ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

Log Service

User Guide

Product Version: v3.16.2

Document Version: 20220915

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.What is Log Service? 14
2.Quick start 15
2.1. Procedure 15
2.2. Log on to the Log Service console 16
2.3. Obtain an AccessKey pair 17
2.4. Manage a project 18
2.5. Manage a Metricstore 19
2.6. Manage Logstores 21
2.7. Manage shards 23
2.8. Terms 25
2.8.1. Terms 25
2.8.2. Log 27
2.8.3. Log group 29
2.8.4. Project 29
2.8.5. Logstore 29
2.8.6. Metricstore 29
2.8.7. Metric 30
2.8.8. Shard 30
2.8.9. Topic 32
3.Data collection 34
3.1. Collection by Logtail 34
3.1.1. Overview 34
3.1.1.1. Logtail overview 34
3.1.1.2. Log collection process of Logtail 37
3.1.1.3. Logtail configuration files and record files 39
3.1.2. Installation 46

3.1.2.1. Install Logtail on a Linux server 46
3.1.2.2. Install Logtail in Windows 48
3.1.2.3. Set Logtail startup parameters 50
3.1.3. Logtail machine group 53
3.1.3.1. Overview 53
3.1.3.2. Configure a user identifier 54
3.1.3.3. Create an IP address-based machine group 55
3.1.3.4. Create a custom ID-based machine group 56
3.1.3.5. View server groups 59
3.1.3.6. Modify a server group 59
3.1.3.7. View the status of a server group 60
3.1.3.8. Delete a machine group 60
3.1.3.9. Manage a Logtail configuration 60
3.1.4. Collect text logs 61
3.1.4.1. Configure text log collection 61
3.1.4.2. Collect logs in simple mode 66
3.1.4.3. Collect logs in full regex mode 70
3.1.4.4. Collect logs in delimiter mode 75
3.1.4.5. Collect logs in JSON mode 80
3.1.4.6. Collect logs in NGINX mode 84
3.1.4.7. Collect logs in IIS mode 89
3.1.4.8. Collect logs in Apache mode 95
3.1.4.9. Configure parsing scripts 101
3.1.4.10. Time formats 103
3.1.4.11. Import historical log files 106
3.1.4.12. Log topics 108
3.1.5. Collect container logs 109
3.1.5.1. Overview 109

3.1.5.2. Install the Logtail component 110
3.1.5.3. Use the Log Service console to collect container tex... 115
3.1.5.4. Use the Log Service console to collect container std... 126
3.1.5.5. Use CRDs to collect container logs in DaemonSet m... 143
3.1.5.6. Use CRDs to collect container text logs in Sidecar m.. 155
3.1.5.7. Use the Log Service console to collect container tex... 169
3.1.5.8. Collect logs from standard Docker containers 174
3.1.5.9. Collect Kubernetes events 177
3.1.5.10. Collect container text logs 180
3.1.5.11. Collect container stdout and stderr logs 186
3.1.5.12. Collect standard Docker logs 198
3.1.6. Custom plug-ins 202
3.1.6.1. Collect MySQL binary logs 203
3.1.6.2. Collect MySQL query results 213
3.1.6.3. Collect syslogs 218
3.1.6.4. Customize Logtail plug-ins to process data 223
3.1.7. Limits 246
3.2. Other collection methods 249
3.2.1. Use the web tracking feature to collect logs 249
3.2.2. Use SDKs to collect logs 252
3.2.2.1. Producer Library 252
3.2.2.2. Log4j Appender 252
3.2.2.3. Logback Appender 252
3.2.2.4. Golang Producer Library 253
3.2.2.5. Python logging 253
3.2.3. Collect common logs 256
3.2.3.1. Collect Log4j logs 256
3.2.3.2. Collect Python logs 258

3.2.3.3. Collect Node.js logs 262
3.2.3.4. Collect WordPress logs 264
3.2.3.5. Collect Unity3D logs 264
4.Query and analysis 267
4.1. Log search overview 267
4.2. Log analysis overview 268
4.3. Configure indexes 270
4.4. Query and analyze logs 273
4.5. Download logs 275
4.6. Index data type 275
4.6.1. Overview 275
4.6.2. Text type 276
4.6.3. Numeric type 277
4.6.4. JSON type 278
4.7. Query syntax and functions 281
4.7.1. Search syntax 281
4.7.2. LiveTail 288
4.7.3. LogReduce 289
4.7.4. Contextual query 292
4.7.5. Saved search 294
4.7.6. Quick analysis 296
4.8. Analysis grammar 298
4.8.1. General aggregate functions 298
4.8.2. Security check functions 300
4.8.3. Map functions and operators 303
4.8.4. Approximate functions 312
4.8.5. Mathematical statistics functions 313
4.8.6. Mathematical calculation functions 314

4.8.7. String functions 316
4.8.8. Date and time functions 320
4.8.9. URL functions 326
4.8.10. Regular expression functions 327
4.8.11. JSON functions 328
4.8.12. Type conversion functions 331
4.8.13. IP functions 332
4.8.14. GROUP BY clause 333
4.8.15. Window functions 335
4.8.16. HAVING clause 337
4.8.17. ORDER BY clause 337
4.8.18. LIMIT syntax 338
4.8.19. Conditional expressions 338
4.8.20. Nested subquery 341
4.8.21. Array functions and operators 342
4.8.22. Binary string functions 360
4.8.23. Bitwise functions 361
4.8.24. Interval-valued comparison and periodicity-valued com.. 361
4.8.25. Comparison functions and operators 365
4.8.26. Lambda expressions 367
4.8.27. Logical functions 369
4.8.28. Column aliases 370
4.8.29. JOIN queries on a Logstore and a MySQL database 371
4.8.30. Geospatial functions 372
4.8.31. Geography functions 375
4.8.32. JOIN clause 376
4.8.33. UNNEST clause 377
4.9. Machine learning syntax and functions 379

4.9.1. Overview 379
4.9.2. Smooth functions 381
4.9.3. Multi-period estimation functions 385
4.9.4. Change point detection functions 387
4.9.5. Maximum value detection function 389
4.9.6. Prediction and anomaly detection functions 390
4.9.7. Time series decomposition function 396
4.9.8. Time series clustering functions 397
4.9.9. Frequent pattern statistics function 402
4.9.10. Differential pattern statistics function 403
4.9.11. Root cause analysis function 404
4.9.12. Correlation analysis functions 407
4.9.13. Kernel density estimation function 410
4.10. Advanced analysis 411
4.10.1. Optimize queries 411
4.10.2. Use cases 412
4.10.3. Examples of time field conversion 415
4.11. Associate Log Service with external data sources 416
4.11.1. Overview 416
4.11.2. Associate Log Service with a MySQL database 417
4.11.3. Associate Log Service with an OSS bucket 419
4.11.4. Associate Log Service with a hosted CSV file 421
4.12. Visual analysis 424
4.12.1. Charts 424
4.12.1.1. Chart overview 424
4.12.1.2. Display query results in a table 425
4.12.1.3. Display query results on a line chart 426
4.12.1.4. Display query results on a column chart 428

4.12.1.5. Display query results on a bar chart 429
4.12.1.6. Display query results on a pie chart 430
4.12.1.7. Display query results on an area chart 433
4.12.1.8. Display query results on a single value chart 434
4.12.1.9. Display query results on a progress bar 438
4.12.1.10. Display query results on a map 440
4.12.1.11. Display query results in a Sankey diagram 440
4.12.1.12. Display query results on a word cloud 442
4.12.1.13. Display query results on a treemap chart 443
4.12.2. Dashboard 444
4.12.2.1. Overview 444
4.12.2.2. Create and delete a dashboard 444
4.12.2.3. Manage a dashboard in display mode 445
4.12.2.4. Manage a dashboard in edit mode 447
4.12.2.5. Configure a drill-down event 449
4.12.2.6. Add a filter 456
4.12.2.7. Manage a Markdown chart 458
5.Alerts 462
5.1. Overview 462
5.2. Configure an alarm 463
5.2.1. Configure an alert rule 463
5.2.2. Authorize a RAM user to manage alert rules 466
5.2.3. Configure alert notification methods 466
5.3. Modify and view an alarm 470
5.3.1. Modify an alert rule 470
5.3.2. View alert statistics 471
5.3.3. Manage alerts 472
5.4. Relevant syntax and fields for reference 473

5.4.1. Syntax of conditional expressions in alert rules 473
5.4.2. Fields in alert logs 477
5.5. FAQ 479
5.5.1. A DingTalk alert notification fails to be sent and the e... 479
6.Real-time consumption 481
6.1. Overview 481
6.2. Consume log data 482
6.3. Consumption by consumer groups 483
6.3.1. Use consumer groups to consume log data 483
6.3.2. View the status of a consumer group 489
6.4. Use Storm to consume log data 490
6.5. Use Flume to consume log data 494
6.6. Use open source Flink to consume log data 497
6.7. Use Logstash to consume log data 503
6.8. Use Spark Streaming to consume log data 505
6.9. Use Realtime Compute to consume log data 510
7.Data shipping 513
7.1. Ship logs to OSS 513
7.1.1. Overview 513
7.1.2. Ship log data from Log Service to OSS 513
7.1.3. Obtain the ARN of a RAM role 517
7.1.4. Storage Formats 518
7.1.5. Decompress Snappy compressed files 520
8.Time series storage 523
8.1. Data import 523
8.1.1. Collect metric data from hosts 523
8.1.2. Import metrics collected by Telegraf 528
8.1.2.1. Telegraf overview 528

8.1.2.2. Collect metric data from MySQL servers 528
8.1.2.3. Collect metric data from Java applications or Tomca... 530
8.1.2.4. Collect metric data from NGINX servers 533
8.1.3. Collect metric data from Prometheus 535
8.1.3.1. Collect metric data from Prometheus by using the R... 536
8.1.3.2. Collect metric data from Prometheus by using a Lo... 538
8.2. Query and analysis 540
8.2.1. Overview of query and analysis of time series data 540
8.2.2. Query and analyze time series data 543
8.3. Visualization 545
8.3.1. Configure a time series chart 545
8.3.2. Send time series data from Log Service to Grafana 545
9.RAM 548
9.1. Overview 548
9.2. Create a RAM role 548
9.3. Create a user 548
9.4. Create a user group 549
9.5. Add a user to a user group 550
9.6. Create a policy 550
9.7. Grant a RAM user the permissions to manage a project 551
9.8. Grant permissions to a RAM role 552
9.9. Use custom policies to grant permissions to a RAM user 552
10.Monitor Log Service 558
10.1. Overview 558
10.2. Manage service logs 559
10.3. Log types 561
10.4. Service log dashboards 571
11.FAQ 572

11.1. Log collection 572
11.1.1. How do I troubleshoot errors that occur when I use L... 572
11.1.2. What can I do if Log Service does not receive heartbe... 572
11.1.3. How do I query the status of local log collection? 575
11.1.4. How do I debug a regular expression? 587
11.1.5. How do I optimize regular expressions? 589
11.1.6. How do I use the full regex mode to collect log entrie... 590
11.1.7. How do I specify time formats for logs? 590
11.1.8. How do I configure non-printable characters in a sam... 591
11.1.9. How do I troubleshoot errors that occur when I collec... 592
11.1.10. How do I obtain the labels and environment variable... 595
11.2. Log search and analysis 597
11.2.1. FAQ about log query 597
11.2.2. What can I do if I cannot obtain the required results... 598
11.2.3. What are the differences between log consumption an... 599
11.2.4. How do I resolve common errors that occur when I q... 600
11.2.5. Why data queries are inaccurate? 602
11.3. Alarm 603
11.3.1. FAQ about alerts 603
11.4. What do I do if the Forbidden.SLS::ListProject error occu... 604

1.What is Log Service?

Log Service is a cloud-native monitoring and analysis platform that provides large-scale, low-cost, and real-time services to process log data. Log Service allows you to collect, query, analyze, visualize, consume, and ship log data. Log Service also allows you to configure alerts. This helps improve the digital capabilities of your business in scenarios such as R&D, O&M, operations, and security.

Log Service is tested and verified by Alibaba Group in various big data scenarios. You can collect, consume, query, and analyze data without the need to develop separate features.

Log Service provides the following features:

- Log collection: Log Service allows you to collect various types of logs in real time, such as events, binary logs, and text logs by using multiple methods. For example, you can use Logtail and JavaScript to collect logs.
- Query and analysis: Log Service allows you to query and analyze the collected logs in real time and view analysis results on charts and dashboards.
- Alerting: Log Service can run query statements at regular intervals after an alert task is created. If the results match the specified conditions, Log Service sends an alert to the specified contacts in real time. You can specify the conditions and contacts when you create the alert task.
- Real-time consumption: Log Service provides real-time consumption interfaces that log consumers can use to consume the collected logs.
- Shipping: Log Service allows you to ship the collected logs to Object Storage Service (OSS) in real time.

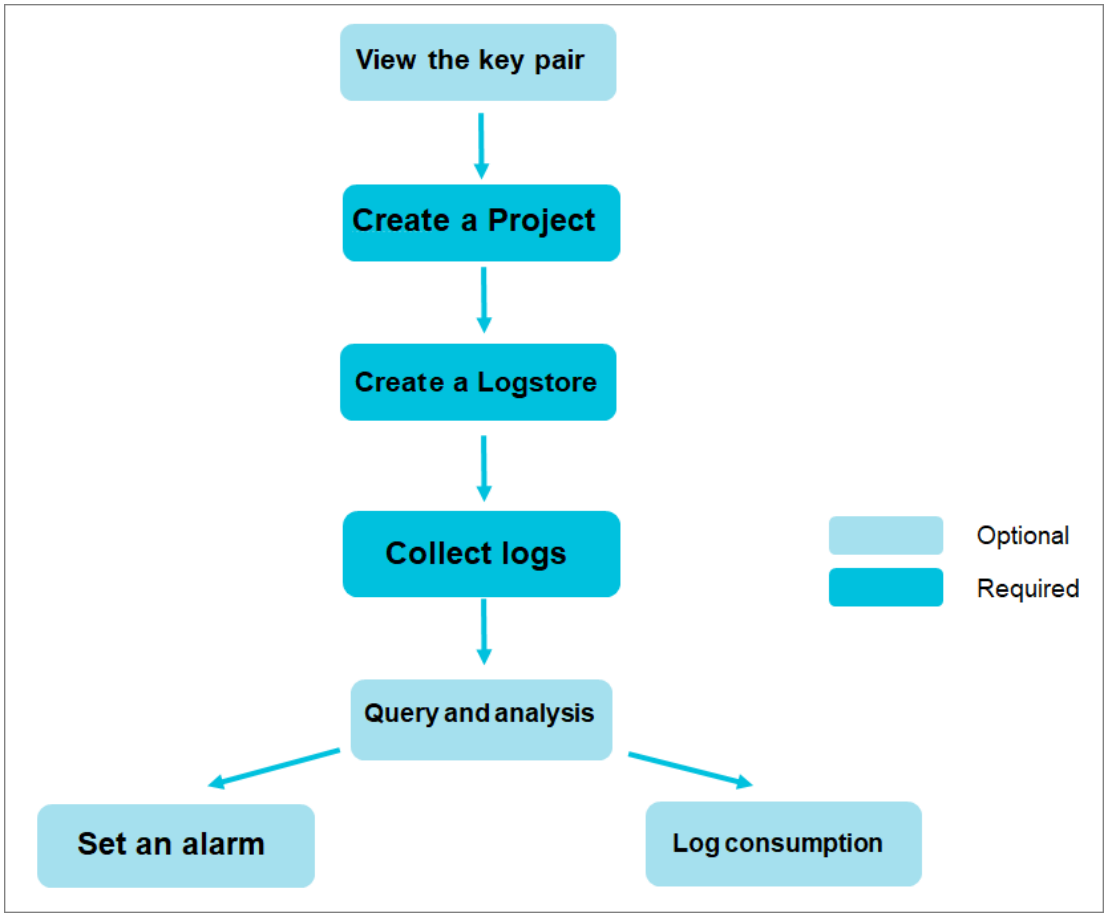
2.Quick start

2.1. Procedure

This topic describes the basic procedure to use Log Service. You can follow this procedure to create projects, create Logstores, and collect log data.

The following figure shows the procedure.

Procedure



1. Optional. Obtain an AccessKey pair.

   An AccessKey pair is a secure identity credential that you can use to call API operations and access your Alibaba Cloud resources. You can use the AccessKey pair to sign API requests and pass security authentication.

2. Create a project.

   Create a project in a specified region. For more information, see Create a project.

3. Create a Logstore.

   Create a Logstore for the project and specify the number of shards. For more information, see Create a Logstore.

4. Collect text logs.

   Select a method to collect log data based on your business requirements. For more information, see Collect text logs.

5. Configure indexes, and query and analyze log data.

   - Before you can query and analyze log data in Log Service, you must enable the indexing feature and

configure indexes. For more information, see Enable the index feature and configure indexes for a Logstore.

- After you enable the indexing feature and configure indexes, you can query and analyze log data in real time. Log Service allows you to query and analyze large amounts of log data in real time. For more information, see Query and analysis.
- After you query and analyze log data, you can configure charts and dashboards to display query and analysis results. For more information, see Overview and Dashboard overview.

6. Configure alert rules.

   Log Service allows you to configure alert rules based on query and analysis results. Log Service sends alert notifications based on the notification methods that you specify. For more information, see Configure an alert rule.

7. Consume logs.

   Log Service allows you to consume logs by using multiple methods, such as a Spark Streaming client, Storm spout, and Flink connector. For more information, see Real-time consumption.

2.2. Log on to the Log Service console

This topic describes how to log on to the Log Service console.

Prerequisites

- Before you log on to the Apsara Uni-manager Management Console, the endpoint of the console is obtained from the deployment staff.
- We recommend that you use Google Chrome.

Procedure

1. Enter the URL of the Apsara Uni-manager Management Console in the address bar and press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you use to log on to the Apsara Uni-manager Management Console from the operations administrator.

   > **Note** The first time that you log on to the Apsara Uni-manager Management Console, you must change the password of your account. For security purposes, your password must meet the minimum complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, including exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

3. Click **Log On**.
4. If your account has multi-factor authentication (MFA) enabled, perform corresponding operations in the following scenarios:
   - You log on to the Apsara Uni-manager Management Console for the first time after MFA is enabled by the administrator:
     a. On the Bind Virtual MFA Device page, bind an MFA device.
     b. Enter the username and password again as in Step 2 and click **Log On**.
     c. Enter a six-digit MFA verification code and click **Authenticate**.
   - You have enabled MFA and bound an MFA device:

     Enter a six-digit MFA verification code and click **Authenticate**.

> **Note** For more information, see the *Bind a virtual MFA device to enable MFA* topic in *Apsara Uni-manager Management Console User Guide*.

5. In the top navigation bar, choose **Products > Log Service**.
6. On the page that appears, select an organization and region, and then click **Access as Administrator**. The home page of the Log Service console is displayed.

2.3. Obtain an AccessKey pair

An AccessKey pair consists of an AccessKey ID and an AccessKey secret. The AccessKey pair is used to implement symmetric encryption to verify the identity of the requester. The AccessKey ID is used to identify a user. The AccessKey secret is used to encrypt the signature string. This topic describes how to obtain an AccessKey pair.

Prerequisites

Only the operation administrators or level-1 organization administrators can obtain the AccessKey pair of an organization.

Context

To call Apsara Uni-manager and cloud service APIs, we recommend that you use the AccessKey pair of a personal account. If you use the AccessKey pair of a personal account, you must configure header parameters as described in the following table for access control.

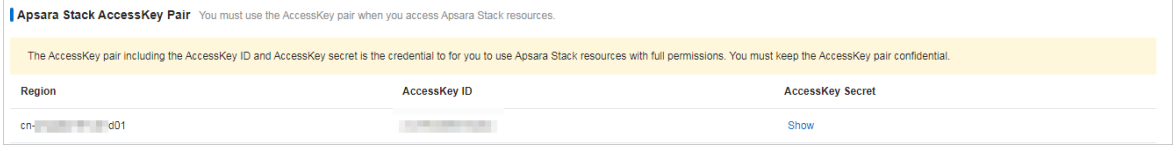
Parameter	Description
x-acs-regionid	The region ID, such as cn-hangzhou-*.
x-acs-organizationid	The ID of the organization in the Apsara Uni-manager Management Console.
x-acs-resourcegroupid	The ID of the resource set in the Apsara Uni-manager Management Console.
x-acs-instanceid	The ID of the instance on which you want to perform operations.

> **Warning** The AccessKey pairs of personal accounts are under control of the Apsara Uni-manager permission system. AccessKey pairs of organization accounts have higher permissions. For security purposes, organization operations must be approved by administrators.

Obtain the AccessKey pair of a personal account

To obtain the AccessKey pair of a personal account, perform the following operations:

1. Log on to the Apsara Uni-manager Management Console.
2. In the upper-right corner of the homepage, move the pointer over the profile picture and click **Personal Information**.
3. In the **Apsara Stack AccessKey Pair** section, view your AccessKey pair.



> **Note** The AccessKey pair consists of an AccessKey ID and an AccessKey secret. AccessKey pairs allow you to access Apsara Stack resources with full permissions for your account. You must keep your AccessKey pair confidential.

Obtain the AccessKey pair of an organization

To obtain the AccessKey pair of an organization, perform the following operations:

1. Log on to the Apsara Uni-manager Management Console as an administrator.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Organizations**.
4. In the organization navigation tree, click an organization name.
5. In the Current Organization section, click **Management Accesskey**.
6. In the Management AccessKey, view the AccessKey pair of the organization.

> **Note** An AccessKey pair is automatically allocated to each level-1 organization. Subordinate organizations use the same AccessKey pair of their level-1 organization.

2.4. Manage a project

A project in Log Service is a resource management unit that is used to separate and manage different resources. This topic describes how to create and delete a project in the Log Service console.

Create a project

> **Note** You can create up to 50 projects for each Apsara Stack tenant account.

1. Log on to the Log Service console.
2. In the Projects section, click **Create Project**.
3. In the **Create Project** panel, configure the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Project Name	The name of the project. The name must be unique in a region. You cannot change the name after you create the project.
Description	The description of the project.
Region	Select a region based on log sources. After you create a project, you cannot change the region where the project resides or migrate the project to another region. If you want to collect logs from an Elastic Compute Service (ECS) instance, we recommend that you select the region where the ECS instance resides. This way, Log Service can use the internal network of Alibaba Cloud to accelerate log collection.

Parameter	Description
Service Logs	Select the service logs that you want to store. After you select the service logs, the logs that are generated in the project are stored in a specified project. For more information, see *Monitor Log Service* in *Log Service Developer Guide*. ◦ If you select **Detailed Logs**, operation logs are stored in the specified project. You are billed by using the pay-as-you-go method. ◦ If you select **Important Logs**, logs of the consumer group latency and Logtail heartbeats are stored in the specified project. This feature is provided free of charge.
Log Storage Location	If you select the service logs that you want to store, you must configure this parameter. Valid values: ◦ Automatic creation (recommended) ◦ Current Project ◦ Other projects in the same region as your project

View the endpoint of a project

After you create a project, you can view the endpoint of the project on the **Overview** page.

1. In the Projects section, click the name of the project.
2. On the **Overview** page, view the endpoint of the project.

   To access the projects that reside in different regions, you must use different endpoints. To access the projects that reside in the same region over a private network or the Internet, you must use different endpoints.

Delete a project

Warning After you delete a project, all log data that is stored in the project and the configurations of the project are deleted and cannot be restored. Proceed with caution.

1. In the Projects section, find the project that you want to delete and click **Delete** in the Actions column.
2. In the **Delete Project** panel, select a reason in the Reason for Deletion section and click **OK**.

Project-related API operations

Action	Operation
Create a project	CreateProject
Delete a project	DeleteProject
Query a project	• Query a specified project: GetProject • Query all projects: ListProject
Modify a project	UpdateProject

For more information, see *API Reference* in *Log Service Developer Guide*.

2.5. Manage a Metricstore

This topic describes how to create, modify, and delete a Metricstore in the Log Service console.

Prerequisites

A project is created. For more information, see Manage a project.

Create a Metricstore

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. Choose **Time Series Storage > Metricstore**. On the Metricstore tab, click the **+** icon.
4. In the **Create Metricstore** panel, configure the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Metricstore Name	The name of the Metricstore. The name must be unique in the project to which the Metricstore belongs. After the Metricstore is created, you cannot change the name of the Metricstore.
Permanent Storage	If you turn on **Permanent Storage**, Log Service permanently stores the collected metric data. **Note** You can use an SDK to query the data retention period. The value 3650 indicates that metric data is permanently stored in the Metricstore.
Data Retention Period	The retention period of the collected metric data in the Metricstore. Valid values: 15 to 3000. Unit: days. Metrics whose retention period exceeds the value of this parameter are automatically deleted. If you do not turn on **Permanent Storage**, you must configure the **Data Retention Period** parameter. **Note** If you shorten the data retention period, Log Service deletes data based on the new retention period within 1 hour. The volume of data that is displayed on the **Storage Size(Log)** card on the homepage of the Log Service console is updated the next day. For example, if you change the value of the Data Retention Period parameter from 5 to 1, Log Service deletes the metric data of the previous four days within 1 hour after the change.
Shards	The number of shards. Log Service provides shards to read and write data. Each shard supports a write speed of 5 MB/s, 500 write operations per second, a read speed of 10 MB/s, and 100 read operations per second. You can create up to 10 shards for each Metricstore. You can create up to 200 shards for each project. For more information, see *Log Service Product Introduction*.

Modify the configurations of a Metricstore

1. Choose **Time Series Storage > Metricstore**. On the Metricstore tab, find the Metricstore whose configurations you want to modify and choose > **Modify**.
2. On the **Metricstore Attribute** page, click **Modify**.
   - Change the data retention period. For more information, see *Create a Metricstore*.
   - Manage shards.

When you create a Metricstore, the Shards parameter is set to 2 by default. You can split or merge the shards based on your business requirements. For more information, see Manage shards.

3. Click **Save**.

Delete a Metricstore

> **Notice**
> - Before you delete a Metricstore, you must delete all Logtail configurations of the Metricstore.
> - If the data shipping feature is enabled for the Metricstore, we recommend that you stop writing data to the Metricstore and make sure that all data in the Metricstore is shipped before you delete the Metricstore.
> - If you are not authorized to delete a Metricstore by using your Alibaba Cloud account, submit a ticket.

1. Choose **Time Series Storage > Metricstore**. On the Metricstore tab, find the Metricstore that you want to delete and choose > **Delete**.

> **Warning** After you delete a Metricstore, all metric data in the Metricstore is deleted and cannot be restored. Proceed with caution.

2. In the message that appears, click **OK**.

2.6. Manage Logstores

A Logstore in Log Service is used to collect, store, and query logs. This topic describes how to create, modify, and delete a Logstore in the Log Service console.

Create a Logstore

> **Note** You can create up to 200 Logstores in each project.

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. On the **Log Storage > Logstores** tab, click the Plus icon.
4. In the **Create Logstore** panel, configure the following parameters and click **OK**.

Parameter	Description
Logstore Name	The name of the Logstore. The name must be unique in the project to which the Logstore belongs. After the Logstore is created, you cannot change the name of the Logstore.
WebTracking	If you turn on **WebTracking**, you can collect and send data from HTML, HTML5, iOS, or Android platforms to Log Service by using the web tracking feature.
Permanent Storage	If you turn on **Permanent Storage**, Log Service permanently stores the collected logs. **Note** You can call an API operation to query the data retention period. If the value is 3650, the logs are permanently stored.

Parameter	Description
Data Retention Period	If you turn off **Permanent Storage**, you must configure this parameter to specify the retention period of logs in the Logstore. Valid values: 1 to 3000. Unit: days. If logs are stored for a period that exceeds the value of this parameter, the logs are automatically deleted. **Note** If you shorten the data retention period, Log Service deletes all expired logs within 1 hour. The data volume that is displayed for **Storage Size(Log)** on the homepage of the Log Service console is updated the next day. For example, if you change the value of the Data Retention Period parameter from 5 to 1, Log Service deletes the logs of the previous four days within 1 hour.
Shards	The number of shards. Log Service provides shards that allow you to read and write data. Each shard supports a write speed of 5 MB/s, 500 write operations per second, a read speed of 10 MB/s, and 100 read operations per second. You can create up to 10 shards in each Logstore. You can create up to 200 shards in each project. For more information, see *Log Service Product Introduction*.
Automatic Sharding	If you turn on **Automatic Sharding**, Log Service increases the number of shards when the existing shards cannot accommodate the data that is written.
Maximum Shards	If you turn on **Automatic Sharding**, you must configure this parameter to specify the maximum number of shards that can be created. Maximum value: 64.
Log Public IP	If you turn on **Log Public IP**, Log Service adds the following information to the Tag field of the collected logs: ◦ __client_ip__: the public IP address of the log source. ◦ __receive_time__: the time at which Log Service receives the log. The value is a UNIX timestamp representing the number of seconds that have elapsed since 00:00:00 Thursday, January 1, 1970.

Modify the configurations of a Logstore

1. In the Projects section, click the project that you want to manage.
2. On the **Log Storage > Logstores** tab, find the Logstore whose configurations you want to modify, click the icon, and then select **Modify**.
3. On the **Logstore Attributes** page, click **Modify**.

   For more information about the parameters, see Create a Logstore.
4. Click **Save**.

Delete a Logstore

If you no longer need a Logstore, you can delete the Logstore in the Log Service console.

Notice

- Before you can delete a Logstore, you must delete all Logtail configurations that are associated with the Logstore.
- If the log shipping feature is enabled for the Logstore, we recommend that you stop writing data to the Logstore and make sure that all data in the Logstore is shipped before you delete the Logstore.
- If you are not authorized to delete a Logstore by using your Alibaba Cloud account, submit a ticket to delete the Logstore.

1. On the **Log Storage > Logstores** tab, find the Logstore that you want to delete, click the icon, and then select **Delete**.

   Warning After you delete a Logstore, all logs in the Logstore are deleted and cannot be restored. Proceed with caution.

2. In the **Delete** message, click **OK**.

Logstore-related API operations

Action	API operation
Create a Logstore	CreateLogstore
Delete a Logstore	DeleteLogstore
Query a Logstore	• Query a Logstore: GetLogstore • Query all Logstores: ListLogstore
Modify a Logstore	UpdateLogstore

2.7. Manage shards

Log data on which read and write operations can be performed is stored in a shard of a Logstore. This topic describes how to split, merge, and delete shards in the Log Service console.

Context

When you create a Logstore, you must specify the number of shards for the Logstore. After the Logstore is created, you can split or merge shards to increase or decrease the number of shards in the Logstore.

- Each shard supports a write speed of up to 5 MB/s and a read speed of up to 10 MB/s. If the read speed or write speed of a shard cannot meet your business requirements, we recommend that you split the shard.

  You can split a shard of a Logstore on the **Logstore Attributes** page of the Logstore.

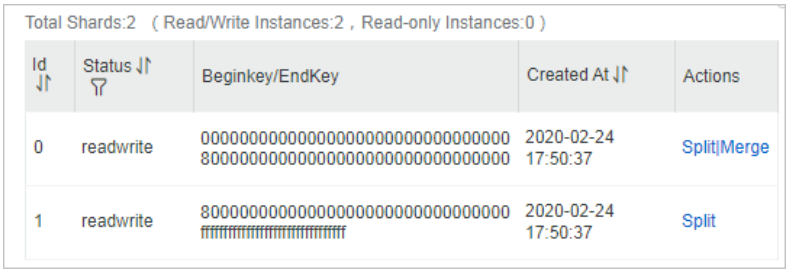
- If the data traffic is very small compared with the maximum read speed or write speed of a shard, we recommend that you merge the shard.

Split a shard

Each shard supports a write speed of up to 5 MB/s and a read speed of up to 10 MB/s. If the read speed or write speed of a shard cannot meet your business requirements, we recommend that you increase the number of shards. You can split a shard to increase the number of shards.

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. On the **Log Storage > Logstores** tab, find the Logstore that you want to manage, click the icon, and then select **Modify**.
4. On the **Logstore Attributes** page, click **Modify**.
5. Find the shard that you want to split and click **Split** in the Actions column.

   Note You can split only a shard that is in the readwrite state.



6. Select the number of new shards that you want to generate after the original shard is split.
7. Click **OK**.
8. Click **Save**.
   After you split the shard, the status of the shard changes from readwrite to readonly. You can continue to consume data from the shard that is in the readonly state. You cannot write data to this shard. New shards are in the readwrite state and are displayed below the original shard. The MD5 hash ranges of the new shards cover the MD5 hash range of the original shard.

Automatic sharding

Log Service provides the automatic sharding feature. After you enable the automatic sharding feature, a shard is automatically split if the following conditions are met:

- The data write speed exceeds the maximum write speed of the current shard for more than 5 minutes.
- The number of shards that are in the readwrite state does not exceed the maximum number of shards that is specified for the Logstore.

> **Note** Automatic sharding is not performed on the shards that are split from a shard in the previous 15 minutes.

You can enable the automatic sharding feature when you create or modify a Logstore. If you turn on Automatic Sharding, you must configure the Maximum Shards parameter.

- Automatic Sharding

  After you turn on Automatic Sharding, if the data write speed exceeds the maximum write speed of the current shard for more than 5 minutes, Log Service automatically splits the shard based on the data volume to increase the number of shards.

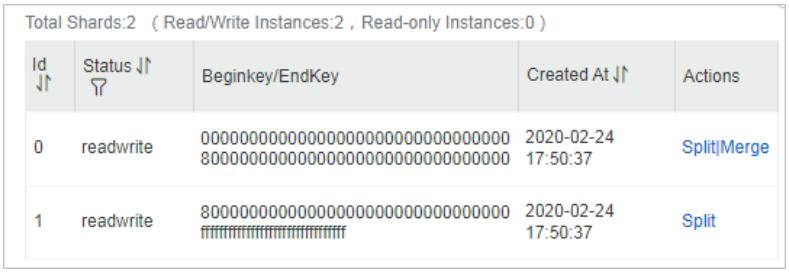
- Maximum Shards

  After you turn on Automatic Sharding, you must configure the Maximum Shards parameter to specify the number of new shards that you want to generate after a shard is automatically split. The maximum value is 64.

Merge shards

You can click Merge in the Actions column of a shard to merge shards. Log Service automatically locates a shard that is next to the specified shard and merges the two shards.

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. On the **Log Storage > Logstores** tab, find the Logstore that you want to manage, click the icon, and then select **Modify**.
4. On the **Logstore Attributes** page, click **Modify**.
5. Find the shard that you want to merge and click **Merge** in the Actions column.



6. Click **Save**.

   After you merge shards, the specified shard and the shard next to the specified shard are in the readonly state. A new shard is generated and is in the readwrite state. The MD5 hash range of the new shard covers the MD5 hash ranges of the original shards.

Delete a shard

- Automatic deletion

  If you specify a data retention period when you create a Logstore, the shards in the Logstore and the data stored in the shards are automatically deleted when the data retention period elapses.

- Manual deletion

  If you turn on Permanent Storage when you create a Logstore, we recommend that you delete the Logstore to delete the shards in the Logstore and the data stored in the shards. For more information, see Delete a Logstore.

Shard-related API operations

Action	API operation
Split a shard	SplitShard
Merge shards	MergeShards
Query shards	ListShard

2.8. Terms

2.8.1. Terms

This topic introduces the terms that are used in Log Service.

Basic resources

Term	Description
project	A project in Log Service is used to isolate the resources of different users and control access to specific resources.
Logstore	A Logstore in Log Service is used to collect, store, and query logs.
Metricstore	A Metricstore in Log Service is used to collect, store, and query metrics.
log	Logs are records of changes that occur in a system during the runtime of the system. The records contain information about the operations that are performed on specified objects and the results of the operations. The records are ordered by time.

Term	Description
log group	A log group is a collection of logs. A log group is the basic unit that is used to write and read logs. Logs in a log group contain the same metadata, such as the IP address and log source.
metric	Metrics are stored as time series.
shard	A shard is used to control the read and write capacities of a Logstore. In Log Service, data is stored in shards. Each shard has an MD5 hash range, and each range is a left-closed, right-open interval. The ranges do not overlap with each other. Each range must be within the entire MD5 hash range [00000000000000000000000000000000,ffffffffffffffffffffffffffffffff).
topic	A topic is a basic management unit in Log Service. You can specify topics when you collect logs. This way, Log Service can classify logs by topic.
endpoint	An endpoint of Log Service is a URL that is used to access a project and the data of the project. To access the projects in different regions, you must use different endpoints. To access the projects in the same region over an internal network or the Internet, you must also use different endpoints. For more information, see *Obtain the endpoint of Log Service* in Developer Guide.
AccessKey pair	An AccessKey pair is an identity credential that consists of an AccessKey ID and an AccessKey secret. The AccessKey ID and AccessKey secret are used for symmetric encryption and identity authentication. The AccessKey ID is used to identify a user. The AccessKey secret is used to encrypt and verify a signature string. The AccessKey secret must be kept confidential. For more information, see *Obtain an AccessKey pair* in Developer Guide.

