display Settings: Title Border Background

Key:

Type: Filter Replace Variable

Alias:

Global filter:

Add Static List Item:

Static List Items:

Add Dynamic List Item:

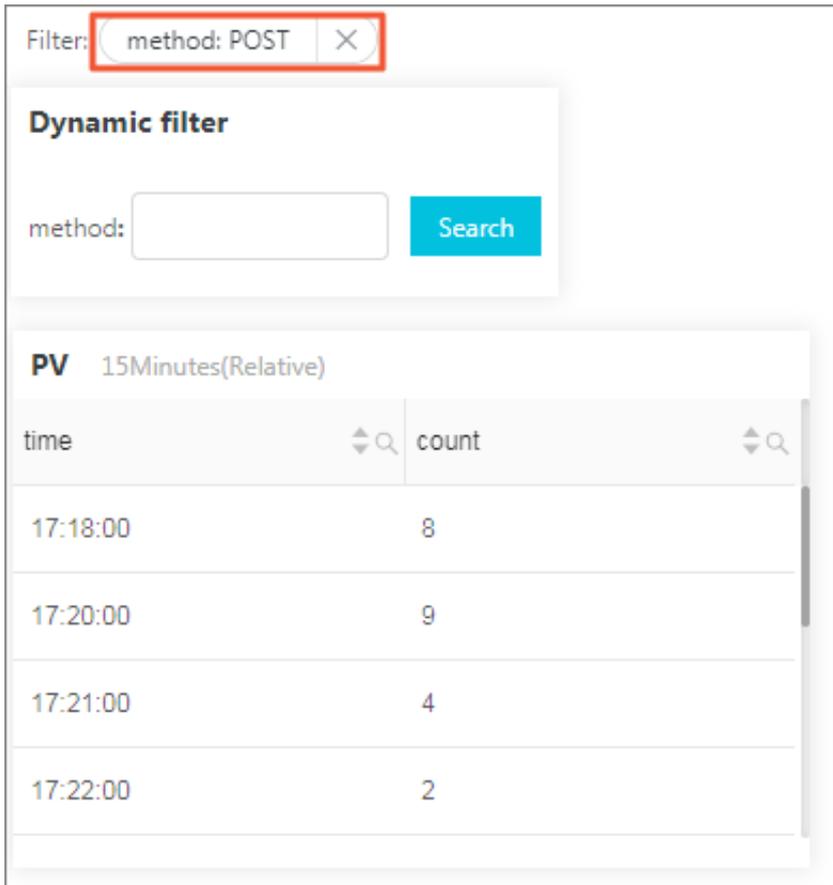
Select Logstore:

1 * | **SELECT DISTINCT** method

Dynamic List Item Preview

method
GET
POST

2. In the target dashboard, select **POST** from the Search drop-down list of the filter.



Filter: method: POST ×

Dynamic filter

method: Search

PV 15Minutes(Relative)

time	count
17:18:00	8
17:20:00	9
17:21:00	4
17:22:00	2



Note:

The chart only displays the data of the accesses that use the `POST` method.

The search-and-analysis statement is then changed to the following:

```
(*) and ( method : POST ) | SELECT date_format ( __time__
- __time__ % 60 , '% H :% i :% s ' ) as time , count ( 1
) as count GROUP BY time ORDER BY time
```

12.2.8 Markdown chart

With Log Service, you can add a markdown chart to the dashboard. In the markdown chart, you can insert images, links, videos, and other elements to make your dashboard page more friendly.

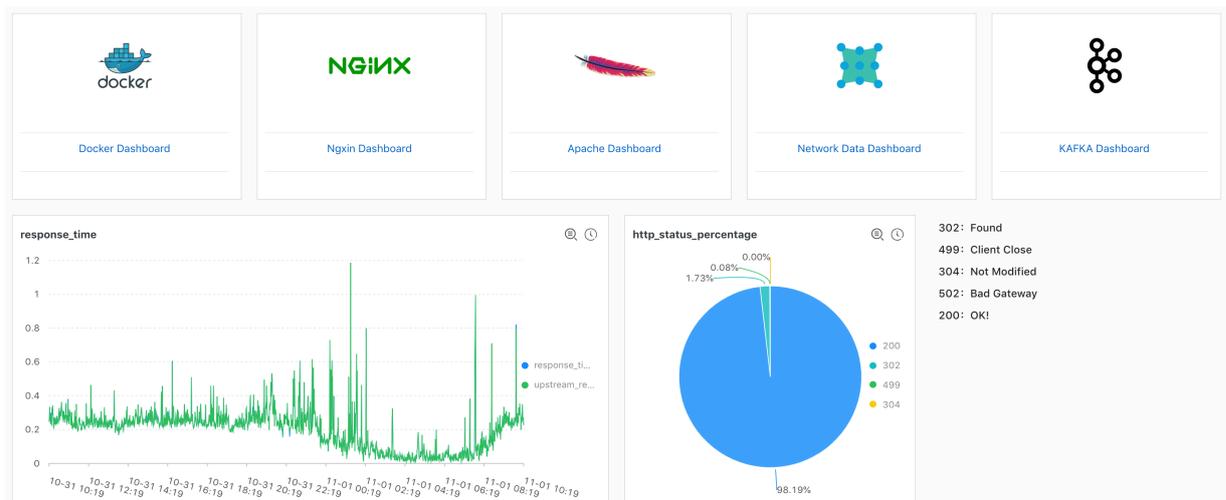
By adding multiple analysis charts to the dashboard when querying and analyzing log data, you can quickly view multiple analysis results and monitor the status of multiple services in real time. With Log Service, you can also add a markdown chart to the dashboard. The markdown chart is edited by using the markdown language. You can insert images, links, videos, and other elements to the markdown chart to make your dashboard page more friendly.

Markdown charts are created according to different requirements. To optimize the dashboard information expression, you can insert text such as background information, chart descriptions, page notes, and extension information in a markdown chart. To easily switch to other query pages, you can insert saved searches or dashboard links of other projects in a markdown chart. To enrich your dashboard information and make your dashboard functions more flexible, you can insert custom images in a markdown chart.

Scenarios

By using a markdown chart, you can customize links that redirect to other dashboards of the current project. You can also insert an image to go with each link to make it easier to tell them apart. You can also insert a markdown chart to describe the parameters in a chart.

Figure 12-20: Scenarios



Prerequisites

1. Log data is collected.
2. A dashboard is configured.

Procedure

1. On the Dashboard page, click Edit in the upper-right corner.
2. Click Create Markdown.

3. In the displayed page, configure markdown chart properties.

Configuration item	Description
Chart name	Name of your markdown chart.
Show border	Turn on the Show Border switch to add borders for your markdown chart.
Show title	Turn on the Show Title switch to display your markdown chart title in the dashboard.
Show background	Turn on the Show Background switch to add white background for your markdown chart.

4. Edit the Markdown Content.

In the Markdown Content area, enter markdown statements. The Show Chart section on the right displays the preview in real time. Modify the markdown statements according to the preview content.

5. After you complete the configuration, click OK.

Figure 12-21: Create a markdown chart

Create Markdown

* Chart Name

* Show Border

* Show Title

* Show Backgr

Markdown Content

```
# level 1 title
## level 2 title
### level 3 title

This is the body.

[Link](https://help.aliyun.com/document_detail/69313.html)
```

level 1 t

level 2 tit

level 3 title

This is the body.

[Link](#)

Markdown tags will be rendered in real-time on the left. [Learn more > Documentation](#)

After you complete the configuration, the created markdown chart is displayed under the current dashboard.

Modify a markdown chart

- **Modify the chart location and size**
 1. On the Dashboard page, click Edit in the upper-right corner.
 2. Drag the markdown chart to adjust its location, and drag the lower-right corner of the chart to adjust its size.
 3. Click Create in the upper-right corner.
- **Modify the chart title**
 1. On the Dashboard page, click Edit in the upper-right corner.
 2. Enter a new title in the chart title box.
 3. On the Dashboard page, click Save in the upper-right corner, and click OK in the displayed dialog box.
- **Modify the chart content**
 1. On the Dashboard page, click Edit in the upper-right corner.
 2. Click Edit in the upper-right corner of the markdown chart.
 3. Modify the chart configuration and click OK.
- **Delete a chart**
 1. On the Dashboard page, click Edit in the upper-right corner.
 2. Click Delete in the upper-right corner of the markdown chart.
 3. On the Dashboard page, click Save in the upper-right corner, and click OK in the displayed dialog box.

Common markdown syntax

- **Title**

Markdown statement:

```
# Level 1 title
## Level 2 title
```

```
### Level 3 title
```

Figure 12-22: Title preview

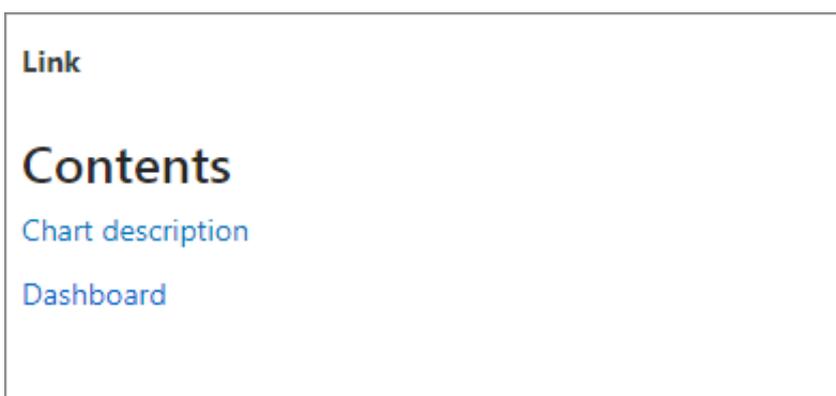


- **Link**

Markdown statement:

```
### Contents  
[ Chart description ]( https :// www . alibabacloud . com /  
help / doc - detail / 69313 . html )  
[ Dashboard ]( https :// www . alibabacloud . com / help / doc -  
detail / 59324 . html )
```

Figure 12-23: Link preview



- **Image**

Markdown statement:

```
< div align = center >  
![ Alt txt ][ id ]
```

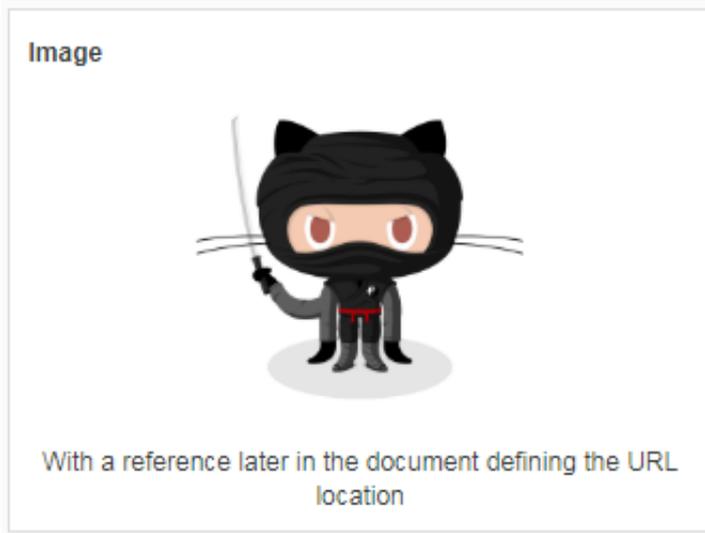
```

With a reference later in the document defining the
URL location

[ id ]: https://octodex.github.com/images/dojocat.jpg
" The Dojocat "

```

Figure 12-24: Image preview



- Special mark

Markdown statement:

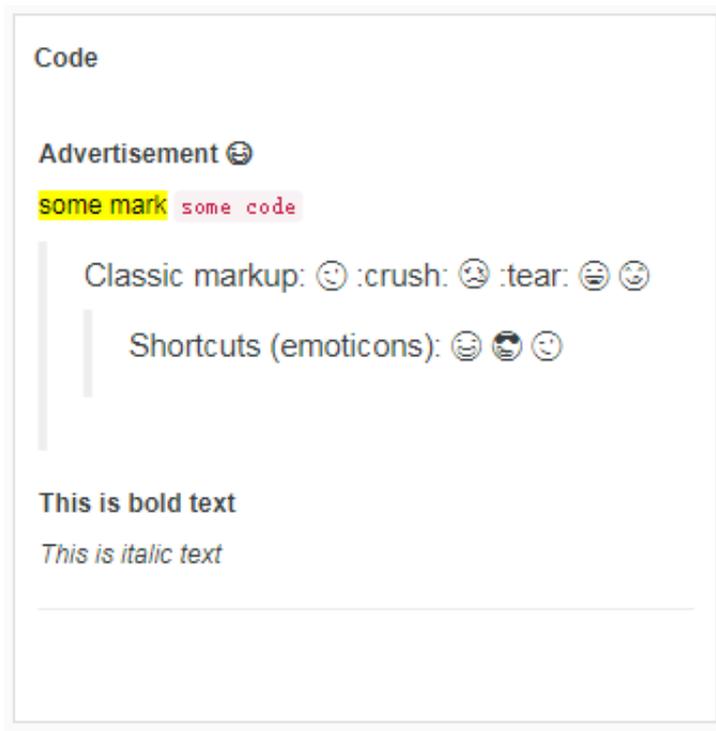
```

---
__Advertis ement :) __
== some mark == ` some code `
> Classic markup :: wink :: crush :: cry :: tear :: laughing
:: yum :
>> Shortcuts ( emoticons ): :-) 8 -) ;)
__This is bold text__
* This is italic text *

```



Figure 12-25: Special mark preview



For more information about markdown syntax, see [Markdown syntax](#).

12.3 Other visualization methods

12.3.1 Console sharing embedment

After configuring the collection and query analysis functions for Log Service, you may want to directly use the log query analysis and dashboard functions, or share these log-related functions with other users. In this case, using RAM for sharing may generate management costs for many subaccounts. To avoid this, Log Service allows you to log on to embedded pages through a single point for integrated query analysis and dashboard.

Context

Benefits

You can embed a specific Logstore query page and dashboard page into a self-built website. This gives you access to the analysis and visualization features of Log Service without logging on to Alibaba Cloud.

- The independent query page and dashboard page can be easily embedded into any website.
- You can generate a logon link by using the security token service (STS) and control the operation permissions, such as ready-only permission, by using remote access management (RAM).

Procedure

1. Log on to your self-built website.

After logon, the Web server STS obtains a temporary identity for you.

- For more information on STS, see [#unique_206](#).
- Grant the user access to specified Logstores. For details, see [#unique_207](#).

2. Request Alibaba Cloud logon service for the logon token.

After getting the temporary AccessKey pair and security token from STS, call the logon service interface to obtain the logon token.



Note:

The security token returned by STS may contain special characters. When the token contains special characters, encode them with URL-encoding method before using the token.

Request example:

```
http://signin.aliyun.com/federation?Action=GetSignInToken
&AccessKeyId=<TemporaryAccessKey
pair returned by the STS>
&AccessKeySecret=<Temporarysecret
returned by the STS>
&SecurityToken=<Security token returned
by the STS>
&TicketType=mini
```

3. Generate a logon-free link.

- Generate an access link along with the link to the embedded page after getting the logon token.

The token is valid for three hours. Therefore, we recommend you generate a new logon token and redirect each access request to an embedded link to your self-built website through a 302 message.

Request example:

```
http://signin.aliyun.com/federation?Action=Login
```

```

    & LoginUrl =< Address to which a
logon request is redirected upon a logon failure
, which is usually configured to the URL on your
self - built website through a 302 message ;>
    & Destination =< Log Service
page to be accessed . Pages for query and
dashboard are supported .>
    & SigninToken =< Logon token
obtained >

```

b) Embedded page.

- A complete page for query and analysis (multiple tags are allowed):

```

https://sls4service.console.aliyun.com/next/
project/<Project name>/logsearch/<Logstore name>?
hideTopbar=true&hideSidebar=true

```

- Query page:

```

https://sls4service.console.aliyun.com/next/
project/<Project name>/logsearch/<Logstore name>?
isShare=true&hideTopbar=true&hideSidebar=true

```

- Dashboard page:

```

https://sls4service.console.aliyun.com/next/
project/<Project name>/dashboard/<Dashboard name>?
isShare=true&hideTopbar=true&hideSidebar=true

```

The sample code in Java, PHP, and Python is as follows:

- **Java:**

```

< dependency >
    < groupId > com . aliyun </ groupId >
    < artifactId > aliyun - java - sdk -
sts </ artifactId >
    < version > 3 . 0 . 0 </ version >
</ dependency >
< dependency >
    < groupId > com . aliyun </ groupId >
    < artifactId > aliyun - java - sdk -
core </ artifactId >
    < version > 3 . 5 . 0 </ version >
</ dependency >
< dependency >
    < groupId > org . apache . httpcompon
ents </ groupId >
    < artifactId > httpclient </
artifactId >
    < version > 4 . 5 . 5 </ version >
</ dependency >
< dependency >
    < groupId > com . alibaba </ groupId >
    < artifactId > fastjson </ artifactId
>
    < version > 1 . 2 . 47 </ version >

```

```
</ dependency >
```

- [PHP](#)
- [Python](#)

12.3.2 Console embedment parameters

You can set relevant parameters to customize the display effects on webpages when the Log Service console is embedded into a self-built website.

Log Service allows you to [embed the console into a self-built website and access the console without logon](#). Then, you can quickly and conveniently query and analyze logs in a visualized manner. In addition, Log Service also provides parameters for you to customize the UI and integrate the console UI with third-party webpages.

URL encoding

All UI parameters are URL-encoded in the following format:

```
https://sls4service.console.aliyun.com/next/project/${ProjectName}/{  
  logsearch | savedsearch | dashboard }/${LogstoreName}/?Parameter  
1&Parameter 2
```

Parameter types are as follows:

- [Common parameters](#)
- [Parameters of the raw log query page](#)
- [Parameters of the saved search page](#)
- [Common dashboard parameters and advanced dashboard parameters](#)

The following example shows the URL of the raw log query page, where the `readOnly` parameter is a UI customization parameter that specifies whether to hide all editing buttons on the page.

```
https://sls4service.console.aliyun.com/next/project/  
projectaaa/logsearch/logstorebbb/?readOnly=true
```



Note:

- All parameters except for `${ProjectName}` and `${LogstoreName}` must be placed after `/?` at the end of a URL.
- You can set multiple parameters in a URL and separate them with an ampersand (&).

Common parameters

Function	Parameter	Type	Required	Description	Example
Hide sidebars	hideTopbar	Boolean	No	Specifies whether to hide the top navigation bar.	hideTopbar=true
	hideSidebar	Boolean	No	Specifies whether to hide the left-side navigation pane.	hideSidebar=true
Hide the back button in the navigation pane	hiddenBack	Boolean	No	Specifies whether to hide the back button on the Logstores page.	hiddenBack=true
Filter the resources listed in the navigation pane	keyFilter	JSON	No	<p>The filtered resources listed in the navigation pane. Valid values:</p> <ul style="list-style-type: none"> logstore savedsearch dashboard <p> Note:</p> <ul style="list-style-type: none"> You can use a hyphen (-) to specify fuzzy match. The parameter value must be in JSON format and URI-encoded. 	<pre>{ "logstore": ["logstore-xx"], "savedsearch": ["savedsearch-xx"], "dashboard": ["dashboard-xx"]} </pre>

Function	Parameter	Type	Required	Description	Example
Configure a time selector	queryTimeType	Integer	No	<p>The time range of queried data.</p> <p>Valid values:</p> <ul style="list-style-type: none"> · [1, 26]: specifies an integer in the interval. Each integer corresponds to a time range. For more information, see the following table. · 99: specifies a custom time range. In this case, you must set the <code>startTime</code> and <code>endTime</code> parameters. 	queryTimeType=1
	startTime	Timestamp (date)	No	The start time of queried data. This parameter is valid only when the <code>queryTimeType</code> parameter is set to 99.	startTime=1547776643
	endTime	Timestamp (date)	No	The end time of queried data. This parameter is valid only when the <code>queryTimeType</code> parameter is set to 99.	endTime=1547776731

queryTimeType	Description
1	1 minute (relative)
2	5 minutes (relative)
3	15 minutes (relative)
4	1 hour (relative)
5	4 hours (relative)
6	1 day (relative)
7	1 week (relative)

queryTimeType	Description
8	30 days (relative)
9	This month (relative)
10	Custom (relative)
11	1 minute (time frame)
12	15 minutes (time frame)
13	1 hour (time frame)
14	4 hours (time frame)
15	1 day (time frame)
16	1 week (time frame)
17	30 days (time frame)
18	Today (time frame)
19	Yesterday (time frame)
20	The day before yesterday (time frame)
21	This week (time frame)
22	Last week (time frame)
23	This month (time frame)
24	This quarter (time frame)
25	This year (time frame)
26	Custom (time frame)
99	Custom time range. In this case, you must set the start <code>Time</code> and end <code>Time</code> parameters.

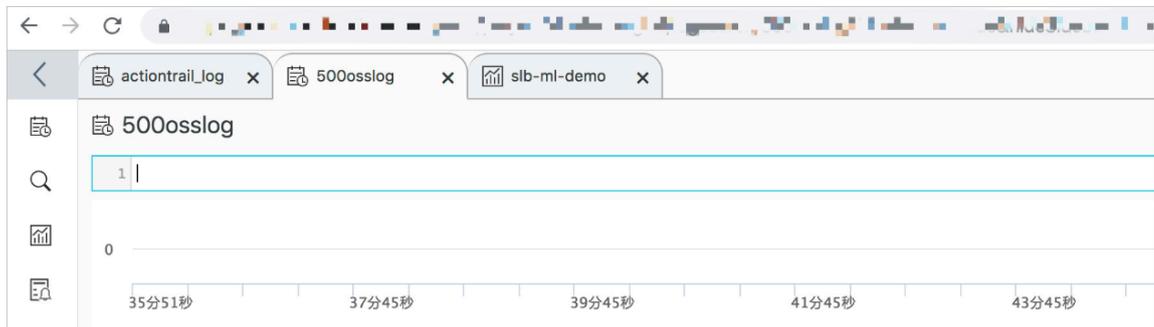
- **Hide sidebars**

- **URL**

```
https://sls4service.console.aliyun.com/next/project/${ProjectName}/logsearch/${LogstoreName}?hideTopbar=true&hideSidebar=true
```

- **Display effects**

To hide the top navigation bar and left-side navigation pane on the search page, set parameters as follows: `hideTopbar=true&hideSidebar=true`. The following figure shows the display effects.



- **Hide the back button in the navigation pane**

- **URL**

To hide the back button in the upper-left corner of the raw log query page, set the `hiddenBack` parameter to true in the URL as follows:

```
https://sls4service.console.aliyun.com/next/project/${ProjectName}/logsearch/${LogstoreName}?hiddenBack=true
```

- **Display effects**



- Filter the resources listed in the navigation pane

- URL

Set the `keyFilter` parameter in JSON format in the URL to filter the resources listed in the left-side navigation pane. For example, you need to display Logstores whose names contain `aegis -` and whose name is `500osslog`, saved search items whose names contain `oss`, and dashboards whose names contain `ddos`.

The JSON-formatted value is `{" logstore ":[" aegis -", " 500osslog "], " savedsearch ":[" oss "], " dashboard ":[" ddos "]}`, where `aegis -` specifies that all Logstores whose names contain `aegis` are queried in fuzzy match mode and `500osslog` specifies that the Logstore whose name is `500osslog` is queried in exact match mode. You can use a hyphen (-) to specify fuzzy match.

```
https://sls4service.console.aliyun.com/next/project/  
/${ProjectName}/logsearch/${LogstoreName}?keyFilter=  
%7B%22logstore%22:%5B%22aegis-%22%2C%22500osslog%22%5D%2C%22savedsearch%22:%5B%22oss%22%5D%2C%22dashboard%22:%5B%22ddos%22%5D%7D
```

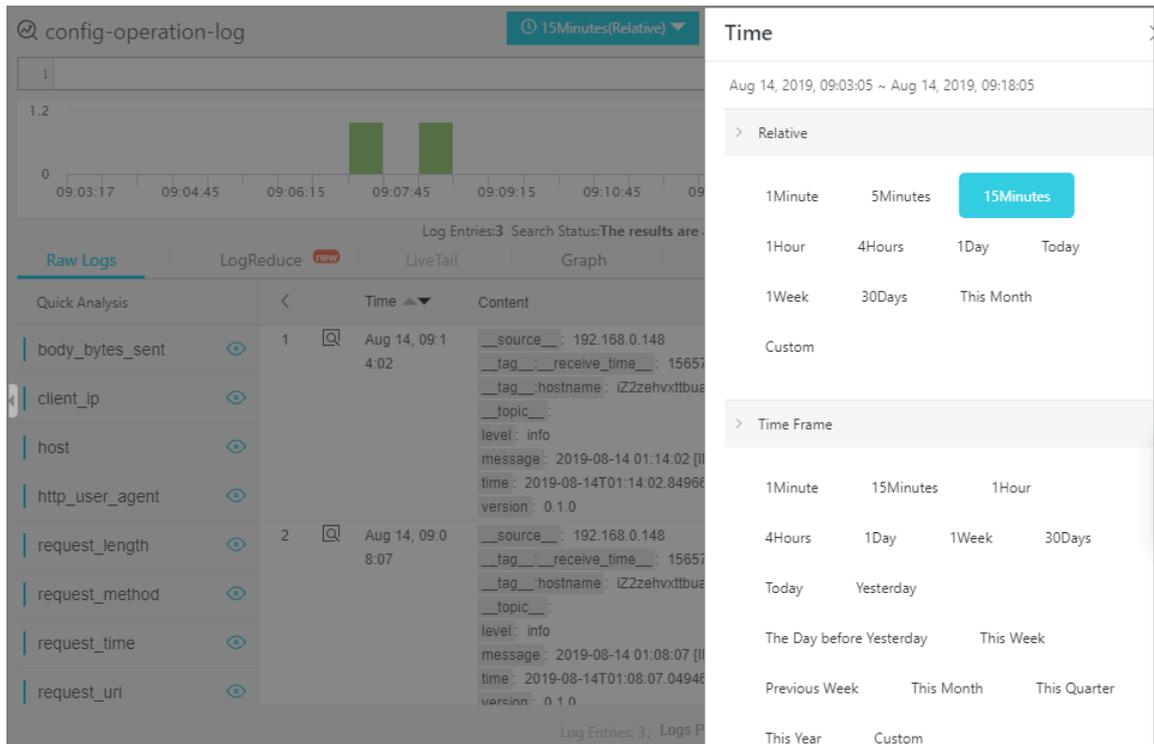
- Display effects

- Configure a time selector

- URL

```
https://sls4service.console.aliyun.com/next/project/${ProjectName}/logsearch/${LogstoreName}?queryTimeType=2
```