Data collection

Term	Description
Logtail	Logtail is used by Log Service to collect logs. For more information, see *Collection by Logtail* in the "Data collection" section.
Logtail configuration	A Logtail configuration is a set of policies that are used by Logtail to collect logs. The configuration includes the log source and collection method. For more information, see *Collection by Logtail* in the "Data collection" section.
machine group	A machine group is a virtual group that contains multiple servers. Log Service uses machine groups to manage the servers from which you want to collect logs by using Logtail. For more information, see *Logtail machine group* in the "Data collection" section.

Data query and analysis

Term	Description
query	You can specify filter conditions in search statements to obtain specific logs. For more information, see *Log search overview* in the "Query and analysis" section.

Term	Description
analysis	You can invoke SQL functions on query results to perform statistical and analytical operations. Then, you can obtain analysis results. • Log Service supports the SQL-92 syntax for log data analysis. For more information, see *Log analysis overview* in the "Query and analysis" section. • Log Service supports the SQL-92 syntax and the PromQL syntax for metric data analysis. For more information, see *Query and analysis* in the "Time series storage" section.
query statement	A query statement is in the `Search statement|Analytic statement` format. A search statement can be executed alone. However, an analytic statement must be executed together with a search statement. The log analysis feature is used to analyze search results or all data in a Logstore. For more information, see *Query and analysis*.
index	Indexes are a structure for storage. Indexes are used to sort one or more columns of data. You can query data only after you create indexes for the data. Log Service provides the following types of indexes: • Full-text index: Log Service splits an entire log into multiple words based on specified delimiters to create indexes. In a search statement, the field names (keys) and field values (values) are plain text. • Field index: After you configure field indexes, you can specify field names and field values in the Key:Value format to search for logs. For more information, see *Configure indexes* in the "Query and analysis" section.

Data consumption and shipping

Term	Description
consumer group	You can use consumer groups to consume data in Log Service. A consumer group consists of multiple consumers. Each consumer consumes different logs that are stored in a Logstore.

Alerting

Term	Description
alert	An alert indicates an alert event. If an alert is triggered based on a specific alert monitoring rule, the alert management system sends the alert event to the notification management system. Log Service also provides alert-related subsystems, features, entities, and modules, such as the alert monitoring system and alert monitoring rules. For more information, see *Alerts*.

2.8.2. Log

Logs are records of changes that occur in a system during the running of the system. The records contain information about the operations that are performed on specific objects and the results of the operations. The records are ordered by time.

Format

Log data is stored in different formats, such as log files, log events, binary logs, and metric data. Log Service uses a semi-structured data model to define logs. A log consists of the following fields: topic, time, content, source, and tags. Log Service has different format requirements on different log fields. The following table describes the log fields and provides the format requirements.

Field	Description	Format
Topic	The custom field in a log. This field can be used to identify the log topic. For example, you can set different log topics, such as access_log and operation_log, for website logs based on log types. For more information, see *Topic*.	The field value can be a string of up to 128 bytes. The value can include an empty string. If the field is an empty string, the log topic is not configured.
Time	The time when the log is generated, or the system time of the host where Logtail resides when the log data is collected. This field is a reserved field.	The value is a UNIX timestamp. It is the number of seconds that have elapsed since 00:00:00 UTC, Thursday, January 1, 1970.
Content	The content of the log. The content consists of one or more items. Each item is a key-value pair.	A key-value pair must comply with the following requirements: • The key is a UTF-8 encoded string of up to 128 bytes. The key can contain letters, digits, and underscores (_). The key cannot start with a digit. The string is 1 to 128 bytes in length. The following fields cannot be used: ◦ __time__ ◦ __source__ ◦ __topic__ ◦ __partition_time__ ◦ _extract_others_ ◦ __extract_others__ • The value can be a string of up to 1 MB.
Source	The source of the log. For example, the value of this field can be the IP address of the server where the log is generated.	The value of this field can be a string of up to 128 bytes.
Tags	The tags of the log. Valid values: • Custom tags: the tags that you add when you call the PutLogs operation to write logs to a specified Logstore. • System tags: the tags that are added by Log Service. The tags include __client_ip__ and __receive_time__.	The field value is in the dictionary format. The keys and the values are strings. The field name is prefixed by __tag__:.

Example

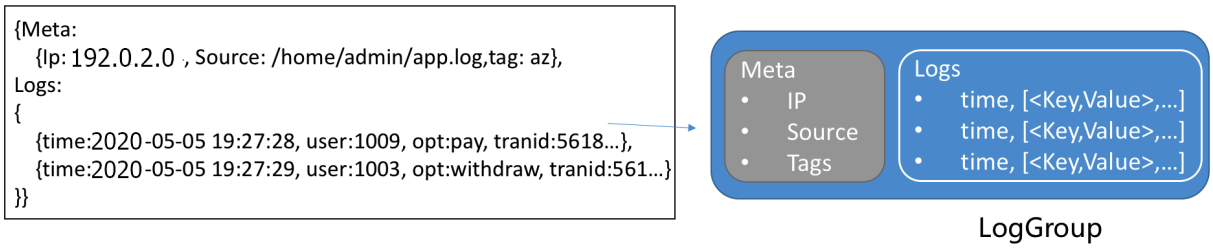
The following sample website access log shows the mapping between the raw log and the data model supported by Log Service.

```
127.0.0.1 - - [01/Mar/2021:12:36:49  0800] "GET /index.html HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Macint
osh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
```

2.8.3. Log group

A log group is a collection of logs. A log group is the basic unit that is used to write and read logs. The logs in a log group contain the same metadata, such as the IP address and log source.

When you write logs to or read logs from Log Service, multiple logs are encapsulated into a log group. This way, you can write and read logs by log group. This method can reduce the number of read and write operations and improve business efficiency. The maximum length of a log group is 5 MB.



2.8.4. Project

A project in Log Service is used to isolate the resources of different users and control access to specific resources.

A project contains resources such as Logstores, Metricstores, and machine groups, and provides an endpoint that you can use to access the resources of Log Service. We recommend that you use different projects to manage the data in different applications, services, or projects.

- You can use a project to organize and manage Logstores or Metricstores. You may need to use Log Service to collect and store the log data of different projects, services, or environments. You can specify different projects to manage the log data. This facilitates the consumption, export, and analysis of the log data.
- You can use a project to perform access control. You can grant the permissions to manage a specified project to a RAM user.
- A project provides an endpoint that you can use to access the resources in the project. Log Service allocates an exclusive endpoint to each project. You can use the endpoint of a project to read, write, and manage log data. For more information about the endpoints, see the "Obtain the endpoint of Log Service" topic in *Developer Guide*.

2.8.5. Logstore

A Logstore in Log Service is used to collect, store, and query log data.

Each Logstore belongs to a project. You can create multiple Logstores in a project. After you create a project, you can create multiple Logstores in the project to store different types of logs that are collected from the same application. For example, if you want to collect the operation logs, application logs, and access logs of App A, you can create a project named app-a and create three Logstores named operation_log, application_log, and access_log in the project to store the logs.

When you perform operations such as writing, querying, analyzing, consuming, or shipping logs, you must specify a Logstore.

- Log Service uses Logstores as a collection unit to collect logs.
- Log Service uses Logstores as a storage unit to support operations such as storing, consuming, and shipping logs.
- Log Service creates indexes in a Logstore to support query and analysis operations.

2.8.6. Metricstore

A Metricstore is a unit that is used to collect, store, and query metrics in Log Service.

Each Metricstore belongs to a project. You can create multiple Metricstores in a project. After you create a project, you can create multiple Metricstores in the project to store different types of metrics that are collected. For example, if you want to collect the metrics of hosts, cloud services, and business applications, you can create a project named demo-monitor and create three Metricstores named host-metrics, cloud-service-metrics, and app-metrics in the project to store the metrics.

You must specify a Metricstore when you write, query, analyze, or consume metrics.

- Log Service uses a Metricstore as a collection unit to collect metrics.
- Log Service uses a Metricstore as a storage unit to store and consume metrics.
- Log Service supports the SQL-92 syntax and the PromQL syntax for you to query and analyze metrics.

2.8.7. Metric

Log Service stores all data in Metricstores as time series. Log Service uses the model of time series data that is defined by Prometheus. Each time series consists of samples with the same metric identifier.

Metric identifier

Each time series has a unique metric identifier that consists of a metric name and a label.

Metric names are strings and must match the **[a-zA-Z_:][a-zA-Z0-9_:]*** regular expression. In most cases, a metric name indicates a description of a time series. For example, http_request_total indicates that each sample of a time series indicates the total number of received HTTP requests.

Labels are key-value pairs. Label keys must match the **[a-zA-Z_][a-zA-Z0-9_]*** regular expression. Label values can contain all characters except vertical bars (|). In most cases, a label indicates an attribute of a time series. For example, the value of the method key may be POST, and the value of the URL key may be /api/v1/get.

Samples

A sample indicates the value of a metric at a point in time. Each sample consists of a timestamp and a value. Timestamps are accurate to the nanosecond, and values are of the DOUBLE type.

Encoding format

When metric data is written to Log Service, the Protocol Buffer (Protobuf) format must be used. This format is also used to write log data. The metric identifier and samples are contained in the content field. The following table describes the related subfields.

Key	Description	Example
__name__	The name of the metric.	nginx_ingress_controller_response_size
__labels__	The labels of the metric. Format: {key}#$#{value}\|{key}#$#{value}|{key}#$#{value}. **Note** The labels must be sorted by key in alphabetical order.	app#$#ingress-nginx\|controller_class#$#nginx|controller_namespace#$#kube-system\|controller_pod#$#nginx-ingress-controller-589877c6b7-hw9cj
__time_nano__	The timestamp of a sample. The value is accurate to the nanosecond.	1585727297293000000
__value__	The value of a sample.	36.0

2.8.8. Shard

Shards are used to manage the read and write capacity of Logstores or Metricstores. In Log Service, data is stored in a shard.

MD5 value range

Each shard has an MD5 value range. and each range is a left-closed, right-open interval in the [BeginKey, EndKey) format. Each range does not overlap the ranges of other shards. The entire MD5 value range is within the following range: [00000000000000000000000000000000,ffffffffffffffffffffffffffffffff). You can specify the number of shards when you create a Logstore or Metricstore. Log Service evenly divides the entire MD5 value range for each shard.

- BeginKey: the start of a shard. The value is included in the MD5 value range of the shard.
- EndKey: the end of a shard. The value is excluded from the MD5 value range of the shard.

In this example, Logstore A has four shards. The following table describes the MD5 value range of each shard.

MD5 value range

Shard ID	Value range
Shard0	[00000000000000000000000000000000,40000000000000000000000000000000)
Shard1	[40000000000000000000000000000000,80000000000000000000000000000000)
Shard2	[80000000000000000000000000000000,c0000000000000000000000000000000)
Shard3	[c0000000000000000000000000000000,ffffffffffffffffffffffffffffffff)

To read data from a shard, you must specify the ID of the shard. To write data to a shard, you can use the load balancing method or specify a hash key.

- If you use the load balancing method, each data packet is randomly written to an available shard.
- If you specify a hash key, data is written to the shard whose MD5 value range includes the value of the specified hash key.

  For example, you use the shard range that is shown in the preceding table. If you specify 5F as a hash key to write data to a Logstore, the data is written to Shard1 because the MD5 value range of Shard1 contains the hash key 5F. If you specify 8C as a hash key, the data is written to Shard2 because the MD5 value range of Shard2 contains the hash key 8C.

Shard capacity

Each shard provides the following read capacity and write capacity:

- Write capacity: 5 MB/s or 500 times/s
- Read capacity: 10 MB/s or 100 times/s

We recommend that you adjust the number of shards based on the actual data traffic. If the data traffic exceeds the read or write capacity of a shard, you can split the shard into multiple shards to increase the capacity. If the data traffic is much lower than the read or write capacity of a shard, you can merge the shard with another shard to reduce the capacity and save costs.

For example, you have two shards that are in the readwrite state and the shards can provide a write capacity of up to 10 MB/s. If you need to write data at a speed of 14 MB/s in real time, we recommend that you split one of your shards into two shards. This way, you can have three shards that are in the readwrite state. If you need to write data at a speed of 3 MB/s in real time, we recommend that you merge your shards.

Notice

- If the error code 403 or 500 is frequently returned when you write data by calling the Log Service API, you can go to the CloudMonitor console to check the traffic and status codes. Then, you can determine whether to increase the number of shards.
- If the data traffic exceeds the capacity of your shards, Log Service attempts to provide services to meet your business requirements. However, Log Service cannot ensure the quality of the services.

Shard status

A shard can be in the readwrite state or readonly state.

When you create a shard, the shard is in the readwrite state. If you split a shard or merge shards, the status of the original shard changes to readonly. The newly generated shards are in the readwrite state. The status of a shard does not affect the read capacity of the shard. Data can be written to the shards that are in the readwrite state, but cannot be written to the shards that are in the readonly state.

Splitting and merging

Log Service allows you to split and merge shards.

- After you split a shard, two more shards are added. The new shards are in the readwrite state and are listed under the original shard. The MD5 value range of the new shards includes the MD5 value range of the original shard.

  You can split only a shard that is in the readwrite state. After you split a shard, the status of the shard changes from readwrite to readonly. This indicates that data can still be read from the shard, but cannot be written to the shard.

- You can merge two shards into one shard. The new shard is in the readwrite state and is listed under the original shards. The MD5 value range of the new shard includes the MD5 value range of the two original shards.

  When you merge shards, you must specify a shard that is in the readwrite state. The shard cannot be the last shard in the shard list. Log Service finds the shard whose MD5 value range is next to the specified shard and then merges the two shards. After you merge the shards, the status of the shards changes from readwrite to readonly. This indicates that data can still be read from the shards, but cannot be written to the shards.

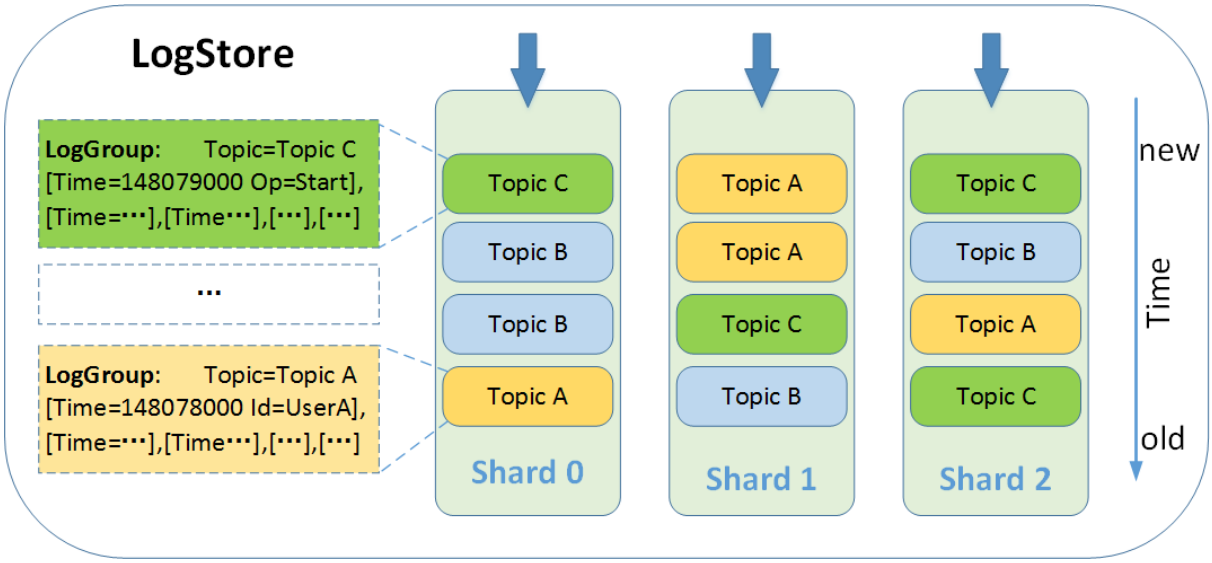
2.8.9. Topic

A topic is a basic management unit in Log Service. When you collect logs, you can specify topics to identify the logs.

You can use topics to identify logs that are generated by different services, users, and instances. For example, System A consists of an HTTP request processing module, a cache module, a logic processing module, and a storage module. You can set the log topic of the HTTP request processing module to http_module, the log topic of the cache module to cache_module, the log topic of the logic processing module to logic_module, and the log topic of the storage module to store_module. After the logs of the preceding modules are collected and saved to the same Logstore, you can identify logs based on the topics.

If you do not need to identify logs in a Logstore, set the topic to *Null - Do not generate topic* when you collect logs. A topic can be an empty string.

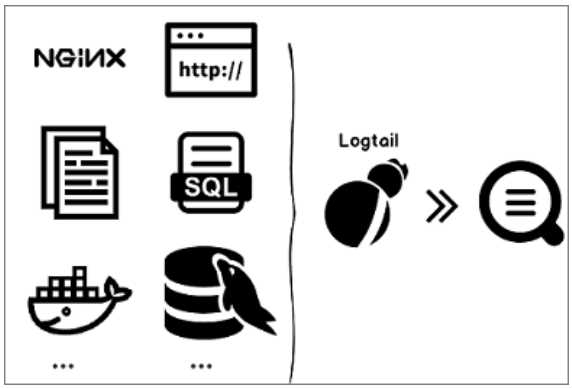
The following figure shows the relationships between Logstores, topics, and shards.


LogStore
LogGroup: Topic=Topic C
[Time=148079000 Op=Start],
[Time=···],[Time···],[···],[···]
...
LogGroup: Topic=Topic A
[Time=148078000 Id=UserA],
[Time=···],[Time···],[···],[···]
Topic C
Topic B
Topic B
Topic A
Shard 0
Topic A
Topic A
Topic C
Topic B
Shard 1
Topic C
Topic B
Topic A
Topic C
Shard 2
new
Time
old

3.Data collection
3.1. Collection by Logtail
3.1.1. Overview
3.1.1.1. Logtail overview

Logtail is a log collection agent that is provided by Log Service. You can use Logtail to collect logs from multiple data sources in real time. These sources include Elastic Compute Service (ECS) instances, data centers, and servers that belong to third-party cloud service providers. This topic describes the features, benefits, limits, and configuration process of Logtail.

Logtail-based log collection



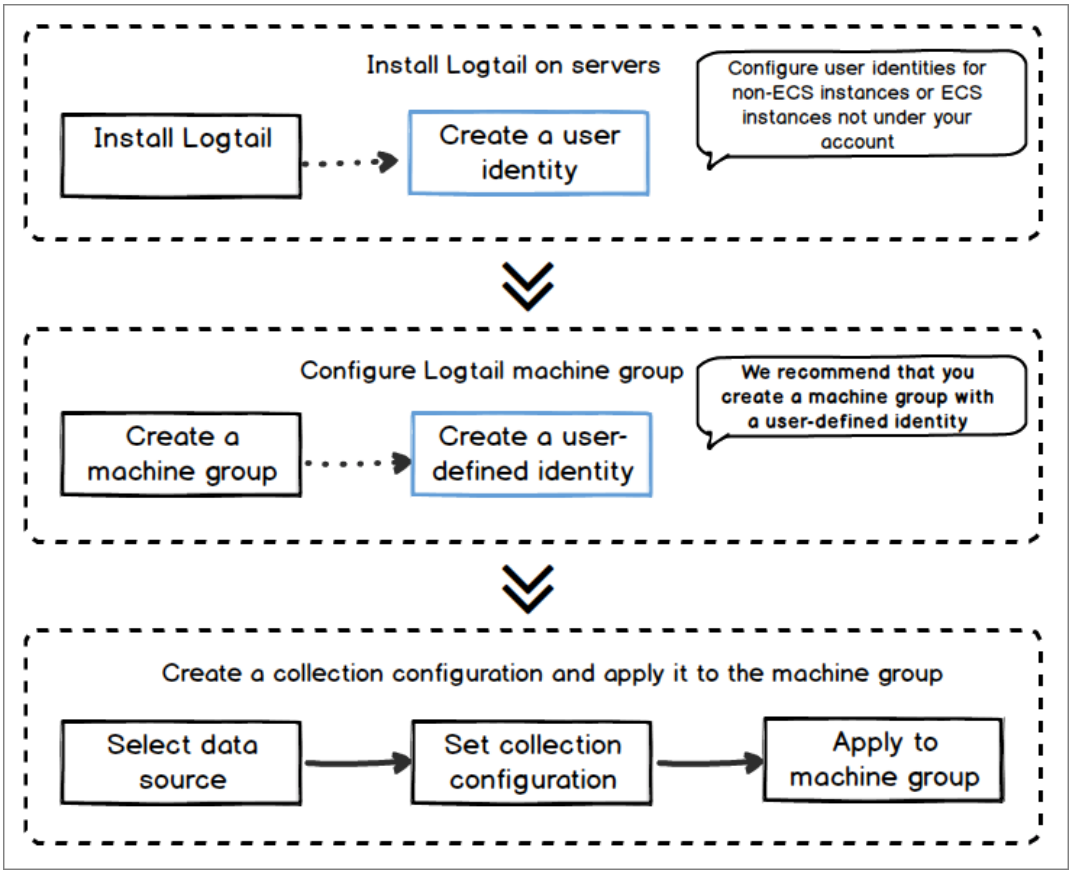
Benefits

- Supports non-intrusive log collection based on log files. You do not need to modify your application code. Your applications are not affected when Logtail collects logs.
- Allows you to collect text logs, binary logs, HTTP data, and container logs.
- Allows you to collect logs from standard containers, swarm clusters, and Kubernetes clusters.
- Handles exceptions during log collection. If a network or server exception occurs, Logtail retries log collection and caches logs on local servers to ensure data security.
- Provides centralized management based on Log Service. After you install Logtail on servers and create a machine group and Logtail configurations, Logtail collects logs from the servers and sends the logs to Log Service.
- Provides a comprehensive self-protection mechanism. The CPU, memory, and network resources that Logtail can use are limited. This ensures that Logtail does not affect the performance of other services on the server.

Limits

For more information about the limits of Logtail, see Limits.

Configuration process



To collect logs from servers by using Logtail, perform the following steps:

1. Install Logtail.

   Install Logtail on servers from which you want to collect logs. For more information, see Install Logtail in Linux and Install Logtail in Windows.

2. Create a machine group.

   Log Service allows you to create a custom ID-based machine group or an IP address-based machine group. For more information, see Create a machine group based on a server IP address and Create a machine group based on a custom ID.

3. Create a Logtail configuration and apply it to the machine group.

After you complete the preceding procedure, Logtail collects logs from your server and sends the logs to the specified Logstore. You can use the Log Service console, call API operations, or use SDKs to query logs.

Terms

- Machine group: A machine group contains one or more servers from which logs of a specific type are collected. After you apply Logtail configurations to a machine group, Log Service collects logs from the servers in the machine group based on the configurations.

  You can set an IP address-based identifier or a custom identifier for a machine group. Then, you can manage the servers in the machine group based on the identifier. You can create and delete a machine group, add servers to a machine group, and remove servers from a machine group in the Log Service console.

- Logtail: Logtail is a log collection agent that is provided by Log Service. Logtail runs on servers to collect logs from the servers. For more information, see Install Logtail in Linux or Install Logtail in Windows.
  - For a Linux-based server, Logtail is installed in the */usr/local/ilogtail* directory. Logtail initiates the following processes whose names start with ilogtail: a log collection process and a daemon process. The logs of Logtail are stored in the */usr/local/ilogtail/ilogtail.LOG* file.

  - For a Windows-based server, Logtail is installed in the *C:\Program Files\Alibaba\Logtail* directory (32-bit system) or *C:\Program Files (x86)\Alibaba\Logtail* directory (64-bit system). Choose **Control Panel > Administrative Tools > Services**. On the Services window, you can view the LogtailDaemon service. The logs of Logtail are stored in the `ilogtail.LOG` file.

- Logtail configurations: Logtail configurations are a set of policies that Logtail uses to collect logs. You can specify the data source and collection mode to create custom Logtail configurations for log collection. The configurations specify how to collect logs from servers, parse the logs, and send the logs to a specified Logstore.

Features

Feature	Description
Real-time log collection	Logtail monitors log files, and reads and parses incremental logs in real time. In most cases, logs are sent to Log Service within 3 seconds after they are generated. **Note** Logtail does not collect historical data. If Logtail reads a log later than 12 hours after the log was generated, Logtail drops the log.
Automatic log rotation	Multiple applications rotate log files based on the file size or date. The original log file is renamed and an empty log file is created during the rotation process. For example, the *app.LOG* file is renamed *app.LOG.1* and *app.LOG.2* during log rotation. You can specify the file to which collected logs are written, for example, *app.LOG*. Logtail monitors the log rotation process to ensure that no logs are lost.
Multiple data sources	Logtail can collect text logs, syslogs, HTTP logs, and MySQL binlogs.
Compatibility with open source collection agents	You can use open source agents such as Logstash and Beats to collect data. Then, you can use Logtail to collect data from the agents and send the data to Log Service.
Automatic exception handling	If data fails to be sent to Log Service due to exceptions, Logtail retries to collect logs based on the scenario. The exceptions include server errors, network errors, and quota exhaustion. If the retry fails, Logtail writes the data to the local cache and resends the data after 3 seconds.
Flexible collection policy configuration	Logtail allows you to create configurations for log collection in a flexible manner. You can specify the directories and files from which logs are collected. You can also specify an exact match or a wildcard match based on your business requirements. You can also specify the log collection mode and customize the fields that you want to extract. You can use a regular expression to extract fields from logs. Log data in Log Service must have the timestamp information. Logtail allows you to customize log time formats and then extract the required timestamps from the time information based on different formats.
Automatic synchronization of Logtail configurations	After you create or update Logtail configurations in the Log Service console, the configurations are synchronized to the servers in which Logtail is installed and take effect within 3 minutes. Logs are collected based on the original configurations during the synchronization.

Feature	Description
Status monitoring	Logtail monitors the CPU and memory resources that are consumed in real time. This ensures that Logtail does not consume an excessive number of resources or affect other services. If the resource consumption exceeds the limit, Logtail is automatically restarted. Logtail also monitors the network bandwidth resources that are consumed. This ensures that Logtail does not consume an excessive amount of bandwidth.
Data transmission with a signature	Logtail retrieves the AccessKey pair of your Apsara Stack tenant account and uses the pair to sign all log data that is sent to Log Service. This way, data tampering is prevented during data transmission. **Note** Logtail obtains the AccessKey pair of your Apsara Stack tenant account by using the HTTPS protocol to ensure the security of your AccessKey pair.

Data collection reliability

Logtail stores checkpoints that are periodically collected to the local server during log collection. If an exception such as an unexpected server shutdown or a process failure occurs, Logtail restarts and then collects data from the last checkpoint. This process avoids incomplete data collection. Logtail runs based on the startup parameters that are specified in the startup configuration file. If the usage of a resource exceeds the limit for more than 5 minutes, Logtail is forcibly restarted. After the restart, a small amount of duplicate data may be collected to the specified Logstore.

To improve log collection reliability, Logtail uses multiple internal mechanisms. However, logs may fail to be collected in the following scenarios:

- Logtail is not running, but logs are rotated multiple times.
- The log rotation rate is high, for example, one rotation per second.
- The log collection rate is lower than the log generation rate for a long period of time.

3.1.1.2. Log collection process of Logtail

This topic describes how Logtail collects logs. The log collection process consists of the following steps: monitor log files, read log files, process logs, filter logs, aggregate logs, and send logs.

Monitor log files

After you install Logtail on a server and create a Logtail configuration that is used to collect logs in the Log Service console, Log Service delivers the Logtail configuration to Logtail in real time. Then, Logtail monitors log files of the server based on the Logtail configuration. Logtail scans log directories and files based on the log file path and the maximum directory depth that you specify for monitoring in the Logtail configuration.

If the log files of the server in a machine group are not updated after you apply the Logtail configuration to the machine group, the log files are considered historical log files. Logtail does not collect historical log files. If log files are updated, Logtail reads and collects the files, and then sends the log files to Log Service. For more information about how to collect historical log files, see Import historical logs.

Logtail registers event listeners to monitor directories from which log files are collected. The event listeners pool the log files in the directories on a regular basis. This ensures that logs are collected at the earliest opportunity in a stable manner. For Linux-based servers, Inotify is used to monitor the directories and pool log files.

Read log files

After Logtail detects updated log files, Logtail reads the log files.

- The first time Logtail reads a log file, Logtail checks the size of the file.

- If the file size is less than 1 MB, Logtail reads data from the beginning of the file.
- If the file size is greater than 1 MB, Logtail reads from the last 1 MB of data in the file.

- If a log file is read before, Logtail reads the file from the previous checkpoint.
- Logtail can read up to 512 KB of data at the same time. Make sure that the size of each log in a log file does not exceed 512 KB.

> **Notice** If you change the system time on the server, you must restart Logtail. Otherwise, the log time becomes invalid and logs are dropped.

Process logs

After Logtail reads a log file, Logtail splits each log in the file into multiple lines, parses the log, and then configures the time field for the log.

- Split a log into multiple lines

  If you specify a regular expression to match the start part in the first line of a log, Logtail splits the log into multiple lines based on the regular expression. If you do not specify a regular expression, a single log line is processed as a log.

- Parse logs

  Logtail parses each log based on the collection mode that you specify in the Logtail configuration.

  > **Note** If you specify complex regular expressions, Logtail may consume an excessive amount of CPU resources. We recommend that you specify regular expressions that allow Logtail to parse logs in an efficient manner.

  If Logtail fails to parse a log, Logtail handles the failure based on the setting of the Drop Failed to Parse Logs parameter in the Logtail configuration.

  - If you turn on **Drop Failed to Parse Logs**, Logtail drops the log and reports an error.
  - If you turn off **Drop Failed to Parse Logs**, Logtail uploads the log. The key of the log is set to **raw_log** and the value is set to the log content.

- Configure the time field for a log
  - If you do not configure the time field for a log, the log time is set to the time when the log is parsed.
  - If you configure the time field for a log, Logtail processes the log based on the following conditions:
    - If the difference between the time when the log is generated and the current time is within 12 hours, the log time is extracted from the parsed log fields.
    - If the difference between the time when the log is generated and the current time is greater than 12 hours, the log is dropped and an error is reported.

Filter logs

After logs are processed, Logtail filters the logs based on the specified filter conditions.

- If you do not specify filter conditions in the **Filter Configuration** field, the logs are not filtered.
- If you specify filter conditions in the **Filter Configuration** field, the fields in each log are traversed.

  Logtail collects only the logs that meet the filter conditions.

Aggregate logs

To reduce the number of network requests, Logtail caches the processed and filtered logs for a specified period of time. Then, Logtail aggregates the logs and sends the logs to Log Service.

If one of the following conditions is met during the cache process, logs are aggregated and sent to Log Service:

- The aggregation duration exceeds 3 seconds.

- The number of aggregated logs exceeds 4,096.
- The total size of aggregated logs exceeds 512 KB.

Send logs

Logtail sends the aggregated logs to Log Service. You can set the max_bytes_per_sec and send_request_concurrency parameters in the Logtail startup configuration file to specify the maximum transmission rate of log data and concurrent requests. For more information, see Set Logtail startup parameters.

If a log fails to be sent, Logtail retries or no longer sends the log based on the error code.

Error code	Description	Handling method
401	Logtail is not authorized to collect data.	Logtail drops the log packets.
404	The project or Logstore that is specified in the Logtail configuration does not exist.	Logtail drops the log packets.
403	The shard quota is exhausted.	Logtail tries again after 3 seconds.
500	A server exception occurs.	Logtail tries again after 3 seconds.

3.1.1.3. Logtail configuration files and record files

This topic describes the basic configuration files and record files of Logtail. When Logtail is active, Logtail uses the configuration files and generates record files.

Startup configuration file (ilogtail_config.json)

The *ilogtail_config.json* file is used to set the startup parameters of Logtail. For more information, see Set Logtail startup parameters.

> Note

- The file must be a valid JSON file. Otherwise, Logtail cannot be started.
- If you modify the file, you must restart Logtail for the modifications to take effect.

After you install Logtail on a server, you can perform the following operations on the *ilogtail_config.json* file:

- Modify the runtime parameters of Logtail.
- Check whether the installation commands are correct.

  The values of the config_server_address and data_server_list parameters in the *ilogtail_config.json* file vary based on the installation command that you select. If the region in the command is different from the region where the Log Service project resides or the address in the command cannot be accessed, the command is incorrect. If the command is incorrect, Logtail cannot collect logs and must be reinstalled.
- File path
  - Linux: The file is stored in */usr/local/ilogtail/ilogtail_config.json*.

- Windows:
  - 64-bit: The file is stored in *C:\Program Files (x86)\Alibaba\Logtail\ilogtail_config.json*.
  - 32-bit: The file is stored in *C:\Program Files\Alibaba\Logtail\ilogtail_config.json*.

> **Note** You can run both 32-bit and 64-bit applications in a 64-bit Windows operating system. To ensure compatibility, the operating system stores 32-bit applications in a separate x86 directory.
>
> Logtail for Windows is a 32-bit application. Therefore, Logtail is installed in the Program Files (x86) directory in 64-bit Windows. If Logtail for 64-bit Windows is available, you can install Logtail in the Program Files directory.

- Containers: The file is stored in a Logtail container. The file path is specified in the environment variable *ALIYUN_LOGTAIL_USER_ID* of the Logtail container. You can run the `docker inspect ${logtail_container_name} | grep ALIYUN_LOGTAIL_CONFIG` command to view the file path. Example: */etc/ilogtail/conf/cn-hangzhou/ilogtail_config.json*.

- Sample file

  If you run the `cat /usr/local/ilogtail/ilogtail_config.json` command, the following output is returned:

```
{
    "config_server_address" : "http://logtail.cn-hangzhou-intranet.log.aliyuncs.com",
    "data_server_list" :
    [
        {
            "cluster" : "cn-hangzhou",
            "endpoint" : "cn-hangzhou-intranet.log.aliyuncs.com"
        }
    ],
    "cpu_usage_limit" : 0.4,
    "mem_usage_limit" : 100,
    "max_bytes_per_sec" : 2097152,
    "process_thread_count" : 1,
    "send_request_concurrency" : 4,
    "streamlog_open" : false
}
```

User identifier file

The user identifier file contains the ID of your Apsara Stack tenant account. The file specifies that the account is authorized to collect logs from the server on which Logtail is installed. For more information, see Configure an account ID on a server.

> **Note**
>
> - If the server is an Elastic Compute Service (ECS) instance that belongs to another Apsara Stack tenant account, a server that is deployed in a self-managed data center, or a server that is provided by a third-party cloud service provider, you must specify the ID of your Apsara Stack tenant account as a user identifier for the server after you install Logtail. Then, you can use Logtail to collect logs from the server by using the account.
> - You must specify the ID of an Apsara Stack tenant account as a user identifier in the user identifier file. You cannot specify the ID of a RAM user as a user identifier.
> - You must specify the name of the user identity file. You do not need to specify the file extension.
> - You can specify multiple user identifiers on a server. However, you can specify only one user identifier for a Logtail container.

- File path

- Linux: The file is stored in */etc/ilogtail/users/*.
- Windows: The file is stored in *C:\LogtailData\users*.
- Containers: The file is stored in a Logtail container. The file path is specified in the environment variable ALIYUN_LOGTAIL_USER_ID of the Logtail container. You can run the `docker inspect ${logtail_container_name} | grep ALIYUN_LOGTAIL_USER_ID` command to view the file path.

- Sample file

  If you run the `ls /etc/ilogtail/users/` command, the following output is returned:

  ```
  782392********** 37292*************
  ```

Custom identifier file (user_defined_id)

The *user_defined_id* file is used to configure a custom ID for a machine group. For more information, see Create a machine group based on a custom ID.

> **Note** When you create a custom ID-based machine group, you must configure the *user_defined_id* file.

- File path
  - Linux: The file is stored in */etc/ilogtail/user_defined_id*.
  - Windows: The file is stored in *C:\LogtailData\user_defined_id*.
  - Containers: The file is stored in a Logtail container. The file path is specified in the environment variable ALIYUN_LOGTAIL_USER_DEFINED_ID of the Logtail container. You can run the `docker inspect ${logtail_container_name} | grep ALIYUN_LOGTAIL_USER_DEFINED_ID` command to view the file path.

- Sample file

  If you run the `cat /etc/ilogtail/user_defined_id` command, the following output is returned:

  ```
  aliyun-ecs-rs1e16355
  ```

Logtail configuration file (user_log_config.json)

The *user_log_config.json* file records the information of a Logtail configuration received by Logtail from Log Service. The file is in the JSON format and is updated along with configuration updates. You can use the *user_log_config.json* file to check whether the Logtail configuration is delivered to the server on which Logtail is installed. If the Logtail configuration file exists and the configurations in the file are the same as the settings of the Logtail configuration in Log Service, the Logtail configuration is delivered.

> **Note** We recommend that you do not modify the Logtail configuration file unless you need to specify sensitive information, such as the AccessKey pair and database password.

- File path
  - Linux: The file is stored in */usr/local/ilogtail/ilogtail_config.json*.
  - Windows
    - 64-bit: The file is stored in *C:\Program Files (x86)\Alibaba\Logtail\user_log_config.json*.
    - 32-bit: The file is stored in *C:\Program Files\Alibaba\Logtail\user_log_config.json*.
  - Containers: The file is stored in */usr/local/ilogtail/user_log_config.json*.