- Display effects



Parameters of the raw log query page

Parameter	Type	Description	Required
ProjectName	String	The name of the project.	Yes
LogstoreName	String	The name of the Logstore.	Yes
queryString	String	The string to be queried.	No
readOnly	Boolean	Specifies whether to hide the editing and modifying buttons on the Logstores page, such as Share, Index Attributes, Save Search, and Saved as Alarm.	No

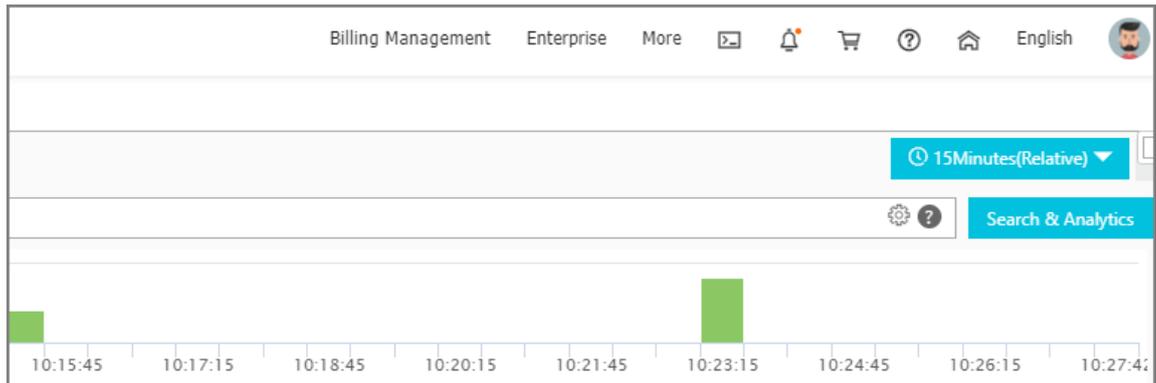
Example

- Set the queryString parameter

- URL

```
https://sls4service.console.aliyun.com/next/project/{ProjectName}/logsearch/{LogstoreName}?queryString=*|selectcount(1)aspv,statusgroupbystatus
```

- Display effects

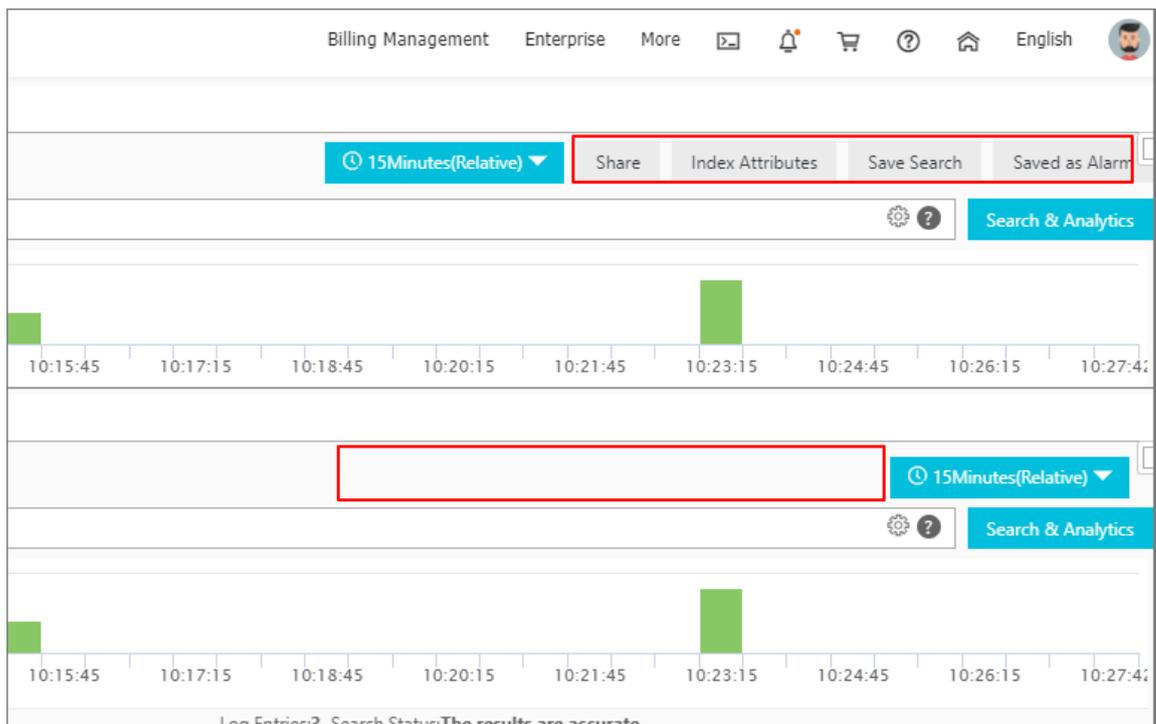


- Set the readOnly parameter

- URL

```
https://sls4service.console.aliyun.com/next/project/{ProjectName}/logsearch/{LogstoreName}?readOnly=true
```

- Display effects



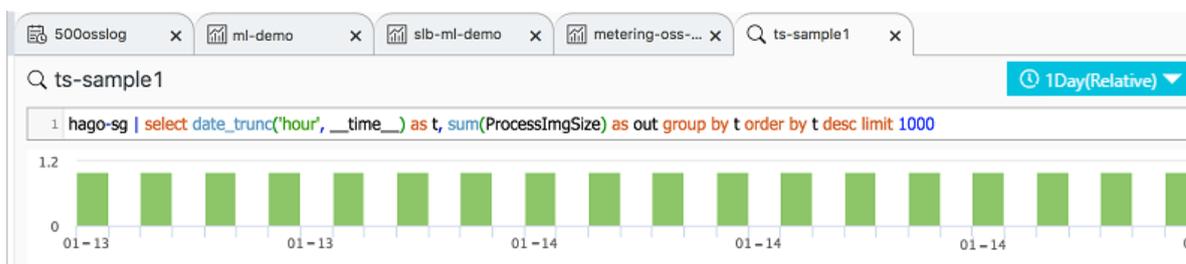
Parameters of the saved search page

Parameter	Type	Description	Required
ProjectName	String	The name of the project.	Yes
savedSearchName	String	The name of the saved search.	Yes

• URL

```
https://sls4service.console.aliyun.com/next/project/{ProjectName}/savedsearch/{savedSearchName}
```

• Display effects



Parameters of the dashboard page

Parameter	Type	Description	Required
ProjectName	String	The name of the project.	Yes
dashboardName	String	The name of the dashboard.	Yes
filters	String	The filter conditions. The value of this parameter must be manually encoded. The original format is <code>encodeURIComponent('filters=key1:value1&filters=key2:value2')</code> After being encoded, the format is " <code>filters=%3Dkey1%3Avalue1%26filters%3Dkey2%3Avalue2</code> ".	No

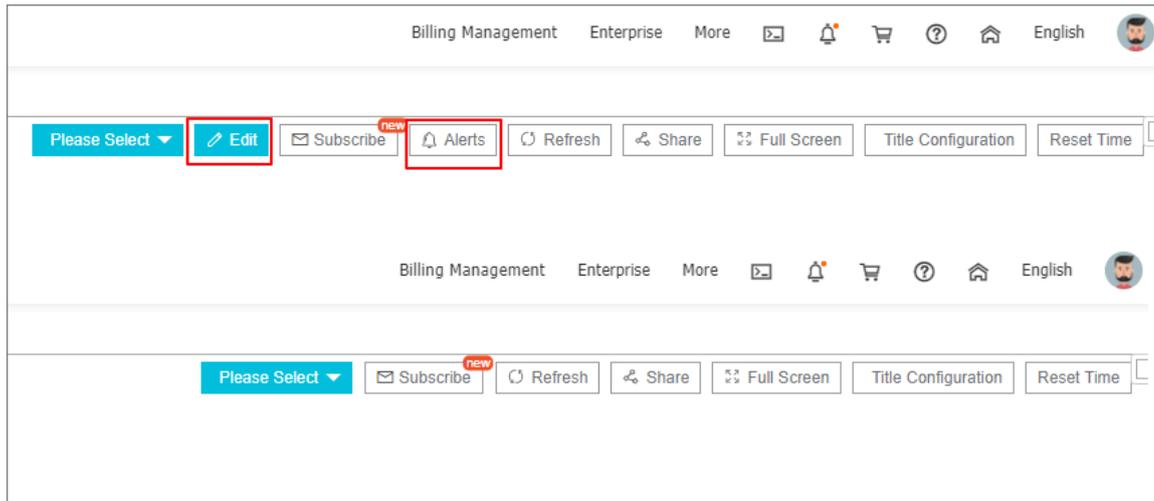
Parameter	Type	Description	Required
token	JSON string	<p>The variables to be replaced. The value of this parameter must be manually encoded.</p> <p>The original format is</p> <pre>encodeURIComponent('token=[{"key": "projectName", "value": "1"}, {"key": "xxx", "value": "yy"}]')</pre> <p>After being encoded, the format is " token % 3D % 5B % 7B % 22key % 22 % 3A % 20 % 22projectName % 22 % 2C % 22value % 22 % 3A % 221 % 22 % 7D % 2C % 20 % 7B % 22key % 22 % 3A % 20 % 22xxx % 22 % 2C % 20 % 22value % 22 % 3A % 20 % 22yy % 22 % 7D % 5D % 22".</p>	No
readOnly	Boolean	Specifies whether to hide the editing and setting buttons on the dashboard page, such as Edit and Alerts.	No
hiddenFilter	Boolean	Specifies whether to hide the filter conditions.	No
hiddenToken	Boolean	Specifies whether to hide the variables to be replaced.	No
autoFresh	String	The interval for refreshing the dashboard page, such as 30 seconds or 5 minutes . The minimum refresh interval is 15 seconds.	No

- Set the readOnly parameter in the URL to hide editing buttons on the dashboard page.

- URL

```
https://sls4service.console.aliyun.com/next/project/{ProjectName}/dashboard/{dashboardId}?readOnly=true
```

- Display effects



- Add two filter conditions, namely, key1=value1 and key2=value2, for the dashboard page. Then, the system uses query and analysis statements to filter all charts based on the filter conditions and continues to use these statements.



Note:

You need to encode the value of the filters parameter.

- URL

```
https://sls4service.console.aliyun.com/next/project/{ProjectName}/dashboard/{dashboardId}?encodeURIComponent('filters=key1:value1&filters=key2:value2')
```

- Display effects

- Add multiple conditions for replacing variables.



Note:

You need to encode the value of the token parameter.

- URL

```
https://sls4service.console.aliyun.com/next/project/${ProjectName}/dashboard/${dashboardId}?encodeURIComponent('token=[{"key":"projectName","value":"1"}, {"key":"xxx","value":"yy"}]')
```

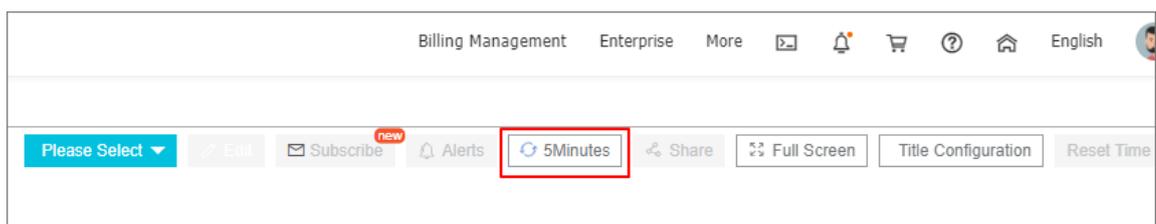
- Display effects

- Set the autoFresh parameter in the URL to refresh the dashboard page every 5 minutes.

- URL

```
https://sls4service.console.aliyun.com/next/project/${ProjectName}/dashboard/${dashboard_name}?autoFresh=5m
```

- Display effects



Advanced parameters of the dashboard page

If you embed an iframe into the dashboard page, the height of the iframe cannot be determined. In this case, two scroll bars may appear at the same time as follows:

- The scroll bar of the dashboard page outside the iframe.
- The scroll bar on the dashboard page in the iframe.

To solve this problem, you can use the postMessage method dashboardHeight of Log Service to obtain the height of the dashboard page and use the obtained value as the height of the iframe.

The sample code is as follows:



Note:

You need to replace parameters such as `${projectName}` with actual values.

```
<!DOCTYPE html >
<html lang="en">
<head >
```

```
< meta charset = " UTF - 8 ">
< title > POST message test </ title >
</ head >
< style >
* {
padding : 0 ;
margin : 0 ;
}

iframe {
display : block ;
width : 100 %;
}
</ style >
< body >
< script >
window . addEventLi stener ( ' message ', function ( e ){
console . log ( e . data . dashboardH eight )
document . getElement ById ( ' test ' ) . style . height = e .
data . dashboardH eight + ' px '
});
</ script >
< div style = " height : 700px ; "> somethings </ div >
< iframe id = " test " src = " http :// sls4servic e . console
. aliyun . com / next / project /${ projectNam e }/ dashboard /${
dashboardN ame }? hideTopbar = true & product =${ productCod e }">
</ body >
</ html >
```

12.3.3 Use JDBC to count and visualize logs

MySQL is a popular relational database. Many softwares support obtaining MySQL data by using MySQL transport protocol and SQL syntax. You can connect to MySQL if you know SQL syntax. Log Service provides MySQL protocol to query and analyze logs. You can use a standard MySQL client to connect to Log Service and use the standard SQL syntax to compute and analyze logs. Clients that support the MySQL transport protocol include MySQL client, JDBC, and Python MySQLdb.

Using bike sharing logs as an example, the following section describes how to use JDBC to connect to Log Service and read log data, the MySQL protocol and SQL syntax to compute logs, and DataV to visualize log data or computation results on a big screen.

JDBC scenarios:

- Use a visualization tool such as DataV, Tableau, or Kibana to connect to Log Service through the MySQL protocol.
- Use libraries such as JDBC in Java or MySQLdb in Python to access Log Service and process query results in the program.

Data example

A bike sharing log contains the user's age, gender, battery usage, vehicle ID, operation latency, latitude, lock type, longitude, operation type, operation result, and unlocking type. Data is stored in `Logstore : ebike` of project: `project : trip_demo`. The region where the project resides is `cn-hangzhou`.

A sample log is as follows:

```
Time : 10 - 12 14 : 26 : 44
__source__ : 11 . 164 . 232 . 105
__topic__ : v1
age : 55
battery : 118497 . 673842
bikeid : 36
gender : male
latency : 17
latitude : 30 . 2931185245
lock_type : smart_lock
longitude : 120 . 052840484
op : unlock
op_result : ok
open_lock : bluetooth
userid : 292
```

Prerequisites

Log indexing and analysis functions have been enabled for each column of Logstore through the console or API.

JDBC statistics

1. Create a Maven project and add JDBC dependency in pom dependency.

```
< dependency >
  < groupId > MySQL </ groupId >
  < artifactId > mysql - connector - java </ artifactId >
  < version > 5 . 1 . 6 </ version >
</ dependency >
```

2. Create a Java class and use JDBC in code for query.

```
/**
 * Created by mayunlei on 2017 / 6 / 19 .
 */
import com . mysql . jdbc .*;
import java . sql .*;
import java . sql . Connection ;
import java . sql . Statement ;
/**
 * Created by mayunlei on 2017 / 6 / 15 .
 */
public class jdbc {
  public static void main ( String args []){
    // Input your configurat ion here .
```

```

    final String endpoint = "cn - hangzhou - intranet . sls
. aliyuncs . com "; // Log Service intranet or VPC domain
name
    final String port = " 10005 "; // The MySQL protocol
port of Log Service .
    final String project = " trip - demo ";
    final String logstore = " ebike ";
    final String accessKeyId = "";
    final String accessKey = "";
    Connection conn = null ;
    Statement stmt = null ;
    try {
        // Step 1 : Load the JDBC driver .
        Class . forName ( " com . mysql . jdbc . Driver " );
        // Step 2 : Create a link .
        conn = DriverManager . getConnect ion ( " jdbc : mysql
://" + endpoint + ":" + port + "/" + project , accessKeyId , accessKey
);
        // Step 3 : Create a statement .
        stmt = conn . createStat ement ( );
        // Step 4 : Define query statements . Query the
number of logs that are generated on October 11
, 2017 and meet the condition op = " unlock ", and
query the average operation latency .
        String sql = " select count ( 1 ) as pv , avg (
latency ) as avg_latenc y from "+ logstore + " " +
" where __date__ >= ' 2017 - 10 - 11 00 : 00 :
00 ' " +
" and __date__ < ' 2017 - 10 - 12 00 : 00 :
00 ' " +
" and op =' unlock '";
        // Step 5 : Execute query conditions .
        ResultSet rs = stmt . executeQue ry ( sql );
        // Step 6 : Extract the query result .
        while ( rs . next ( ) ) {
            // Retrieve by column name
            System . out . print ( " pv : " );
            // Obtain pv from the result .
            System . out . print ( rs . getLong ( " pv " ) );
            System . out . print ( " ; avg_latenc y : " );
            // Obtain avg_latenc y in the result .
            System . out . println ( rs . getDouble ( " avg_latenc
y " ) );
            System . out . println ( );
        }
        rs . close ( );
    } catch ( ClassNotFo undExcepti on e ) {
        e . printStack Trace ( );
    } catch ( SQLExcepti on e ) {
        e . printStack Trace ( );
    } catch ( Exception e ) {
        e . printStack Trace ( );
    } finally {
        if ( stmt != null ) {
            try {
                stmt . close ( );
            } catch ( SQLExcepti on e ) {
                e . printStack Trace ( );
            }
        }
        if ( conn != null ) {
            try {
                conn . close ( );
            } catch ( SQLExcepti on e ) {

```

```
        e . printStack Trace ();  
    }  
} } }
```

Use DavaV to access and display data

Visualized large-screen DataV displays data and connects to Log Service to read log data or display log computation results.

1. Create data sources

You can select MySQL for RDS or Log Service as a data source as per your needs. The following section uses the MySQL protocol as an example to describe how to connect to Log Service.

As shown in the figure, select the corresponding region and the intranet, and enter an AccessKey for the username and password. The AccessKey can be of a main

account or a sub-account that has the read permission to Log Service. Set the port number to 10005 and the database name to the project name.

Figure 12-26: Editing data

New Data Source

* Type
RDS for MySQL

Intranet China East 1

VPC(Virtual Private Cloud)(Tutorial)

* Name
log_analytics

* Host
cn-hangzhou-intranet.log.aliyuncs.com

* Username
L7A4U8b2b0X0g0fL

* Password

* Port
10005

* Database
Database List
trip-demo
Test Connection

⚠ Before submitting, please ensure: IP Address White List

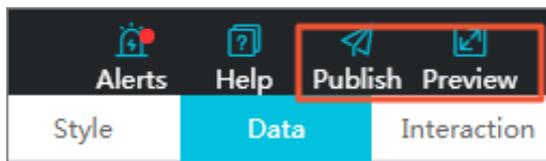
OK

2. Creates a view.

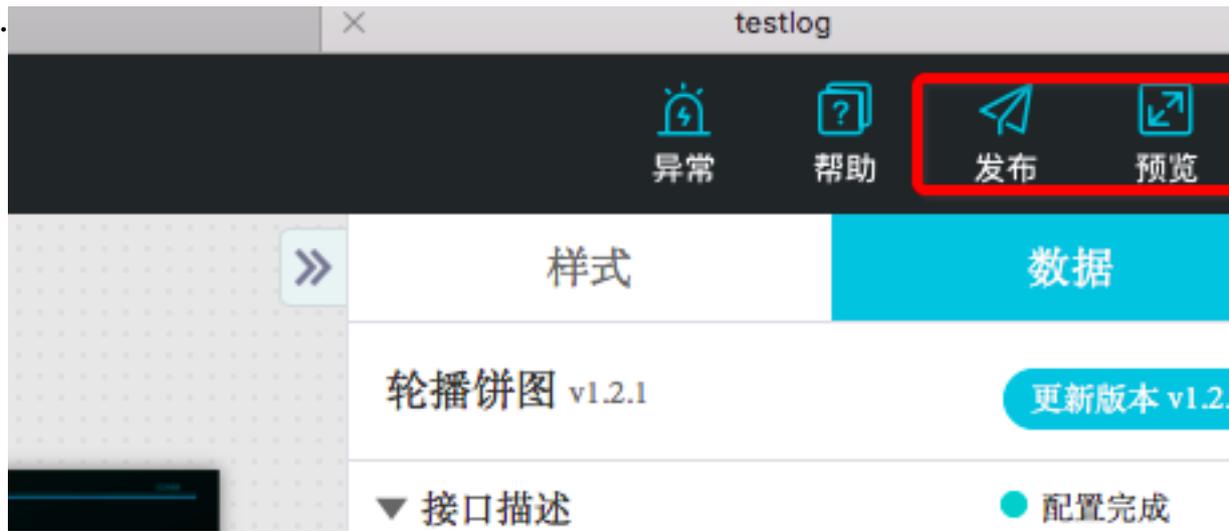
Select a service template in the view and click any view on the large screen. Right-click the view to modify data or the data source of the view.

As shown in the figure, select the database created in preceding steps as the data source, enter the queried SQL, and enter mappings between the query results and view fields in preceding field mappings.

Figure 12-27: Select the database



3. Preview the view and publish.



Click the Preview button to preview the view.

Figure 12-28: Preview

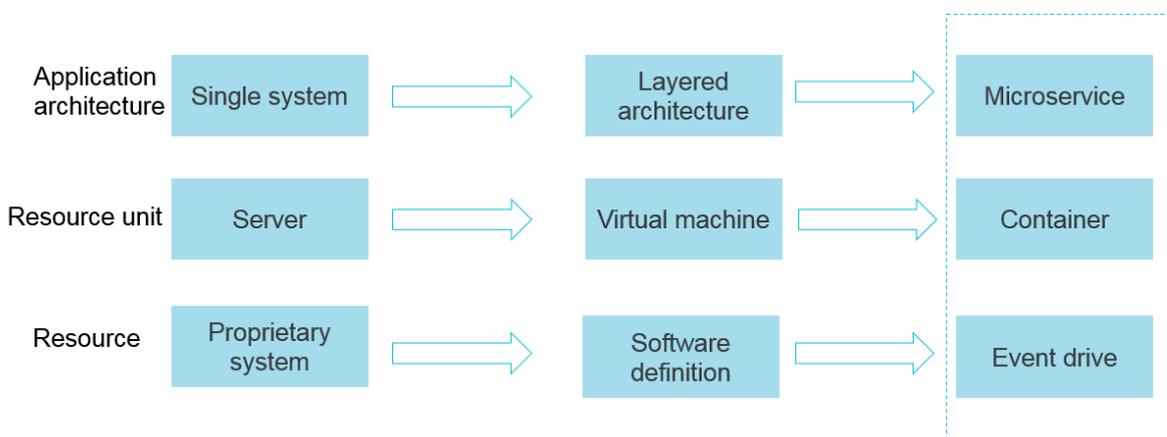


12.3.4 OpenTracing implementation of Jaeger

The advent of containers and serverless programming methods greatly increased the efficiency of software delivery and deployment. The evolution of the architecture has shown these changes:

- The application architecture is changing from a single system to microservices. Then, the business logic changes to the call and request between microservices.
- In terms of resources, traditional physical servers are fading out and changing to the invisible virtual resources.

Figure 12-29: Architectural evolution



According to these changes, behind the elastic and standardized architecture, the operations and maintenance (O&M) and diagnostics requirements become more and more complex. To address this trend, a series of development and operations (DevOps)-oriented diagnostic and analysis systems have emerged, including centralized logging systems, centralized metrics systems, and distributed tracing systems.

In addition to Jaeger, Alibaba Cloud also provides the OpenTracing link tracing service [XTrace](#).

Logging, metrics, and tracing systems

The features of logging, metrics, and tracing systems are described as follows:

- A logging system is used to record discrete events.

The system records data such as the debugging or error information of an application. This data is the basis of diagnostics.

- A metrics system is used to record data that can be aggregated.

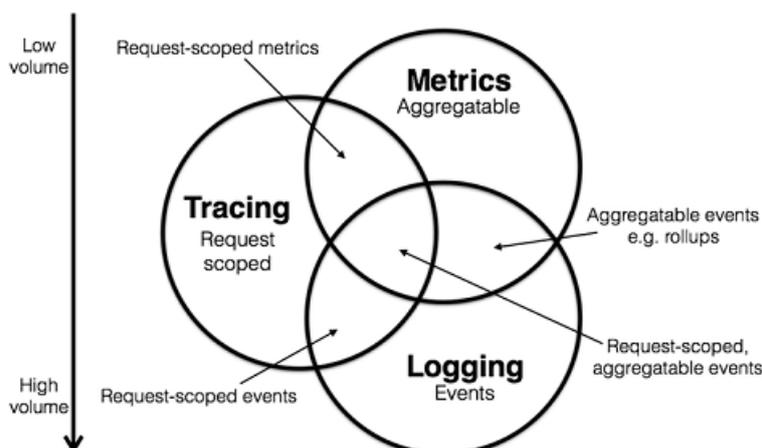
For example, the current depth of a queue can be defined as a metric and updated when an element is added to or removed from the queue. The number of HTTP requests can be defined as a counter that accumulates the number when new requests arrive.

- A tracing system is used to record information within the request scope.

The system records data such as the process and consumed time for a remote method call. This data is the tool we use to investigate system performance issues.

The logging, metrics, and tracing systems provide overlapped features as follows.

Figure 12-30: Logging, metrics, and tracing systems



Based on these descriptions, we can classify existing systems. For example, Zipkin focuses on tracing. Prometheus begins to focus on metrics and may integrate with more tracing features in the future, but can hardly deal with logging. Systems such as ELK and Alibaba Cloud Log Service begin to focus on logging, continuously integrate with features of other fields, and are moving toward the intersection of all three systems.

For more information, see [Metrics, tracing, and logging](#). Tracing systems are described in the following sections.

Background

The tracing technology has emerged since the 1990s. However, this technology moved into the mainstream with Google's article "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure". Meanwhile, the article "Uncertainty in Aggregate Estimates from Sampled Distributed Traces" described the more detailed analysis of sampling

. After these articles were published, a group of excellent tracing software programs were developed.