- Sample file

  If you run the `cat /usr/local/ilogtail/user_log_config.json` command, the following output is returned:

```
{
   "metrics" : {
      "##1.0##k8s-log-c12ba2028*****939f0b$app-java" : {
         "aliuid" : "16542189*****50",
         "category" : "app-java",
         "create_time" : 1534739165,
         "defaultEndpoint" : "cn-hangzhou-intranet.log.aliyuncs.com",
         "delay_alarm_bytes" : 0,
         "enable" : true,
         "enable_tag" : true,
         "filter_keys" : [],
         "filter_regs" : [],
         "group_topic" : "",
         "local_storage" : true,
         "log_type" : "plugin",
         "log_tz" : "",
         "max_send_rate" : -1,
         "merge_type" : "topic",
         "plugin" : {
            "inputs" : [
               {
                  "detail" : {
                     "IncludeEnv" : {
                        "aliyun_logs_app-java" : "stdout"
                     },
                     "IncludeLable" : {
                        "io.kubernetes.container.name" : "java-log-demo-2",
                        "io.kubernetes.pod.namespace" : "default"
                     },
                     "Stderr" : true,
                     "Stdout" : true
                  },
                  "type" : "service_docker_stdout"
               }
            ]
         },
         "priority" : 0,
         "project_name" : "k8s-log-c12ba2028c*****ac1286939f0b",
         "raw_log" : false,
         "region" : "cn-hangzhou",
         "send_rate_expire" : 0,
         "sensitive_keys" : [],
         "tz_adjust" : false,
         "version" : 1
      }
   }
}
```


AppInfo record file (app_info.json)

The *app_info.json* file records the information of Logtail, such as the startup time and the IP address and hostname obtained by Logtail.

If the IP address of a server is associated with the hostname in the */etc/hosts* file of the server, Logtail obtains the IP address. If you do not associate the IP address of a server with the hostname, Logtail obtains the IP address of the first network interface controller (NIC) on the server.

> **Note**
> - The AppInfo record file records only the basic information of Logtail.
> - If you modify the hostname or other network settings of the server, you must restart Logtail to obtain a new IP address.

- File path
  - Linux: The file is stored in */usr/local/ilogtail/app_info.json*.
  - Windows
    - 64-bit: The file is stored in *C:\Program Files (x86)\Alibaba\Logtail\app_info.json*.
    - 32-bit: The file is stored in *C:\Program Files\Alibaba\Logtail\app_info.json*.
  - Containers: The file is stored in the */usr/local/ilogtail/app_info.json*.
- Sample file

  If you run the `cat /usr/local/ilogtail/app_info.json` command, the following output is returned:

```
{
   "UUID" : "",
   "hostname" : "logtail-ds-slpn8",
   "instance_id" : "E5F93BC6-B024-11E8-8831-0A58AC14039E_1**.***.***.***_1536053315",
   "ip" : "1**.***.***.***",
   "logtail_version" : "0.16.13",
   "os" : "Linux; 3.10.0-693.2.2.el7.x86_64; #1 SMP Tue Sep 12 22:26:13 UTC 2017; x86_64",
   "update_time" : "2018-09-04 09:28:36"
}
```

Field	Description
UUID	The serial number of the server.
hostname	The hostname.
instance_id	The unique identifier of Logtail. This identifier is randomly generated.
ip	The IP address that is obtained by Logtail. If Logtail does not obtain an IP address, the value of this parameter is null. Logtail cannot run as expected. You must specify an IP address for the server and then restart Logtail. **Note** If you create an IP address-based machine group, you must make sure that the IP address that you specify for the machine group is the same as the value of this field. If the two IP addressed are different, modify the IP address that you specify for the machine group in the Log Service console. Check the IP addresses again after 1 minute.
logtail_version	The version of Logtail.
os	The version of the operating system.
update_time	The last startup time of Logtail.

Logtail operational log file (ilogtail.LOG)

The *logtail_plugin.LOG* file records the operational logs of Logtail plug-ins. The levels of logs in ascending order include INFO, WARN, ERROR. You can ignore logs at the INFO level.

- File path
  - Linux: The file is stored in */usr/local/ilogtail/ilogtail.LOG*.
  - Windows
    - 64-bit: The file is stored in *C:\Program Files (x86)\Alibaba\Logtail\ilogtail.LOG*.
    - 32-bit: The file is stored in *C:\Program Files\Alibaba\Logtail\ilogtail.LOG*.
  - Containers: The file is stored in */usr/local/ilogtail/ilogtail.LOG*.
- Sample file

  If you run the `tail /usr/local/ilogtail/ilogtail.LOG` command, the following output is returned:

```
[2018-09-13 01:13:59.024679]    [INFO]    [3155]    [build/release64/sls/ilogtail/elogtail.cpp:123]
change working dir:/usr/local/ilogtail/
[2018-09-13 01:13:59.025443]    [INFO]    [3155]    [build/release64/sls/ilogtail/AppConfig.cpp:175
]    load logtail config file, path:/etc/ilogtail/conf/ap-southeast-2/ilogtail_config.json
[2018-09-13 01:13:59.025460]    [INFO]    [3155]    [build/release64/sls/ilogtail/AppConfig.cpp:176
]    load logtail config file, detail:{
   "config_server_address" : "http://logtail.ap-southeast-2-intranet.log.aliyuncs.com",
   "data_server_list" : [
      {
         "cluster" : "ap-southeast-2",
         "endpoint" : "ap-southeast-2-intranet.log.aliyuncs.com"
      }
]
```

Logtail plug-in log file (logtail_plugin.LOG)

The *logtail_plugin.LOG* file records the operational logs of Logtail plug-ins. The levels of logs in ascending order include INFO, WARN, ERROR. You can ignore logs at the INFO level.

If an exception such as **CANAL_RUNTIME_ALARM** occurs, you can troubleshoot the exception based on the *logtail_plugin.LOG* file.

- File path
  - Linux: The file is stored in */usr/local/ilogtail/logtail_plugin.LOG*.
  - Windows:
    - 64-bit: The file is stored in *C:\Program Files (x86)\Alibaba\Logtail\logtail_plugin.LOG*.
    - 32-bit: The file is stored in *C:\Program Files\Alibaba\Logtail\logtail_plugin.LOG*.
  - Containers: The file is stored in */usr/local/ilogtail/logtail_plugin.LOG*.
- Sample file

  If you run the `tail /usr/local/ilogtail/logtail_plugin.LOG` command, the following output is returned:

```
2018-09-13 02:55:30 [INF] [docker_center.go:525] [func1] docker fetch all:start
2018-09-13 02:55:30 [INF] [docker_center.go:529] [func1] docker fetch all:stop
2018-09-13 03:00:30 [INF] [docker_center.go:525] [func1] docker fetch all:start
2018-09-13 03:00:30 [INF] [docker_center.go:529] [func1] docker fetch all:stop
2018-09-13 03:03:26 [INF] [log_file_reader.go:221] [ReadOpen] [##1.0##sls-zc-test-hz-pub$docker-std
out-config,k8s-stdout]    open file for read, file:/logtail_host/var/lib/docker/containers/7f46afec
6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f114
8b2e2f31bd3410f5b2d624-json.log    offset:40379573    status:794354-64769-40379963
2018-09-13 03:03:26 [INF] [log_file_reader.go:221] [ReadOpen] [##1.0##k8s-log-c12ba2028cfb444238cd9
ac1286939f0b$docker-stdout-config,k8s-stdout]    open file for read, file:/logtail_host/var/lib/doc
ker/containers/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624/7f46afec6a14de39b59
ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624-json.log    offset:40379573    status:794354-64769-40
379963
2018-09-13 03:04:26 [INF] [log_file_reader.go:308] [CloseFile] [##1.0##sls-zc-test-hz-pub$docker-st
dout-config,k8s-stdout]    close file, reason:no read timeout    file:/logtail_host/var/lib/docker/
containers/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624/7f46afec6a14de39b59ee9c
dfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624-json.log    offset:40379963    status:794354-64769-403799
63
2018-09-13 03:04:27 [INF] [log_file_reader.go:308] [CloseFile] [##1.0##k8s-log-c12ba2028cfb444238cd
9ac1286939f0b$docker-stdout-config,k8s-stdout]    close file, reason:no read timeout    file:/logta
il_host/var/lib/docker/containers/7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624/
7f46afec6a14de39b59ee9cdfbfa8a70c2fa26f1148b2e2f31bd3410f5b2d624-json.log    offset:40379963    sta
tus:794354-64769-40379963
2018-09-13 03:05:30 [INF] [docker_center.go:525] [func1] docker fetch all:start
2018-09-13 03:05:30 [INF] [docker_center.go:529] [func1] docker fetch all:stop
```

Container path mapping file (docker_path_config.json)

The *docker_path_config.json* file is created only when you collect container logs. The file records the path mappings between container log files and host log files. The file is in the JSON format.

If the **DOCKER_FILE_MAPPING_ALARM** message appears when you troubleshoot a log collection exception, Docker files fail to be mapped to host files. You can use the *docker_path_config.json* file to troubleshoot the exception.

> **Note** This file is an information record file. Modifications to this file do not take effect. If you delete this file, another file is automatically created without service interruptions.

- File path

  /usr/local/ilogtail/docker_path_config.json

- Sample file

  If you run the `cat /usr/local/ilogtail/docker_path_config.json` command, the following output is returned:

```
{
   "detail" : [
      {
         "config_name" : "##1.0##k8s-log-c12ba2028cfb444238cd9ac1286939f0b$nginx",
         "container_id" : "df19c06e854a0725ea7fca7e0378b0450f7bd3122f94fe3e754d8483fd330d10",
         "params" : "{\n   \"ID\" : \"df19c06e854a0725ea7fca7e0378b0450f7bd3122f94fe3e754d8483fd330
d10\",\n   \"Path\" : \"/logtail_host/var/lib/docker/overlay2/947db346695a1f65e63e582ecfd10ae1f5701
9a1b99260b6c83d00fcd1892874/diff/var/log\",\n   \"Tags\" : [\n      \"nginx-type\",\n      \"access
-log\",\n      \"_image_name_\",\n      \"registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-
test:latest\",\n      \"_container_name_\",\n      \"nginx-log-demo\",\n      \"_pod_name_\",\n
\"nginx-log-demo-h2lzc\",\n      \"_namespace_\",\n      \"default\",\n      \"_pod_uid_\",\n
\"87e56ac3-b65b-11e8-b172-00163f008685\",\n      \"_container_ip_\",\n      \"172.20.4.224\",\n
\"purpose\",\n      \"test\"\n   ]\n}\n"
      }
   ],
   "version" : "0.1.0"
}
```


3.1.2. Installation

3.1.2.1. Install Logtail on a Linux server

This topic describes how to install, upgrade, and uninstall Logtail on a Linux server.

Supported operating systems

You can install Logtail on servers that run one of the following x86-64 Linux operating systems:

- Aliyun Linux 2
- Red Hat Enterprise Linux 6, Red Hat Enterprise Linux 7, and Red Hat Enterprise Linux 8
- CentOS Linux 6, CentOS Linux 7, and CentOS Linux 8
- Debian GNU/Linux 8, Debian GNU/Linux 9, and Debian GNU/Linux 10
- Ubuntu 14.04, Ubuntu 16.04, Ubuntu 18.04, and Ubuntu 20.04
- SUSE Linux Enterprise Server 11, SUSE Linux Enterprise Server 12, and SUSE Linux Enterprise Server 15
- openSUSE Leap 15.1, openSUSE Leap 15.2, and openSUSE Leap 42.3
- Glibc 2.5 or later

Procedure

> **Note** If you run the installation command on a server on which Logtail is installed, the installer uninstalls Logtail from the server, deletes the `/usr/local/ilogtail` directory, and then reinstalls Logtail. If the installation is successful, Logtail starts when the system reboots.

1. Run the following command to download the Logtail installer:

   ```
   wget http://${service:sls-backend-server:sls_data.endpoint}/logtail.sh -O logtail.sh; chmod 755 lo
   gtail.sh
   ```

   > **Note** You must replace `${service:sls-backend-server:sls_data.endpoint}` in the command with the actual endpoint. You can view the endpoint information on the Overview page of a Logstore.

2. Run the installation command.

   Start Linux PowerShell and run the following command as an administrator to install Logtail:

```
./logtail.sh install
```

3. Configure a user identifier.

View the version of Logtail

Go to the installation directory and open the `/usr/local/ilogtail/app_info.json` file. The value of the `logtail_version` field indicates the version of Logtail. Run the `cat /usr/local/ilogtail/app_info.json` command to view the version of Logtail. The following example shows a response:

```
{
   "UUID" : "0DF18E97-0F2D-486F-B77F-*********",
   "hostname" : "david*******",
   "instance_id" : "F4FAFADA-F1D7-11E7-846C-00163E30349E_*********_1515129548",
   "ip" : "**********",
   "logtail_version" : "0.16.0",
   "os" : "Linux; 2.6.32-220.23.2.ali1113.el5.x86_64; #1 SMP Thu Jul 4 20:09:15 CST 2013; x86_64",
   "update_time" : "2018-01-05 13:19:08"
}
```

Upgrade Logtail

You can use the Logtail installation script logtail.sh to upgrade Logtail. The installation script selects an upgrade method based on the configurations of the Logtail that is installed.

Note During the upgrade, Logtail is temporarily stopped, and all files, except the configuration files and checkpoint files, are overwritten.

Run the following command to upgrade Logtail:

```
# Download the Logtail installer.
wget http://${service:sls-backend-server:sls_data.endpoint}/logtail.sh -O logtail.sh;
chmod 755 logtail.sh
# Upgrade Logtail.
sudo ./logtail.sh upgrade
```

Check the upgrade result:

```
# The upgrade is successful.
Stop logtail successfully.
ilogtail is running
Upgrade logtail success
{
   "UUID" : "***",
   "hostname" : "***",
   "instance_id" : "***",
   "ip" : "***",
   "logtail_version" : "0.16.11",
   "os" : "Linux; 3.10.0-693.2.2.el7.x86_64; #1 SMP Tue Sep 12 22:26:13 UTC 2017; x86_64",
   "update_time" : "2018-08-29 15:01:36"
}
# The upgrade fails because the current version is the latest version.
[Error]:    Already up to date.
```

Start and stop Logtail

- Start Logtail

Run the following command as an administrator to start Logtail:

```
/etc/init.d/ilogtaild start
```

- Stop Logtail

  Run the following command as an administrator to stop Logtail:

  ```
  /etc/init.d/ilogtaild stop
  ```

Uninstall Logtail

Run the following command as an administrator to uninstall Logtail in Linux PowerShell:

```
wget http://${service:sls-backend-server:sls_data.endpoint}/logtail.sh -O logtail.sh
chmod 755 logtail.sh
./logtail.sh uninstall
```

3.1.2.2. Install Logtail in Windows

This topic describes how to install Logtail on a Windows server.

Supported systems

Logtail supports the following Windows operating systems:

- Windows 7 (Client) 32-bit
- Windows 7 (Client) 64-bit
- Windows Server 2008 32-bit
- Windows Server 2008 64-bit
- Windows Server 2012 64-bit
- Windows Server 2016 64-bit

Procedure

1. Download the installation package.

   Run the following command to download the installation package:

   ```
   wget http://${service:sls-backend-server:sls_data.endpoint}/windows/logtail_installer.zip
   ```

   > **Note** You must replace `${service:sls-backend-server:sls_data.endpoint}` in the command with the actual endpoint. For more information about endpoints, see View the information of a project.

2. Decompress the `logtail_installer.zip` package to the current directory.
3. Run the installation command.

   Run Windows PowerShell or Command Prompt as an administrator. Enter the `logtail_installer` directory, and then run the installation command based on the network type.

   ```
   .\logtail_installer.exe install me-east-1
   ```

   > **Note** You must replace `${region}` in the command with the actual endpoint. For more information about endpoints, see View the information of a project.

4. Configure an account ID for a server.

Installation directory

After you run the installation command, Logtail is installed in the specified directory. The directory cannot be changed. In the directory, you can View the version of Logtail in the *app_info.json* file or Uninstall Logtail.

The installation directory is as follows:

- 32-bit Windows: *C:\Program Files\Alibaba\Logtail*
- 64-bit Windows: *C:\Program Files (x86)\Alibaba\Logtail*

> **Note** You can run 32-bit or 64-bit applications in a 64-bit Windows operating system. However, the operating system stores 32-bit applications in separate x86 folders to ensure compatibility.
>
> Logtail for Windows is a 32-bit application. Therefore, it is installed in the *Program Files (x86)* folder in 64-bit Windows. If Logtail for 64-bit Windows becomes available in the future, it will be installed in the *Program Files* folder.

View the version of Logtail

To view the version of Logtail, go to the default installation directory, and then use the notepad or another text editor to open the *app_info.json* file. The `logtail_version` field shows the version of Logtail.

In the following example, the version of Logtail is 1.0.0.0:

```
{
    "logtail_version" : "1.0.0.0"
}
```

Upgrade Logtail

- Automatic upgrade

  Logtail later than 1.0.0.0 is automatically upgraded in Windows.

- Manual upgrade

  Logtail earlier than 1.0.0.0 must be manually upgraded. The manual upgrade procedure is the same as the installation procedure.

  > **Note** During a manual upgrade, the files in the original installation directory are deleted. We recommend that you back up the files before you perform a manual upgrade.

Start and stop Logtail

Open the **Control Panel**, choose System and Security > **Administrative Tools**, and then double-click **Services**.

Find the service based on your Logtail version.

- Logtail 0.x.x.x: LogtailWorker.
- Logtail 1.0.0.0 and later: LogtailDaemon.

Perform the following operations as required:

- Start Logtail

  Right-click the service and select **Start** from the shortcut menu.

- Stop Logtail

  Right-click the service and select **Stop** from the shortcut menu.

- Restart Logtail

  Right-click the service and select **Restart** from the shortcut menu.

Uninstall Logtail

Run Windows PowerShell or Command Prompt as an administrator. Enter the `logtail_installer` directory, and then run the following command:

```
.\logtail_installer.exe uninstall
```

After Logtail is uninstalled, the installation directory is deleted. However, some residual configuration data is still maintained in the *C:\LogtailData* directory. You can manually delete the data. The residual configuration data includes the following information:

- *checkpoint*: checkpoints of all plug-ins, for example, the Windows event log plug-in.
- *logtail_check_point*: checkpoints of Logtail.
- *users*: IDs of Apsara Stack tenant accounts.

3.1.2.3. Set Logtail startup parameters

This topic describes how to set Logtail startup parameters.

Context

You may need to set Logtail startup parameters in the following scenarios:

- You need to collect a large number of log files that consume much memory. You want to maintain the metadata (such as the file signature, collection location, and file name) of each file in the memory.
- The CPU usage is high due to heavy log data traffic.
- The traffic sent to Log Service is heavy due to a large amount of log data.
- You want to collect syslogs or TCP data streams.

Startup configurations

- File path

  ```
  /usr/local/ilogtail/ilogtail_config.json
  ```

- File format

  JSON

- Sample file (only partial configurations are provided)

  ```
  {
      ...
      "cpu_usage_limit" : 0.4,
      "mem_usage_limit" : 100,
      "max_bytes_per_sec" : 2097152,
      "process_thread_count" : 1,
      "send_request_concurrency" : 4,
      "streamlog_open" : false,
      "streamlog_pool_size_in_mb" : 50,
      "streamlog_rcv_size_each_call" : 1024,
      "streamlog_formats":[],
      "streamlog_tcp_port" : 11111,
      "buffer_file_num" : 25,
      "buffer_file_size" : 20971520,
      "buffer_file_path" : "",
      ...
  }
  ```

Startup parameters

Parameter	Description	Example
cpu_usage_limit	The CPU usage threshold for a single core. Data type: double.	For example, the value 0.4 indicates that the CPU usage of Logtail is limited to 40% processing capacity of a single core. In most cases, the processing capacity of a single core is about 24 MB/s in the simple mode and 12 MB/s in the full regex mode.
mem_usage_limit	The usage threshold of the resident memory. Data type: integer. Unit: MB.	For example, the value 100 indicates that the memory usage of Logtail is limited to 100 MB. If the threshold is exceeded, Logtail restarts. If you want to collect more than 1,000 log files, you can increase the threshold value.
max_bytes_per_sec	The traffic limit on the raw data that is sent by Logtail. Data type: integer. Unit: bytes/s.	For example, the value 2097152 indicates that the data transfer rate of Logtail is limited to 2 MB/s.
process_thread_count	The number of threads that Logtail uses to process data.	Default value: 1. Each thread provides a write speed of 24 MB/s in the simple mode and 12 MB/s in the full regex mode. We recommend that you do not modify the default value.
send_request_concurrency	Logtail sends data packets asynchronously by default. If the write transactions per second (TPS) is high, you can set this parameter to a greater value.	Twenty asynchronous concurrencies are provided by default. Each concurrency can provide 0.5 MB/s to 1 MB/s network throughput. The number of concurrencies varies with the network delay.
streamlog_open	Specifies whether to receive syslogs. Data type: Boolean.	The value false indicates that syslogs are not received. The value true indicates that syslogs are received.
streamlog_pool_size_in_mb	The size of memory pool that the syslog server uses to cache syslogs. Unit: MB.	Logtail requests memory when it starts. Set the memory pool size based on the server memory size and your business requirements.
streamlog_rcv_size_each_call	The size of the buffer that Logtail uses when the linux socket rcv API is called. Unit: bytes. Valid values: 1024 to 8192.	You can set a greater value if the syslog traffic is high.
streamlog_formats	The method that is used to parse received syslogs.	N/A
streamlog_tcp_addr	The associated address that Logtail uses to receive syslogs. Default value: 0.0.0.0.	N/A
streamlog_tcp_port	The TCP port that Logtail uses to receive syslogs.	Default value: 11111.

Parameter	Description	Example
buffer_file_num	The maximum number of cached files. If a network exception occurs or the writing quota is exceeded, Logtail writes parsed logs to local files in the installation directory. After the network recovers or a new writing quota is available, Logtail retries to send the logs to Log Service.	Default value: 25.
buffer_file_size	The maximum number of bytes that can be contained in each cache file. The maximum disk space available for cache files is the value of buffer_file_num multiplied by the value of buffer_file_size.	Default value: 20971520 bytes (20 MB).
buffer_file_path	The directory in which cached files are stored. If you modify this parameter, you must move the files (for example, *logtail_buffer_file_**) in the old cache directory to the new directory. Then, Logtail can read, send, and delete the cache files.	The default value is null, which indicates that the cached files are stored in the Logtail installation directory */usr/local/ilogtail*.
bind_interface	The name of the NIC associated with the local machine, for example, eth1. This parameter is valid only for Logtail that runs in Linux.	By default, the available NICs are automatically associated with the local machine. If you specify this parameter, Logtail will use the specified NIC to upload logs.
check_point_filename	The full path in which the checkpoint file is stored. This parameter is used to customize the path to store the checkpoint file of Logtail.	Default value: */tmp/logtail_check_point*. We recommend that Docker users modify this path and mount the directory where the checkpoint file resides to the host. Otherwise, duplicate collection occurs due to checkpoint data loss when the container is released. For example, you can set check_point_filename to `/data/logtail/check_point.dat` in Docker and add `-v /data/docker1/logtail:/data/logtail` to the Docker startup command. Then, the */data/docker1/logtail* directory of the host is mounted to the */data/logtail* directory of Docker.

> **Note**
>
> - The preceding table lists only the common startup parameters. If the *ilogtail_config.json* file contains parameters that are not listed in the table, the default settings are used for these parameters.
> - We recommend that you do not add unnecessary parameters to the *ilogtail_config.json* file.

Modify configurations

1. Configure the *ilogtail_config.json* file as needed.

Ensure that the modified configurations are in the valid JSON format.

2. Restart Logtail to apply the modified configurations.

```
/etc/init.d/ilogtaild stop
/etc/init.d/ilogtaild start
/etc/init.d/ilogtaild status
```

3.1.3. Logtail machine group

3.1.3.1. Overview

Log Service uses machine groups to manage the servers from which you want to collect logs by using Logtail.

A machine group is a virtual group that contains multiple servers. If you want to use a Logtail configuration file to collect logs from multiple servers, you can add the servers to a machine group. Then, you can apply the Logtail configuration file to the machine group.

To define a machine group, you can use one of the following methods:

- IP address: Add the IP addresses of all servers to a machine group. Each server can be identified by using its unique IP address.
- Custom ID: Use a custom ID to identify the machine group and use the same ID for servers in the machine group.

> **Note** Windows and Linux servers cannot be added to the same machine group.

IP address-based machine groups

You can add multiple servers to a machine group by adding their IP addresses to the machine group. Then, you can create a Logtail configuration file for all the servers at the same time.

- If you use ECS instances and have not associated them with hostnames or changed their network types, you can add their private IP addresses to the machine group.
- In other cases, you must add the server IP addresses obtained by Logtail to a machine group. The IP address of each server is recorded in the IP address field of the *app_info.json* file on the server.

> **Note** The *app_info.json* file records the internal information of Logtail. This file includes the server IP addresses obtained by Logtail. If you modify the IP address field of the file, the IP addresses obtained by Logtail remain unchanged.

Logtail obtains a server IP address by using the following methods:

- If the IP address of a server is associated with the hostname in the */etc/hosts* file of the server, Logtail obtains this IP address.
- If the IP address of a server is not associated with the hostname, Logtail obtains the IP address of the first network interface controller (NIC) on the server.

For more information, see Create an IP address-based server group.

Custom ID-based machine groups

You can use custom IDs to dynamically define machine groups.

An application system consists of multiple modules. You can scale out each module by adding multiple servers to the module. If you want to collect logs by module, you can create a machine group for each module. Therefore, you must specify a custom ID for each server in each module. For example, a website consists of an HTTP request processing module, a caching module, a logic processing module, and a storage module. The custom IDs of these modules can be http_module, cache_module, logic_module, and store_module.

For more information, see Create a machine group based on a custom ID.

3.1.3.2. Configure a user identifier

This topic describes how to specify the ID of an Apsara Stack tenant account as a user identifier on a server.

Prerequisites

- A server is available.

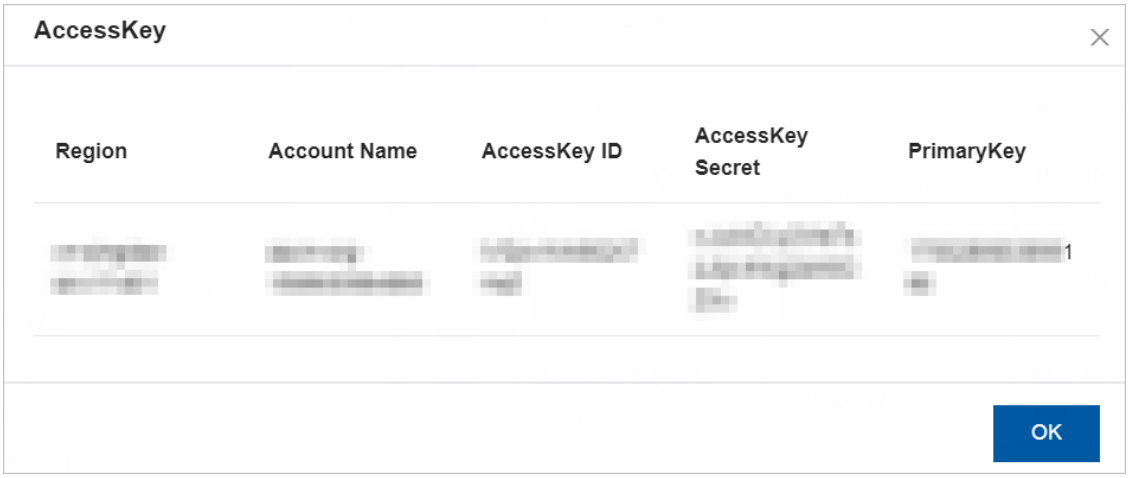
  The server can be an Elastic Compute Service (ECS) instance that belongs to another Apsara Stack tenant account, a server that is provided by a third-party cloud service provider, or a self-managed data center.
- Logtail is installed on the server from which you want to collect logs. For more information, see Install Logtail in Linux or Install Logtail in Windows.

Context

If your server is an ECS instance that belongs to another Apsara Stack tenant account, a server that is provided by a third-party cloud service provider, or a self-managed data center, you must specify the ID of your Apsara Stack tenant account as a user identifier on your server after Logtail is installed. Then, the Apsara Stack tenant account can use Logtail to collect logs from the server. If you do not configure a user identifier on your server, Log Service cannot receive the heartbeat of the server and Logtail cannot collect logs from the server.

Procedure

1. View the ID of your Apsara Stack tenant account.
   i. Log on to the Log Service console. For more information, see Log on to the Log Service console.
   ii. In the top navigation bar, click **Enterprise**.
   iii. In the left-side navigation pane, click **Organizations**.
   iv. Select the account that you want to view and click **Obtain an accesskey**
   v. In the **AccessKey** dialog box, view the ID of the Apsara Stack tenant account.



2. Log on to the server and specify the ID of the Apsara Stack tenant account as a user identifier.
   - Linux server:

     In the */etc/ilogtail/users* directory, create a file and set the name of the file to the ID of the Apsara Stack tenant account. Example:

     ```
     touch /etc/ilogtail/users/1**************
     touch /etc/ilogtail/users/1**************
     ```

   - Windows server:

In the *C:\LogtailData\users* directory, create a file and set the name of the file to the ID of the Apsara Stack tenant account. For example, you can set the file name to *C:\LogtailData\users\1*************.*

Note

- If the */etc/ilogtail/users* directory does not exist, you must create the directory.
- You can configure the IDs of multiple Apsara Stack tenant accounts on the same server.
- After you configure or delete a user identifier, the change takes effect within 1 minute.
- If you no longer need a user identity file on a server, we recommend that you delete the file from the server at the earliest opportunity.

3.1.3.3. Create an IP address-based machine group

Log Service allows you to create an IP address-based machine group. This topic describes how to create an IP address-based machine group in the Log Service console.

Prerequisites

- A project and a Logstore are created.
- At least one server is available.
  - If you use an Elastic Compute Service (ECS) instance, you must make sure that the ECS instance and Log Service project belong to the same Apsara Stack tenant account and reside in the same region.
  - If the ECS instance belongs to another Apsara Stack tenant account, you must configure a user identifier for the ECS instance before you create an IP address-based machine group. If the server is provided by a third-party cloud service provider or is deployed in a self-managed data center, you must also configure a user identifier. For more information, see Configure an account ID on a server.
- Logtail is installed on the server. For more information, see Install Logtail in Linux and Install Logtail in Windows.

Procedure

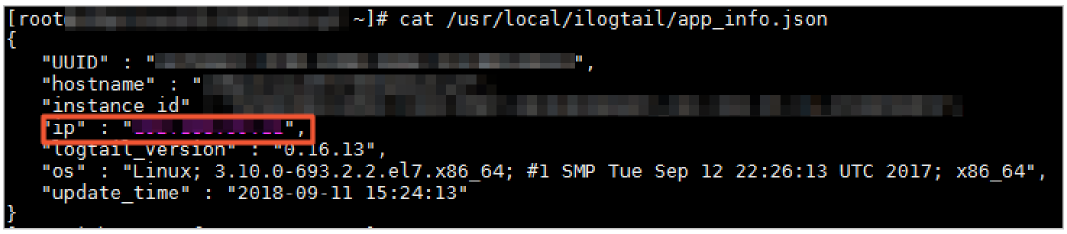
1. Obtain the IP addresses of the servers.

   The IP addresses that are obtained by Logtail are recorded in the ip field of the *app_info.json* file.

   On the servers where Logtail is installed, you can go to the following path to check the *app_info.json* file:

   - Linux: */usr/local/ilogtail/app_info.json*
   - 64-bit Windows: *C:\Program Files (x86)\Alibaba\Logtail\app_info.json*
   - 32-bit Windows: *C:\Program Files\Alibaba\Logtail\app_info.json*

   The following figure shows the IP address of the server in Linux.



2. Log on to the Log Service console.
3. In the Projects section, click the project in which you want to create a machine group.
4. In the left-side navigation pane, click the **Machine Groups** icon.
5. Click the icon next to Machine Groups and select **Create Machine Group**.

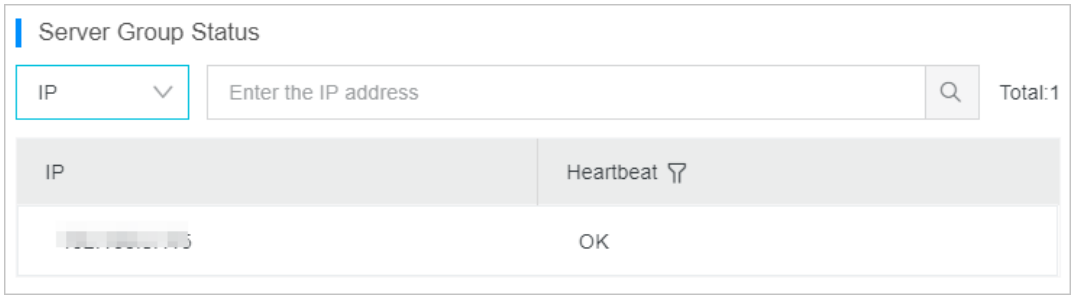
   You can also create a machine group in the Logtail configuration wizard.

6. In the **Create Machine Group** panel, set the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Name	The name of the machine group. The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Notice** After the machine group is created, you cannot modify its name. Proceed with caution.
Identifier	Select **IP Addresses**.
Topic	The topic of the machine group. This topic is used to differentiate log data that is generated on different servers. For more information, see Log topic.
IP address	Enter the IP addresses that are obtained in Step 1. **Note** - If a machine group contains multiple servers, you must separate the IP addresses with line feeds. - You cannot add Windows and Linux servers to the same machine group.

7. View the status of the machine group.
   i. In the Machine Groups list, click the machine group that you create.
   ii. On the **Machine Group Settings** page, view the status of the machine group.

   If the **Heartbeat** status is **OK**, the server is connected to Logtail. If the status is **FAIL**, click **Automatic Retry**.



Result

You can view the created machine group in the Machine Groups list.

3.1.3.4. Create a custom ID-based machine group

Log Service allows you to create a custom ID-based machine group. This topic describes how to create a custom ID-based machine group in the Log Service console.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- At least one server is available.
  - If you use an Elastic Compute Service (ECS) instance, you must make sure that the ECS instance and Log Service project belong to the same Apsara Stack tenant account and reside in the same region. If the ECS instance belongs to another Apsara Stack tenant account, you must configure a user identifier for the ECS instance before you create a custom ID-based machine group. If the server is provided by a third-party cloud service provider or is deployed in a self-managed data center, you must also configure a user identifier. For more information, see Configure an account ID on a server.
  - Logtail is installed on the server. For more information, see Install Logtail on ECS instances.

Context

Custom ID-based machine groups offer significant benefits in the following scenarios:

- If your servers reside in multiple custom network environments such as virtual private clouds (VPCs), some IP addresses of the servers may conflict. In this case, Logtail cannot collect logs as expected. You can use a custom ID to prevent this issue.
- If you want to add multiple servers to a machine group, you can set the same custom ID for new servers as the machine group. Log Service identifies the custom ID and adds the servers with the same custom ID to the machine group.

Procedure

1. Create a file named *user_defined_id* in a specified directory.
   - Linux: Store the file in the */etc/ilogtail/user_defined_id* directory.
   - Windows: Store the file in the *C:\LogtailData\user_defined_ided_id* directory.
2. Set a custom ID for the server.

   > **Note**
   > - You cannot add Windows and Linux servers to the same machine group. You cannot set the same custom ID for Linux and Windows servers.
   > - You can set one or more custom IDs for a single server and separate custom IDs with line feeds.
   > - In the Linux server, if the */etc/ilogtail/* directory or the */etc/ilogtail/user_defined_id* file does not exist, you can create the directory and file. In the Windows server, if the *C:\LogtailData* directory or the *C:\LogtailData\user_defined_id* file does not exist, you can also create the directory and file.

   - Linux:

     Set the custom ID in the */etc/ilogtail/user_defined_id* file. For example, if you want to set the custom ID to `userdefined`, run the following command to edit the file. In the file, enter `userdefined`.

     ```
     vim /etc/ilogtail/user_defined_id
     ```

   - Windows:

     Set the custom ID in the *C:\LogtailData\user_defined_id* file. For example, if you want to set the custom ID to `userdefined_windows`, enter `userdefined_windows` in the *C:\LogtailData\user_defined_id* file.
3. Create a machine group.
   i. Log on to the Log Service console.
   ii. In the Projects section, click the name of the project in which you want to create a machine group.
   iii. In the left-side navigation pane, click the **Machine Groups** icon.

iv. Click the icon next to **Machine Groups**, and then select **Create Machine Group**.

v. In the Create Machine Group panel, set the parameters. The following table describes the parameters.