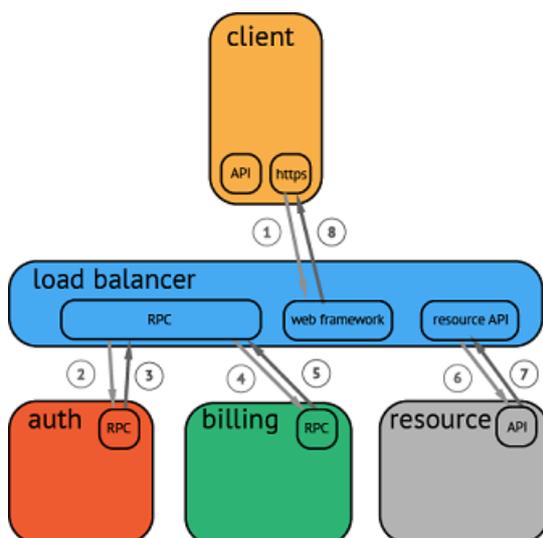
The following tracing software programs have been widely used:

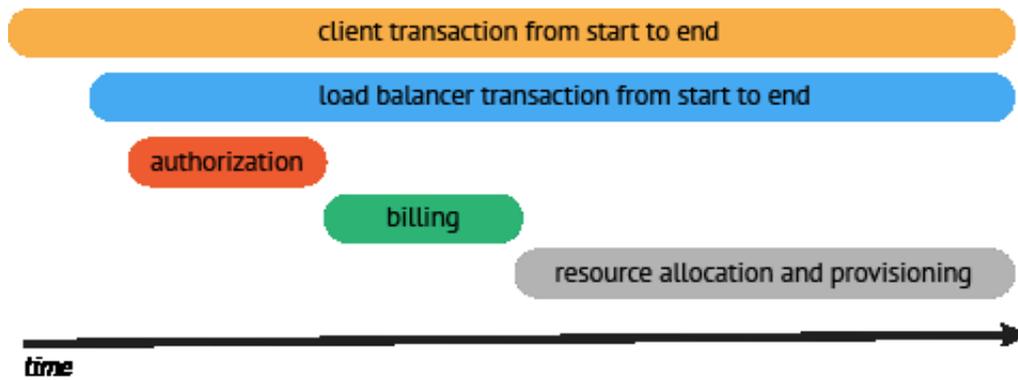
- Dapper (Google): the foundation for all tracers
- Stackdriver Trace (Google)
- Zipkin (Twitter)
- Appdash (Golang)
- EagleEye (Taobao)
- Ditecting (Pangu, the tracing system used by Alibaba Cloud cloud services)
- Cloud Map (Ant tracing system)
- sTrace (Shenma)
- X-Ray (AWS)

Distributed tracing systems have developed rapidly into many variants. However, they generally have three steps: code tracking, data storage, and query display.

The following example shows a distributed call. When a client initiates a request, the request first goes to the load balancer and then passes through the authentication service, billing service, and finally to the requested resources. Afterward, the system returns a result.

Figure 12-31: Example of a distributed call



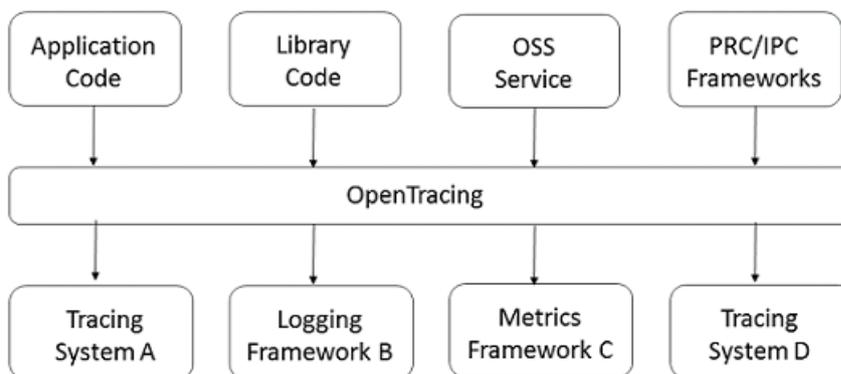


After collecting and storing the data, the distributed tracing system presents the traces in a timing diagram that contains a timeline. However, during data collection, the system has to intrude on user code and the API operations of different systems are not compatible. This causes great changes if you want to switch tracing systems.

OpenTracing

The [OpenTracing](#) standard was introduced to prevent API compatibility issues among different distributed tracing systems. OpenTracing is a lightweight standardization layer. This layer is located between applications or class libraries and tracing or log analysis programs.

Figure 12-32: OpenTracing



Benefits

- OpenTracing already enters Cloud Native Computing Foundation (CNCF) and provides unified concepts and data standards for global distributed tracing systems.
- OpenTracing provides APIs with no relation to platforms or vendors. This allows developers to easily add or change the implementation of tracing systems.

Data model

In a key-value pair, the key must be a string and the value can be of any type. However, be aware that tracers that support OpenTracing do not support all value types.

- SpanContext. This is the context of the span.
- References. They indicate the relationship between zero or multiple related spans. Spans establish the relationship based on the SpanContext.

Each SpanContext contains the following statuses:

- Any OpenTracing implementation must transmit the current call chain status such as trace and span identifiers across process boundaries based on a unique span.
- Baggage items that are the data accompanying a trace. They are a collection of key-value pairs stored in a trace and must be transmitted across process boundaries.

For more information about OpenTracing data models, see the OpenTracing semantic standards.

Implementations

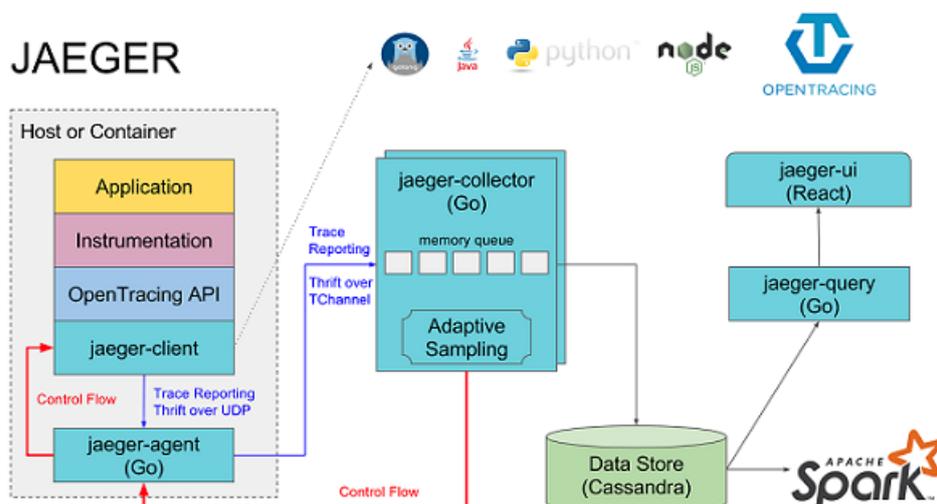
The document [Supported tracers](#) lists all OpenTracing implementations. In these implementations, [Jaeger](#) and [Zipkin](#) are widely used.

Jaeger

Jaeger is an open-source distributed tracing system released by Uber. This system is compatible with OpenTracing API operations.

Architecture

Figure 12-33: Architecture



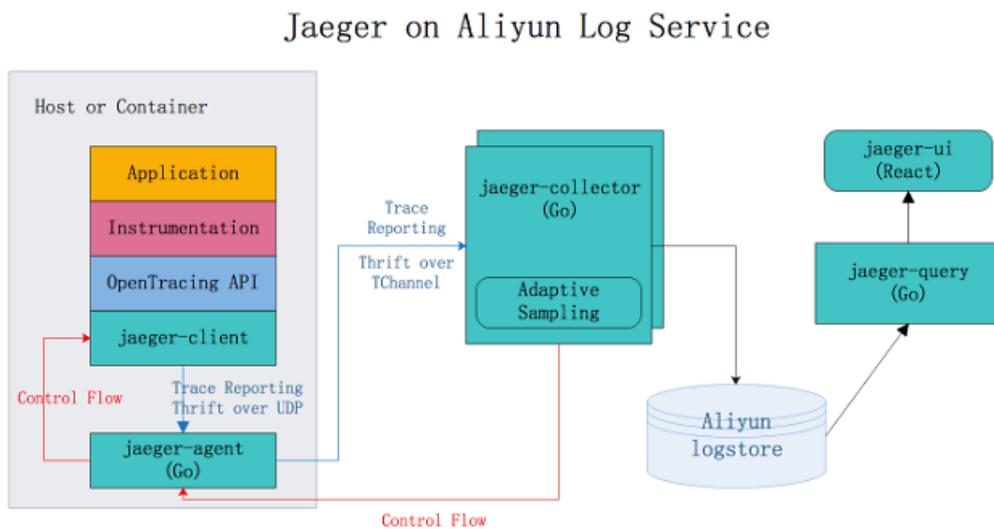
As shown in this figure, Jaeger consists of the following components:

- **Jaeger client:** implements SDKs that are compatible with OpenTracing standards for different programming languages. An application uses an API operation to write data. The client library transmits trace information to the jaeger-agent according to the sampling policy specified in the application.
- **Agent:** a network daemon that monitors span data received by the User Datagram Protocol (UDP) port and that sends multiple items of data to the collector at the same time. The agent is designed as a basic component and deployed on all hosts. The agent decouples the client library and the collector, and shields the client library from collector routing and discovery details.
- **Collector:** receives data from the jaeger-agent and then writes the data to backend storage. The collector is designed as a stateless component. Therefore, you can simultaneously run an arbitrary number of jaeger-collectors.
- **Data store:** the backend storage designed as a pluggable component that supports writing data to Cassandra and Elasticsearch.
- **Query:** receives query requests, retrieves trace information from the backend storage system, and then displays the result in the user interface (UI). Query is stateless. You can start multiple instances, and deploy the instances behind NGINX load balancers.

Jaeger on Alibaba Cloud Log Service

Jaeger on Alibaba Cloud Log Service is a Jaeger-based distributed tracing system. This system persists tracing data in Log Service, and queries and displays data by using the Jaeger native API operations.

Figure 12-34: Jaeger



Benefits

- The native Jaeger only supports persistent storage in Cassandra and Elasticsearch . You must maintain the stability of the backend storage system and adjust the storage capacity. Jaeger on Alibaba Cloud Log Service uses Log Service of Alibaba Cloud to process large amounts of data. In this way, you can easily benefit from the Jaeger distributed tracing technology, and save more efforts on the backend storage system.
- The Jaeger UI can query and display traces, but requires more support for analysis and troubleshooting. By using Jaeger on Alibaba Cloud Log Service, you can use the powerful query and analysis features of Log Service to efficiently analyze system issues.
- Compared with Jaeger that uses Elasticsearch as the backend storage, Log Service costs only 13% of the price of Elasticsearch when you use the pay-as-you-go billing method. For more information, see [Compare LogSearch/Analytics with ELK in log query and analysis](#).

Procedure

For more information, see [GitHub](#).

Example

HotROD is an application that consists of multiple microservices and uses the OpenTracing API operations to record trace information.

To use Jaeger on Alibaba Cloud Log Service to diagnose issues in HotROD, follow these steps as shown in the tutorial video:

1. Configure Log Service.
2. Run Jaeger by running the docker-compose command.
3. Run HotROD.
4. Use the Jaeger UI to retrieve the specified trace information.
5. Use the Jaeger UI to view detailed trace information.
6. Use the Jaeger UI to locate application performance bottlenecks.
7. Use the Log Service console to locate application performance bottlenecks.
8. The application calls the OpenTracing API operations.

Tutorial

<http://cloud.video.taobao.com//play/u/2143829456/p/1/e/6/t/1/50081772711.mp4>

You can use the following query statements in this example:

- Collect statistics on the average latency and requests of frontend service HTTP GET/dispatch operations in minutes.

```
process . serviceName : " frontend " and operationName : "
HTTP GET / dispatch " |
select from_unixtime ( __time__ - __time__ % 60 ) as
time ,
truncate ( avg ( duration ) / 1000 / 1000 ) as avg_duration_ms
,
count ( 1 ) as count
group by __time__ - __time__ % 60 order by time
desc limit 60
```

- Compare the time consumed by two trace operations.

```
traceID : " trace1 " or traceID : " trace2 " |
select operationName ,
( max ( duration ) - min ( duration ) ) / 1000 / 1000 as
duration_diff_ms
group by operationName
order by duration_diff_ms desc
```

- Collect statistics on trace IP addresses with latencies of more than 1.5 seconds.

```
process . serviceName : " frontend " and operationName : "
HTTP GET / dispatch " and duration > 1500000000 |
select " process . tags . ip " as IP ,
```

```
truncate ( avg ( duration ) / 1000 / 1000 ) as avg_durati on_ms  
,  
count ( 1 ) as count  
group by " process . tags . ip "
```

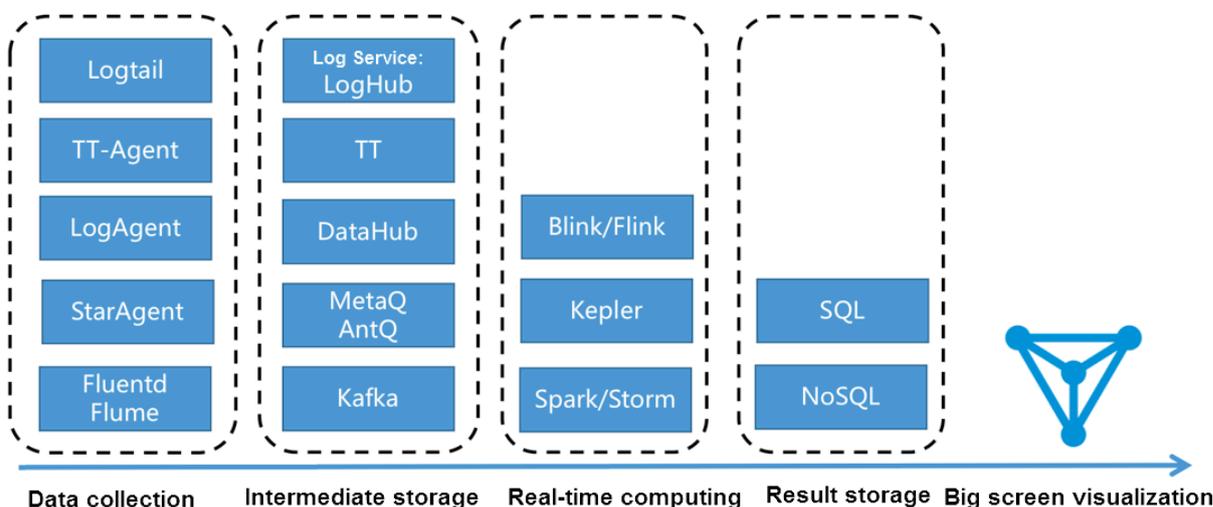
12.3.5 Interconnect with DataV big screen

People will think of the outstanding Tmall real-time big screen when talking about the Double 11 shopping campaign. The real-time big screen is impressive for its most typical stream computing architecture:

- Data collection: Collect data from each source in real time.
- Data collection: Collect data from each source in real time.
- Real-time computing: Subscribe to real-time data and compute data in windows by using the computing rules. This is the most important part in the process.
- Result storage: Store the computing results in SQL and NoSQL databases.
- Visualization: Call the results by using APIs for demonstration.

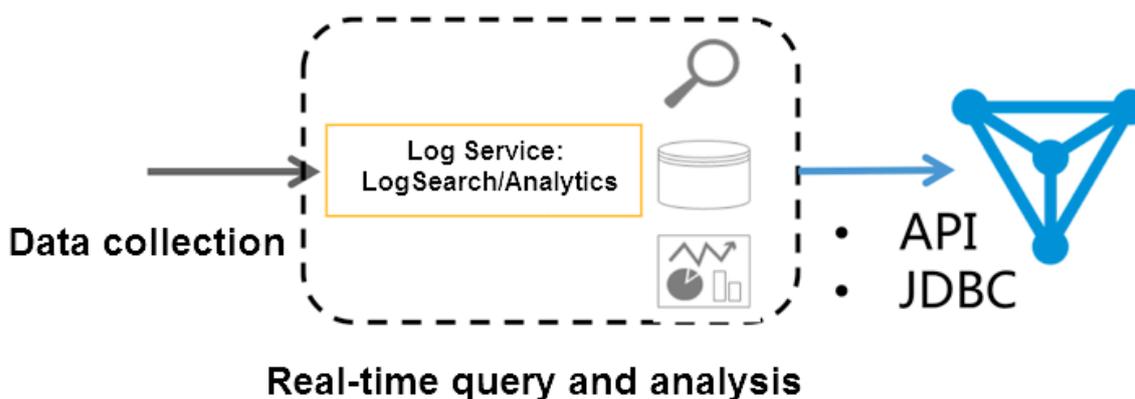
In Alibaba Group, many mature products can be used to complete such work. The following figure shows the products generally used.

Figure 12-35: Related products



Besides the preceding solution, you can also use the LogSearch/Analytics APIs of Log Service to directly interconnect with DataV to display data on a big screen.

Figure 12-36: Log Service + DataV



In September 2017, Log Service enhanced the real-time log analysis function (LogSearch/Analytics), which allows you to analyze logs in real time by using query and SQL92 syntax. Besides the built-in dashboard, Log Service supports the interconnection methods such as Grafana and Tableau (JDBC) to achieve result analysis visualization.

Features

Based on the data volume, timeliness, and business needs, computing is generally divided into two modes:

- Real-time computing (stream computing): Fixed computing + variable data.
- Offline computing (data warehouse + offline computing): Variable computing + fixed data.

Log Service provides two interconnection methods to collect data in real time. In addition, in log analysis scenarios that has timeliness needs, LogHub data can be indexed in real time. Then, you can use LogSearch/Analytics to directly query and analyze data. This method has the following advantages:

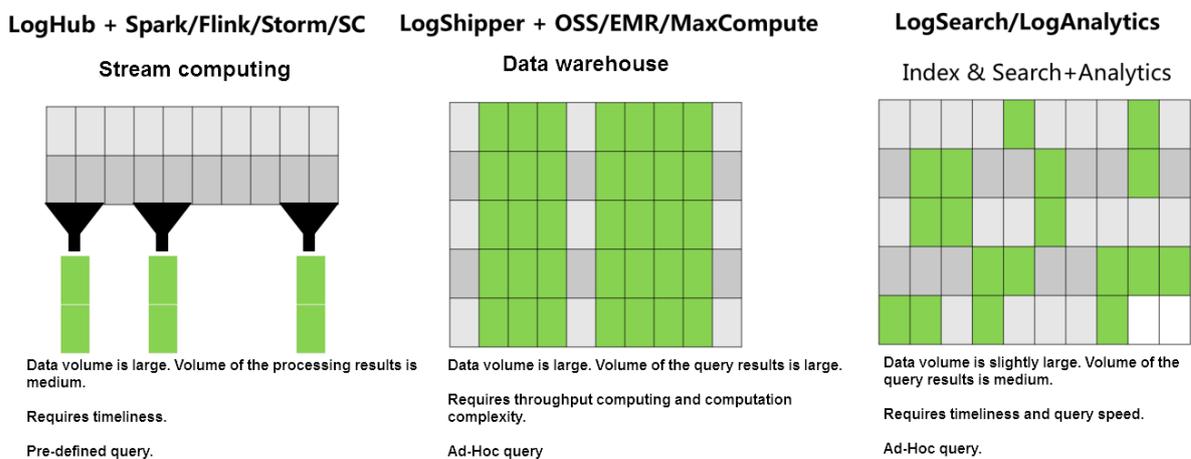
- Fast: You can obtain the results immediately after query is passed in by using APIs, without waiting or pre-computing the results.
- Real-time: In 99.9% cases, the generated logs are displayed on the big screen within 1s.

- **Dynamic:** Whether statistic method modification or supplementary data, the display results are refreshed in real time, without waiting for recomputation.

However, no computing system is omnipotent. This method has the following limits:

- **Data volume:** Up to 10 billion GB data can be computed at a time. You must set the time limit if the data volume is exceeded.
- **Computing flexibility:** Currently, only the SQL92 syntax is supported for computing . Custom UDF is unsupported.

Figure 12-37: Log service advantage



Configuration process

Operation Demonstration:

To interconnect Log Service data with DataV big screen, follow these steps:

1. Collect data. See [5-minute quick start](#) to access the data source to Log Service.
2. Set the index See [Index settings and visualization](#) or Use case for [website log analysis](#) in Best Practices.
3. Interconnect with the DataV plug-in to convert the real-time results queried by using the SQL statement to a view.

After completing steps 1 and 2, you can view the raw logs on the search page. This document mainly describes how to perform step 3.

Procedure

Step 1 Create a DataV data source

Click **Data Sources** in the left-side navigation pane. Click **Add Source**. The **New Data Source** dialog box appears. Enter the basic information of the data source. The following table describes the definition of each configuration item.

Figure 12-38: New data

Configuration item	Description
Type	Select Log Service.
Name	Configure a name for the data source.
AK ID	The AccessKey ID of the main account, or the AccessKey ID of the sub-account that has the permission to read Log Service.
AK Secret	The AccessKey Secret of the main account, or the AccessKey Secret of the sub-account that has the permission to read Log Service.

Configuration item	Description
Endpoint	The address of the region where the Log Service project resides. In the preceding figure, the address of region Hangzhou is entered.

Step 2 Create a line chart

1. Create a line chart.

In the data configuration of the line chart, set the data source type to Log Service, select the data source `log_service_api` created in the previous step, and enter the parameters in the Query text box.

Figure 12-39: Data source

Data Source Type

Log Service (SLS)

Select Source :

log_service_api Create

Query :

```
{
  "projectName": "dashboard-demo",
  "logStoreName": "access-log",
  "topic": "",
  "from": ":from",
  "to": ":to"
}
```

Data filter: Add filter

Auto Data Request: Every 1 Second

View Data Response

An example of the query parameters is as follows and the following table describes the parameters.

```
{
  "projectName": "dashboard - demo ",
  "logStoreName": " access - log ",
  "topic": "",
  "from": ": from ",
  "to": ": to ",
  "query": "*| select approx_distinct ( remote_addr ) as
uv , count ( 1 ) as pv , date_format ( from_unixtime
( date_trunc ( ' hour ', __time__ ) ), '% Y /% m /% d % H :% i :% s'"
}
```

```
s ') as time group by time order by time limit
1000 " ,
" line ": 100 ,
" offset ": 0
}
```

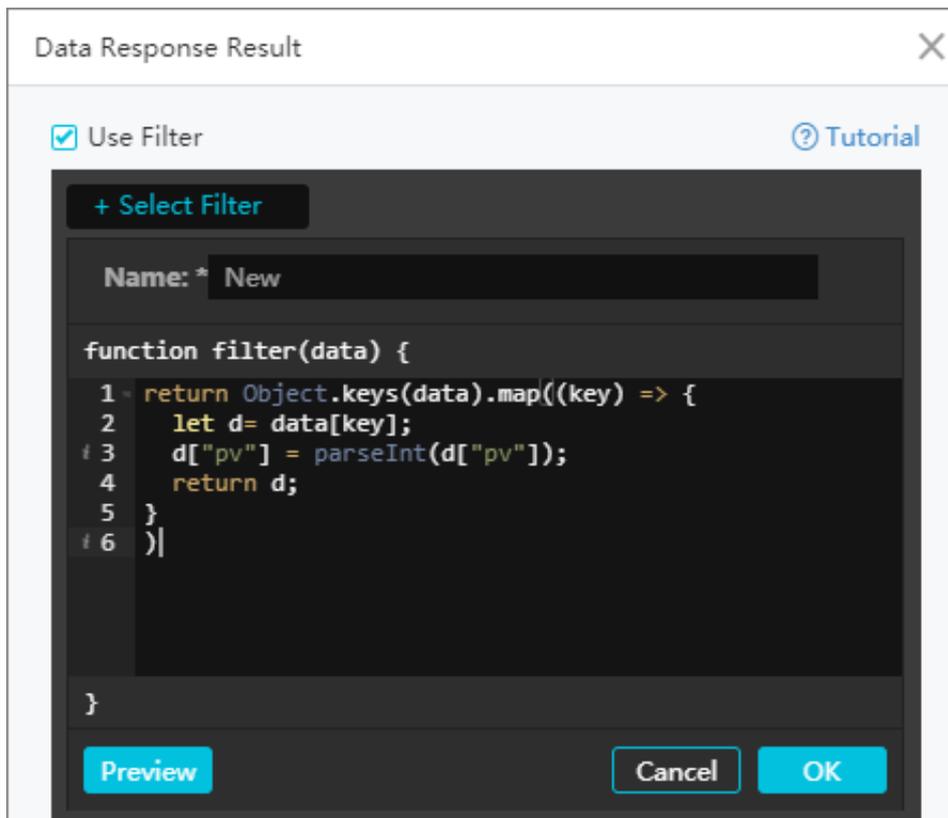
Configuration item	Description
projectName	The name of your project.
logstoreName	The name of your Logstore.
topic	Your log topic. If you have not set the topic, leave the parameter value empty.
from, to	<p>from and to specify the start time and end time of the log respectively.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> Note: In the preceding example, the parameter values are respectively set to : : from and : to . During the test, you can enter the time in UNIX format, for example, 1509897600. After the release, convert the time to : from and : to , and set the specific time ranges of the values in the URL parameter. For example, the previewed URL is http :// datav . aliyun . com / screen / 86312 . After http :// datav . aliyun . com / screen / 86312 ? from = 1510796077 & to = 1510798877 is opened, the values are computed based on the specified time.</p> </div>

Configuration item	Description
<p>query</p>	<p>Your query condition. In the preceding example, the query condition is the pv quantity per minute. For more information about the query syntax, see #unique_13.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> Note: The time in the query must be in the format like 2017/07/11 12:00:00. Therefore, use <code>date_format(from_unixtime(date_trunc('hour', __time__)), '%Y/%m/%d %H:%i:%s')</code> to align the time on the hour, and then convert it to the target format.</p> <pre style="background-color: #f0f0f0; padding: 5px;">date_format (from_unixtime (date_trunc (' hour ', __time__)) , '% Y /% m /% d % H :% i :% s ')</pre> </div>
<p>line</p>	<p>Enter the default value 100.</p>

Configuration item	Description
offset	Enter the default value 0.

After the configurations, click View Data Response.

Figure 12-40: View Data Response



2. Create a filter.

The Data Response Result dialog box appears after you click View Data Response. Select the Use Filter check box and click Select Filter > New Filter to create a filter.

Enter the filter content in the following format:

```
return Object . keys ( data ) . map ( ( key ) => {
let d = data [ key ];
d [ " pv " ] = parseInt ( d [ " pv " ] );
return d ;
}
)
```

In the filter, convert the result used by y-axis to the int type. In the preceding example, the y-axis indicates the pv. Therefore, the pv column must be converted.