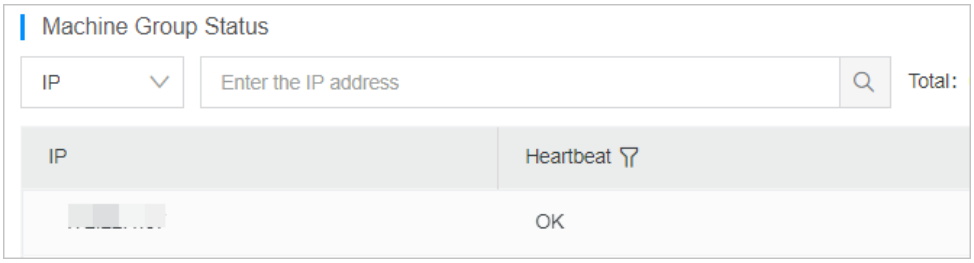
Parameter	Description
Name	The name of the machine group. The name must be 2 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Notice** After the machine group is created, you cannot modify its name. Proceed with caution.
Identifier	The identifier of the server. Select **Custom ID**.
Topic	The topic of the machine group. This topic is used to differentiate log data that is generated in different servers. For more information, see Log topic.
Custom Identifier	Enter the custom ID that is set in .

vi. Click **OK**.

Note If you need to add a server to the machine group, set a custom ID of the server to the custom ID of the machine group.

4. In the Machine Groups list, click the name of the machine group to view the status of the machine group.

   On the **Machine Group Settings** page, you can view the IP address list of the servers in the machine group and the heartbeat status in the **Machine Group Status** section.



   - The Machine Group Status section lists the IP addresses of the servers whose custom ID is the same as the custom ID that you set for the machine group.

     For example, the custom ID of a machine group is userdefined and the IP addresses in the Machine Group Status section are 10.10.10.10, 10.10.10.11, and 10.10.10.12. This indicates that you specified the same custom ID for the servers in this machine group. If you want to add another server to the machine group and the IP address of the server is 10.10.10.13, set the custom ID to userdefined for the server. Then, you can view the IP address of the server that you added in the Machine Group Status section.

   - If the Heartbeat status is OK, the server is connected to Log Service. If the status is FAIL, see What can I do if no heartbeat packet is received from a Logtail client?.

Disable a custom ID

If you want to set the Identifier parameter to IP Addresses, delete the user_defined_id file. The new configurations take effect within 1 minute.

- Linux:

```
rm -f /etc/ilogtail/user_defined_id
```

- In Windows, run the following command:

```
del C:\LogtailData\user_defined_id
```

Time to take effect

By default, after you add, delete, or modify the user_defined_id file, the new configurations take effect within 1 minute. If you want the configurations to immediately take effect, run the following command to restart Logtail.

- Linux:

```
/etc/init.d/ilogtaild stop
/etc/init.d/ilogtaild start
```

- Windows:
  i. Choose **Start Menu > Control Panel > Administrative Tools > Services**.
  ii. In the **Services** window, select the required service.
    - For Logtail V0.x.x.x, select LogtailWorker.
    - For Logtail V1.0.0.0 or later, select LogtailDaemon.
  iii. Right-click the service and then select **Restart** to validate the configurations.

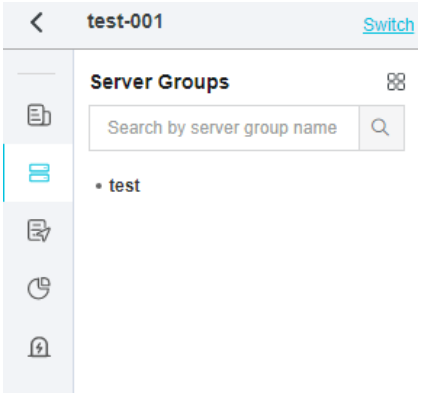
3.1.3.5. View server groups

This topic describes how to view the server groups of a project on the **Server Groups** page in the Log Service console.

Procedure

1. Log on to the Log Service console.
2. Find the target project in the project list and click the project name.
3. In the left-side navigation pane of the page that appears, click the **Server Groups** icon to display the list of server groups.

   You can view all server groups of the project.



3.1.3.6. Modify a server group

This topic describes how to modify a server group in the Log Service console. After you create a server group, you can modify the parameters of the server group.

Procedure

1. Log on to the Log Service console.
2. Find the target project in the project list and click the project name.

3. In the left-side navigation pane of the page that appears, click the **Server Groups** icon to display the list of server groups.
4. Click the name of the server group to be modified. On the **Server Group Settings** page, click **Modify**.

   > **Note** The name of the server group cannot be modified.

5. Modify the parameters of the server group, and then click **Save**.

3.1.3.7. View the status of a server group

This topic describes how to view the status of a server group in the Log Service console. You can view the heartbeat information of Logtail to check whether Logtail is installed on the servers in a server group.

Procedure

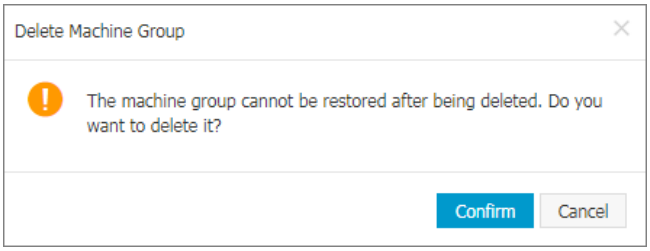
1. Log on to the Log Service console.
2. Find the target project in the project list and click the project name.
3. In the left-side navigation pane of the page that appears, click the **Server Groups** icon to display the list of server groups.
4. Click the name of the server group. On the **Server Group Settings** page, check the server group status.
   - If the heartbeat is OK, Logtail is installed on the servers in the server group and Logtail is connected to Log Service.
   - If the heartbeat status is FAIL, Logtail fails to connect to Log Service. If the FAIL state persists, perform troubleshooting based on the instructions provided in What can I do if no heartbeat packet is received from a Logtail client?

3.1.3.8. Delete a machine group

This topic describes how to delete a machine group in the Log Service console. You can delete a machine group if you no longer need to collect logs from the machine group.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to delete a machine group.
3. In the left-side navigation pane, click the icon. The Machine Groups list is displayed.
4. In the Machine Groups list, find the machine group that you want to delete, click the icon next to the machine group, and then select **Delete**.
5. In the message that appears, click **OK**.



3.1.3.9. Manage a Logtail configuration

This topic describes how to create, view, modify, and delete a Logtail configuration in the Log Service console.

View a list of Logtail configurations

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to view Logtail configurations.
3. Choose **Log Storage > Logstores**. On the Logstores tab, Click the > icon of the Logstore in which you want to view Logtail configurations. Then, choose **Data Import > Logtail Configurations**.
4. Click the Logtail configuration that you want to view.
5. On the **Logtail Config** page, view the details of the Logtail configuration.

Create a Logtail configuration

You can create a Logtail configuration in the Log Service console. For more information, see Configure text log collection.

Modify a Logtail configuration

Click the name of the Logtail configuration that you want to modify. On the **Logtail Config** page, click **Modify**.

You can also change the log collection mode of the Logtail configuration, and then apply the Logtail configuration to the related machine group again. The procedure to modify a Logtail configuration is the same as the procedure to create a Logtail configuration.

Delete a Logtail configuration

In the **Logtail Configurations** list, find the Logtail configuration that you want to delete, click the icon next to the Logtail configuration, and then select **Delete**.

Warning After you delete a Logtail configuration, the Logtail configuration is disassociated from the related machine group. Logtail no longer collects the logs that are specified by the Logtail configuration. Proceed with caution.

3.1.4. Collect text logs

3.1.4.1. Configure text log collection

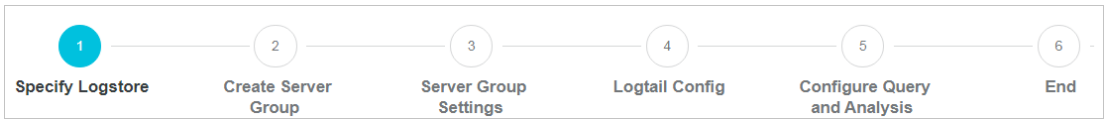
This topic describes the configuration process and collection modes when you use Logtail to collect text logs from servers.

Prerequisites

A project and a Logstore are created. For more information, see Create a project and Create a Logstore.

Configuration process

Log Service provides a configuration wizard that you can use to configure log collection.



Collection modes

Logtail supports various collection modes, such as simple mode, full regex mode, delimiter mode, JSON mode, NGINX configuration mode, IIS configuration mode, and Apache configuration mode.

- Collect logs in simple mode
- Collect logs in full regex mode

- Collect logs in delimiter mode
- Collect logs in JSON mode
- Collect logs in NGINX mode
- Collect logs in IIS mode
- Collect logs in Apache mode

Procedure

1. Log on to the Log Service console.
2. Select a data source.

   Select a data source based on your business requirements. Log Service supports the following data sources of text logs: **RegEx - Text Log**, **Single Line - Text Log**, **Multi-Line - Text Log**, **Delimiter Mode - Text Log**, **JSON - Text Log**, **Nginx - Text Log**, **IIS - Text Log**, and **Apache - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

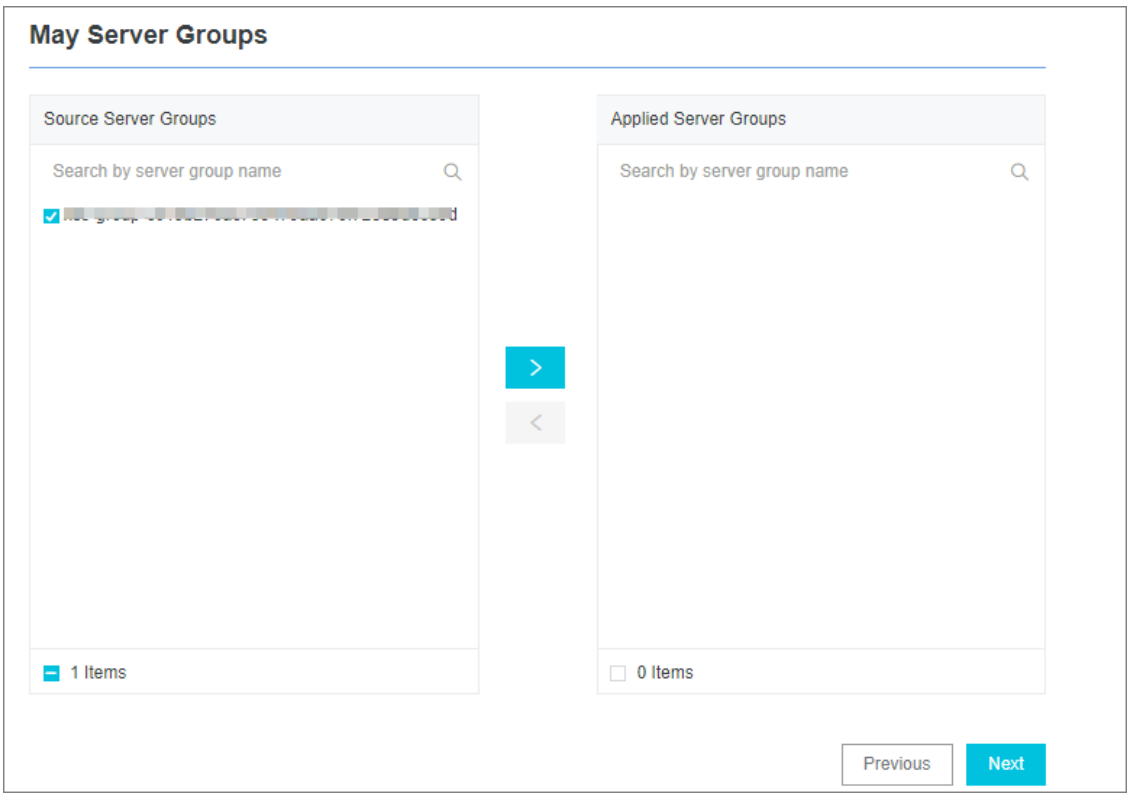
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration and click **Next**.

   Logtail parameters vary based on collection modes. For more information, see the related parameters for specific collection methods in Collection modes.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be unique in a project. After the Logtail configuration is created, you cannot change the name of the Logtail configuration. You can also click **Import Other Configuration** to import a Logtail configuration from another project.
Log Path	The directory and name of the log file. The specified log file name can be a complete file name or a file name that contains wildcards. Log Service scans all levels of the specified directory to match log files. Examples: ◦ If you specify */apsara/nuwa/.../*.log*, Log Service matches the files whose name is suffixed by *.log* in the */apsara/nuwa* directory and its recursive subdirectories. ◦ If you specify */var/logs/app_*/*.log*, Log Service matches the files that meet the following conditions: The file name contains *.log*. The file is stored in a subdirectory of the */var/logs* directory or in a recursive subdirectory of the subdirectory. The name of the subdirectory matches the *app_**pattern. **Note** ◦ By default, each log file can be collected by using only one Logtail configuration. ◦ To use multiple Logtail configurations to collect one log file, we recommend that you create a symbolic link that points to the directory where the file is located. For example, you want to collect two copies of the log.log file from the */home/log/nginx/log/log.log* directory. You can run the following command to create a symbolic link that points to the directory. When you configure the Logtail configurations, use the original path in one Logtail configuration and use the symbolic link in the other Logtail configuration. `ln -s /home/log/nginx/log /home/log/nginx/link_log` ◦ You can use only asterisks (*) and question marks (?) as wildcards in the log path.
Blacklist	If you turn on **Blacklist**, you can configure a blacklist to skip the specified directories or files when Logtail collects logs. You can use exact match or wildcard match to specify directories and files. Examples: ◦ If you select **Filter by Directory** from the Filter Type drop-down list and enter */tmp/mydir* in the Content column, all files in the directory are skipped. ◦ If you select **Filter by File** from the Filter Type drop-down list and enter */tmp/mydir/file* in the Content column, only the specified file is skipped.

Parameter	Description
Docker File	If you want to collect logs from Docker containers, you can turn on **Docker File** and specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs. For more information about container text logs, see Collect Kubernetes logs.
Mode	The default mode is **Simple Mode - Multi-line**. You can change the mode.

Log Sample	Enter a sample log that is retrieved from a log source in an actual scenario. Then, Log Service can automatically generate a regular expression to match the start part in the first line of the log. Example:
	`[2020-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened`
	`at TestPrintStackTrace.f(TestPrintStackTrace.java:3)`
	`at TestPrintStackTrace.g(TestPrintStackTrace.java:7)`
	`at TestPrintStackTrace.main(TestPrintStackTrace.java:16)`
	If you collect single-line text logs in simple mode, you do not need to set this parameter.

Regex to Match First Line	The regular expression that Logtail uses to match the start part in the first line of a log. The unmatched lines are collected as part of a log. You can specify a regular expression to match the start part in the first line of a log. You can also use the regular expression that is automatically generated by Log Service. ◦ Automatically generate a regular expression to match the start part in the first line of a log. After you enter a sample log, click **Auto Generate**. Log Service automatically generates a regular expression to match the start part in the first line of the log. ◦ Specify a regular expression to match the start part in the first line of a log. After you enter a sample log, click **Manual** and specify a regular expression to match the start part in the first line of the log. Then, click **Validate** to check whether the regular expression is valid. If you collect single-line text logs in simple mode, you do not need to set this parameter.
Drop Failed to Parse Logs	Specifies whether to drop logs that fail to be parsed. ◦ If you turn on **Drop Failed to Parse Logs**, logs that fail to be parsed are not uploaded to Log Service. ◦ If you turn off **Drop Failed to Parse Logs**, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum depth at which the log directory is monitored. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ◦ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ◦ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ◦ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ◦ utf8: UTF-8 encoding format ◦ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ◦ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ◦ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ◦ Never: All log files are continuously monitored and never time out. ◦ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.

Parameter	Description
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ◦ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ◦ Filter out logs that do not meet specified conditions. ▪ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ▪ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ◦ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ◦ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs.

3.1.4.2. Collect logs in simple mode

When you collect logs in simple mode, the logs are not parsed. Each log is collected and uploaded to Log Service as a whole. This simplifies the process of log collection. This topic describes how to create a Logtail configuration in simple mode in the Log Service console to collect logs.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- Ports 80 and 443 are enabled for the server from which you want to collect logs.

Context

The simple mode supports the following types of text logs:

- Single-line text log

  Each log line is collected as a log. Two logs in a log file are separated by a line feed. In single-line mode, you must specify the directories and names of log files. Then, Logtail collects logs by line from the specified files.

- Multi-line text log

  Multiple log lines are collected as a log by default. In multi-line mode, you must specify the directories and names of log files. In addition, you must enter a sample log and configure a regular expression to match the start part in the first line of a log. Logtail uses the regular expression to match the start part in the first line of a log and reckons unmatched lines as part of the log.

Note If you collect logs in simple mode, the timestamp of a log indicates the system time of the server when the log is collected.

Procedure

1. Log on to the Log Service console.
2. Select a data source.

   Select **Single Line - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

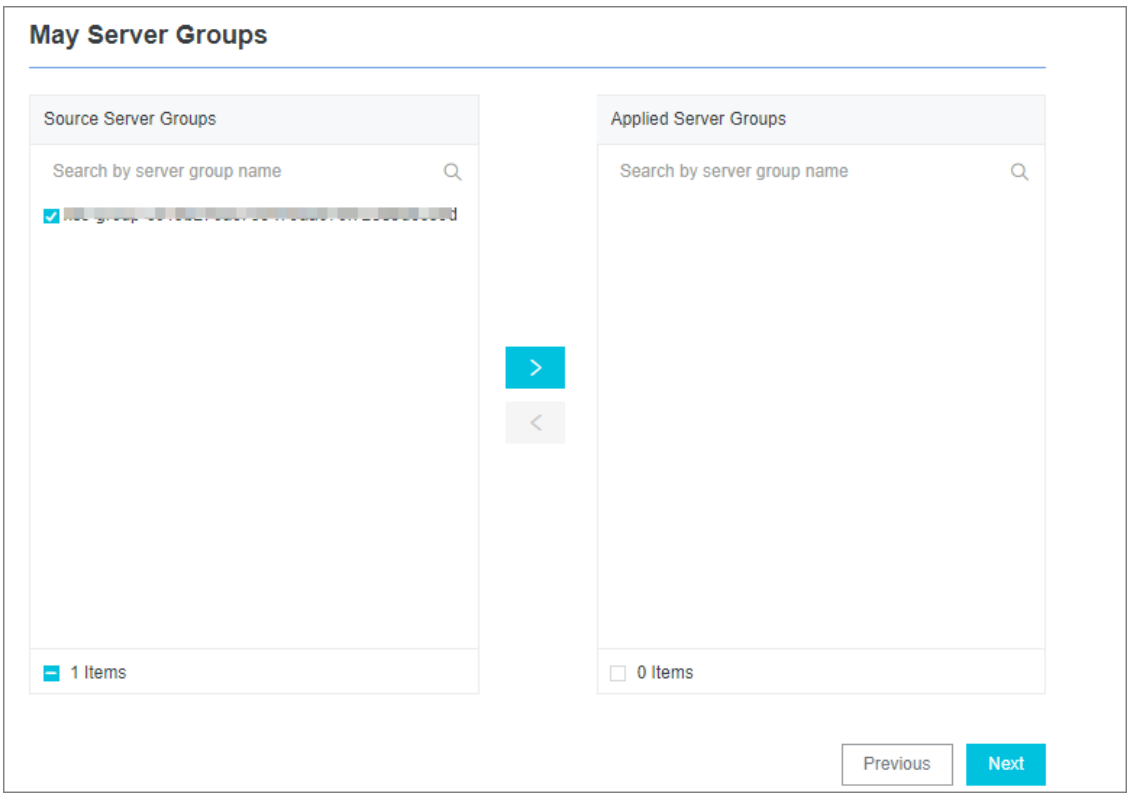
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



Notice If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be FAIL. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Note** After the Logtail configuration is created, you cannot change the name of the Logtail configuration.
Log Path	The directory and name of the log file. ○ The specified log file name can be a complete file name or a file name that contains wildcards. ○ Recursive directory matching is used in the log file search. If this matching method is applied, all files that match the specified file name in the specified directory and its sub-directories are monitored. ■ If you specify `/apsara/nuwa/ ** /*.log`, Log Service matches the files whose name is suffixed by `.log` in the `/apsara/nuwa` directory and its recursive subdirectories. ■ If you specify `/var/logs/app_* … /*.log*`, Log Service matches the files that meet the following conditions: The file name contains `.log`. The file is stored in a subdirectory of the `/var/logs` directory or in a recursive subdirectory of the subdirectory. The name of the subdirectory matches the `app_*` pattern. **Note** ■ By default, each log file can be collected by using only one Logtail configuration. ■ You can use only asterisks (`*`) and question marks (`?`) as wildcards in the log path.
Docker File	If you want to collect logs from Docker containers, you can specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs.
Mode	If you have specified **Single Line - Text Log** for the data source, the default mode is **Simple Mode**. You can change the mode.
Maximum Directory Monitoring Depth	The maximum number of directory layers that can be recursively monitored when logs are collected from the data source. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ○ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ○ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ○ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ○ utf8: UTF-8 encoding format ○ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ○ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ○ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ○ Never: All log files are continuously monitored and never time out. ○ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.

Parameter	Description
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ○ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ○ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in full regex mode.

3.1.4.3. Collect logs in full regex mode

You can use the full regex mode to extract custom fields from logs. This topic describes how to create a Logtail configuration in full regex mode in the Log Service console to collect logs.

Context

If you want to collect multi-line logs and extract fields from the logs, we recommend that you use regular expressions. Log Service can generate a regular expression based on a sample log that you specify in the Import Data wizard. However, you must modify a regular expression before it can match fields in the sample log as expected. For more information, see How do I test a regular expression?.

Procedure

1. Log on to the Log Service console.
2. Select a data source.

   Select **RegEx - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

   You can also click **Create Now** to create a project and a Logstore.

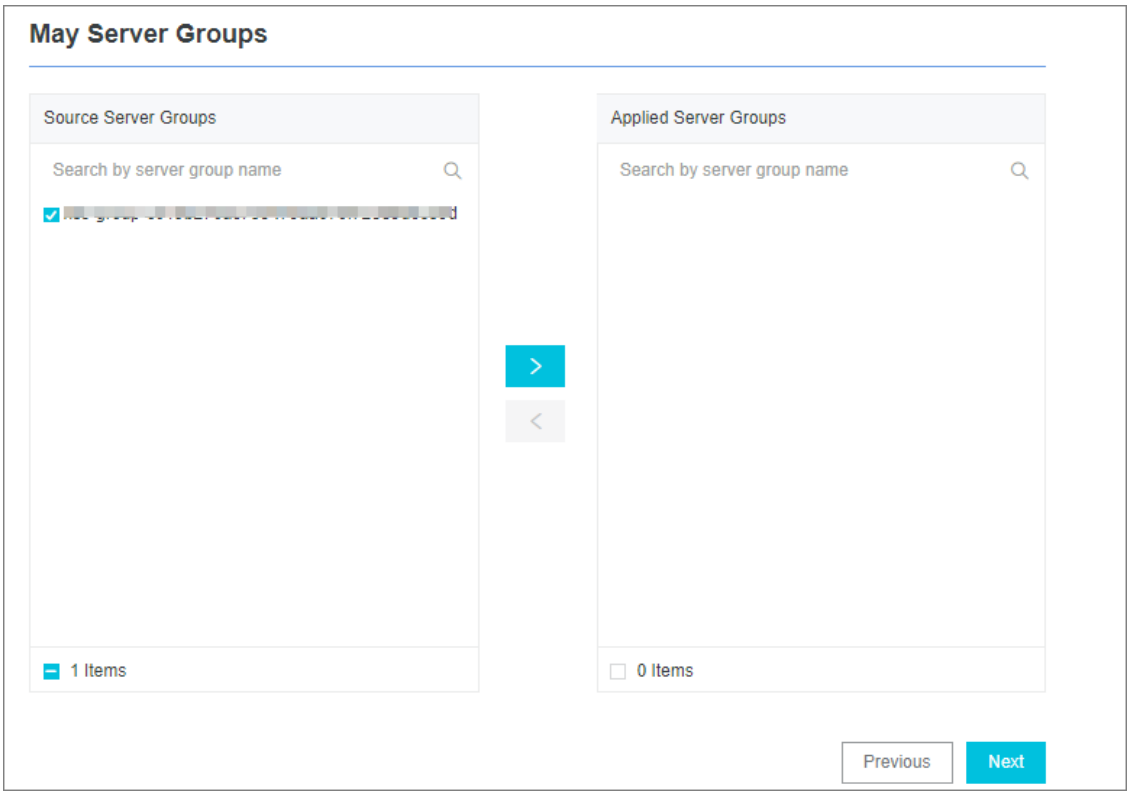
   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.

5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be unique in a project. After the Logtail configuration is created, you cannot change the name of the Logtail configuration. You can also click **Import Other Configuration** to import a Logtail configuration from another project.

Parameter	Description
Log Path	The directory and name of the log file. ◦ The specified log file name can be a complete file name or a file name that contains wildcards. ◦ Log Services scans all levels of the specified directory to match log files. Examples: ■ If you specify */apsara/nuwa/**/*.log*, Log Service matches the files whose name is suffixed by *.log* in the */apsara/nuwa* directory and its recursive subdirectories. ■ If you specify */var/logs/app_*/*.log*, Log Service matches the files that meet the following conditions: The file name contains *.log*. The file is stored in a subdirectory of the */var/logs* directory or in a recursive subdirectory of the subdirectory. The name of the subdirectory matches the *app_** pattern. **Note** ■ By default, each log file can be collected by using only one Logtail configuration. ■ To use multiple Logtail configurations to collect one log file, we recommend that you create a symbolic link that points to the directory where the file is located. For example, you want to collect two copies of the log.log file from the */home/log/nginx/log/log.log* directory. You can run the following command to create a symbolic link that points to the directory. When you configure the Logtail configurations, use the original path in one Logtail configuration and use the symbolic link in the other Logtail configuration. `ln -s /home/log/nginx/log /home/log/nginx/link_log` ■ You can use only asterisks (*) and question marks (?) as wildcards in the log path.
Docker File	If you collect logs from Docker containers, you can configure the paths and tags of the containers. Logtail monitors the containers when they are created and destroyed, filters the logs of the containers by tag, and collects the filtered logs.
Mode	If you have specified **RegEx - Text Log** for the data source, the default mode is **Full Regex Mode**. You can change the mode.
Singleline	The singleline mode is enabled by default. In this mode, logs are separated by line. To collect multi-line logs, such as Java program logs, you must disable the **Singleline** mode and configure **Regex to Match First Line**.
Log Sample	Enter a sample log that is retrieved from a log source in an actual scenario. Then, Log Service can automatically generate a regular expression.
Regex to Match First Line	You can click **Auto Generate** or **Manual**. After you enter a sample log and click **Auto Generate**, Log Service automatically generates a regular expression. If no regular expression is generated, you can switch to the manual mode and enter a regular expression for verification.

Parameter	Description
Extract Field	To analyze and process specific fields in logs, you can turn on **Extract Field**. Then, the specified fields are converted to key-value pairs and sent to Log Service. You must specify a regular expression to parse the log content.
Regular Expression	If you turn on Extract Field, you must specify this parameter. ○ Automatically generate a regular expression. You can select the fields to be extracted from the sample log and then click Generate Regular Expression. Log Service automatically generates a regular expression. ○ Specify a regular expression. You can also enter a regular expression. Click **Manual** to switch to the manual mode. After you enter a regular expression, click **Validate** to check whether Log Service can parse the log content by using the regular expression. For more information, see How do I test a regular expression?.
Extracted Content	If you turn on Extract Field, you must specify this parameter. After a regular expression is automatically generated or manually specified, you must specify the key name for each extracted field.
Use System Time	If you turn on Extract Field, you must specify this parameter. If you turn off Use System Time, you must specify a field as the time field and name the field `time`. After you specify the `time` field, click **Auto Generate** in the **Time Conversion Format** field to parse the time. For more information, see Configure the time format.
Drop Failed to Parse Logs	Specifies whether to upload logs to Log Service if the logs fail to be parsed. ○ If you turn on this switch, logs that fail to be parsed are not uploaded to Log Service. ○ If you turn off this switch, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum number of directory levels that can be recursively monitored when Log Service collects logs from the data source. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.

Parameter	Description
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ○ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ○ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ○ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ○ utf8: UTF-8 encoding format ○ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ○ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ○ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ○ Never: All log files are continuously monitored and never time out. ○ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

> **Note**
>
> - To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
> - If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in full regex mode.

3.1.4.4. Collect logs in delimiter mode

Log Service allows you to collect logs in delimiter mode. After logs are collected, you can transform and ship the logs, and perform multidimensional log analysis. This topic describes how to create a Logtail configuration in delimiter mode in the Log Service console to collect logs.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- Ports 80 and 443 are enabled for the server from which you want to collect logs.

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, select **Delimiter Mode - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

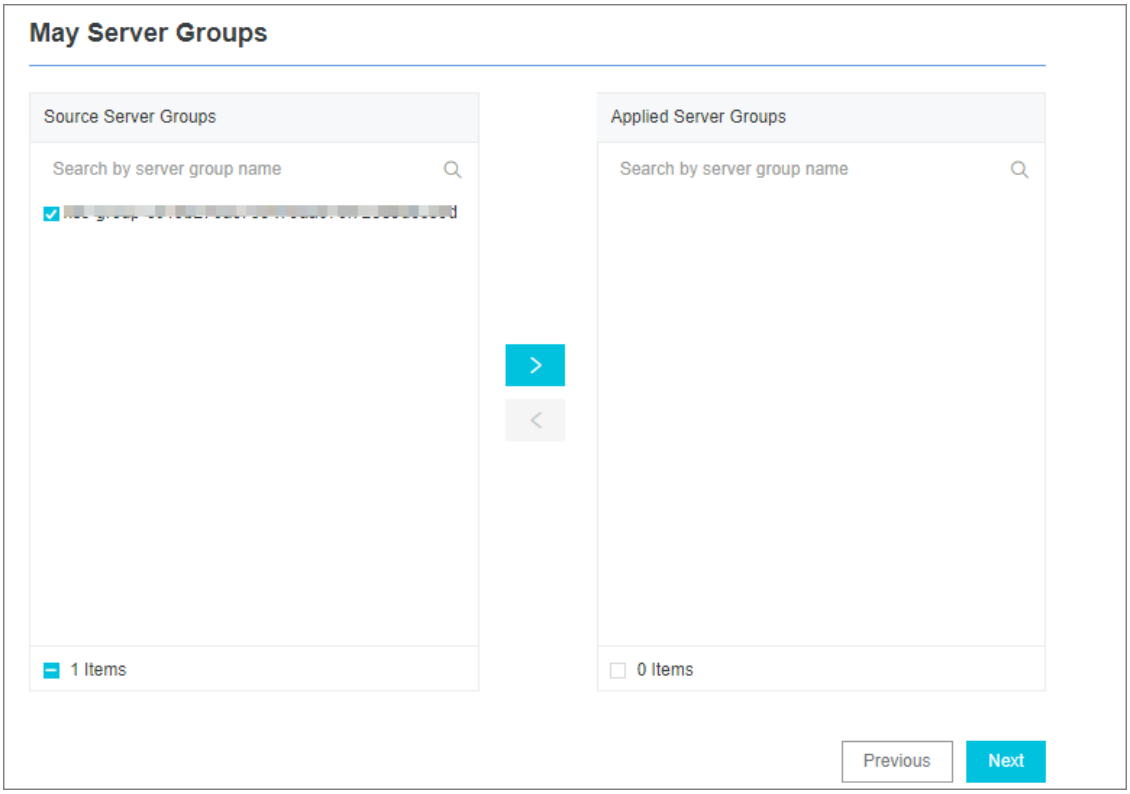
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



Notice If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration and click **Next**.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Note** After the Logtail configuration is created, you cannot change the name of the Logtail configuration.
Log Path	The directory and name of the log file. The specified log file name can be a complete file name or a file name that contains wildcards. For more information, see Wildcard matching. Log Services scans all levels of the specified directory to match log files. Examples:
Docker File	If you want to collect logs from Docker containers, you can specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs.
Mode	The default mode is **Delimiter Mode**. You can change the mode.

Parameter	Description
Log Sample	Enter a sample log that is retrieved from a log source in an actual scenario. Example: `127.0.0.1|#|-|#|13/Apr/2020:09:44:41 +0800|#|GET /1 HTTP/1.1|#|0.000|#|74|#|404|#|3650|#|-|#|curl/7.29.0` **Note** The delimiter mode applies only to single-line logs. If you want to collect multi-line logs, we recommend that you select Simple Mode - Multi-line or Full Regex Mode.
Delimiter	Select a delimiter based on the log format. For example, if you select Vertical Line, a vertical bar (|) is used as the delimiter. For more information, see Appendix: delimiters and sample logs. **Note** If you set the Delimiter parameter to **Hidden Characters**, you must enter a character in the following format: `0x`*Hexadecimal ASCII code of the non-printable character*. For example, if you want to use the non-printable character whose hexadecimal ASCII code is 01, you must enter **0x01**.
Quote	If a log field contains delimiters, you must specify a pair of quotes to enclose the field. Log Service parses the content that is enclosed in a pair of quotes into a complete field. Select a quote based on the log format. **Note** If you set the Quote parameter to **Hidden Characters**, you must enter a character in the following format: `0x`*Hexadecimal ASCII code of the non-printable character*. For example, if you want to use the non-printable character whose hexadecimal ASCII code is 01, you must enter **0x01**.
Extracted Content	Log Service extracts the log content based on the sample log and delimiter that you specify. The extracted log content is delimited into values. You must specify a key for each value.
Incomplete Entry Upload	Specifies whether to upload a log whose number of parsed fields is less than the number of the specified keys. If you turn on this switch, the log is uploaded. If you turn off this switch, the log is dropped. For example, if you specify a vertical bar (|) as the delimiter, the log 11|22|33|44|55 is parsed into the following fields: 11, 22, 33, 44, and 55. You can set the keys to A, B, C, D, and E. ◦ If you turn on **Incomplete Entry Upload**, 55 is uploaded as the value of the D key when Log Service collects the log 11|22|33|55. ◦ If you turn off **Incomplete Entry Upload**, the log 11|22|33|55 is dropped because the number of fields parsed from the log does not match the number of the specified keys.

Parameter	Description
Use System Time	Specifies whether to use the system time. ◦ If you turn on **Use System Time**, the timestamp of a log indicates the system time of the server when the log is collected. ◦ If you turn off **Use System Time**, you must set the **Specify Time Key** and **Time Format** parameters based on the value of the time field that is specified in **Extracted Content**. For example, if you set the **Specify Time Key** parameter to **time_local** and the **Time Format** parameter to **%d/%b/%Y:%H:%M:%S**, the timestamp of a log is the value of the **time_local** field.
Drop Failed to Parse Logs	Specifies whether to upload logs to Log Service if the logs fail to be parsed. ◦ If you turn on **Drop Failed to Parse Logs**, logs that fail to be parsed are not uploaded to Log Service. ◦ If you turn off **Drop Failed to Parse Logs**, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum depth at which the log directory is monitored. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ◦ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ◦ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ◦ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.

Parameter	Description
Log File Encoding	The encoding format of log files. Valid values: ○ utf8: UTF-8 encoding format ○ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ○ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ○ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ○ Never: All log files are continuously monitored and never time out. ○ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ○ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ○ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in delimiter mode.

Appendix: delimiters and sample logs

Logs that are in the delimiter-separated values (DSV) format use line feeds as boundaries. Each line indicates a log. Each log is parsed into multiple fields by using delimiters. Both single-character and multi-character delimiters are supported. If a field contains delimiters, you can enclose the field in a pair of quotes.

- Single-character delimiter

The following example shows sample logs with single-character delimiters:

```
05/May/2016:13:30:28,10.10.*.*,"POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=**********
**&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=**********************
******* HTTP/1.1",200,18204,aliyun-sdk-java
05/May/2016:13:31:23,10.10.*.*,"POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=**********
**&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=**********************
******* HTTP/1.1",401,23472,aliyun-sdk-java
```

If a log contains single-character delimiters, you must specify the delimiter. You can also specify a quote.

- Delimiter: Available single-character delimiters include the tab character (\t), vertical bar (|), space character, comma (,), and semicolon (;). You can also specify a non-printable character as the delimiter. You cannot specify a double quotation mark (") as the delimiter.

  However, a double quotation mark (") can be used as a quote. You can place the double quotation mark at the border of a field, or in the field. If a double quotation mark (") is included in a log field but is not used as a quote, it must be escaped as double quotation marks (""). When Log Service parses log fields, the double quotation marks ("") are automatically converted to a double quotation mark ("). For example, you can specify a comma (,) as the delimiter and a double quotation mark (") as the quote in a log field. You must enclose the field that contains commas (,) in a pair of quotes. In addition, you must escape the double quotation mark (") in the field to double quotation marks (""). If a processed log is in the 1999,Chevy,"Venture ""Extended Edition, Very Large""","",5000.00 format, the log can be parsed into the following five fields: 1999, Chevy, Venture "Extended Edition, Very Large", an empty field, and 5000.00.

- Quote: If a log field contains delimiters, you must specify a pair of quotes to enclose the field. Log Service parses the content that is enclosed in a pair of quotes into a new complete field.