The results contain both the t and pv columns. You can set the x-axis to t and the y-axis to pv.

Step 3 Configure a pie chart

1. Create a carousel pie chart.

Figure 12-41: Query text box

Enter the following contents in the Query text box:

```
{
  "projectName": "dashboard - demo ",
  "logStoreName": "access - log ",
  "topic": "",
  "from": 1509897600 ,
  "to": 1509984000 ,
  "query": "*| select count ( 1 ) as pv , method group
by method ",
  "line": 100 ,
  "offset": 0
}
```

During the query, the ratios of different methods can be computed.

2. Add a filter and enter the following contents in the filter:

```
return Object . keys ( data ) . map ( ( key ) => {
  let d = data [ key ];
  d [ " pv " ] = parseInt ( d [ " pv " ] );
  return d ;
}
)
```

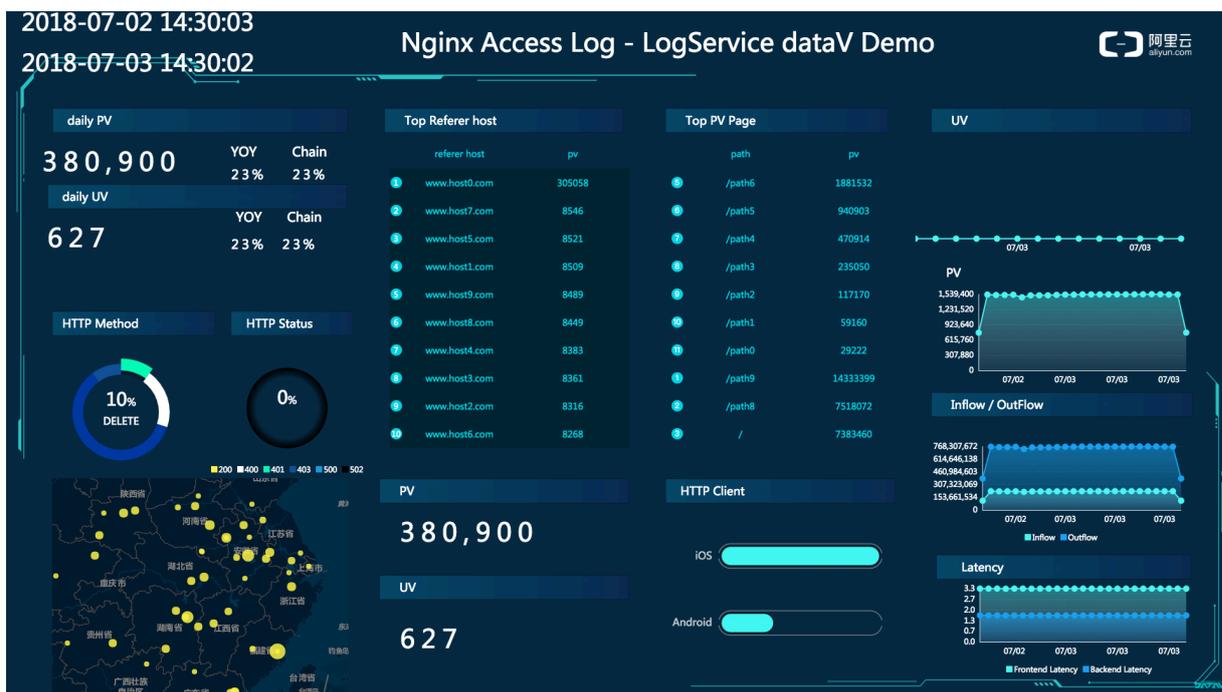
Enter method in the type text box and pv in the value text box for the pie chart.

Step 4 Preview and release

Click Preview and Publish to create a big screen. Developers and business personnel can view their business access conditions in real time in the Double 11 shopping campaign.

Trial: [Demo](#). You can set the values of the parameters from and to in the URL to any time.

Figure 12-42: Real-Time Screen



Use case: Continuously adjust the real-time big screen under the statistic criteria

For example, a temporary requirement is raised during the Computing Conference, which is to count the online (website) traffic across China. Total log data collection is configured and LogSearch/Analytics is enabled in Log Service. Therefore, you only need to enter your query condition.

1. For example, to count the UV, obtain the unique count of the forward field under Nginx in all access logs from October 11 to the present.

```
* | select approx_dis tinct ( forward ) as uv
```

2. After the system runs online for one day, the requirement is changed. Currently, only data under the domain yunqi needs to be counted. You can add a filter condition (host) for real-time query.

```
host : yunqi . aliyun . com | select approx_dis tinct ( forward ) as uv
```

3. It is detected that the Nginx access logs contain multiple IP addresses. By default, only the first IP address is required. Therefore, process the query condition in the query.

```
host : yunqi . aliyun . com | select approx_dis tinct ( split_part ( forward , ',', 1 ) ) as uv
```

4. According to the requirement in the third day, the advertisement access in uc must be removed from access computing. In this case, you can add a filter condition not ... to obtain the latest result immediately.

```
host : yunqi . aliyun . com not url : uc - iflow | select approx_dis tinct ( split_part ( forward , ',', 1 ) ) as uv
```

12.3.6 Interconnection with Grafana

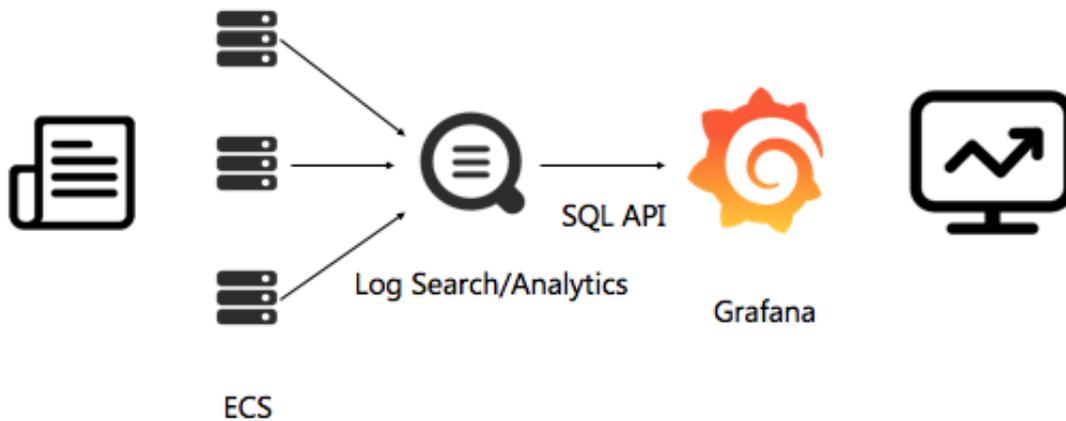
Alibaba Cloud Log Service is an all-in-one service for log-type data. Users can leave trivial jobs like data collection, storage computing interconnection, and data index and query to Log Service to focus on the analysis. In September 2017, Log Service upgraded the LogSearch/Analytics, allowing real-time log analysis using query + SQL92 syntax.

Besides the built-in Dashboard, interconnections with DataV, Grafana, Tableau, and Quick are also available to visualize the results analysis. This article demonstrates how to analyze and visualize Nginx logs using Log Service by giving an example of Grafana.

Process structure

Process from log collection to analysis is structured as follows.

Figure 12-43: Process structure



Procedure

1. Log data collection. For detailed procedures, see [#unique_224](#).
2. Index settings and console query configuration. For detailed procedures, see [#unique_15](#) and [#unique_225](#) or Website Log Analysis Case.
3. Install the Grafana plug-in to convert real-time query SQL into views.

After completing Step 1 and 2, you can view the raw log on the query page.

This document demonstrates Step 3.

Procedure

1. [Install Grafana](#)
2. [Install the Log Service plug-in](#)
3. [Configure the log data source](#)

4. **Add Dashboard**
 - a. **Configure the template variables**
 - b. **Configure PV and UV**
 - c. **Configure inbound and outbound bandwidth**
 - d. **Percentage of HTTP methods**
 - e. **Percentage of HTTP status codes**
 - f. **Page of top sources**
 - g. **Pages of the maximum latency**
 - h. **Top pages**
 - i. **Top pages of non-200 requests**
 - j. **Average frontend and backend latency**
 - k. **Client statistics**
 - l. **Save and release the dashboard**
5. **View the results**

1 Install Grafana

For detailed installation steps, see [Grafana official document](#).

To Ubuntu, for example, the installation command is:

```
wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_4.5.2_amd64.deb
sudo apt-get install -y adduser libfontconfig
sudo dpkg -i grafana_4.5.2_amd64.deb
```

To use the pie chart, you must install the pie chart plug-in. For detailed procedures, see [Grafana official document](#).

The installation command is as follows:

```
grafana -cli plugins install grafana -piechart -panel
```

2 install the Log service plug-in

Install the Log Service plug-in Confirm the directory location of Grafana plug-in. The location of Ubuntu plug-in is `/var/lib/grafana/plugins/`. Restart the grafana-server after installing the plug-in.

Taking Ubuntu system as an example, run the following command to install the plug-in and restart the grafana-server.

```
cd /var/lib/grafana/plugins/
```

```
git clone https://github.com/aliyun/aliyun-log-grafana-datasource-plugin
service grafana-server restart
```

3 Configure the log data source

For the deployment in a local machine, the default installation location is Port 3000. Open the Port 3000 in a browser.

1. Click on Grafana's logo in the upper left corner and select Data Sources in the dialog box.
2. Click Add data source, and use Grafana and Alibaba Cloud Log Service for log visual analysis.
3. Complete the configuration items for the new data source.

Each part is configured as follows:

Configuration item	Configuration content
datasource	The name can be customized and the type is LogService.
Http Setting	URL input example: <code>http://dashboard-demo.cn-hangzhou.log.aliyuncs.com</code> The <code>dashboard-demo</code> (project name) and <code>cn-hangzhou.log.aliyuncs.com</code> (endpoint of the project region) must be replaced with your project and region addresses when configuring your data source. You can select Direct or Proxy for Access.
Http Auth	Use the default configuration.

Configuration item	Configuration content
log service details	For detailed configuration of Log Service, enter Project, Logstore, and AccessKey that has the read permission respectively. AccessKey can belong to the primary account or the subaccount.

Configuration example

Figure 12-44: Configuration example

After the configuration is complete, click Add to add DataSource. Next, add Dashboard.

4 Add Dashboard

Click to open the menu in the upper left corner, select Dashboards and click New. Add a new Dashboard to the menu in the upper left corner.

4.1 Configure the template variables

You can configure the template variables in Grafana to show different views in the same view by choosing different variable values. This document describes the configuration of each time interval and the access of different domain names.

1. Click the Settings icon at the top of the page, and then click Templating.

2. On the current page, the configured template variables are displayed. Click New to create a new template. First, configure a time interval.

The name of the variable is the one you used in the configuration, which is called \$myinterval here. You must write \$ myinterval in the query condition. Please refer to the following table for configuration.

Configuration items	Configuration content
Name	The variable name, which you can name myinterval.
Type	Please select Interval
Lable.	Please enter time interval
Internal Options	Please enter 1m , 10m , 30m , 1h , 6h , 12h , 1d , 7d , 14d , 30d in Value entry.

3. Configure a domain name template.

Generally, multiple domain names can be mounted on one vps. You must view the accesses of different domain names. For the template value, enter *, www . host . com , www . host0 . com , www . host1 . com to view all the domain names, or view the accesses of www . host . com , www . host0 . com or www . host1 . com respectively.

The domain name template configuration as below.

Configuration items	Configuration content
Name	Variable name, you can name it hostname.
Type	Please select Custom
Lable	Please enter a domain name
Custom Options	Please enter *, www . host . com , www . host0 . com , www . host1 . com forValue entry.