  Available quotes include the tab character (\t), vertical bar (|), space character, comma (,), semicolon (;), and non-printable characters.

  For example, if you specify a comma (,) as the delimiter and a double quotation mark (") as the quote, the log 1997,Ford,E350,"ac, abs, moon",3000.00 is parsed into the following five fields: 1997, Ford, E350, ac, abs, moon, and 3000.00.

- Multi-character delimiter

  The following example shows sample logs with multi-character delimiters:

```
05/May/2016:13:30:28&&10.200.**.**&&POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=**********
******&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=pD12XYLmGxKQ%2Bmkd6x7
hAgQ7b1c%3D HTTP/1.1&&200&&18204&&aliyun-sdk-java
05/May/2016:13:31:23&&10.200.**.**&&POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=**********
******&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=**********************
********** HTTP/1.1&&401&&23472&&aliyun-sdk-java
```

  A multi-character delimiter can contain two or three characters, such as ||, &&&, and ^_^. Log Service parses logs based on delimiters. You do not need to use quotes to enclose log fields.

  > **Note** You must make sure that the delimiters in a field cannot be parsed into a new field. Otherwise, Log Service cannot parse the fields as expected.

  For example, if you specify && as the delimiter, the log 1997&&Ford&&E350&&ac&abs&moon&&3000.00 is parsed into the following five fields: 1997, Ford, E350, ac&abs&moon, and 3000.00.

3.1.4.5. Collect logs in JSON mode

Log Service allows you to collect JSON logs in JSON mode by using Logtail. After logs are collected, you can transform and ship the logs, and perform multidimensional log analysis. This topic describes how to create a Logtail configuration in JSON mode in the Log Service console to collect logs.

Context

JSON logs can be written in the object or array structure. A log in the object structure contains key-value pairs, and a log in the array structure contains an ordered list of values.

In JSON mode, Logtail can parse JSON logs in the object structure and extract the keys and values from the first layer of each object. The extracted keys are used as field names, and the extracted values are used as field values. Logtail cannot parse JSON logs in the array structure. If you want to parse JSON logs in the array structure, you can collect data from the JSON logs in full regex or simple mode. For more information, see Collect logs by line or Use regular expressions to collect logs.

Sample JSON logs:

```
{"url": "POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=U0Ujpek********&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=pD12XYLmGxKQ%2Bmkd6x7hAgQ7b1c%3D HTTP/1.1", "ip": "10.200.98.220", "user-agent": "aliyun-sdk-java", "request": {"status": "200", "latency": "18204"}, "time": "05/Jan/2020:13:30:28"}
{"url": "POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=U0Ujpek********&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=pD12XYLmGxKQ%2Bmkd6x7hAgQ7b1c%3D HTTP/1.1", "ip": "10.200.98.210", "user-agent": "aliyun-sdk-java", "request": {"status": "200", "latency": "10204"}, "time": "05/Jan/2020:13:30:29"}
```

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, select **JSON - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

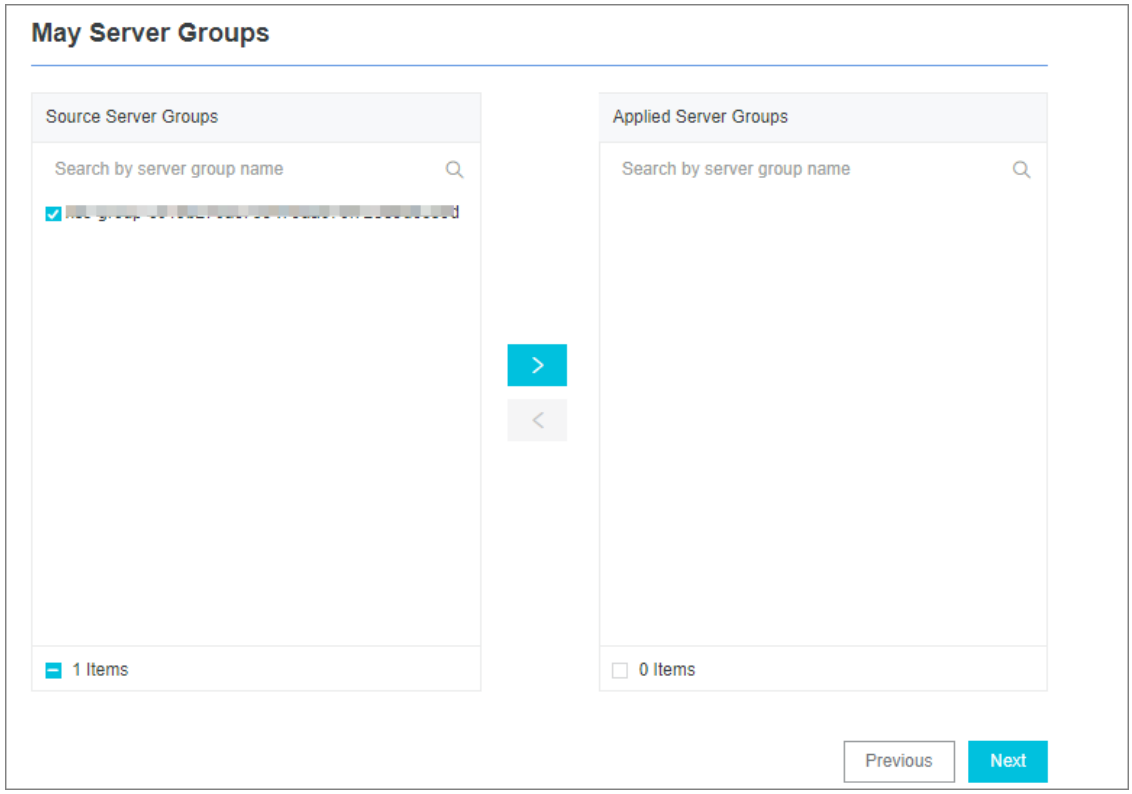
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



Notice If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration and click **Next**.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Note** After the Logtail configuration is created, you cannot change the name of the Logtail configuration.
Log Path	The directory and name of the log file. The specified log file name can be a complete file name or a file name that contains wildcards. For more information, see Wildcard matching. Log Services scans all levels of the specified directory to match log files. Examples:
Docker File	If you want to collect logs from Docker containers, you can specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs.
Mode	The default mode is **JSON Mode**. You can change the mode.

Parameter	Description
Use System Time	Specifies whether to use the system time. ◦ If you turn on **Use System Time**, the timestamp of a log indicates the system time of the server when the log is collected. ◦ If you turn off **Use System Time**, you must set the **Specify Time Key** and **Time Format** parameters based on the time field of JSON logs. For example, if the time information in a JSON log is **"time": "05/May/2016:13:30:28"**, you can set the **Specify Time Key** parameter to **time** and the **Time Format** parameter to **%d/%b/%Y:%H:%M:%S**.
Drop Failed to Parse Logs	Specifies whether to upload logs to Log Service if the logs fail to be parsed. ◦ If you turn on **Drop Failed to Parse Logs**, logs that fail to be parsed are not uploaded to Log Service. ◦ If you turn off **Drop Failed to Parse Logs**, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum depth at which the log directory is monitored. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ◦ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ◦ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ◦ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.

Parameter	Description
Log File Encoding	The encoding format of log files. Valid values: ○ utf8: UTF-8 encoding format ○ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ○ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ○ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ○ Never: All log files are continuously monitored and never time out. ○ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ○ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ○ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in JSON mode.

3.1.4.6. Collect logs in NGINX mode

Log Service allows you to collect NGINX logs and perform multidimensional log analysis. This topic describes how to create a Logtail configuration in NGINX mode in the Log Service console to collect logs.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- Ports 80 and 443 are enabled for the server from which you want to collect logs.

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, select **Nginx - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

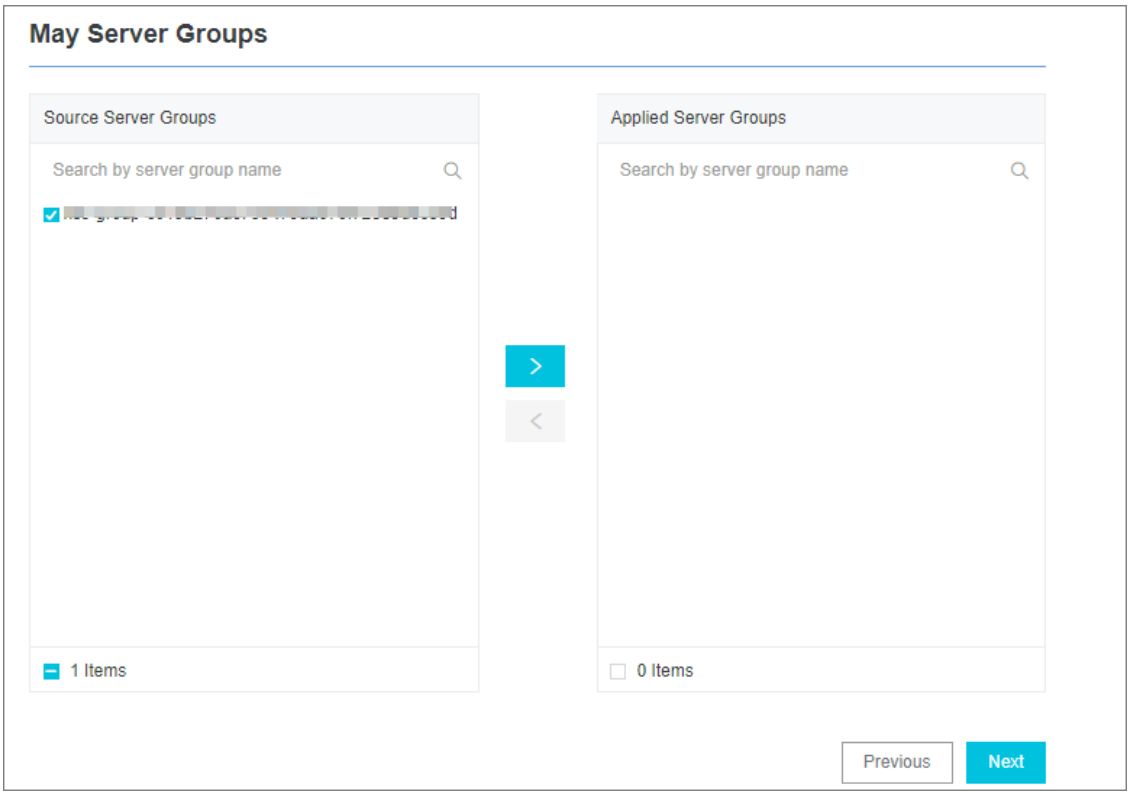
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be FAIL. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration and click **Next**.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Note** After the Logtail configuration is created, you cannot change the name of the Logtail configuration.
Log Path	The directory and name of the log file. The specified log file name can be a complete file name or a file name that contains wildcards. For more information, see Wildcard matching. Log Services scans all levels of the specified directory to match log files. Examples:
Docker File	If you want to collect logs from Docker containers, you can specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs.
Mode	The default mode is **NGINX Configuration Mode**. You can change the mode.
NGINX Log Configuration	Enter the log configuration section that is specified in the NGINX configuration file. The section starts with **log_format**. Example: `log_format main '$remote_addr - $remote_user [$time_local] "$request" ' '$request_time $request_length ' '$status $body_bytes_sent "$http_referer" ' '"$http_user_agent"';` For more information, see Appendix: log formats and sample logs.
NGINX Key	The NGINX keys and values are automatically generated based on the content of NGINX Log Configuration and Log Sample.
Drop Failed to Parse Logs	Specifies whether to upload logs to Log Service if the logs fail to be parsed. ○ If you turn on **Drop Failed to Parse Logs**, logs that fail to be parsed are not uploaded to Log Service. ○ If you turn off **Drop Failed to Parse Logs**, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum depth at which the log directory is monitored. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ◦ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ◦ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ◦ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ◦ utf8: UTF-8 encoding format ◦ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ◦ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ◦ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ◦ Never: All log files are continuously monitored and never time out. ◦ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.

Parameter	Description
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ○ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ○ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in NGINX mode.

Appendix: log formats and sample logs

Before you collect NGINX access logs, you must specify log_format and access_log in the */etc/nginx/nginx.conf* file. The log_format parameter is used to specify the log format. The access_log parameter is used to specify the path in which the NGINX log files are stored.

- Log format

  The following sample code shows the default values of the log_format and access_log parameters:

```
log_format main  '$remote_addr - $remote_user [$time_local] "$request" '
                 '$request_time $request_length '
                 '$status $body_bytes_sent "$http_referer" '
                 '"$http_user_agent"';
access_log /var/logs/nginx/access.log main
```

  The following table describes the log fields.

Log field	Description
remote_addr	The IP address of the client.
remote_user	The username of the client.
time_local	The system time of the server. The value must be enclosed in brackets [].
request	The URI and HTTP protocol of a request.

Log field	Description
request_time	The time that is required to process a request. Unit: seconds.
request_length	The length of a request. The length includes the request line, request header, and request body.
status	The status of a request.
body_bytes_sent	The number of bytes in a response that is sent to the client. The size of the response header is excluded.
http_referer	The URL of the source web page.
http_user_agent	The browser information of the client.

- Sample log

```
192.168.1.2 - - [10/Jul/2020:15:51:09 +0800] "GET /ubuntu.iso HTTP/1.0" 0.000 129 404 168 "-" "Wget
/1.11.4 Red Hat modified"
```

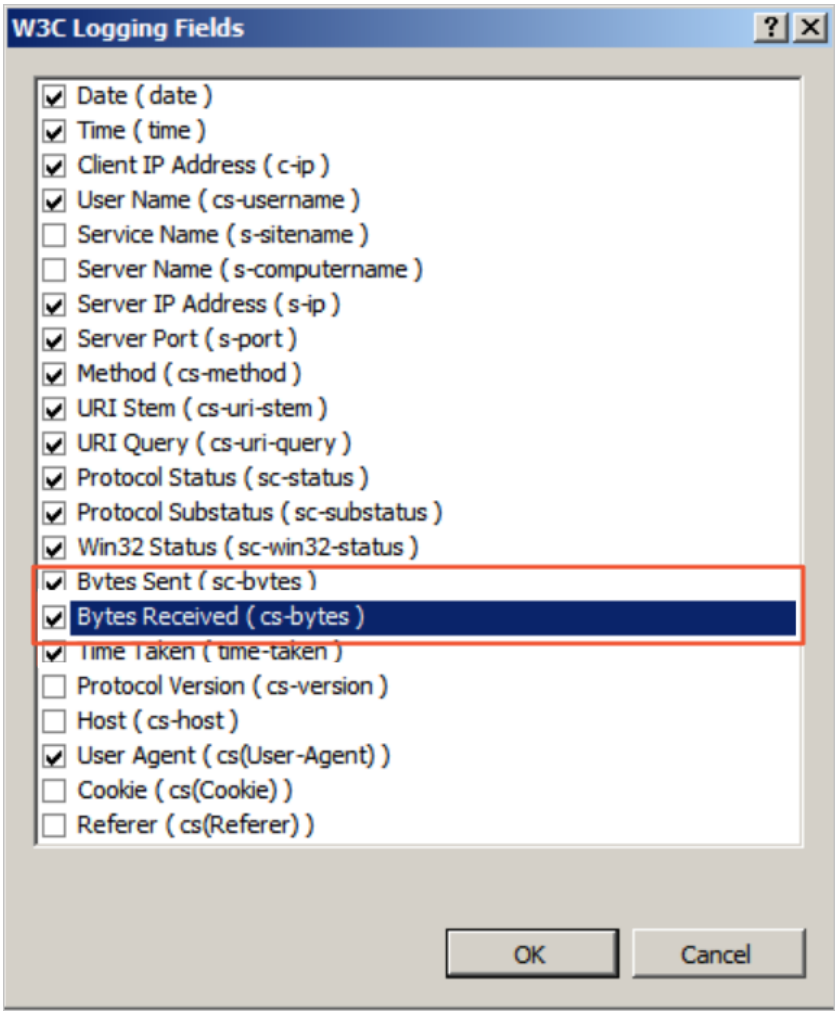
3.1.4.7. Collect logs in IIS mode

Log Service allows you to collect Internet Information Services (IIS) logs and perform multidimensional log analysis. This topic describes how to create a Logtail configuration in IIS mode in the Log Service console to collect logs.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- Ports 80 and 443 are enabled for the server from which you want to collect logs.
- Logs are generated on the server in the IIS, NCSA Common, or W3C Extended format.

We recommend that you use the W3C Extended log format. If you select the W3C Extended format, you must configure the fields in the W3C Logging Fields dialog box. To do so, you must select **Bytes Sent (sc-bytes)** and **Bytes Received (cs-bytes)** and use the default settings for other fields.



Procedure

1. Log on to the Log Service console.
2. Select a data source.

   In the Import Data section, select **IIS - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

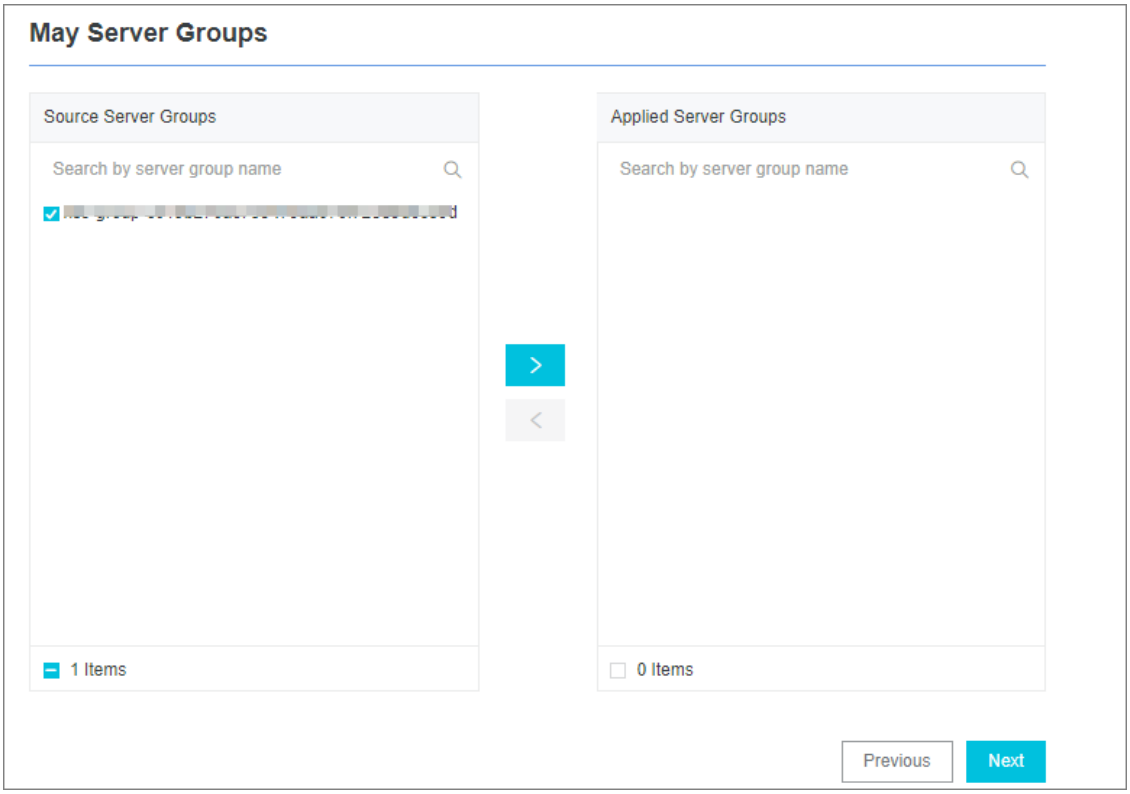
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



Notice If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Note** After the Logtail configuration is created, you cannot change the name of the Logtail configuration.
Log Path	The directory and name of the log file. The specified log file name can be a complete file name or a file name that contains wildcards. For more information, see Wildcard matching. Log Services scans all levels of the specified directory to match log files. Examples:
Docker File	If you want to collect logs from Docker containers, you can specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs. For more information, see Collect container text logs.

Parameter	Description
Mode	If you have specified **IIS - Text Log** for the data source, the default mode is **IIS Configuration Mode**. You can change the mode.
Log format	Select the format of logs that are generated on the IIS server. ○ IIS: Microsoft IIS log file format ○ NCSA: NCSA Common log file format ○ W3C: W3C Extended log file format
IIS Configuration	Specify the IIS configuration fields. ○ If you set the Log format parameter to IIS or NCSA, the IIS configuration fields are automatically generated. ○ If you set the Log format parameter to W3C, enter the content that is specified in the logFile logExtFileFlags field of the IIS configuration file. `logExtFileFlags="Date, Time, ClientIP, UserName, SiteName, ComputerName, ServerIP, Method, UriStem, UriQuery, HttpStatus, Win32Status, BytesSent, BytesRecv, TimeTaken, ServerPort, UserAgent, Cookie, Referer, ProtocolVersion, Host, HttpSubStatus"` ■ Default path of the IIS5 configuration file: C:\WINNT\system32\inetsrv\MetaBase.bin ■ Default path of the IIS6 configuration file: C:\WINDOWS\system32\inetsrv\MetaBase.xml ■ Default path of the IIS7 configuration file: C:\Windows\System32\inetsrv\config\applicationHost.config
IIS Key Name	Log Service automatically extracts IIS keys based on the content of **IIS Configuration**.
Drop Failed to Parse Logs	○ If you turn on **Drop Failed to Parse Logs**, logs that fail to be parsed are not uploaded to Log Service. ○ If you turn off **Drop Failed to Parse Logs**, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum depth at which the log directory is monitored. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ○ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ○ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ○ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ○ utf8: UTF-8 encoding format ○ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ○ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ○ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ○ Never: All log files are continuously monitored and never time out. ○ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.

Parameter	Description
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ○ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ○ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in IIS mode.

Appendix: sample logs and field descriptions

The following example shows a sample IIS log:

```
#Software: Microsoft Internet Information Services 7.5
#Version: 1.0
#Date: 2020-09-08 09:30:26
#Fields: date time s-sitename s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status sc-bytes cs-bytes time-taken
2009-11-26 06:14:21 W3SVC692644773 125.67.67.* GET /index.html - 80 - 10.10.10.10 Baiduspider+(+http://www.baidu.com)200 0 64 185173 296 0
```

- Field prefixes

Prefix	Description
s-	The server action.
c-	The client action.
cs-	The client-to-server action.
sc-	The server-to-client action.

- Fields

Log field	Description
date	The date on which the client sends the request.
time	The point in time at which the client sends the request.
s-sitename	The Internet service name and instance ID of the site that is visited by the client.
s-computername	The name of the server on which the log is generated.
s-ip	The IP address of the server on which the log is generated.
cs-method	The HTTP request method that is used by the client, for example, GET or POST.
cs-uri-stem	The URI resource that is requested by the client.
cs-uri-query	The query string that follows the question mark (?) in the HTTP request.
s-port	The port number of the server.
cs-username	The authenticated domain name or username that is used by the client to access the server. ◦ Authenticated users are referenced as `domain\username`. ◦ Anonymous users are indicated by a hyphen (-).
c-ip	The real IP address of the client that sends the request.
cs-version	The protocol version that is used by the client, for example, HTTP 1.0 or HTTP 1.1.
cs(User-Agent)	The browser that is used by the client.
Cookie	The content of the sent or received cookie. If no cookie is sent or received, a hyphen (-) is displayed.
referer	The previous site that is visited by the user.
cs-host	The host information.
sc-status	The HTTP status code that is returned by the server.
sc-substatus	The HTTP substatus code that is returned by the server.
sc-win32-status	The Windows status code that is returned by the server.
sc-bytes	The number of bytes that are sent by the server.
cs-bytes	The number of bytes that are received by the server.
time-taken	The time that is required to process the request. Unit: milliseconds.

3.1.4.8. Collect logs in Apache mode

Log Service allows you to collect Apache logs and perform multidimensional log analysis. This topic describes how to create a Logtail configuration in Apache mode in the Log Service console to collect logs.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- Ports 80 and 443 are enabled for the server from which you want to collect logs.
- The print format, log path, and log file name are specified in the Apache configuration file. For more information, see Appendix: log formats and sample logs.

Procedure

1. Log on to the Log Service console.
2. Select a data source.

   In the Import Data section, select **Apache - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

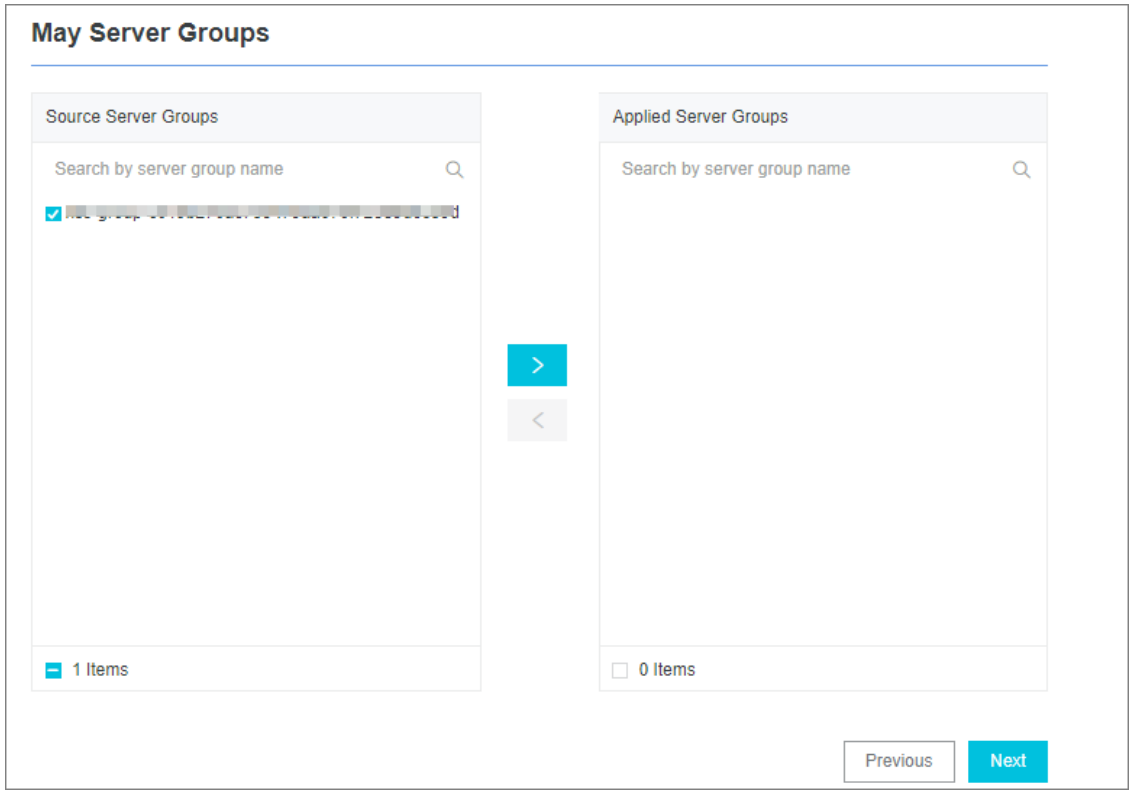
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be FAIL. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration.

   The following table describes the Logtail parameters.

Parameter	Description
Config Name	The name of the Logtail configuration. The name must be 3 to 128 characters in length, and can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or a digit. **Note** After the Logtail configuration is created, you cannot change the name of the Logtail configuration.
Log Path	The directory and name of the log file. The specified log file name can be a complete file name or a file name that contains wildcards. For more information, see Wildcard matching. Log Services scans all levels of the specified directory to match log files. Examples: ◦ If you specify */apsara/nuwa/**/*.log*, Log Service matches the files whose name is suffixed by *.log* in the */apsara/nuwa* directory and its recursive subdirectories. ◦ If you specify */var/logs/app_*/*.log*, Log Service matches the files that meet the following conditions: The file name contains *.log*. The file is stored in a subdirectory of the */var/logs* directory or in a recursive subdirectory of the subdirectory. The name of the subdirectory matches the *app_** pattern. **Note** ◦ By default, each log file can be collected by using only one Logtail configuration. ◦ To use multiple Logtail configurations to collect one log file, we recommend that you create a symbolic link that points to the directory where the file is located. For example, you want to collect two copies of the log.log file from the */home/log/nginx/log/log.log* directory. You can run the following command to create a symbolic link that points to the directory. When you configure the Logtail configurations, use the original path in one Logtail configuration and use the symbolic link in the other Logtail configuration. `ln -s /home/log/nginx/log /home/log/nginx/link_log` ◦ You can use only asterisks (*) and question marks (?) as wildcards in the log path.
Docker File	If you want to collect logs from Docker containers, you can specify the paths and tags of the containers. Logtail monitors the containers to check whether the containers are created or destroyed. Then, Logtail filters the logs of the containers based on tags and collects the filtered logs. For more information about container text logs, see Collect container text logs.
Mode	If you have specified **Apache - Text Log** for the data source, the default mode is **Apache Configuration Mode**. You can change the mode.
Log format	Select a log format based on the format specified in the Apache configuration file. Valid values: common, combined, and Custom.

Parameter	Description
APACHE Logformat Configuration	Enter the log configuration section that is specified in the Apache configuration file. The section starts with LogFormat. For more information, see Appendix: log formats and sample logs. ◦ If you set **Log format** to **common** or **combined**, the system automatically inserts a value into this field. Check whether the value is the same as the value specified in the Apache configuration file. ◦ If you set **Log format** to **Custom**, specify a value based on your business requirements. For example, you can enter **LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %D %f %k %p %q %R %T %I %O" customized**.
APACHE Key Name	Log Service automatically reads Apache keys from the value of the **APACHE Logformat Configuration** field.
Drop Failed to Parse Logs	Specifies whether to drop logs that fail to be parsed. ◦ If you turn on **Drop Failed to Parse Logs**, logs that fail to be parsed are not uploaded to Log Service. ◦ If you turn off **Drop Failed to Parse Logs**, raw logs are uploaded to Log Service if the logs fail to be parsed.
Maximum Directory Monitoring Depth	The maximum depth at which the log directory is monitored. Valid values: 0 to 1000. The value 0 indicates that only the specified log file directory is monitored.

7. (Optional)Specify **Advanced Options** and click **Next**.

   Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.

Parameter	Description
Topic Generation Mode	The topic generation mode. ◦ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ◦ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ◦ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ◦ utf8: UTF-8 encoding format ◦ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ◦ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ◦ Custom: If you select this value, you must select a time zone.
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ◦ Never: All log files are continuously monitored and never time out. ◦ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ◦ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ◦ Filter out logs that do not meet specified conditions. ▪ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ▪ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

> **Note**
>
> - To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
> - If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect logs in Apache mode.

Appendix: log formats and sample logs

Before you collect Apache logs, you must specify the print format, log path, and log file name. For example, **CustomLog "/var/log/apache2/access_log" combined** indicates that the combined format is used when logs are printed. The log file path is */var/log/apache2/access_log*. Log Service supports the following log formats. A sample log is also provided.

- Log formats
  - The combined log format:

    ```
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    ```

  - The common log format:

    ```
    LogFormat "%h %l %u %t \"%r\" %>s %b"
    ```

  - A custom log format:

    ```
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %D %f %k %p %q %R %T %I %O" customized
    ```

  The following table describes the related log fields. For more information, see mod_log_config.

Format string	Log field	Description
%a	client_addr	The IP address of the client.
%A	local_addr	The local IP address.
%b	response_size_bytes	The number of bytes in a response. If no bytes are sent, a hyphen (-) is displayed for this field.
%B	response_bytes	The number of bytes in a response. If no bytes are sent, the digit 0 is displayed for this field.
%D	request_time_msec	The time required to process a request. Unit: microseconds.
%f	filename	The file name.
%h	remote_addr	The name of the remote host.
%H	request_protocol_supple	The request protocol.
%I	bytes_received	The number of bytes that are received by the server. This field is recorded in logs only after you enable the mod_logio module.
%k	keep_alive	The number of keep-alive requests handled on the connection.

Format string	Log field	Description
%l	remote_ident	The information that is provided by the remote host for identification.
%m	request_method_supple	The HTTP request method.
%O	bytes_sent	The number of bytes that are sent by the server. This field is recorded in logs only after you enable the mod_logio module.
%p	remote_port	The port number of the server.
%P	child_process	The ID of the child process.
%q	request_query	The query string. If no query strings exist, an empty string is displayed.
%r	request	The content of the request. The content includes the method name, address, and HTTP protocol.
%R	response_handler	The type of the handler that generates a response on the server.
%s	status	The initial HTTP status of a response.
%>s	status	The final HTTP status of a response.
%t	time_local	The point in time at which the server receives a request.
%T	request_time_sec	The time required to process a request. Unit: seconds.
%u	remote_user	The username of the client.
%U	request_uri_supple	The URI in a request. The URI does not include the query string.
%v	server_name	The name of the server.
%V	server_name_canonical	The name of the server. The name is specified by using the UseCanonicalName directive.
"%{User-Agent}i"	http_user_agent	The information of the client.
"%{Rererer}i"	http_referer	The URL of the web page. The URL is linked to the resource that is being requested.

- Sample log

```
192.168.1.2 - - [02/Feb/2020:17:44:13 +0800] "GET /favicon.ico HTTP/1.1" 404 209 "http://localhost/x1.html" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.97 Safari/537.36"
```

3.1.4.9. Configure parsing scripts

When Log Service collects logs, Log Service extracts some fields in raw logs as log content based on specific parsing methods. This way, you can collect logs based on your business requirements. This topic describes the parsing methods that are supported by Log Service.

Specify a method to separate log lines

A complete access log such as an NGINX access log occupies a line. Separate multiple logs with line feeds. The following example shows two access logs:

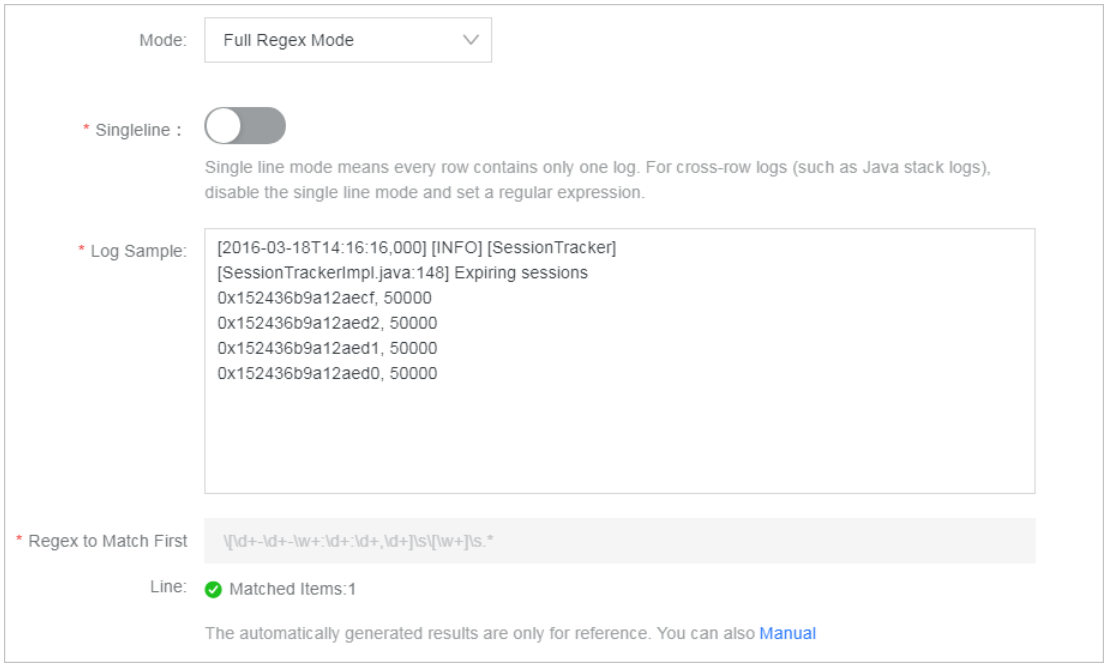
```
203.0.113.10 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (com
patible; MSIE 6.0; Windows NT 5.1; 360se)"
203.0.113.10 - - [13/Mar/2016:10:00:11 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (com
patible; MSIE 6.0; Windows NT 5.1; 360se)"
```

In most cases, logs for Java applications contain multiple lines. Therefore, logs are separated based on the start part in the first line of a log. The following example shows a Java application log:

```
[2016-03-18T14:16:16,000] [INFO] [SessionTracker] [SessionTrackerImpl.java:148] Expiring sessions
0x152436b9a12aecf, 50000
0x152436b9a12aed2, 50000
0x152436b9a12aed1, 50000
0x152436b9a12aed0, 50000
```

The preceding Java application log starts with a datetime value. To ensure the accuracy of log collection, you can specify a regular expression to match the start part in the first line of a log. In this example, the regular expression that is used to match the start part in the first line of a log is `[\d+-\d+-\w+:\d+:\d+,\d+]\s.*`. The following figure shows how to enter a sample log and specify a regular expression in the Log Service console.

Full regex mode



Extract log fields

A log contains one or more key-value pairs based on the data model of Log Service. If you want to extract specific fields for analysis, you must specify a regular expression to match the content that you want to extract. If you do not need to process the content of a log, you can process the log as a key-value pair.