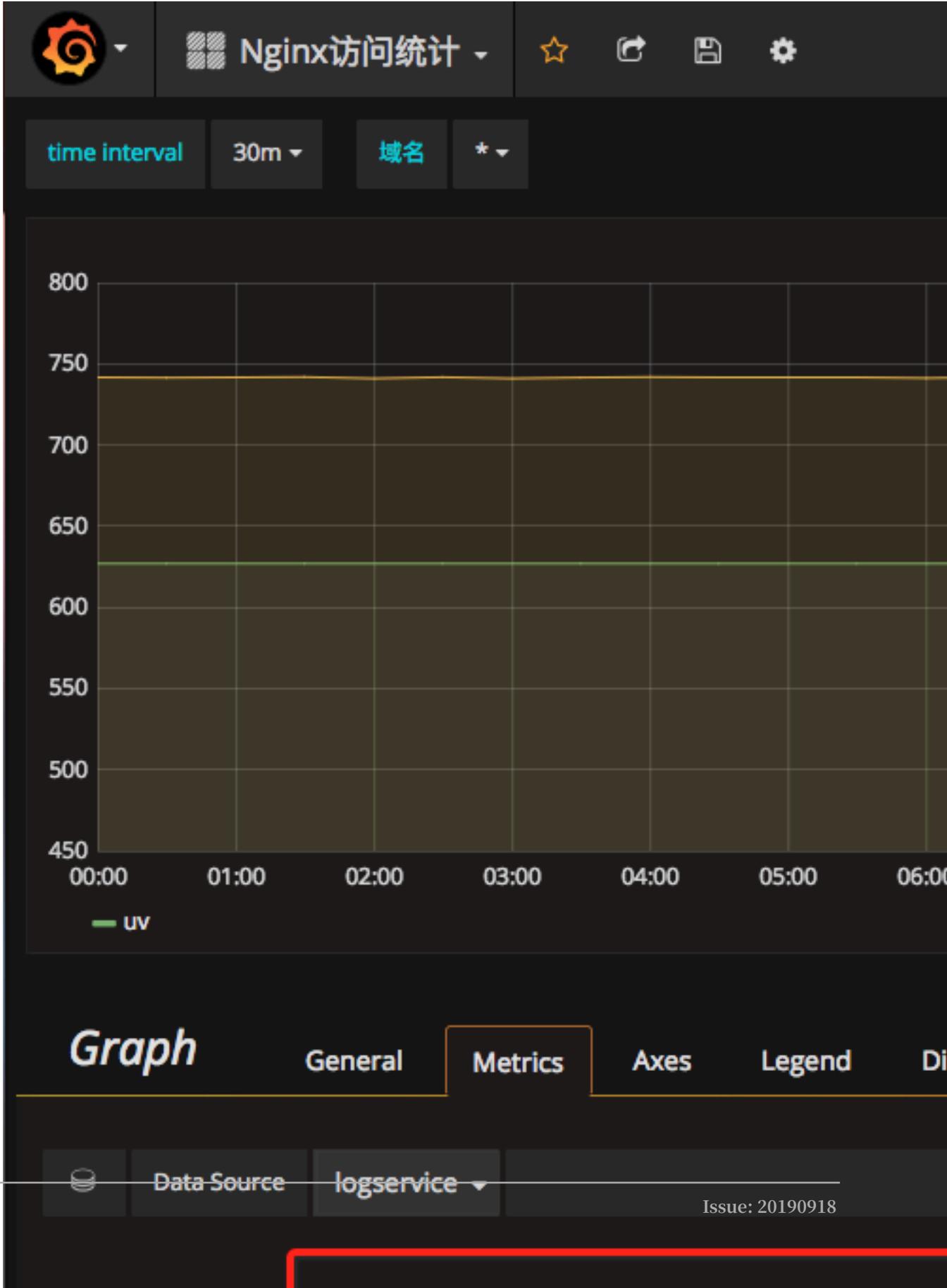
After the configuration is complete, the template variables configured appears on the top of the Dashboard page. You can select any value from the drop-down box. For example, the following values can be selected for time interval.

4.2 Configure PV and UV

1. Click **ADD ROW** on the left to create a new Row. If a Row already exists, you can select **Add Panel** in the left dialog box.
2. Grafana supports multiple types of views. For PV and UV data, create a Graphview.
3. Click **Pannel Title** and click **Edit** in the dialog box.

- 4. In Metrics configuration, select `logservice` for datasource and enter Query, Y axis, and X axis:

Figure 12-45: Configure PV and UV

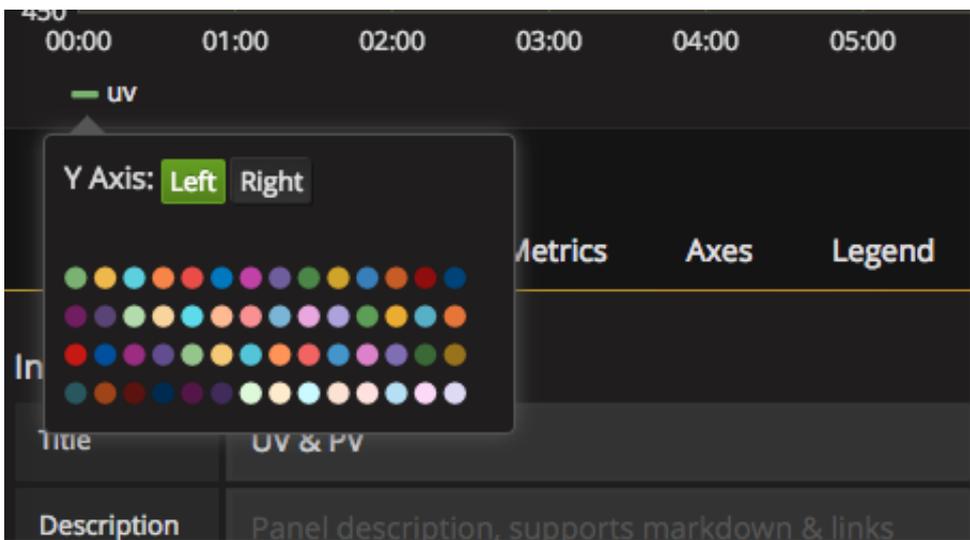


5. In the dataSource drop-down box, select the configured: `logservice` .

Configuration items	Configuration content
Query	<pre>\$ hostname select approx_distinct (remote_address) as uv , count (1) as pv , __time__ - time group by time order by time limit 1 , 000</pre> <p>The \$hostname in the preceding Query is replaced with the domain name selected by the user for actual display. The \$ \$myinterval is replaced with a time interval. Note that the number of \$ before myinterval is two and for hostname is one.</p>
X-Column	time
Y-Column	uv,pv

UV and PV values are displayed with two Y axes due to their large difference. Click the colored line on the left of UV below the icon to decide whether the UV is displayed on the left or the right Y axis:

Figure 12-46: Y-axis display



The default view for the title is Panel Title. To modify, open the General tab and enter a new title in the Title configuration item, such as `PV & UV` .

4.3 Configure inbound and outbound bandwidth

You can add inbound and outbound bandwidth traffic in the same way with [4.2 Configure PV and UV](#).

The main configuration items are as follows:

Configuration items	Configuration content
Query	<pre>\$ hostname select sum (body_byte_ sent) as net_out , sum (request_ le ngth) as net_in , __time__ - __time__ % \$\$ myinterval as time group by __time__ - __time__ % \$\$ myinterval limit 10000</pre>
X-Column	Time
Y-Column	net_in,net_out

4.4 Percentage of HTTP methods

You can configure the percentage of HTTP methods in the same way with [4.2 Configure PV and UV](#).

Create a new Row, select Pie Chart , and enter Query, X axis, and Y axis in the configuration.

The main configuration items are as follows:

Configuration items	Configuration content
Query	<pre>\$ hostname select count (1) as pv , method group by method</pre>
X-Column	pie
Y-Column	method,pv

4.5 Percentage of HTTP status codes

You can configure the percentage of HTTP status codes in the same way with [4.2 Configure PV and UV](#).

Create a new Row, and select Pie Chart for the view.

The main configuration items are as follows:

Configuration items	Configuration content
Query	<pre>\$ hostname select count (1) as pv , status group by status</pre>
X-Column	pie
Y-Column	status,pv

4.6 Page of top sources

You can configure the Page of top sources in the same way with [4.2 Configure PV and UV](#).

Create a new Row, and select Pie Chart for the view:

The main configuration items are as follows:

Configuration items	Configuration content
Query	<pre>\$ hostname select count (1) as pv , referer group by referer order by pv desc</pre>
X-Column	pie
Y-Column	referer,pv

4.7 Pages of the maximum latency

You can configure the pages of the maximum latency in the same way with [4.2 Configure PV and UV](#).

To show URL and its latency in a table, you must specify the view as Table at the time of creation.

The main configuration items are as follows:

Configuration items	Configuration content
Query	<pre>\$ hostname select url as top_latency_url , request_time order by request_time desc limit 10</pre>
X-Column	X-Column is left blank
Y-Column	top_latency_url,request_time

4.8 Top pages

You can add top pages in the same way with [4.2 Configure PV and UV](#).

Create a new Table view. Data Source select logservice, query content, X-axis, and y-axis please refer to the following table.

Configuration items	Configuration content
Query	<pre>\$ hostname select count (1) as pv , split_part (url , '?' , 1) as path group by split_part (url , '?' , 1) order by pv desc limit 20</pre>
X-Column	X-Column is left blank
Y-Column	path,pv

4.9 Top pages of non-200 requests

You can add top pages of non-200 requests in the same way with [4.2 Configure PV and UV](#).

Create a new Table view. Data Source select logservice, query content, X-axis, and y-axis please refer to the following table.

Configuration items	Configuration content
Query	<pre>\$ hostname not status : 200 select count (1) as pv , url group by url order by pv desc</pre>
X-Column	X-Column is left blank
Y-Column	url,pv

4.10 Average frontend and backend latency

You can add average frontend in the same way with [4.2 Configure PV and UV](#).

Create a new Graph view: Data Source select logservice, query content, X-axis, and y-axis please refer to the following table.

Configuration items	Configuration content
Query	<pre>\$ hostname select avg (request_t i me) as response_t ime , avg (upstream_r esponse_t i me) as upstream_r esponse_t i me , __time__ - __time__ % \$\$ myinterval as time group by __time__ - __time__ % \$\$ myinterval limit 10000</pre>
X-Column	time
Y-Column	upstream_response_time,response_time

4.11 Client statistics

You can add client statistics in the same way with [4.2 Configure PV and UV](#).

Create a new Pie Chart. Data Source select logservice, query content, X-axis, and y-axis please refer to the following table.

Configuration items	Configuration content
Query	<pre>\$ hostname select count (1) as pv , case when regexp_like (http_user_ agent , ' okhttp ') then ' okhttp ' when regexp_like (http_user_ agent , ' iPhone ') then ' iPhone ' when regexp_like (http_user_ agent , ' Android ') then ' Android ' else ' unKnown ' end as http_user_ agent group by http_user_ agent order by pv desc limit 10</pre>
X-Column	pie
Y-Column	http_user_agent,pv

4.12 Save and release the dashboard

Click the Save button at the top of the page to release the Dashboard.

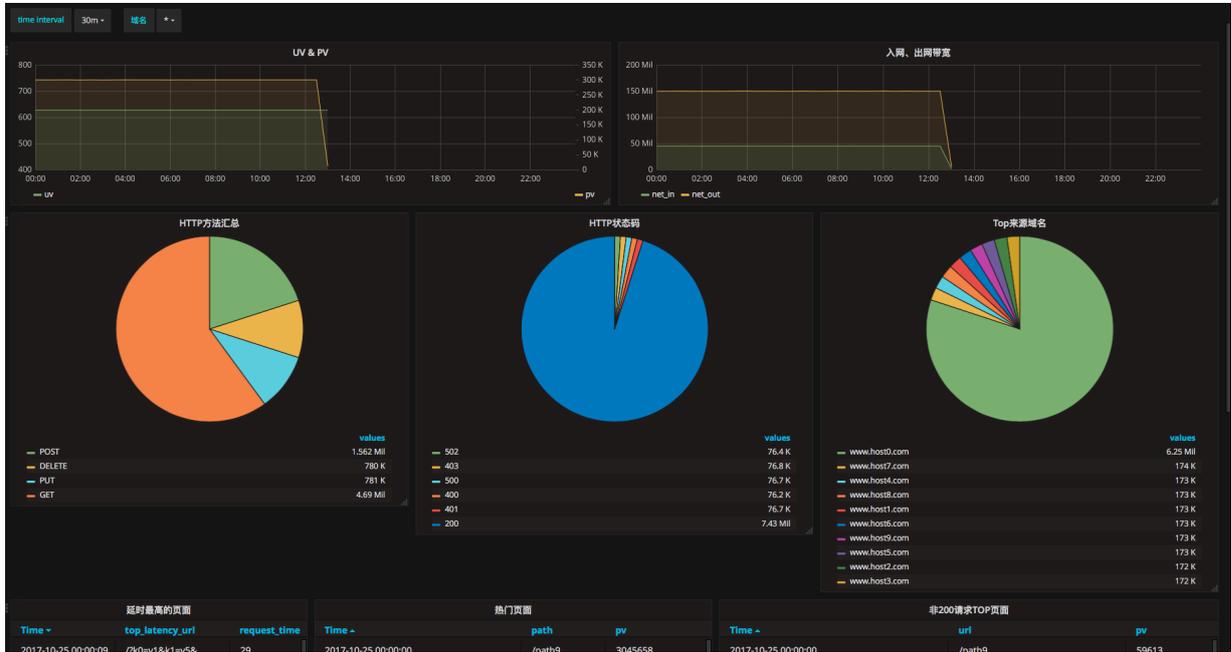
5 View the results

Open the home page of Dashboard to view the results. Demo address: [Demo](#).

At the top of the page, you can choose the time range or time granularity of statistics, or you can choose a different domain name.

Now, the configuration of Dashboard for Nginx access statistics is completed, enabling you to mine valuable information from your views.

Figure 12-47: Check the result.



13 FAQ

13.1 What can cause an inaccurate query result to return?

When you query or analyze logs, a message indicating **The results are inaccurate.** may be displayed. This is displayed when only partial logs are scanned for query and analysis results, meaning the results do not include scans of full-log queries or analysis, and are therefore considered inaccurate.

Possible causes include:

1. The time range for queries is excessive.

Cause: The time range for queries is excessively wide, for example, three months or a year. In this case, Log Service cannot scan all logs generated within this time period.

Solution: Narrow down the time range for queries and perform multiple queries.

2. The query condition is exceedingly complex.

Cause: The query condition is exceedingly complex, or Log Service cannot read query results because the query condition contains multiple frequently used words.

Solution: Narrow down the query scope and perform multiple queries.

3. The SQL database reads an abnormally large amount of data.

Cause: The SQL database reads an abnormally large amount of data, which leads to inaccurate query results. For example, if the SQL database reads strings from multiple columns, it can read only 1 GB of data from each Shard. If this threshold is exceeded, inaccurate query results will be returned.

Solution: Narrow down the query scope and perform multiple queries.