The following example shows two access logs. You can use one of the following two methods to parse the logs.

```
203.0.113.10 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (com
patible; MSIE 6.0; Windows NT 5.1; 360se)"
203.0.113.10 - - [13/Mar/2016:10:00:11 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (com
patible; MSIE 6.0; Windows NT 5.1; 360se)"
```

- Extract specific fields.

  In this example, the regular expression is `(\S+)\s-\s-\s\[(\S+)\s[^]]+]\s"(\w+).*`. The extracted fields are `203.0.113.10`, `13/Mar/2016:10:00`, and `GET`.

- Extract all.

  In this example, the regular expression is `(.*)`. The extracted content is `203.0.113.10 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; 360se)"`.

Specify the log time

A log must contain a time field whose value is a UNIX timestamp based on the data model of Log Service. You can use the system time when Logtail collects a log or the time in the log content as the log time.

The following example shows two access logs. You can use one of the following two methods to parse the logs.

```
203.0.113.10 - - [13/Mar/2016:10:00:10 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (com
patible; MSIE 6.0; Windows NT 5.1; 360se)"
203.0.113.10 - - [13/Mar/2016:10:00:11 +0800] "GET / HTTP/1.1" 0.011 180 404 570 "-" "Mozilla/4.0 (com
patible; MSIE 6.0; Windows NT 5.1; 360se)"
```

- Extract the time in the log content as the log time.

  In this example, the time in the log content is `13/Mar/2016:10:00:10`. To extract the time, use the following time expression: `%d/%b/%Y:%H:%M:%S`.

- Use the system time when the log was collected by Logtail as the log time.

  If you use the system time when the log was collected by Logtail as the log time, the time is converted to a timestamp.

3.1.4.10. Time formats

When you use Logtail to collect logs, you must specify time formats based on the time strings of raw logs. Logtail extracts a time string from a raw log and parses the string into a UNIX timestamp. This topic describes the commonly used time formats and provides related examples.

Commonly used time formats of logs

The following table describes the time formats that are supported by Logtail.

> **Note**

- The timestamp of a log in Log Service is accurate to seconds. Therefore, you can specify the time format only to seconds.
- You need to specify the time format only for the time in a time string. Other parameters such as the time zone are not required.
- In Linux, Logtail supports all time formats provided by the strftime function. Logtail can parse and use all log time strings that can be formatted by using the strftime function.

Format	Description	Example
%a	The abbreviated day name.	Fri
%A	The full day name.	Friday
%b	The abbreviated month name.	Jan
%B	The full month name.	January
%d	The day of a month. Valid values: 01 to 31.	07, 31
%h	The abbreviated month name. The format is equivalent to %b.	Jan
%H	The hour in the 24-hour format.	22
%I	The hour in the 12-hour format.	11
%m	The month. Valid values: 01 to 12.	08
%M	The minute. Valid values: 00 to 59.	59
%n	The line feed.	A line feed
%p	The abbreviation that indicates the morning or afternoon. Valid values: AM and PM.	AM or PM
%r	The time in the 12-hour format. The format is equivalent to %I:%M:%S %p.	11:59:59 AM
%R	The time expressed in hours and minutes. The format is equivalent to %H:%M.	23:59
%S	The second. Valid values: 00 to 59.	59
%t	The tab character.	None
%y	The two-digit year number. Valid values: 00 to 99.	04 or 98
%Y	The four-digit year number.	2004 or 1998
%C	The two-digit century number. Valid values: 00 to 99.	16
%e	The day of a month. Valid values: 1 to 31. Prefix a single-digit number with a space character.	7 or 31
%j	The day of a year. Valid values: 001 to 366.	365

Format	Description	Example
%u	The day of a week as a number. Valid values: 1 to 7. The value 1 indicates Monday.	2
%U	The week of a year. Sunday is the first day of each week. Valid values: 00 to 53.	23
%V	The week of a year. Monday is the first day of each week. Valid values: 01 to 53. If a week that contains January 1 has four or more January days, the week is the first week of a year. Otherwise, the next week is considered the first week of a year.	24
%w	The day of a week as a number. Valid values: 0 to 6. The value 0 indicates Sunday.	5
%W	The week of a year. Monday is the first day of each week. Valid values: 00 to 53.	23
%c	The date and time in the ISO 8601 format.	Tue Nov 20 14:12:58 2020
%x	The date in the ISO 8601 format.	Tue Nov 20 2020
%X	The time in the ISO 8601 format.	11:59:59
%s	The UNIX timestamp.	1476187251

Examples

The following table lists commonly used time formats. It also provides related examples and time expressions.

Example	Time expression	Time format
2017-12-11 15:05:07	%Y-%m-%d %H:%M:%S	Custom
[2017-12-11 15:05:07.012]	[%Y-%m-%d %H:%M:%S]	Custom
02 Jan 06 15:04 MST	%d %b %y %H:%M	RFC822
02 Jan 06 15:04 -0700	%d %b %y %H:%M	RFC822Z
Monday, 02-Jan-06 15:04:05 MST	%A, %d-%b-%y %H:%M:%S	RFC850
Mon, 02 Jan 2006 15:04:05 MST	%A, %d %b %Y %H:%M:%S	RFC1123
2006-01-02T15:04:05Z07:00	%Y-%m-%dT%H:%M:%S	RFC3339
2006-01-02T15:04:05.999999999Z07:00	%Y-%m-%dT%H:%M:%S	RFC3339Nano

3.1.4.11. Import historical log files

This topic describes how to import historical log files from a server to Log Service. By default, Logtail collects only incremental logs from servers. You can configure Logtail to collect historical logs.

Prerequisites

- Logtail V0.16.15 (Linux), Logtail V1.0.0.1 (Windows), or later is installed on the server. For more information, see Install Logtail in Linux and Install Logtail in Windows.
- A Logtail configuration is created and applied to a machine group. For more information, see Configure text log collection.

  If you use the Logtail configuration to import only historical files, you can specify a log collection path that does not exist.

Context

Logtail collects logs based on the modifications in the log files that are monitored. Logtail can also collect logs by loading events from local files. Logtail collects historical logs by loading local events.

You must import historical log files to the installation directory of Logtail. The directory varies based on the operating system.

- Linux: */usr/local/ilogtail*
- Windows:
  - 32-bit: *C:\Program Files\Alibaba\Logtail*
  - 64-bit: *C:\Program Files (x86)\Alibaba\Logtail*

> **Note**
> - The maximum interval between the time when a local event is generated and the time when the local event is imported is 1 minute.
> - If a local event is loaded, Logtail sends the `LOAD_LOCAL_EVENT_ALARM` message to the server.
> - If you want to import a large number of log files, we recommend that you modify the startup parameters of Logtail to increase the value of the cpu_usage_limit parameter to 2 or more and increase the value of the mem_usage_limit parameter to 512 MB or more. For more information, see Set Logtail startup parameters.

Procedure

1. Obtain the unique identifier of the Logtail configuration.

   Open the *user_log_config.json* file in the directory where Logtail is installed. You can obtain the unique identifier of the Logtail configuration from this file.

   For example, to obtain the unique identifier of the Logtail configuration in a Linux server, run the following command:

   ```
   grep "##" /usr/local/ilogtail/user_log_config.json | awk '{print $1}'
                       "##1.0##log-config-test$multi"
                       "##1.0##log-config-test$ecs-test"
                       "##1.0##log-config-test$metric_system_test"
                       "##1.0##log-config-test$redis-status"
   ```

2. Add a local event.

   i. Create the *local_event.json* file in the Logtail installation directory.

ii. Add the local event in the JSON format to the *local_event.json* file of the Logtail installation directory. The following example shows the format of the local event:

```
[
  {
    "config" : "${your_config_unique_id}",
    "dir" : "${your_log_dir}",
    "name" : "${your_log_file_name}"
   },
  {
   ...
   }
   ...
]
```

> **Note** To prevent Logtail from loading invalid JSON files, we recommend that you save the configurations of the local event to a temporary file. Then, edit and copy the configurations to the *local_event.json* file.

Parameter	Description
config	Enter the unique identifier that is obtained in Step 1. Example: ##1.0##log-config-test$ecs-test.
dir	The directory in which historical log files are saved. Example: */data/logs*. **Note** The directory cannot end with a forward slash (/).
name	The name of the historical log file. The name can contain wildcards. Example: access.log.2018-08-08 and access.log*.

The following example shows how to configure a local event in Linux by using the `cat /usr/local/ilogtail/local_event.json` command:

```
[
  {
    "config": "##1.0##log-config-test$ecs-test",
    "dir": "/data/log",
    "name": "access.log*"
  },
  {
    "config": "##1.0##log-config-test$tmp-test",
    "dir": "/tmp",
    "name": "access.log.2017-08-09"
  }
]
```

FAQ

- How do I check whether Logtail loads a Logtail configuration?

  After you save the *local_event.json* file, Logtail loads the configurations of the local event to the memory within 1 minute. Then, the content of the *local_event.json* file is deleted.

  You can use the following methods to check whether the Logtail configuration is loaded.

i. If no content exists in the *local_event.json* file, Logtail reads the local event from the file.

ii. Check whether the *ilogtail.LOG* file in the Logtail installation directory contains the process local event parameter. If the content in the *local_event.json* file is cleared but the process local event parameter does not exist, the content of the *local_event.json* file may be invalid and filtered out.

- Why am I unable to collect a log file after a Logtail configuration is loaded?
  - The Logtail configuration is invalid.
  - The configurations of the local event in the *local_event.json* file are invalid.
  - The log file does not exist in the path that is specified in the Logtail configuration.
  - The log file has been collected by Logtail.

3.1.4.12. Log topics

Logs can be identified by log topics. When you collect logs, you can specify a topic for the logs.

You can specify a topic in the following scenarios: when you use Logtail to collect logs, when you call API operations, or when you use an SDK to upload log data. In the Log Service console, you can set the topic generation mode to Null - Do not generate topic, Machine Group Topic Attributes, or File Path RegEx.

- Null - Do not generate topic

  In this mode, the topic is an empty string. You can query logs without the need to specify a topic.

- Machine Group Topic Attributes

  You can use this mode to identify logs that are generated on different servers. If the logs are saved with the same file name or the logs are saved in the same directory, you can specify different topics to identify the logs.

  You can add servers to different machine groups, and configure different topic attributes for the machine groups. When you create a Logtail configuration, set **Topic Generation Mode** to **Machine Group Topic Attributes**. If Logtail sends the logs of a server in a machine group to Log Service, Logtail uploads the topic attributes of the machine group as topic names. You can use the topic attributes as filters to query logs.

- File Path RegEx

  You can use this mode to identify logs that are generated by different users or instances. Log Service stores logs in different directories for different users or instances. However, if duplicate sub-directory names or log file names exist in these directories, Log service cannot identify which user or instance generates the logs.

  To resolve this issue, you can set **Topic generation modes** to **File Path Regex** and enter the regular expression of the log file path when you create a Logtail configuration. The regular expression must match the log file path. When Logtail sends logs to Log Service, Logtail uploads the username or the instance name as the topic name. You can use the topic name as a filter to query logs.

  Logs that are generated by different users or instances may be stored in different files with the same name. Each file is stored in a different directory. For example, three log files are all named *service.log* and you only specify the *service.log* file in the */logs* directory as the log source when you collect logs from these files. After the logs are sent to Log Service, Log Service cannot identify which users or instances generate the logs. To resolve this issue, you can set **Topic Generation Mode** to **File Path RegEx**, and enter the `\/(.*)\/serviceA\/.*` regular expression. Then, Log Service generates the following topics for logs that are in different directories: userA, userB, and userC.

```
/logs
  | - /userA/serviceA
    | - service.log
  | - /userB/serviceA
    | - service.log
  | - /userC/serviceA
    | - service.log
```

> **Note** You must escape the forward slash (/) in the regular expression that is used to match file paths.

To extract multiple fields from a file path, you can use the `?P<key>` sub-expression to extract fields from the layers of the file path. The value of the key parameter can only contain lowercase letters and digits. Example:

```
/home/admin/serviceA/userB/access.log
\/home\/admin\/(?P<service>[^\/]+)/(?P<user>[^/]+)/.*
```

The following custom tags are created for logs:

```
"__tag__ : service : serviceA"
"__tag__ : user : userB"
```

- Static topic generation

  You can set **Topic Generation Mode** to **File Path RegEx**. In the **Custom RegEx** field, enter `customized:// + custom topic name`.

  > **Note** Static topic generation is supported by Logtail V0.16.21 (Linux) and later.

3.1.5. Collect container logs

3.1.5.1. Overview

Log Service allows you to collect Kubernetes container logs in DaemonSet mode or Sidecar mode. This topic describes the procedures and differences of log collection in the two modes.

Log collection modes

Log collection in DaemonSet mode features simple O&M, low resource usage, and flexible configurations. You can collect container stdout and stderr. You can also collect container text logs. In DaemonSet mode, Logtail collects logs from all containers on the DaemonSet-specific node. However, in this mode, performance bottleneck issues may occur on Logtail, and containers are loosely isolated. In Sidecar mode, a Sidecar container is created for each container from which you want to collect logs. In this mode, Logtail provides good performance, and tenants are completely isolated.

Log collection configurations

You can create log collection configurations by using the Log Service console or custom resource definitions (CRDs). The following table describes the differences between the two modes.

Item	CRD	Log Service console
Operation complexity	Low	Moderate
Feature diversity	All configurations that the console supports and advanced configurations that the console does not support	Moderate
Ease of use	Moderate	Low
Network connection	Connected to a Kubernetes cluster	Connected to the Internet
Integration with container applications	Supported	Not supported

Item	CRD	Log Service console
Authentication method	Kubernetes authentication	Authentication based on Alibaba Cloud accounts

Log collection procedures

The following procedure describes how to collect logs in DaemonSet mode:

1. Collect Kubernetes logs.
2. Create a log collection configuration.

   Log Service allows you to create log collection configurations by using CRDs or the Log Service console to collect container logs from Kubernetes clusters.

   - Use CRDs to collect container logs in DaemonSet mode.
   - Use the Log Service console to collect container text logs in DaemonSet mode.
   - Use the Log Service console to collect container stdout and stderr in DaemonSet mode.

   > **Note** If you use CRDs, resources such as projects, Logstores, indexes, machine groups, and Logtail configurations are automatically created. In addition, this method leads to better integration with Kubernetes. We recommend that you use this method. If you use the Log Service console, you need to only perform simple operations. The first time you use Log Service to collect container logs, we recommend that you use this method.

The following procedure describes how to collect logs in Sidecar mode:

1. Collect Kubernetes logs.
2. Install Sidecar and create a log collection configuration.

   Log Service allows you to create log collection configurations by using CRDs or the Log Service console to collect container logs from Kubernetes clusters.

   - Use CRDs to collect container text logs in Sidecar mode.
   - Use the Log Service console to collect container text logs in Sidecar mode.

3.1.5.2. Install the Logtail component

This topic describes how to install the Logtail component in a Kubernetes cluster.

Context

Before you can collect container logs from a Kubernetes cluster, you must install the Logtail component.

When you install the Logtail component, the following operations are automatically completed:

1. The alibaba-log-configuration ConfigMap is created. This ConfigMap stores the configuration information about Log Service, such as project information.
2. Optional. The AliyunLogConfig custom resource definition (CRD) is created.
3. Optional. The alibaba-log-controller Deployment is created. This Deployment is used to monitor the changes in the AliyunLogConfig CRD and create Logtail configurations.
4. The logtail-ds DaemonSet is created. This DaemonSet is used to collect logs from nodes.

Alibaba Cloud Container Service for Kubernetes (ACK) clusters

You can install the Logtail component in an existing ACK cluster. You can also install the Logtail component when you create an ACK cluster. To install the Logtail component when you create an ACK cluster, you must select **Enable Log Service**.

Install the Logtail component in an existing ACK cluster

1. Log on to the Log Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find and click the cluster in which you want to install the Logtail component.
4. In the left-side navigation pane of the page that appears, choose **Operations > Add-ons**.
5. On the **Logs and Monitoring** tab, find the **logtail-ds** component and click **Install**.

   After the component is installed, a machine group named `k8s-group-${your_k8s_cluster_id}` and a Logstore named `config-operation-log` are automatically created in the project that you use.

   > **Notice** Do not delete the `config-operation-log` Logstore.

Install the Logtail component when you create an ACK cluster

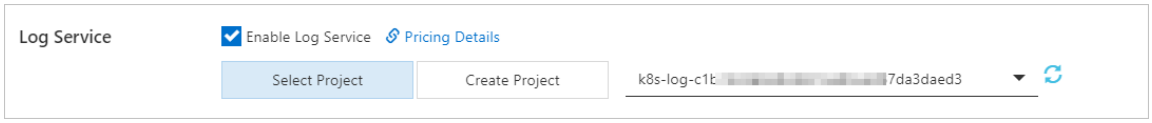
1. Log on to the Log Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, click **Create Kubernetes Cluster**.
4. In the **Component Configurations** step, select **Enable Log Service**.

   > **Note** In this example, only the steps that are required to enable Log Service are provided. For more information about how to create an ACK cluster, see *Create a Kubernetes cluster* in *Container Service for Kubernetes User Guide*.

   If you select **Enable Log Service**, the system prompts you to create a Log Service project. You can use one of the following methods to create a project:

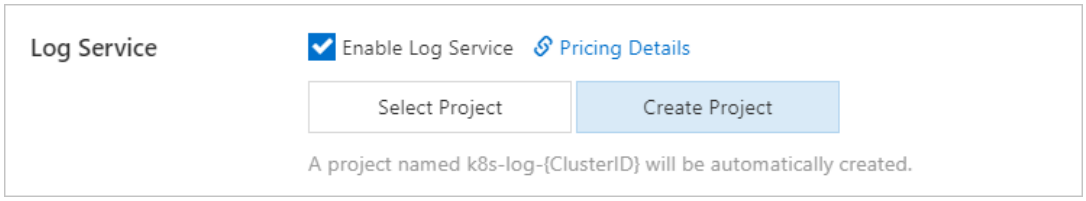
   - **Select Project**

     You can select an existing project to manage the container logs that are collected.

   - **Create Project**

     Log Service automatically creates a project named `k8s-log-{ClusterID}` to manage the container logs that are collected. `ClusterID` indicates the unique ID of the ACK cluster that is created.

   After the component is installed, a machine group named `k8s-group-${your_k8s_cluster_id}` and a Logstore named `config-operation-log` are automatically created in the project that you use.

   > **Notice** Do not delete the `config-operation-log` Logstore.

Self-managed Kubernetes clusters

1. Log on to the Log Service console
2. Create a project whose name starts with `k8s-log-custom-`.

   Example: k8s-log-custom-sd89ehdq. For more information, see Create a project..

3. Log on to your Kubernetes cluster.
4. Run the following command to install the alibaba-log-controller component.

   > **Notice** Make sure that the kubectl command-line tool is installed on the machine on which you want to run the command.

   i. Download the alicloud-log-k8s-custom-install.sh script.

   ii. Upload the script to the machine.

   iii. Go to the directory of the script and modify permissions.

   ```
   chmod 744 ./alicloud-log-k8s-custom-install.sh;
   ```

   iv. Run the following installation command:

   ```
   ./alicloud-k8s-log-installer.sh --cluster-id ${your_k8s_cluster_id} --ali-uid ${your_ali_uid}
   --region-id ${your_k8s_cluster_region_id} {access-key-id} {access-key-secret}
   ```

   You can configure the parameters in the command based on your business requirements. The following table describes the parameters.

Parameter	Description
your_k8s_cluster_id	The ID of your Kubernetes cluster.
your_ali_uid	The ID of your Alibaba Cloud account. **Note** The ID of an Alibaba Cloud account is a string. For more information about how to view the ID of an Alibaba Cloud account, see Configure an account ID on a server.
your_k8s_cluster_region_id	The ID of the region where your Kubernetes cluster resides.
access-key-id	The AccessKey ID of your Alibaba Cloud account. For more information, see Obtain an AccessKey pair.
access-key-secret	The AccessKey secret of your Alibaba Cloud account. For more information, see Obtain an AccessKey pair.

   After the component is installed, a machine group named `k8s-group-${your_k8s_cluster_id}` and a Logstore named `config-operation-log` are automatically created in the project that you use.

   > **Notice**
   > - Do not delete the `config-operation-log` Logstore.
   > - If you install the component in a self-managed Kubernetes cluster, Logtail is granted the `privileged` permissions. This helps prevent the `container text file busy` error that occurs when other pods are deleted. For more information, see Bug 1468249, Bug 1441737, and Issue 34538.

FAQ

- How do I collect and send container logs from multiple Kubernetes clusters to the same Log Service project?
  - Alibaba Cloud ACK clusters

    If you want to collect and send container logs from multiple ACK clusters to the same Log Service project, you must select the same project when you create the ACK clusters.

- Self-managed Kubernetes clusters

  If you want to collect and send container logs from multiple self-managed Kubernetes clusters to the same Log Service project, you must set the {your-project-suffix} parameter to the same value when you install the Logtail component in each of the Kubernetes clusters.

  > **Note** You can collect and send container logs from multiple self-managed Kubernetes clusters to the same Log Service project only if the Kubernetes clusters reside in the same region.

- How do I view the logs of Logtail?

  The logs of Logtail are stored in the files named *ilogtail.LOG* and *logtail_plugin.LOG* in the */usr/local/ilogtail/* directory of a Logtail container.

  The stdout and stderr of the Logtail container are not for reference. You can ignore the following stdout and stderr:

  ```
  start umount useless mount points, /shm$|/merged$|/mqueue$
  umount: /logtail_host/var/lib/docker/overlay2/3fd0043af174cb0273c3c7869500fbe2bdb95d13b1e110172ef57
  fe840c82155/merged: must be superuser to unmount
  umount: /logtail_host/var/lib/docker/overlay2/d5b10aa19399992755de1f85d25009528daa749c1bf8c16edff44
  beab6e69718/merged: must be superuser to unmount
  umount: /logtail_host/var/lib/docker/overlay2/5c3125daddacedec29df72ad0c52fac800cd56c6e880dc4e8a640
  b1e16c22dbe/merged: must be superuser to unmount
  ......
  xargs: umount: exited with status 255; aborting
  umount done
  start logtail
  ilogtail is running
  logtail status:
  ilogtail is running
  ```

- How do I view the status of Log Service components in Kubernetes clusters?

  Run the following commands:

  ```
  kubectl get deploy alibaba-log-controller -n kube-system
  kubectl get ds logtail-ds -n kube-system
  ```

- What do I do if alibaba-log-controller fails to start?

  Check whether alibaba-log-controller is installed by using the following method:

  - Run the installation command on the control plane of your Kubernetes cluster.
  - Specify the ID of your Kubernetes cluster in the installation command.

  If alibaba-log-controller is not installed by using the preceding method, run the `kubectl delete -f deploy` command to delete the installation template that is generated. Then, run the installation command again.

- How do I view the status of the Logtail DaemonSet in a Kubernetes cluster?

  Run the `kubectl get ds -n kube-system` command to view the status of the Logtail DaemonSet.

  > **Note** The default namespace to which a Logtail container belongs is kube-system.

- How do I view the version number, IP address, startup time, and status of Logtail?

- Run the following command to view the status of Logtail:

  ```
  kubectl get po -n kube-system | grep logtail
  ```

  The following output is returned:

  ```
  NAME            READY     STATUS    RESTARTS   AGE
  logtail-ds-gb92k   1/1       Running   0          2h
  logtail-ds-wm7lw   1/1       Running   0          4d
  ```

- Run the following command to view the version number and IP address of Logtail:

  ```
  kubectl exec logtail-ds-gb92k -n kube-system cat /usr/local/ilogtail/app_info.json
  ```

  The following output is returned:

  ```
  {
     "UUID" : "",
     "hostname" : "logtail-ds-gb92k",
     "instance_id" : "0EBB2B0E-0A3B-11E8-B0CE-0A58AC140402_172.20.4.2_1517810940",
     "ip" : "192.0.2.0",
     "logtail_version" : "0.16.2",
     "os" : "Linux; 3.10.0-693.2.2.el7.x86_64; #1 SMP Tue Sep 12 22:26:13 UTC 2017; x86_64",
     "update_time" : "2021-02-05 06:09:01"
  }
  ```

- How do I view the run logs of Logtail?

  The run logs of Logtail are stored in the *ilogtail.LOG* file in the */usr/local/ilogtail/* directory. If the log file is rotated, the generated files are compressed and stored as *ilogtail.LOG.x.gz*.

  Run the `kubectl exec logtail-ds-gb92k -n kube-system tail /usr/local/ilogtail/ilogtail.LOG` command to view the logs. Example output:

  ```
  [2018-02-05 06:09:02.168693] [INFO] [9] [build/release64/sls/ilogtail/LogtailPlugin.cpp:104] logtai
  l plugin Resume:start
  [2018-02-05 06:09:02.168807] [INFO] [9] [build/release64/sls/ilogtail/LogtailPlugin.cpp:106] logtai
  l plugin Resume:success
  [2018-02-05 06:09:02.168822] [INFO] [9] [build/release64/sls/ilogtail/EventDispatcher.cpp:369] star
  t add existed check point events, size:0
  [2018-02-05 06:09:02.168827] [INFO] [9] [build/release64/sls/ilogtail/EventDispatcher.cpp:511] add
  existed check point events, size:0 cache size:0 event size:0 success count:0
  ```

- How do I restart Logtail for a pod?

  i. Stop Logtail.

  In the following command, `logtail-ds-gb92k -n` specifies the container, and `kube-system` specifies the namespace. Configure the parameters based on your business requirements.

  ```
  kubectl exec logtail-ds-gb92k -n kube-system /etc/init.d/ilogtaild stop
  ```

  If the following output is returned, Logtail is stopped:

  ```
  kill process Name: ilogtail pid: 7
  kill process Name: ilogtail pid: 9
  stop success
  ```

  ii. Start Logtail.

  In the following command, `logtail-ds-gb92k -n` specifies the container, and `kube-system` specifies the namespace. Configure the parameters based on your business requirements.

```
kubectl exec logtail-ds-gb92k -n kube-system /etc/init.d/ilogtaild start
```

If the following output is returned, Logtail is started:

```
ilogtail is running
```

What's next

Create Logtail configurations to collect container logs.

- DaemonSet mode
  - For more information about how to collect container logs by using CRDs, see Use CRDs to collect container logs in DaemonSet mode.
  - For more information about how to collect container stdout and stderr by using the Log Service console, see Use the Log Service console to collect container stdout and stderr in DaemonSet mode.
  - For more information about how to collect container text logs by using the Log Service console, see Use the Log Service console to collect container text logs in DaemonSet mode.
- Sidecar mode
  - For more information about how to collect container text logs by using CRDs, see Use CRDs to collect container text logs in Sidecar mode.
  - For more information about how to collect container text logs by using the Log Service console, see Use the Log Service console to collect container text logs in Sidecar mode.

3.1.5.3. Use the Log Service console to collect container text logs in DaemonSet mode

This topic describes how to create a Logtail configuration in the Log Service console and use the Logtail configuration to collect container text logs in DaemonSet mode.

Prerequisites

The Logtail component is installed. For more information, see Install the Logtail component.

Features

Logtail can collect container text logs, and then upload the text logs together with container metadata to Log Service. Logtail supports the following features:

- Allows you to specify a log file path in a container. You do not need to manually map the log file path to a path on the host.
- Uses the container label whitelist to specify containers from which text logs are collected.
- Uses the container label blacklist to specify containers from which text logs are not collected.
- Uses the environment variable whitelist to specify containers from which text logs are collected.
- Uses the environment variable blacklist to specify containers from which text logs are not collected.
- Collects multi-line logs. For example, Logtail can collect Java stack logs.
- Automatically associates container metadata that needs to be uploaded together with the collected container text logs. The metadata includes container names, image names, pod names, namespaces, and environment variables.
- If a container runs in a Kubernetes cluster, Logtail also supports the following features:
  - Uses Kubernetes namespaces, pod names, and container names to specify containers from which text logs are collected.
  - Uses the Kubernetes label whitelist to specify containers from which text logs are collected.
  - Uses the Kubernetes label blacklist to specify containers from which text logs are not collected.

- Automatically associates Kubernetes labels that need to be uploaded together with the collected container text logs.

Limits

- If Logtail detects the `die` event on a container that is stopped, Logtail no longer collects text logs from the container. If collection latency exists, some text logs that are collected before the container is stopped may be lost.
- For Docker containers, only overlay and overlay2 storage drivers are supported. If other storage drivers are used, you must mount a volume to the directory of logs. Then, a temporary directory is generated.
- Logtail cannot access the symbolic link of a container. You must specify an actual path as the collection directory.
- If a volume is mounted to the data directory of a container, Logtail cannot collect data from the parent directory of the data directory. You must specify the complete path of the data directory as the collection directory.

  For example, if a volume is mounted to the */var/log/service* directory and you set the collection directory to */var/log*, Logtail cannot collect logs from the /var/log directory. You must specify */var/log/service* as the collection directory.
- By default, Kubernetes mounts the root directory of the host to the `/logtail_host` directory of the Logtail container. If you want to collect text logs from the host, you must specify `/logtail_host` as the prefix of the log file path.

  For example, if you want to collect logs from the `/home/logs/app_log/` directory of the host, you must specify `/logtail_host/home/logs/app_log/` as the log file path.
- Logtail collects data from containers that use the Docker engine or containerd engine.
  - Docker: Logtail accesses the Docker engine in the */run/docker.sock* directory. Make sure that the directory exists and Logtail has the permissions to access the directory.
  - containerd: Logtail accesses the containerd engine in the */run/containerd/containerd.sock* directory. Make sure that the directory exists and Logtail has the permissions to access the directory.

Create a Logtail configuration

1. Log on to the Log Service console.
2. In the **Import Data** section, click **Kubernetes - Object**.
3. Select a project and a Logstore. Then, click **Next**.

   In this example, select the project that you use to install the Logtail component and the Logstore that you create.
4. Click **Use Existing Machine Groups**.

   After you install the Logtail component, Log Service automatically creates a machine group named `k8s-group-${your_k8s_cluster_id}`. You can select this machine group.
5. Select the `k8s-group-${your_k8s_cluster_id}` machine group from **Source Server Groups** and move the machine group to **Applied Server Groups**. Then, click **Next**.

   > **Notice** If the heartbeat status of the machine group is **FAIL**, you can click **Automatic Retry**. If the issue persists, see *What do I do if no heartbeat connections are detected on Logtail?* in the FAQ.
6. Configure the parameters for the Logtail configuration and click **Next**.
   i. Configure the basic settings, such as the name, log path, and mode. For more information, see Collect text logs.
   ii. Turn on **Docker File**.
   iii. (Optional)Specify conditions to filter containers.

- For versions earlier than Logtail V1.0.29, containers can be filtered only by using environment variables and container labels. The following table describes the parameters.

  A namespace of a Kubernetes cluster and the name of a container in a Kubernetes cluster can be mapped to container labels. The value of the LabelKey parameter for a namespace is `io.kubernetes.pod.namespace`. The value of the LabelKey parameter for a container name is `io.kubernetes.container.name`. We recommend that you use the two container labels to filter containers. If the container labels do not meet your business requirements, you can use the environment variable whitelist or the environment variable blacklist to filter containers. For example, the namespace of a pod is backend-prod, and the name of a container in the pod is worker-server. If you want the logs of the worker-server container to be collected, you can specify `io.kubernetes.pod.namespace : backend-prod` or `io.kubernetes.container.name : worker-server` in the container label whitelist.

  > **Notice**
  > - Container labels are retrieved by running the docker inspect command. Container labels are different from Kubernetes labels. For more information, see Obtain container labels.
  > - Environment variables are the same as the environment variables that are configured to start containers. For more information, see Obtain container environment variables.
  > - Do not specify duplicate values for the LabelKey parameter. If you specify duplicate values for the LabelKey parameter, only one of the values takes effect.

Parameter	Description
Label Whitelist	The container label whitelist. The whitelist specifies the containers from which text logs are collected. When you configure the container label whitelist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose container labels contain the keys specified by LabelKey are matched. ■ If the LabelValue parameter is not empty, containers whose container labels consist of the key-value pairs specified by LabelKey and LabelValue are matched. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the container labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the LabelValue parameter, regular expression matching is performed. For example, if you set the LabelKey parameter to *io.kubernetes.container.name* and set the LabelValue parameter to *^(nginx\|cube)$*, a container named nginx and a container named cube are matched. Key-value pairs are connected by using the OR operator. If a container label consists of one of the specified key-value pairs, the container to which the container label belongs is matched.

Parameter	Description
Label Blacklist	The container label blacklist. The blacklist specifies the containers from which text logs are not collected. When you configure the container label blacklist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose container labels contain the keys specified by LabelKey are filtered out. ■ If the LabelValue parameter is not empty, containers whose container labels consist of the key-value pairs specified by LabelKey and LabelValue are filtered out. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the container labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the LabelValue parameter, regular expression matching is performed. For example, if you set the LabelKey parameter to *io.kubernetes.container.name* and set the LabelValue parameter to *^(nginx\|cube)$*, a container named nginx and a container named cube are matched. Key-value pairs are connected by using the OR operator. If a container label consists of one of the specified key-value pairs, the container to which the container label belongs is filtered out.
Environment Variable Whitelist	The environment variable whitelist. The whitelist specifies the containers from which text logs are collected. When you configure the environment variable whitelist, the EnvKey parameter is required, and the EnvValue parameter is optional. ■ If the EnvValue parameter is empty, containers whose environment variables contain the keys specified by EnvKey are matched. ■ If the EnvValue parameter is not empty, containers whose environment variables consist of the key-value pairs specified by EnvKey and EnvValue are matched. By default, string matching is performed for the values of the EnvValue parameter. Containers are matched only if the values of the environment variables are the same as the values of the EnvValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the EnvValue parameter, regular expression matching is performed. For example, if you set the EnvKey parameter to *NGINX_SERVICE_PORT* and set the EnvValue parameter to *^(80\|6379)$*, containers whose port number is 80 and containers whose port number is 6379 are matched. Key-value pairs are connected by using the OR operator. If an environment variable consists of one of the specified key-value pairs, the container to which the environment variable belongs is matched.

Parameter	Description
Environment Variable Blacklist	The environment variable blacklist. The blacklist specifies the containers from which text logs are not collected. When you configure the environment variable blacklist, the EnvKey parameter is required, and the EnvValue parameter is optional. ■ If the EnvValue parameter is empty, containers whose environment variables contain the keys specified by EnvKey are filtered out. ■ If the EnvValue parameter is not empty, containers whose environment variables consist of the key-value pairs specified by EnvKey and EnvValue are filtered out. By default, string matching is performed for the values of the EnvValue parameter. Containers are matched only if the values of the environment variables are the same as the values of the EnvValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the EnvValue parameter, regular expression matching is performed. For example, if you set the EnvKey parameter to *NGINX_SERVICE_PORT* and set the EnvValue parameter to *^(80\|6379)$*, containers whose port number is 80 and containers whose port number is 6379 are matched. Key-value pairs are connected by using the OR operator. If an environment variable consists of one of the specified key-value pairs, the container to which the environment variable belongs is filtered out.

■ For Logtail V1.0.29 or later, we recommend that you use different levels of Kubernetes information, such as pod names, namespaces, container names, and labels to filter containers.

Turn on **Deployed in K8s** and configure the following parameters to filter containers.

> **Note** If you change Kubernetes labels when Kubernetes control resources, such as Deployments, are running, the operational pod is not restarted. Therefore, the pod cannot detect the change. This may cause a matching rule to become invalid. When you specify the Kubernetes label whitelist and the Kubernetes label blacklist, we recommend that you use the Kubernetes labels of pods.

Parameter	Description
K8s Pod Name Regular Matching	The pod name. The pod name specifies the containers from which text logs are collected. Regular expression matching is supported. For example, if you specify *^(nginx-log-demo.*)$*, all containers in the pod whose name starts with nginx-log-demo are matched.
K8s Namespace Regular Matching	The namespace. The namespace specifies the containers from which text logs are collected. Regular expression matching is supported. For example, if you specify *^(default|nginx)$*, all containers in the nginx and default namespaces are matched.

Parameter	Description
K8s Container Name Regular Matching	The container name. The container name specifies the containers from which text logs are collected. Regular expression matching is supported. Kubernetes container names are defined in spec.containers. For example, if you specify *^(container-test)$*, all containers whose name is container-test are matched.
K8s Label Whitelist	The Kubernetes label whitelist. The whitelist specifies the containers from which text logs are collected. When you configure the Kubernetes label whitelist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose Kubernetes labels contain the keys specified by LabelKey are matched. ■ If the LabelValue parameter is not empty, containers whose Kubernetes labels consist of the key-value pairs specified by LabelKey and LabelValue are matched. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the Kubernetes labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`), regular expression matching is performed. For example, if you set the **LabelKey** parameter to *app* and set the **LabelValue** parameter to *^(test1\|test2)$*, containers whose Kubernetes labels consist of app:test1 or app:test2 are matched. Key-value pairs are connected by using the OR operator. If a Kubernetes label consists of one of the specified key-value pairs, the container to which the Kubernetes label belongs is matched.
K8s Label Blacklist	The Kubernetes label blacklist. The blacklist specifies the containers from which text logs are not collected. When you configure the Kubernetes label blacklist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose Kubernetes labels contain the keys specified by LabelKey are filtered out. ■ If the LabelValue parameter is not empty, containers whose Kubernetes labels consist of the key-value pairs specified by LabelKey and LabelValue are filtered out. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the Kubernetes labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`), regular expression matching is performed. For example, if you set the **LabelKey** parameter to *app* and set the **LabelValue** parameter to *^(test1\|test2)$*, regular expression matching is performed. Key-value pairs are connected by using the OR operator. If a Kubernetes label consists of one of the specified key-value pairs, the container to which the Kubernetes label belongs is filtered out.

iv. (Optional)Specify log labels.

For Logtail V1.0.29 or later, we recommend that you specify environment variables and Kubernetes labels for logs as log labels.

Parameter	Description
Environment Variable Log Tag	After you specify environment variables as log labels, Log Service adds environment variable-related fields to logs. For example, if you set the **EnvKey** parameter to *VERSION* and set the **EnvValue** parameter to *env_version*, Log Service adds the __tag__:__env_version__: v1.0.0 field to logs if the environment variable configurations of a container include `VERSION=v1.0.0`.
K8s Label Log Tag	After you specify Kubernetes labels as log labels, Log Service adds Kubernetes label-related fields to logs. For example, if you set the **LabelKey** parameter to *app* and set the **LabelValue** parameter to `k8s_label_app`, Log Service adds the __tag__:__k8s_label_app__: serviceA field to logs if the label configurations of a Kubernetes cluster include `app=serviceA`.

7. Preview data, configure indexes, and then click **Next**.

By default, full-text indexing is enabled for Log Service. You can also configure field indexes based on collected logs in manual or automatic mode. For more information, see Configure indexes.

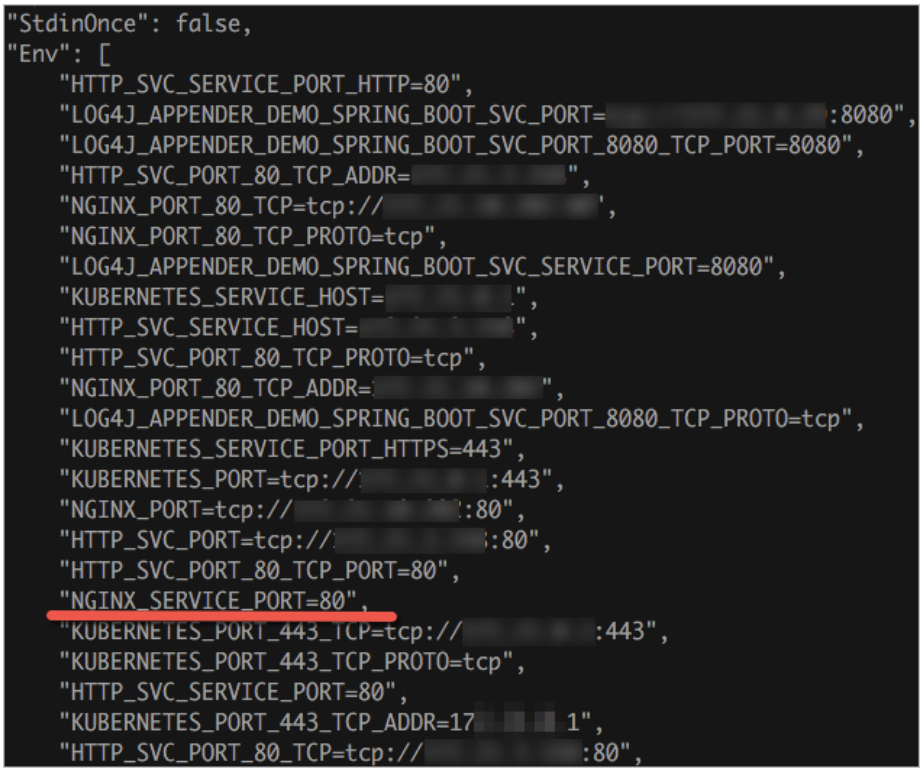
Configuration examples

Example 1: Filter containers based on the environment variable whitelist and the environment variable blacklist

Collect text logs from the containers whose environment variable configurations include `NGINX_SERVICE_PORT=80` but exclude `POD_NAMESPACE=kube-system`. The log file path is `/var/log/nginx/access.log`. The logs are parsed in **simple mode**.

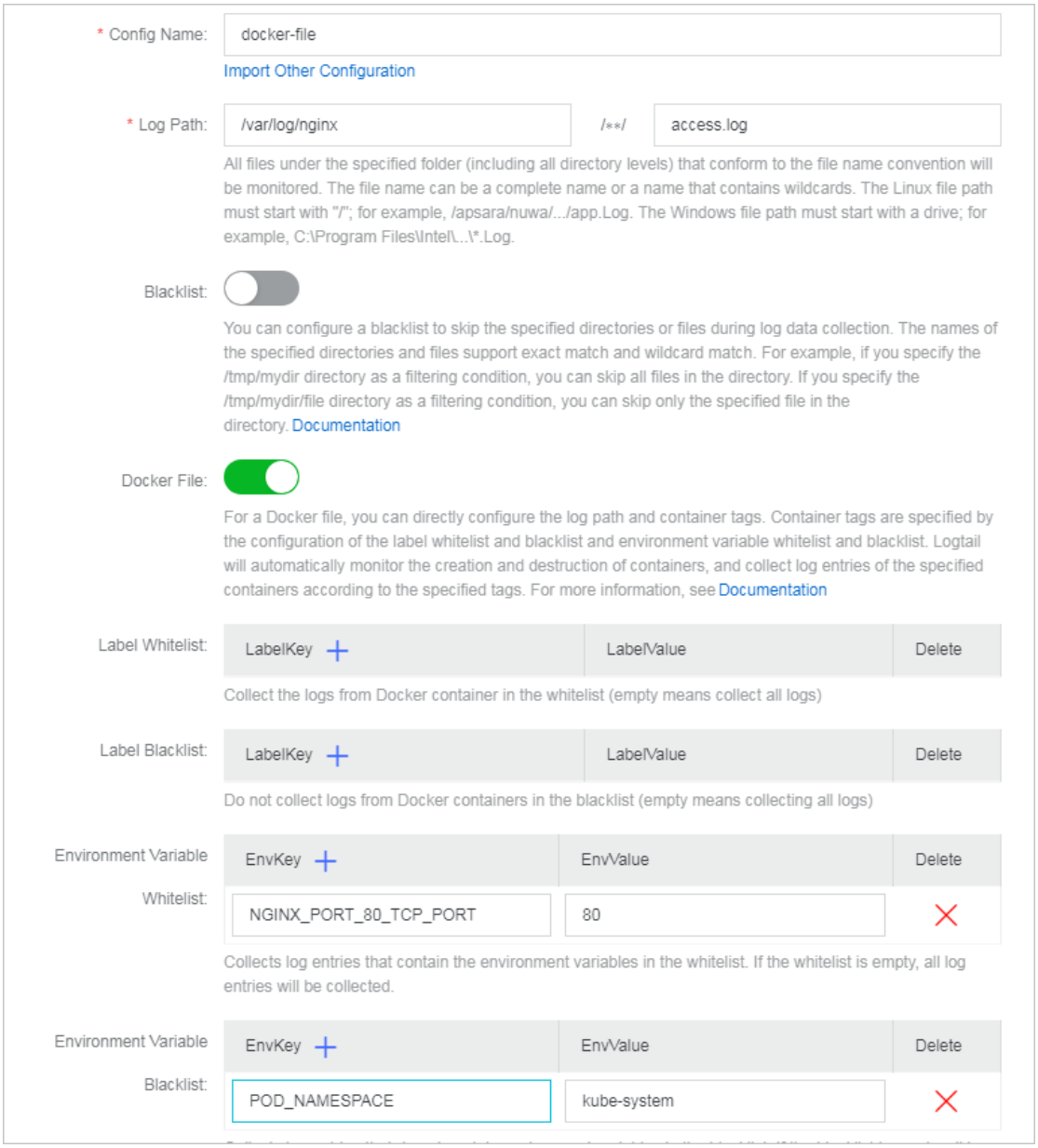
1. Obtain environment variables.

To view the environment variables of a container, you can log on to the host on which the container resides.



2. Create a Logtail configuration.

The following figure shows an example of a Logtail configuration. For more information about how to create a Logtail configuration that is used to collect logs in simple mode, see Collect logs by line.



Example 2: Filter containers based on the container label whitelist and the container label blacklist

Collect text logs from the containers whose container label is `io.kubernetes.container.name=nginx`. The log file path is `/var/log/nginx/access.log`. The logs are parsed in **simple mode**.

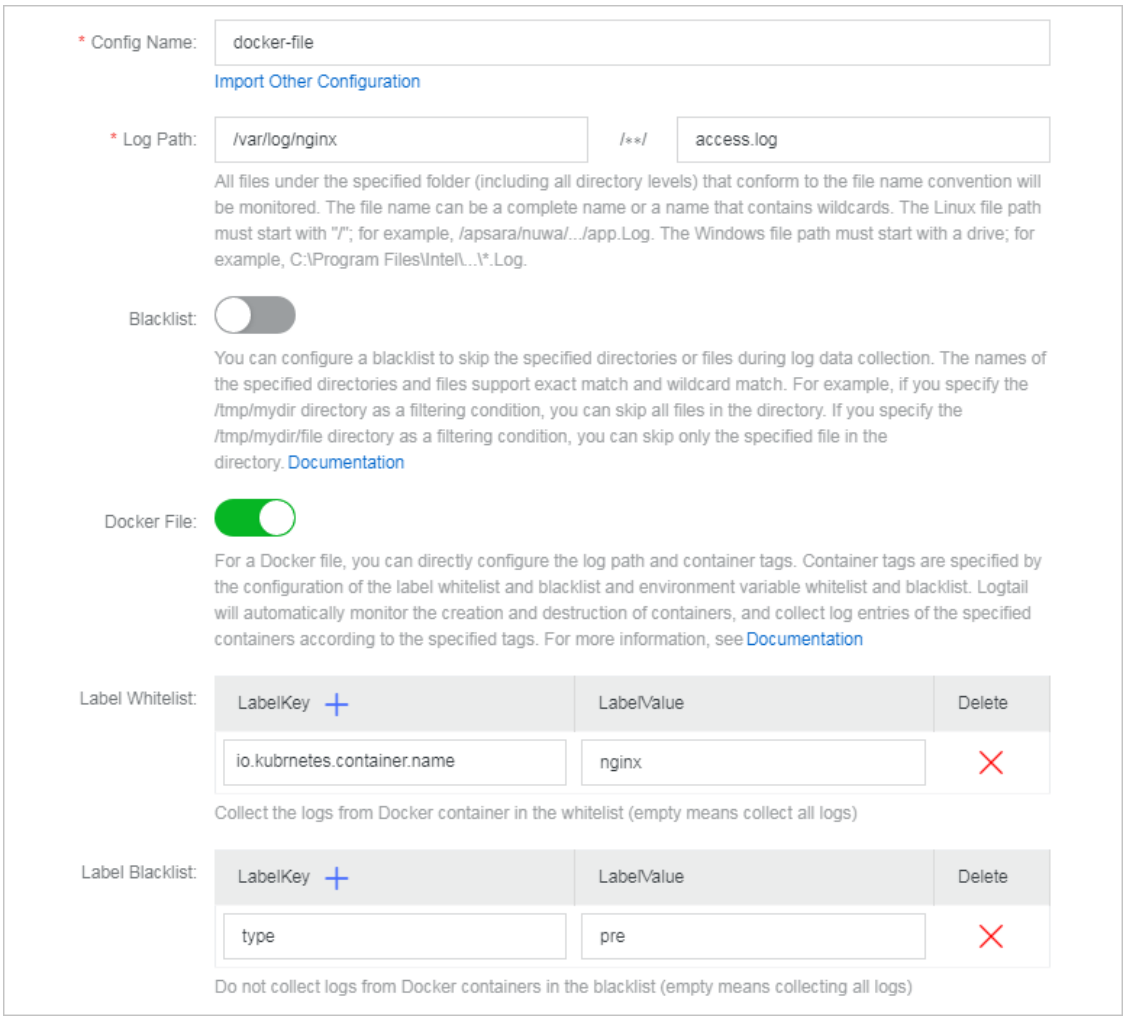
1. Obtain container labels.

   To view the container labels of a container, you can log on to the host on which the container resides.

```
"OnBuild": null,
"Labels": {
    "annotation.io.kubernetes.container.hash": "53073f5a",
    "annotation.io.kubernetes.container.restartCount": "0",
    "annotation.io.kubernetes.container.terminationMessagePath": "/dev/termination-log",
    "annotation.io.kubernetes.container.terminationMessagePolicy": "File",
    "annotation.io.kubernetes.pod.terminationGracePeriod": "30",
    "io.kubernetes.container.logpath": "/var/log/pods/ad00a078-4182                    585/nginx_0.log",
    "io.kubernetes.container.name": "nginx",
    "io.kubernetes.docker.type": "container",
    "io.kubernetes.pod.name": "example-foo-86ccd54874-r4mfh",
    "io.kubernetes.pod.namespace": "default",
    "io.kubernetes.pod.uid": "ad0
    "io.kubernetes.sandbox.id": "                                                    1dfa6da112969",
    "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
},
"StopSignal": "SIGTERM"
```


2. Create a Logtail configuration.

   The following figure shows an example of a Logtail configuration. For more information about how to create a Logtail configuration that is used to collect logs in simple mode, see Collect logs by line.



Example 3: Filter containers by using Kubernetes namespaces, pod names, and container names

Collect text logs from the nginx-log-demo-0 container in pods whose name starts with nginx-log-demo in the default namespace.

1. Obtain different levels of Kubernetes information.

- Obtain information about pods.

```
~/.kube » kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
nginx-log-demo-0-bxl79    1/1     Running   0          48d
nginx-log-demo-1-qmrqk    1/1     Running   0          48d
nginx-log-demo-2-7khv9    1/1     Running   0          48d
nginx-log-demo-3-j24xc    1/1     Running   0          48d
```

- Obtain information about namespaces.

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/psp: ack.privileged
  creationTimestamp: "2022-01-06T18:42:43Z"
  generateName: nginx-log-demo-0-
  labels:
    controller-uid: ae3eedc4-1667-458b-a6fe-39888576dbf4
    job-name: nginx-log-demo-0
  name: nginx-log-demo-0-bxl79
  namespace: default
  ownerReferences:
  - apiVersion: batch/v1
    blockOwnerDeletion: true
    controller: true
    kind: Job
    name: nginx-log-demo-0
    uid: ae3eedc4-1667-458b-a6fe-39888576dbf4
  resourceVersion: "50566856"
  uid: ee10fb7d-d989-47b3-bc2a-e9ffbe767849
spec:
  containers:
  - args:
    - --log-type=nginx
    - --stdout=true
    - --total-count=100000000000
    - --log-file-size=1000000000
    - --log-file-count=2
    - --logs-per-sec=2778
    command:
    - /bin/mock_log
    image: registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test:latest
    imagePullPolicy: Always
    name: nginx-log-demo-0
    resources:
```

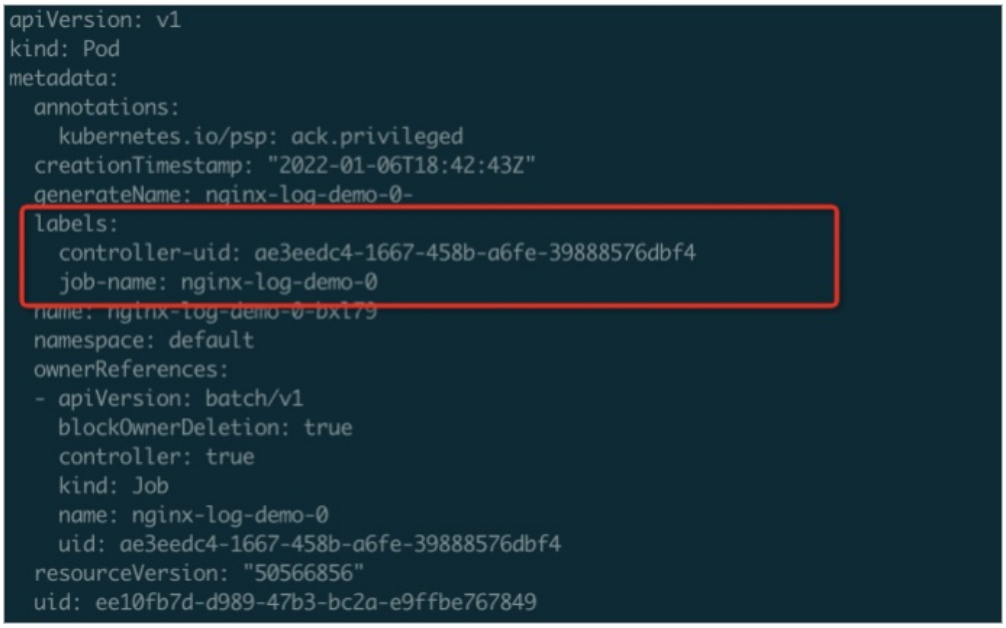
2. Create a Logtail configuration.

   For more information about how to create a Logtail configuration that is used to collect logs in simple mode, see Collect logs by line.

Example 4: Filter containers by using Kubernetes labels

Collect text logs from containers whose Kubernetes labels contain the job-name key and a specific value. The value starts with nginx-log-demo.

1. Obtain Kubernetes labels.



2. Create a Logtail configuration.

   For more information about how to create a Logtail configuration that is used to collect logs in simple mode, see Collect logs by line.

Default fields

The following table describes the fields that are included by default in each container text log.

Log field	Description
_image_name_	The name of the image.
_container_name_	The name of the container.
_pod_name_	The name of the pod.
namespace	The namespace of the pod.
_pod_uid_	The unique identifier of the pod.
_container_ip_	The IP address of the pod.

3.1.5.4. Use the Log Service console to collect container stdout and stderr in DaemonSet mode

This topic describes how to create a Logtail configuration in the Log Service console and use the Logtail configuration to collect container stdout and stderr in DaemonSet mode.

Prerequisites

- The Logtail component is installed. For more information, see Collect Kubernetes logs.
- A Logstore is created in the project that you use to install the Logtail component. For more information, see Create a Logstore.

Features

Logtail can collect container stdout and stderr, and then upload the stdout and stderr together with container metadata to Log Service. Logtail supports the following features:

- Collects stdout and stderr.
- Uses the container label whitelist to specify containers from which stdout and stderr are collected.
- Uses the container label blacklist to specify containers from which stdout and stderr are not collected.
- Uses the environment variable whitelist to specify containers from which stdout and stderr are collected.
- Uses the environment variable blacklist to specify containers from which stdout and stderr are not collected.
- Collects multi-line logs. For example, Logtail can collect Java stack logs.
- Automatically associates container metadata that needs to be uploaded together with the collected container stdout and stderr. The metadata includes container names, image names, pod names, namespaces, and environment variables.
- If a container runs in a Kubernetes cluster, Logtail also supports the following features:
  - Uses Kubernetes namespaces, pod names, and container names to specify containers from which stdout and stderr are collected.
  - Uses the Kubernetes label whitelist to specify containers from which stdout and stderr are collected.
  - Uses the Kubernetes label blacklist to specify containers from which stdout and stderr are not collected.
  - Automatically associates Kubernetes labels that need to be uploaded together with the collected container stdout and stderr.

Implementation

Logtail communicates with the domain socket of Docker. Logtail queries all Docker containers and identifies the containers from which stdout and stderr are collected by using the specified labels and environment variables. Logtail runs the `docker logs` command to collect logs from the specified containers.

When Logtail collects stdout and stderr from a container, Logtail periodically stores checkpoints to a checkpoint file. If Logtail is stopped and then started, Logtail collects logs from the last checkpoint.

Limits

- You can use the Log Service console to collect stdout and stderr in DaemonSet mode only if Logtail runs V0.16.0 or later and runs on Linux. For more information about Logtail versions and version updates, see Install Logtail in Linux.
- Logtail collects data from containers that use the Docker engine or containerd engine.
  - Docker: Logtail accesses the Docker engine in the */run/docker.sock* directory. Make sure that the directory exists and Logtail has the permissions to access the directory.
  - containerd: Logtail accesses the containerd engine in the */run/containerd/containerd.sock* directory. Make sure that the directory exists and Logtail has the permissions to access the directory.
- By default, the last multi-line log that is collected by Logtail is cached for 3 seconds. This prevents the multi-line log from being split into multiple logs due to output latency. You can change the cache time by modifying the `BeginLineTimeoutMs` parameter. We recommend that you do not specify a value less than 1000 with millisecond precision. If you specify a value that is less than 1000, an error may occur.
- If Logtail detects the `die` event on a container that is stopped, Logtail no longer collects stdout or stderr from the container. If collection latency exists, some stdout and stderr that are collected before the container is stopped may be lost.
- The logging driver collects stdout and stderr only in the JSON format from containers that use the Docker engine.
- By default, stdout and stderr that are collected from different containers by using the same Logtail configuration have the same context. If you want to specify a different context for the stdout and stderr that are collected from each container, you must create a Logtail configuration for each container.
- By default, the collected data is stored in the `content` field. Logtail can process the collected data. For more information, see Configure data processing methods.

Create a Logtail configuration

1. Log on to the Log Service console.
2. In the **Import Data** section, click **Kubernetes - Standard Output**.
3. Select a project and a Logstore. Then, click **Next**.

   In this example, select the project that you use to install the Logtail component and the Logstore that you create.
4. Click **Use Existing Machine Groups**.

   After you install the Logtail component, Log Service automatically creates a machine group named `k8s-group-${your_k8s_cluster_id}`. You can select this machine group.
5. Select the `k8s-group-${your_k8s_cluster_id}` machine group from **Source Server Groups** and move the machine group to **Applied Server Groups**. Then, click **Next**.

   > **Notice** If the heartbeat status of the machine group is **FAIL**, you can click **Automatic Retry**. If the issue persists, see *What do I do if no heartbeat connections are detected on Logtail?* in the FAQ.
6. In the Specify Data Source step, specify the data source and click **Next**.

   Configure the parameters that are used to collect logs in the **Plug-in Config** field. Example:

```
{
    "inputs":[
        {
            "type":"service_docker_stdout",
            "detail":{
                "Stdout":true,
                "Stderr":true,
                "IncludeContainerLabel":{
                    "LabelKey":"LabelValue"
                },
                "ExcludeContainerLabel":{
                    "LabelKey":"LabelValue"
                },
                "IncludeK8sLabel":{
                    "LabelKey":"LabelValue"
                },
                "ExcludeK8sLabel":{
                    "LabelKey":"LabelValue"
                },
                "IncludeEnv":{
                    "EnvKey":"EnvValue"
                },
                "ExcludeEnv":{
                    "EnvKey":"EnvValue"
                },
                "ExternalK8sLabelTag":{
                    "EnvKey":"EnvValue"
                },
                "ExternalEnvTag":{
                    "EnvKey":"EnvValue"
                },
                "K8sNamespaceRegex":"^(default|kube-system)$",
                "K8sPodRegex":"^(deploy.*)$",
                "K8sContainerRegex":"^(container1|container2)$"
            }
        }
    ]
}
```

Configure the following parameters:

- Data source type

  The type of the data source is fixed as service_docker_stdout.

- Parameters related to container filtering
  - For versions earlier than Logtail V1.0.29, you can filter containers only by using environment variables or container labels. You can take note of the following descriptions.

    The namespace of a Kubernetes cluster and the name of a container in the Kubernetes cluster can be mapped to container labels. The value of the LabelKey parameter for the namespace is `io.kubernetes.pod.namespace`. The value of the LabelKey parameter for the container name is `io.kubernetes.container.name`. We recommend that you use the two container labels to filter containers. If the container labels do not meet your business requirements, you can use the environment variable whitelist or the environment variable blacklist to filter containers. For example, the namespace of a pod is backend-prod, and the name of a container in the pod is worker-server. If you want the logs of the worker-server container to be collected, you can specify `"io.kubernetes.pod.namespace" : "backend-prod"` or `"io.kubernetes.container.name" : "worker-server"` in the container label whitelist.

Notice

- Container labels are retrieved by running the docker inspect command. Container labels are different from Kubernetes labels. For more information, see Obtain container labels.
- Environment variables are the same as the environment variables that are configured to start containers. For more information, see Obtain environment variables.
- Do not specify duplicate values for the LabelKey parameter. If you specify duplicate values for the LabelKey parameter, only one of the values takes effect.

Parameter	Type	Required	Description
IncludeLabel	Map (The values of the LabelKey and LabelValue parameters are strings.)	No	The container label whitelist. The whitelist specifies the containers from which stdout and stderr are collected. This parameter is empty by default, which indicates that stdout and stderr are collected from all containers. When you configure the container label whitelist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose container labels contain the keys specified by LabelKey are matched. ■ If the LabelValue parameter is not empty, containers whose container labels consist of the key-value pairs specified by LabelKey and LabelValue are matched. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the container labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the LabelValue parameter, regular expression matching is performed. For example, if you set the LabelKey parameter to *io.kubernetes.container.name* and set the LabelValue parameter to *^(nginx\|cube)$*, a container named nginx and a container named cube are matched. Key-value pairs are connected by using the OR operator. If a container label consists of one of the specified key-value pairs, the container to which the container label belongs is matched.

Parameter	Type	Required	Description
ExcludeLabel	Map (The values of the LabelKey and LabelValue parameters are strings.)	No	The container label blacklist. The blacklist specifies the containers from which stdout and stderr are not collected. This parameter is empty by default, which indicates that stdout and stderr are collected from all containers. When you configure the container label blacklist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose container labels contain the keys specified by LabelKey are filtered out. ■ If the LabelValue parameter is not empty, containers whose container labels consist of the key-value pairs specified by LabelKey and LabelValue are filtered out. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the container labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the LabelValue parameter, regular expression matching is performed. For example, if you set the LabelKey parameter to *io.kubernetes.container.name* and set the LabelValue parameter to *^(nginx\|cube)$*, a container named nginx and a container named cube are matched. Key-value pairs are connected by using the OR operator. If a container label consists of one of the specified key-value pairs, the container to which the container label belongs is filtered out.

Parameter	Type	Required	Description
IncludeEnv	Map (The values of the EnvKey and EnvValue parameters are strings.)	No	The environment variable whitelist. The whitelist specifies the containers from which stdout and stderr are collected. This parameter is empty by default, which indicates that stdout and stderr are collected from all containers. When you configure the environment variable whitelist, the EnvKey parameter is required, and the EnvValue parameter is optional. ■ If the EnvValue parameter is empty, containers whose environment variables contain the keys specified by EnvKey are matched. ■ If the EnvValue parameter is not empty, containers whose environment variables consist of the key-value pairs specified by EnvKey and EnvValue are matched. By default, string matching is performed for the values of the EnvValue parameter. Containers are matched only if the values of the environment variables are the same as the values of the EnvValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the EnvValue parameter, regular expression matching is performed. For example, if you set the EnvKey parameter to *NGINX_SERVICE_PORT* and set the EnvValue parameter to *^(80\|6379)$*, containers whose port number is 80 and containers whose port number is 6379 are matched. Key-value pairs are connected by using the OR operator. If an environment variable consists of one of the specified key-value pairs, the container to which the environment variable belongs is matched.

Parameter	Type	Required	Description
ExcludeEnv	Map (The values of the EnvKey and EnvValue parameters are strings.)	No	The environment variable blacklist. The blacklist specifies the containers from which stdout and stderr are not collected. This parameter is empty by default, which indicates that stdout and stderr are collected from all containers. When you configure the environment variable blacklist, the EnvKey parameter is required, and the EnvValue parameter is optional. ■ If the EnvValue parameter is empty, containers whose environment variables contain the keys specified by EnvKey are filtered out. ■ If the EnvValue parameter is not empty, containers whose environment variables consist of the key-value pairs specified by EnvKey and EnvValue are filtered out. By default, string matching is performed for the values of the EnvValue parameter. Containers are matched only if the values of the environment variables are the same as the values of the EnvValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`) for the EnvValue parameter, regular expression matching is performed. For example, if you set the EnvKey parameter to *NGINX_SERVICE_PORT* and set the EnvValue parameter to *^(80\|6379)$*, containers whose port number is 80 and containers whose port number is 6379 are matched. Key-value pairs are connected by using the OR operator. If an environment variable consists of one of the specified key-value pairs, the container to which the environment variable belongs is filtered out.

- For Logtail V1.0.29 or later, we recommend that you use different levels of Kubernetes information, such as pod names, namespaces, container names, and labels to filter containers.

  Note If you change Kubernetes labels when Kubernetes control resources, such as Deployments, are running, the operational pod is not restarted. Therefore, the pod cannot detect the change. This may cause a matching rule to become invalid. When you configure the Kubernetes label whitelist or the Kubernetes label blacklist, we recommend that you use the Kubernetes labels of pods. For more information about Kubernetes labels, see Labels and Selectors.

Parameter	Type	Required	Description

Parameter	Type	Required	Description
IncludeK8sLabel	Map (The values of the LabelKey and LabelValue parameters are strings.)	No	The Kubernetes label whitelist. The whitelist specifies the containers from which stdout and stderr are collected. When you configure the Kubernetes label whitelist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose Kubernetes labels contain the keys specified by LabelKey are matched. ■ If the LabelValue parameter is not empty, containers whose Kubernetes labels consist of the key-value pairs specified by LabelKey and LabelValue are matched. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the Kubernetes labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`), regular expression matching is performed. For example, if you set the **LabelKey** parameter to *app* and set the **LabelValue** parameter to *^(test1\|test2)$*, containers whose Kubernetes labels consist of app:test1 or app:test2 are matched. Key-value pairs are connected by using the OR operator. If a Kubernetes label consists of one of the specified key-value pairs, the container to which the Kubernetes label belongs is matched.

Parameter	Type	Required	Description
ExcludeK8sLabel	Map (The values of the LabelKey and LabelValue parameters are strings.)	No	The Kubernetes label blacklist. The blacklist specifies the containers from which stdout and stderr are not collected. When you configure the Kubernetes label blacklist, the LabelKey parameter is required, and the LabelValue parameter is optional. ■ If the LabelValue parameter is empty, containers whose Kubernetes labels contain the keys specified by LabelKey are filtered out. ■ If the LabelValue parameter is not empty, containers whose Kubernetes labels consist of the key-value pairs specified by LabelKey and LabelValue are filtered out. By default, string matching is performed for the values of the LabelValue parameter. Containers are matched only if the values of the Kubernetes labels are the same as the values of the LabelValue parameter. If you specify a value that starts with a caret (`^`) and ends with a dollar sign (`$`), regular expression matching is performed. For example, if you set the **LabelKey** parameter to *app* and set the **LabelValue** parameter to *^(test1\|test2)$*, containers whose Kubernetes labels consist of app:test1 or app:test2 are matched. Key-value pairs are connected by using the OR operator. If a Kubernetes label consists of one of the specified key-value pairs, the container to which the Kubernetes label belongs is filtered out.
K8sNamespaceRegex	string	No	The namespace. The namespace specifies the containers from which stdout and stderr are collected. Regular expression matching is supported. For example, if you specify "K8sNamespaceRegex":"^(default|nginx)$", all containers in the nginx and default namespaces are matched.
K8sPodRegex	string	No	The pod name. The pod name specifies the containers from which stdout and stderr are collected. Regular expression matching is supported. For example, if you specify "K8sPodRegex":"^(nginx-log-demo.*)$",, all containers in the pod whose name starts with nginx-log-demo are matched.

Parameter	Type	Required	Description
K8sContainerRegex	string	No	The container name. The container name specifies the containers from which stdout and stderr are collected. Regular expression matching is supported. Kubernetes container names are defined in spec.containers. For example, if you specify "K8sContainerRegex":"^(container-test)$", all containers whose name is container-test are matched.

- Parameters related to log labels

  For Logtail V1.0.29 or later, we recommend that you specify environment variables or Kubernetes labels for logs as log labels.

Parameter	Type	Required	Description
ExternalEnvTag	Map (The values of the EnvKey and EnvValue parameters are strings.)	No	After you specify environment variables as log labels, Log Service adds environment variable-related fields to logs. For example, if you set the **EnvKey** parameter to *VERSION* and set the **EnvValue** parameter to *env_version*, Log Service adds the __tag__:__env_version__: v1.0.0 field to logs if the environment variable configurations of a container include `VERSION=v1.0.0`.
ExternalK8sLabelTag	Map (The values of the LabelKey and LabelValue parameters are strings.)	No	After you specify Kubernetes labels as log labels, Log Service adds Kubernetes label-related fields to logs. For example, if you set the **LabelKey** parameter to *app* and set the **LabelValue** parameter to `k8s_label_app`, Log Service adds the __tag__:__k8s_label_app__: serviceA field to logs if the label configurations of a Kubernetes cluster include `app=serviceA`.

- Other parameters

Parameter	Type	Required	Description
Stdout	boolean	No	Specifies whether to collect stdout. This parameter is empty by default, which indicates that stdout is collected.
Stderr	boolean	No	Specifies whether to collect stderr. This parameter is empty by default, which indicates that stderr is collected.
BeginLineRegex	string	No	The regular expression that is used to match the beginning of the first line of a log. This parameter is empty by default, which indicates that each line is regarded as a log. If the beginning of a line matches the specified regular expression, the line is regarded as the first line of a new log. If the beginning of a line does not match the specified regular expression, the line is regarded as a part of the last log.

Parameter	Type	Required	Description
BeginLineTimeoutMs	int	No	The timeout period for matching the beginning of the first line of a log based on the specified regular expression. This parameter is empty by default, which indicates that the timeout period is 3,000 milliseconds. If no new log is generated within 3,000 milliseconds, Logtail stops matching the beginning of the first line of a log and uploads the last log to Log Service.
BeginLineCheckLength	int	No	The size of the beginning of the first line of a log that matches the specified regular expression. This parameter is empty by default, which indicates that the size of the beginning of the first line of a log is 10,240 bytes. You can configure this parameter to check whether the beginning of the first line of a log matches the specified regular expression. We recommend that you configure this parameter to improve the match efficiency.
MaxLogSize	int	No	The maximum size of a log. This parameter is empty by default, which indicates that the maximum size of a log is 524,288 bytes. If the size of a log exceeds the value of this parameter, Logtail stops matching the beginning of the first line of a log and uploads the log to Log Service.
StartLogMaxOffset	int	No	The maximum size of historical data that can be traced the first time Logtail collects logs from a log file. Valid values: [131072,1048576]. Unit: bytes. This parameter is empty by default. In this case, the maximum size of historical data that can be traced is 131,072 bytes, equivalent to 128 KB.

7. Preview data, configure indexes, and then click **Next**.

   By default, full-text indexing is enabled for Log Service. You can also configure field indexes based on collected logs in manual or automatic mode. For more information, see Configure indexes.

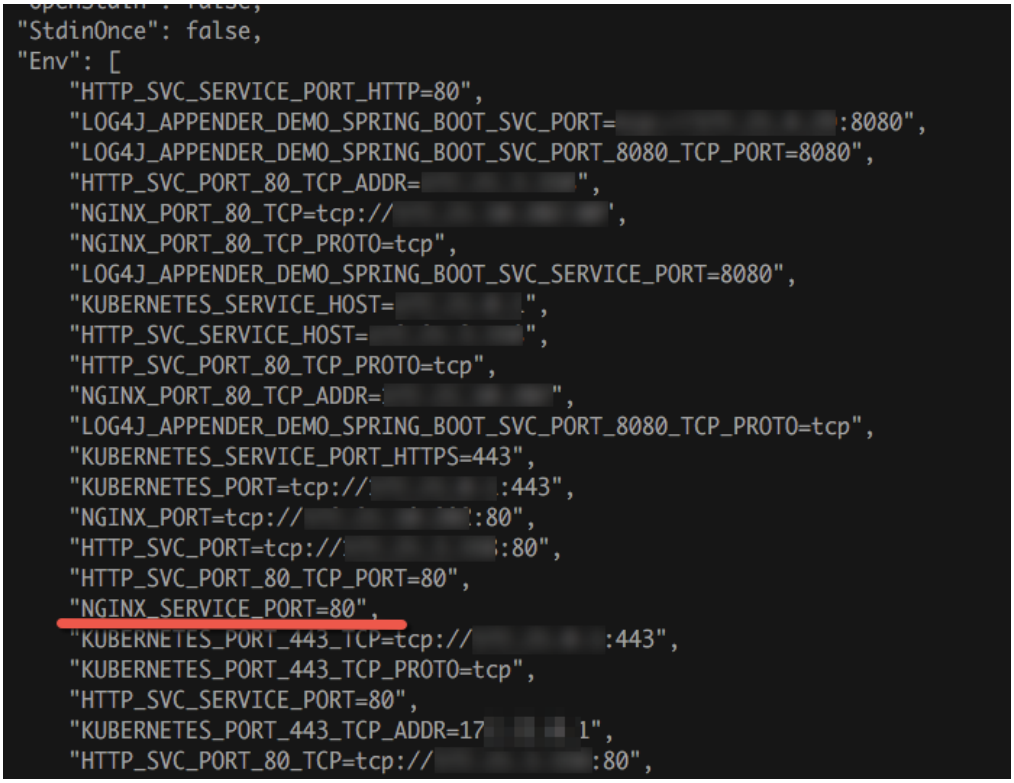
Examples of Logtail configurations for single-line logs

Example 1: Filter containers based on the environment variable whitelist and the environment variable blacklist

Collect stdout and stderr from the containers whose environment variable configurations include `NGINX_SERVICE_PORT=80` but exclude `POD_NAMESPACE=kube-system`.

1. Obtain environment variables.

   To view the environment variables of a container, you can log on to the host on which the container resides.



2. Create a Logtail configuration.

   Example:

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeEnv": {
                    "NGINX_SERVICE_PORT": "80"
                },
                "ExcludeEnv": {
                    "POD_NAMESPACE": "kube-system"
                }
            }
        }
    ]
}
```

Example 2: Filter containers based on the container label whitelist and the container label blacklist

Collect stdout and stderr from the containers whose container label is `io.kubernetes.container.name=nginx`.

1. Obtain container labels.

   To view the container labels of a container, you can log on to the host on which the container resides.

```
"OnBuild": null,
"Labels": {
    "annotation.io.kubernetes.container.hash": "53073f5a",
    "annotation.io.kubernetes.container.restartCount": "0",
    "annotation.io.kubernetes.container.terminationMessagePath": "/dev/termination-log",
    "annotation.io.kubernetes.container.terminationMessagePolicy": "File",
    "annotation.io.kubernetes.pod.terminationGracePeriod": "30",
    "io.kubernetes.container.logpath": "/var/log/pods/ad00a078                                    85/nginx_0.log",
    "io.kubernetes.container.name": "nginx",
    "io.kubernetes.docker.type": "container",
    "io.kubernetes.pod.name": "example-foo-86ccd54874-r4mfh",
    "io.kubernetes.pod.namespace": "default",
    "io.kubernetes.pod.uid": "ad00a07
    "io.kubernetes.sandbox.id": "5216                                                a8d0b6891dfa6da112969",
    "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
},
"StopSignal": "SIGTERM"
```


2. Create a Logtail configuration.

   Example:

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeLabel": {
                    "io.kubernetes.container.name": "nginx"
                }
            }
        }
    ]
}
```

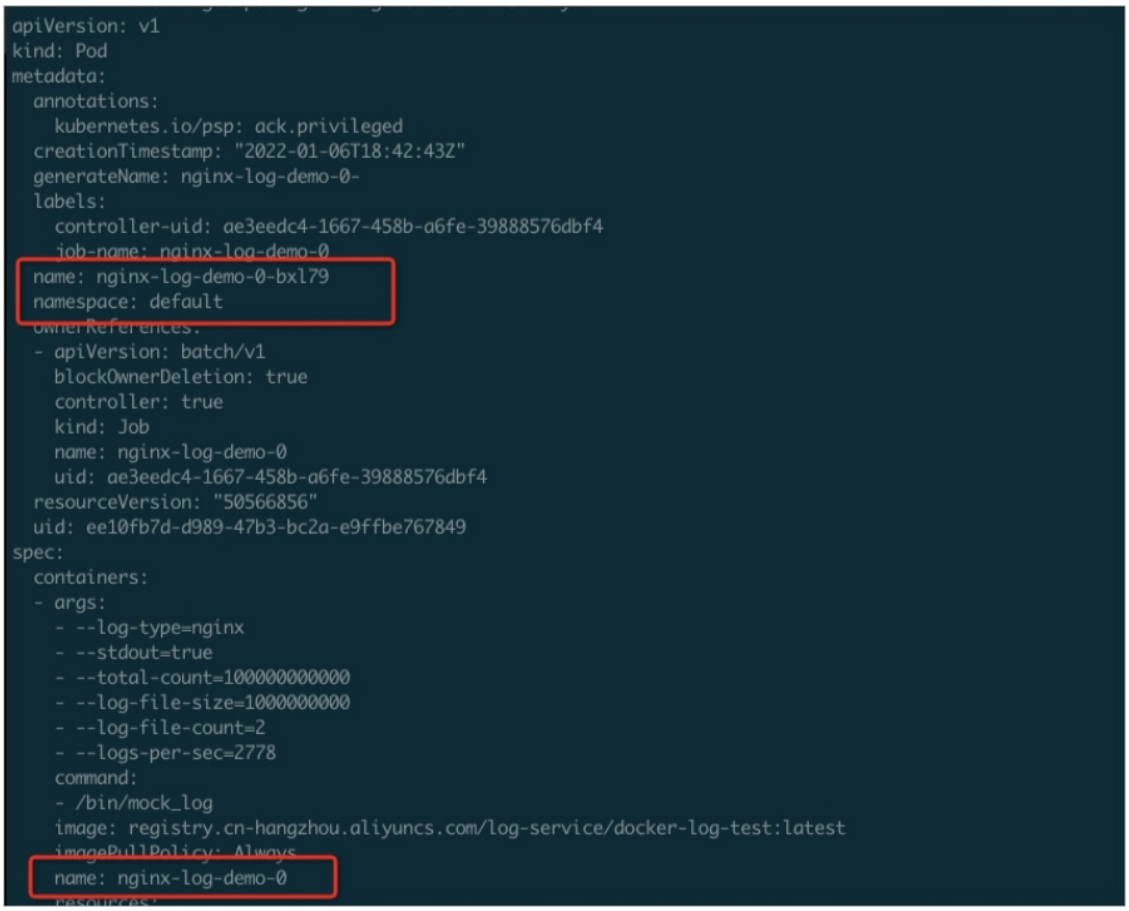
Example 3: Filter containers by using Kubernetes namespaces, pod names, and container names

Collect stdout and stderr from the nginx-log-demo-0 container in pods whose name starts with nginx-log-demo in the default namespace.

1. Obtain different levels of Kubernetes information.
   i. Obtain information about pods.

```
~/.kube » kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
nginx-log-demo-0-bxl79  1/1     Running   0          48d
nginx-log-demo-1-qmrqk  1/1     Running   0          48d
nginx-log-demo-2-7khv9  1/1     Running   0          48d
nginx-log-demo-3-j24xc  1/1     Running   0          48d
```

ii. Obtain information about namespaces.



2. Create a Logtail configuration.

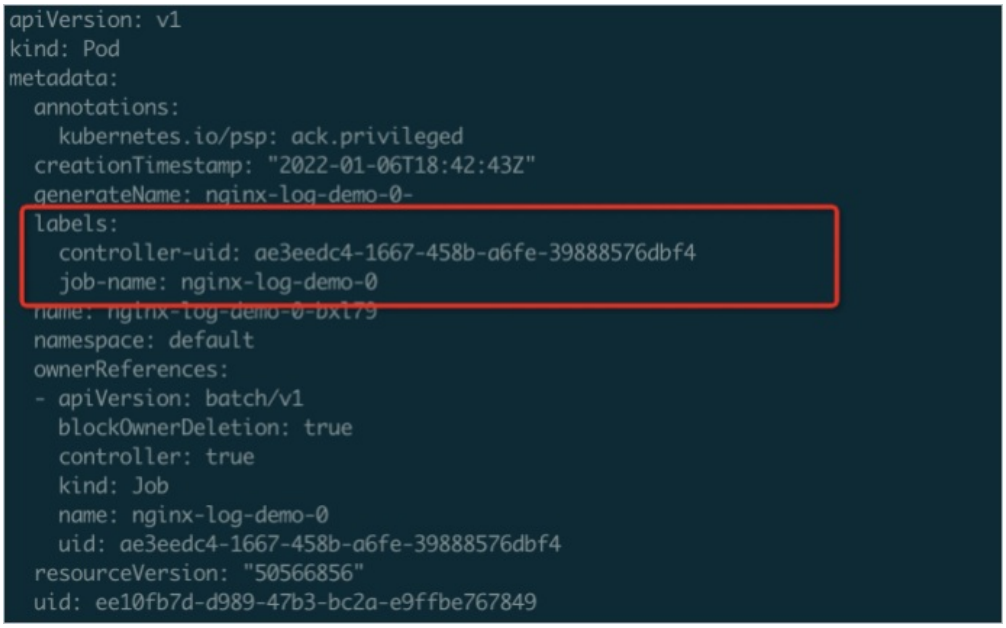
   Example:

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "K8sNamespaceRegex":"^(default)$",
                "K8sPodRegex":"^(nginx-log-demo.*)$",
                "K8sContainerRegex":"^(nginx-log-demo-0)$"
            }
        }
    ]
}
```

Example 4: Filter containers by using Kubernetes labels

Collect stdout and stderr from containers whose Kubernetes labels contain the job-name key and a specific value. The value starts with nginx-log-demo.

1. Obtain Kubernetes labels.



2. Create a Logtail configuration.

   Example:

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeK8sLabel":{
                    "job-name":"^(nginx-log-demo.*)$"
                }
            }
        }
    ]
}
```

Examples of Logtail configurations for multi-line logs

Java exception stack logs are multi-line logs. You can create a Logtail configuration to collect the Java exception stack logs based on the following descriptions:

- Sample logs

```
2021-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoCon
troller : service start
2021-02-03 14:18:41.969 ERROR [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoCon
troller : java.lang.NullPointerException
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193
)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:199)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96)
...
2021-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoCon
troller : service start done
```

- Logtail configuration

Collect the Java exception stack logs of the containers whose container label is `app=monitor`. The Java exception stack logs start with a date that is in a fixed format. Logtail matches only the first 10 bytes of each line to improve match efficiency. After the logs are collected and sent to Log Service, Log Service uses regular expressions to parse the logs into fields such as time, level, module, thread, and message.

- inputs is required and is used to configure the data collection settings for the Logtail configuration. You must configure inputs based on your data source.

  > **Note** You can specify only one type of data source in inputs.

- processors is optional and is used to configure the data processing settings for the Logtail configuration. You can specify one or more processing methods. For more information, see Configure data processing methods.

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
],
"processors": [
    {
        "type": "processor_regex",
        "detail": {
            "SourceKey": "content",
            "Regex": "(\\d+-\\d+-\\d+ \\d+:\\d+:\\d+\\.\\d+)\\s+(\\w+)\\s+\\[([^]]+)]\\s+\\[([^]]+)
]\\s+([\\s\\S]*)",
            "Keys": [
                "time",
                "level",
                "module",
                "thread",
                "message"
            ],
            "NoKeyError": true,
            "NoMatchError": true,
            "KeepSource": false
        }
    }
]
}
```

- Parsed logs

  For example, if the collected log is `2018-02-03 14:18:41.968 INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoController : service start done`, the log is parsed into the following fields:

```
__tag__:__hostname__:logtail-dfgef
_container_name_:monitor
_image_name_:example.com-hangzhou.aliyuncs.xxxxxxxxxxxxxx
_namespace_:default
_pod_name_:monitor-6f54bd5d74-rtzc7
_pod_uid_:7f012b72-04c7-11e8-84aa-00163f00c369
_source_:stdout
_time_:2018-02-02T14:18:41.979147844Z
time:2018-02-02 02:18:41.968
level:INFO
module:spring-cloud-monitor
thread:nio-8080-exec-4
class:c.g.s.web.controller.DemoController
message:service start done
```

Log fields

The following table describes the fields that are uploaded by default for each log in a Kubernetes cluster.

Log field	Description
time	The time at which the data is uploaded. Example: `2021-02-02T02:18:41.979147844Z`.
source	The type of the data source. Valid values: stdout and stderr.
_image_name_	The name of the image.
_container_name_	The name of the container.
_pod_name_	The name of the pod.
namespace	The namespace of the pod.
_pod_uid_	The unique identifier of the pod.
_container_id_	The IP address of the pod.

3.1.5.5. Use CRDs to collect container logs in DaemonSet mode

After you install Logtail in a container in DaemonSet mode, you can use a custom resource definition (CRD) to create a Logtail configuration and use the Logtail configuration to collect container logs.

Prerequisites

The alibaba-log-controller component is installed. For more information, see Collect Kubernetes logs.

Implementation

The following list describes the process in which logs are collected by using a CRD:

1. The `kubectl` tool or other tools are used to apply an AliyunLogConfig CRD.
2. The alibaba-log-controller detects the update in CRD configurations.
3. The alibaba-log-controller sends requests to Log Service to create a Logstore, create a Logtail configuration, and apply the Logtail configuration to a machine group based on the content of the CRD and the status of the Logtail configurations in Log Service.
4. Logtail periodically sends a request to the server on which the Logtail configuration is created to obtain the

new or updated Logtail configuration and perform hot reloading.

5. Logtail collects stdout and stderr logs or text logs from each container based on the obtained Logtail configuration.
6. Logtail sends the collected container logs to Log Service.

Limits

- Limits on text log collection
  - If Logtail detects the `die` event on a container that is stopped, Logtail stops collecting text logs from the container. If collection latency occurs, some text logs that are generated before the container is stopped may be lost.
  - Logtail cannot access the symbolic link of a container. You must specify an actual path as the collection directory.
  - If a volume is mounted on the data directory of a container, Logtail cannot collect data from the parent directory of the data directory. You must specify the complete path of the data directory as the collection directory.

    For example, if a volume is mounted on the */var/log/service* directory and you set the collection directory to */var/log*, Logtail cannot collect logs from the /var/log directory. You must specify */var/log/service* as the collection directory.
  - By default, Kubernetes mounts the root directory of the host on the `/logtail_host` directory of the Logtail container. If you want to collect text logs from the host, you must specify `/logtail_host` as the prefix of the log file path.

    For example, if you want to collect logs from the `/home/logs/app_log/` directory of the host, you must specify `/logtail_host/home/logs/app_log/` as the log file path.
  - For Docker containers, only overlay and overlay2 storage drivers are supported. If other storage drivers are used, you must mount a volume on the directory of logs. Then, a temporary directory is generated.
- Limits on stdout and stderr log collection

  The logging driver collects stdout and stderr logs only in the JSON format from containers that use the Docker engine.
- General limits

  Logtail collects data from containers that use the Docker engine or containerd engine.
  - Docker: Logtail accesses the Docker engine in the */run/docker.sock* directory. Make sure that the directory exists and Logtail has the permissions to access the directory.
  - containerd: Logtail accesses the containerd engine in the */run/containerd/containerd.sock* directory. Make sure that the directory exists and Logtail has the permissions to access the directory.

Create a Logtail configuration

To create a Logtail configuration, you need to only create an AliyunLogConfig CRD. After you create a Logtail configuration, the system automatically applies the Logtail configuration. If you want to delete the Logtail configuration, you need to only delete the CRD.

1. Log on to your Kubernetes cluster.
2. Run the following command to create a YAML file.

   In this example, the file name is *cube.yaml*. Replace the file name with an actual file name.

   ```
   vim cube.yaml
   ```
3. Enter the following script in the YAML file and configure the parameters based on your business scenario.

Notice

- The value of the configName parameter must be unique in the Log Service project that you use.
- If multiple CRDs are associated with the same Logtail configuration, the Logtail configuration is affected when you delete or modify one of the CRDs. After the deletion or modification, the status of the other CRDs that are associated with the Logtail configuration becomes inconsistent with the status of the Logtail configuration in Log Service.

```
apiVersion: log.alibabacloud.com/v1alpha1      # The default value is used. You do not need to mod
ify this parameter.
kind: AliyunLogConfig                          # The default value is used. You do not need to mod
ify this parameter.
metadata:
  name: simple-stdout-example                  # The name of the resource. The name must be unique
in the current Kubernetes cluster.
spec:
  project: k8s-my-project                      # Optional. The name of the project. The default va
lue is the name of the project that you use to install the Logtail component.
  logstore: k8s-stdout                         # The name of the Logstore. If the Logstore that yo
u specify does not exist, Log Service automatically creates a Logstore.
  shardCount: 2                                # Optional. The number of shards. Valid values: 1 t
o 10. Default value: 2.
  lifeCycle: 90                                # Optional. The data retention period of the Logsto
re. The value of this parameter takes effect only when you create a Logstore. Valid values: 1 to 3
650. Unit: days. Default value: 90. The value 3650 specifies that log data is permanently stored i
n the Logstore.
  logtailConfig:                               # The Logtail configuration.
    inputType: plugin                          # The type of the data source. Valid values: file a
nd plugin. file specifies text logs. plugin specifies stdout and stderr logs.
    configName: simple-stdout-example          # The name of the Logtail configuration. The name m
ust be the same as the resource name that is specified in metadata.name.
    inputDetail:                               # The detailed settings of the Logtail configuratio
n. For more information, see the following configuration examples.
      ...
```

Parameter	Type	Required	Description
project	string	No	The name of the project. The default value is the name of the project that you use to install the Logtail component.
logstore	string	Yes	The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatically creates a Logstore.
shardCount	int	No	The number of shards. Valid values: 1 to 10. Default value: 2.

Parameter	Type	Required	Description
lifeCycle	int	No	The data retention period of the Logstore. Valid values: 1 to 3650. Unit: days. Default value: 90. The value 3650 specifies that log data is permanently stored in the Logstore. **Notice** The value of this parameter takes effect only when you create a Logstore. If you change the value of the lifeCycle parameter for an existing Logstore that is specified by the logstore parameter, the new value does not take effect.
machineGroups	array	No	The machine group to which the Logtail configuration is applied. The default value is the machine group named `k8s-group-${your_k8s_cluster_id}`. This machine group is automatically created by Log Service when you install the Logtail component.
logtailConfig	object	Yes	The detailed settings of the Logtail configuration. In most cases, you need to configure only the inputType, configName, and inputDetail parameters.

4. Run the following command to apply the Logtail configuration.

   In this example, the file name is *cube.yaml*. Replace the file name with an actual file name.

   ```
   kubectl apply -f cube.yaml
   ```

   After the Logtail configuration is applied, Logtail collects stdout and stderr logs or text logs from each container, and then sends the collected logs to Log Service.

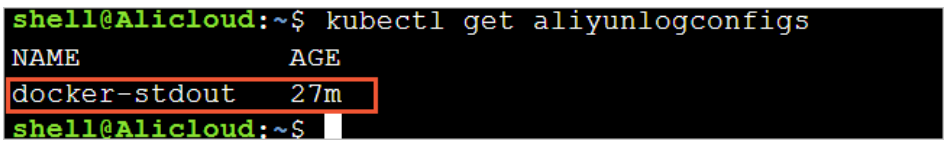
View Logtail configurations

You can view Logtail configurations in the Log Service console or by using CRDs. For more information about how to view Logtail configurations in the Log Service console, see Manage a Logtail configuration.

Notice If you modify the settings of a Logtail configuration in the Log Service console and view the Logtail configuration by using a CRD, the modification is not displayed in the output of the CRD. If you modify the settings of a Logtail configuration by using a CRD and view the Logtail configuration in the Log Service console, the modification is displayed in the Log Service console.

View all Logtail configurations in the current Kubernetes cluster

You can run the `kubectl get aliyunlogconfigs` command to view all Logtail configurations. The following figure shows the output.



View the details and status of a Logtail configuration

You can run the `kubectl get aliyunlogconfigs config_name -o yaml` command to view the details and status of a Logtail configuration. The *config_name* parameter in the command specifies the name of the Logtail configuration that you want to view. Replace the configuration name with an actual configuration name. The following figure shows the output.

The status and statusCode parameters in the output indicate the status of the Logtail configuration.

- If the value of the statusCode parameter is 200, the Logtail configuration is applied.
- If the value of the statusCode parameter is not 200, the Logtail configuration fails to be applied.

Examples of Logtail configurations that are used to collect stdout and stderr logs

If you want to collect container stdout and stderr logs, you must set the inputType parameter to `plugin` and add detailed settings to the `plugin` field of the `inputDetail` parameter. For more information about the parameters and the descriptions of the parameters, see Use the Log Service console to collect container stdout and stderr in DaemonSet mode.

Example 1: Collect container stdout and stderr logs in simple mode

Collect stdout and stderr logs from all containers except the containers whose environment variable configurations include `COLLECT_STDOUT_FLAG=false`. To view the environment variables of a container, you can log on to the host on which the container resides. CRD configuration example:

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: simple-stdout-example
spec:
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatical
ly creates a Logstore.
  logstore: k8s-stdout
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. If you want to collect stdout and stderr logs, you must set the val
ue to plugin.
    inputType: plugin
    # The name of the Logtail configuration. The name must be the same as the resource name that is sp
ecified in metadata.name.
    configName: simple-stdout-example
    inputDetail:
      plugin:
        inputs:
          -
            # input type
            type: service_docker_stdout
            detail:
              # The settings that allow Logtail to collect both stdout and stderr logs.
              Stdout: true
              Stderr: true
              # The environment variable denylist. In this example, stdout and stderr logs are collect
ed from all containers except the containers whose environment variable configurations include COLLECT
_STDOUT_FLAG=false.
              ExcludeEnv:
                COLLECT_STDOUT_FLAG: "false"
```

Example 2: Collect container stdout and stderr logs in simple mode and process the logs by using regular expressions

To view the environment variables of a container, you can log on to the host on which the container resides.

Collect the access logs of Grafana from containers in simple mode and parse the access logs into structured data by using regular expressions. The environment variable configurations of the container where Grafana resides include `GF_INSTALL_PLUGINS=grafana-piechart-....`. To view the environment variables of the container, you can log on to the host on which the container resides.

- CRD configuration

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: regex-stdout-example
spec:
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automati
cally creates a Logstore.
  logstore: k8s-stdout-regex
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. If you want to collect stdout logs, you must set the value to pl
ugin.
    inputType: plugin
    # The name of the Logtail configuration. The name must be the same as the resource name that is
specified in metadata.name.
    configName: regex-stdout-example
    inputDetail:
      plugin:
        inputs:
          -
            # input type
            type: service_docker_stdout
            detail:
              # The settings that allow Logtail to collect only stdout logs.
              Stdout: true
              Stderr: false
              # The environment variable allowlist. In this example, stdout logs are collected only
from containers whose environment variable configurations include a key of GF_INSTALL_PLUGINS.
              IncludeEnv:
                GF_INSTALL_PLUGINS: ''
        processors:
          -
            # The settings that allow Logtail to parse the collected stdout logs by using a regular
expression.
            type: processor_regex
            detail:
              # The name of the source field. By default, the collected stdout logs are stored in t
he content field.
              SourceKey: content
              # The regular expression that is used to extract log content.
              Regex: 't=(\d+-\d+-\w+:\d+:\d+\+\d+) lvl=(\w+) msg="([^"]+)" logger=(\w+) userId=(\w+
) orgId=(\w+) uname=(\S*) method=(\w+) path=(\S+) status=(\d+) remote_addr=(\S+) time_ms=(\d+) size
=(\d+) referer=(\S*).*'
              # The keys that you want to extract from logs.
              Keys: ['time', 'level', 'message', 'logger', 'userId', 'orgId', 'uname', 'method', 'p
ath', 'status', 'remote_addr', 'time_ms', 'size', 'referer']
              # The settings that allow Logtail to retain the source field.
              KeepSource: true
              # The settings that allow Logtail to report an error when the specified source field
does not exist.
              NoKeyError: true
              # The settings that allow Logtail to report an error when the specified regular expre
ssion does not match the value of the specified source field.
              NoMatchError: true
```

- Raw log

```
t=2018-03-09T07:14:03+0000 lvl=info msg="Request Completed" logger=context userId=0 orgId=0 uname=
method=GET path=/ status=302 remote_addr=172.16.64.154 time_ms=0 size=29 referer=
```

- Parsed log

Examples of Logtail configurations that are used to collect text logs

If you want to collect container text logs, you must set the `inputType` parameter to `file` and add detailed settings to the `inputDetail` parameter. For more information about the parameters and the descriptions of the parameters, see Use the Log Service console to collect container text logs in DaemonSet mode.

Example 1: Collect container text logs in simple mode

Collect container text logs whose environment variable configurations include a key of `ALIYUN_LOGTAIL_USER_DEFINED_ID`. The log file path is */data/logs/app_1/simple.LOG*.

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: simple-file-example
spec:
   # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatica
lly creates a Logstore.
  logstore: k8s-file
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. If you want to collect text logs, you must set the value to file.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that is sp
ecified by the metadata.name parameter.
    configName: simple-file-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in simple mode.
      logType: common_reg_log
      # The log file path.
      logPath: /data/logs/app_1
      # The log file name. You can use wildcard characters such as asterisks (*) and question marks (?
) when you specify the log file name. Example: log_*.log.
      filePattern: simple.LOG
      # If you want to collect container text logs, you must set the dockerFile parameter to true.
      dockerFile: true
      # The environment variable allowlist. In this example, text logs are collected only from contain
ers whose environment variable configurations include a key of ALIYUN_LOGTAIL_USER_DEFINED_ID.
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

Example 2: Collect container text logs in full regex mode

A Java program generates a multi-line log that contains error stack information. You can collect the log in full regex mode and specify a regular expression that is used to match the start part in the first line of the log in the Logtail configuration.

- Sample log

```
[2018-05-11T20:10:16,000] [INFO] [SessionTracker] [SessionTrackerImpl.java:148] Expiring sessions
java.sql.SQLException: Incorrect string value: '\xF0\x9F\x8E\x8F",...' for column 'data' at row 1
at org.springframework.jdbc.support.AbstractFallbackSQLExceptionTranslator.translate(AbstractFallba
ckSQLExceptionTranslator.java:84)
at org.springframework.jdbc.support.AbstractFallbackSQLException
```

- CRD configuration

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: regex-file-example
spec:
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automati
cally creates a Logstore.
  logstore: k8s-file
  logtailConfig:
    # The type of the data source. If you want to collect text logs, you must set the value to file
.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that is
specified in metadata.name.
    configName: regex-file-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in full regex mode.
      logType: common_reg_log
      # The log file path.
      logPath: /app/logs
      # The log file name. You can use wildcard characters such as asterisks (*) and question marks
(?) when you specify the log file name. Example: log_*.log.
      filePattern: error.LOG
      # The regular expression that is used to match the start part in the first line of the log.
      logBeginRegex: '\[\d+-\d+-\w+:\d+:\d+,\d+]\s\[\w+]\s.*'
      # The regular expression that is used to extract log content.
      regex: '\[([^]]+)]\s\[(\w+)]\s\[(\w+)]\s\[([^:]+):(\d+)]\s(.*)'
      # The keys that you want to extract from logs.
      key : ["time", "level", "method", "file", "line", "message"]
      # The format of the time values that are extracted from logs. By default, time values are ext
racted from the time field of logs that are collected in full regex mode. If you do not want to ext
ract time values, you can leave this parameter empty. If you configure the timeFormat parameter, yo
u must also configure the adjustTimezone and logTimezone parameters.
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # By default, Logtail uses UTC. You must configure the following parameter before you can for
cefully change the time zone:
      adjustTimezone: true
      # The time zone offset. The time zone of logs is UTC+8. You can change the value of this para
meter to change the time zone.
      logTimezone: "GMT+08:00"
      # The settings that allow Logtail to upload raw logs if the logs fail to be parsed.
      discardUnmatch: false
      # If you want to collect container text logs, you must set the dockerFile parameter to true.
      dockerFile: true
      # The environment variable allowlist. In this example, text logs are collected only from cont
ainers whose environment variable configurations include a key of ALIYUN_LOGTAIL_USER_DEFINED_ID.
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

- Collected log

Example 3: Collect container text logs in delimiter mode

If the container text logs that you want to collect contain delimiters, you can collect the container text logs in delimiter mode. Logs that are in the delimiter-separated values (DSV) format use line feeds as boundaries. Each log is placed in a separate line. Each log is parsed into multiple fields by using delimiters.

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: delimiter-file-example
spec:
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatical
ly creates a Logstore.
  logstore: k8s-file
  logtailConfig:
    # The type of the data source. If you want to collect text logs, you must set the value to file.
    inputType: file
    configName: delimiter-file-example
    # The name of the Logtail configuration. The name must be the same as the resource name that is sp
ecified in metadata.name.
    inputDetail:
      # The settings that allow Logtail to collect text logs in delimiter mode.
      logType: delimiter_log
      # The log file path.
      logPath: /usr/local/ilogtail
      # The log file name. You can use wildcard characters such as asterisks (*) and question marks (?
) when you specify the log file name. Example: log_*.log.
      filePattern: delimiter_log.LOG
      # The delimiter.
      separator: '|&|'
      # The keys that you want to extract from logs.
      key : ['time', 'level', 'method', 'file', 'line', 'message']
      # The name of the field from which time values are extracted.
      timeKey: 'time'
      # The format of the time values that are extracted from logs. By default, time values are extrac
ted from the time field of logs that are collected in delimiter mode. If you do not want to extract ti
me values, you can leave this parameter empty. If you configure the timeFormat parameter, you must als
o configure the adjustTimezone and logTimezone parameters.
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # By default, Logtail uses UTC. You must configure the following parameter before you can forcef
ully change the time zone:
      adjustTimezone: true
      # The time zone offset. The time zone of logs is UTC+8. You can change the value of this paramet
er to change the time zone.
      logTimezone: "GMT+08:00"
      # The settings that allow Logtail to upload raw logs if the logs fail to be parsed.
      discardUnmatch: false
      # If you want to collect container text logs, you must set the dockerFile parameter to true.
      dockerFile: true
      # The environment variable allowlist. In this example, text logs are collected only from contain
ers whose environment variable configurations include a key of ALIYUN_LOGTAIL_USER_DEFINED_ID.
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ''
```

Example 4: Collect container text logs in JSON mode

If the container text logs that you want to collect are JSON logs of the object type, you can collect the container text logs in JSON mode.

- Raw log

```
{"url": "POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=U0Ujpek********&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=pD12XYLmGxKQ%2Bmkd6x7hAgQ7b1c%3D HTTP/1.1", "ip": "10.200.98.220", "user-agent": "aliyun-sdk-java", "request": {"status": "200", "latency": "18204"}, "time": "05/Jan/2020:13:30:28"}
```

- CRD configuration

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: json-file-example
spec:
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatically creates a Logstore.
  logstore: k8s-file
  logtailConfig:
    # The type of the data source. If you want to collect text logs, you must set the value to file.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that is specified in metadata.name.
    configName: json-file-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in JSON mode.
      logType: json_log
      # The log file path.
      logPath: /usr/local/ilogtail
      # The log file name. You can use wildcard characters such as asterisks (*) and question marks (?) when you specify the log file name. Example: log_*.log.
      filePattern: json_log.LOG
      # The name of the field from which time values are extracted. If no requirements are specified, set the value to timeFormat: ''.
      timeKey: 'time'
      # The format of the time values that are extracted from logs. If no requirements are specified, set the value to timeFormat: ''.
      timeFormat: '%Y-%m-%dT%H:%M:%S'
      # If you want to collect container text logs, you must set the dockerFile parameter to true.
      dockerFile: true
      # The environment variable allowlist. In this example, text logs are collected only from containers whose environment variable configurations include a key of ALIYUN_LOGTAIL_USER_DEFINED_ID.
      dockerIncludeEnv:
        ALIYUN_LOGTAIL_USER_DEFINED_ID: ""
```

3.1.5.6. Use CRDs to collect container text logs in Sidecar mode

This topic describes how to install Sidecar. This topic also describes how to use a custom resource definition (CRD) to create a Logtail configuration that is used to collect container text logs in Sidecar mode.

Prerequisites

The alibaba-log-controller component is installed. For more information, see Collect Kubernetes logs.

Context

In Sidecar mode, the Logtail container shares a log directory with an application container. The application container writes logs to the shared directory. Logtail monitors changes to the log files in the shared directory and collects logs. For more information, see Sidecar container with a logging agent and How Pods manage multiple containers.

Step 1: Install Sidecar

1. Log on to your Kubernetes cluster.
2. Create a YAML file.

   In this command, the file name is *sidecar.yaml*. Replace the file name with an actual file name.

   ```
   vim sidecar.yaml
   ```

3. Enter the following script in the YAML file and configure the parameters based on your business scenario.

   Notice Make sure that the time zone you specify for the *TZ* field in the *env* parameter is valid. If the time zones in raw logs and processed logs in a Log Service project are inconsistent, the time that is recorded for the collected logs may be a point in time in the past or in the future. For example, if the Log Service project resides in greater China, you can set the time zone to Asia/Shanghai.

   ```
   apiVersion: batch/v1
   kind: Job
   metadata:
     name: nginx-log-sidecar-demo
     namespace: default
   spec:
     template:
       metadata:
         name: nginx-log-sidecar-demo
       spec:
         restartPolicy: Never
         containers:
         - name: nginx-log-demo
           image: registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test:latest
           command: ["/bin/mock_log"]
           args: ["--log-type=nginx", "--stdout=false", "--stderr=true", "--path=/var/log/nginx/acces
   s.log", "--total-count=1000000000", "--logs-per-sec=100"]
           volumeMounts:
           - name: nginx-log
             mountPath: /var/log/nginx
         ##### logtail sidecar container
         - name: logtail
           # more info: https://cr.console.aliyun.com/repository/cn-hangzhou/log-service/logtail/deta
   il
           # this images is released for every region
           image: registry.cn-hangzhou.aliyuncs.com/log-service/logtail:latest
           # when recevie sigterm, logtail will delay 10 seconds and then stop
           command:
           - sh
           - -c
           - /usr/local/ilogtail/run_logtail.sh 10
           livenessProbe:
             exec:
               command:
               - /etc/init.d/ilogtaild
               - status
             initialDelaySeconds: 30
             periodSeconds: 30
           resources:
   ```

```
    resources:
      limits:
        memory: 512Mi
      requests:
        cpu: 10m
        memory: 30Mi
    env:
      ##### base config
      # user id
      - name: "ALIYUN_LOGTAIL_USER_ID"
        value: "${your_aliyun_user_id}"
      # user defined id
      - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
        value: "${your_machine_group_user_defined_id}"
      # config file path in logtail's container
      - name: "ALIYUN_LOGTAIL_CONFIG"
        value: "/etc/ilogtail/conf/${your_region_config}/ilogtail_config.json"
      ##### env tags config
      - name: "ALIYUN_LOG_ENV_TAGS"
        value: "_pod_name_|_pod_ip_|_namespace_|_node_name_|_node_ip_"
      - name: "_pod_name_"
        valueFrom:
          fieldRef:
            fieldPath: metadata.name
      - name: "_pod_ip_"
        valueFrom:
          fieldRef:
            fieldPath: status.podIP
      - name: "_namespace_"
        valueFrom:
          fieldRef:
            fieldPath: metadata.namespace
      - name: "_node_name_"
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      - name: "_node_ip_"
        valueFrom:
          fieldRef:
            fieldPath: status.hostIP
    volumeMounts:
    - name: nginx-log
      mountPath: /var/log/nginx
  ##### share this volume
  volumes:
  - name: nginx-log
    emptyDir: {}
```

i. Configure the basic variables in the configuration script. The following table describes the variables.

```
##### base config
          # user id
          - name: "ALIYUN_LOGTAIL_USER_ID"
            value: "${your_aliyun_user_id}"
          # user defined id
          - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
            value: "${your_machine_group_user_defined_id}"
          # config file path in logtail's container
          - name: "ALIYUN_LOGTAIL_CONFIG"
            value: "/etc/ilogtail/conf/${your_region_config}/ilogtail_config.json"
```

Variable	Description
${your_aliyun_user_id}	The ID of your Apsara Stack tenant account. For more information, see Configure a user identifier.
${your_machine_group_user_defined_id}	The custom identifier of your machine group. The identifier must be unique in the region where your project resides. Example: nginx-log-sidecar. For more information, see Create a custom ID-based machine group.
${your_region_config}	The ID of the region where your project resides and the type of the network that your project uses. For more information about regions, see Manage a Logtail configuration.

ii. Specify the mount path in the configuration script.

> **Note** We recommend that you mount containers on a volume of the emptyDir type.

```
        volumeMounts:
        - name: nginx-log
          mountPath: /var/log/nginx
      ##### share this volume
      volumes:
      - name: nginx-log
        emptyDir: {}
```

Parameter	Description
name	The name of the volume. You can specify a name based on your business requirements. **Notice** The value of the name parameter in the volumeMounts node and the value of the name parameter in the volumes node must be the same. This ensures that the Logtail container and the application container are mounted on the same volume.
mountPath	The mount path. You can enter the path of files in which container text logs are recorded.

iii. Specify a waiting period for the Logtail container in the configuration script.

In most cases, the waiting period is 10 seconds. This value specifies that the Logtail container exits 10 seconds after the container receives a stop command. This setting helps prevent incomplete data collection.

```
command:
- sh
- -c
- /usr/local/ilogtail/run_logtail.sh 10
```

4. Run the following command to apply the configurations in the *sidecar.yaml* file.

In this command, the file name is *sidecar.yaml*. Replace the file name with an actual file name.

```
kubectl apply -f sidecar.yaml
```

Step 2: Create a Logtail configuration

To create a Logtail configuration, you only need to create an AliyunLogConfig CRD. After you create a Logtail configuration, the system automatically applies the Logtail configuration. If you want to delete the Logtail configuration, you only need to delete the CRD.

1. Log on to your Kubernetes cluster.
2. Run the following command to create a YAML file.

In this command, the file name is *cube.yaml*. Replace the file name with an actual file name.

```
vim cube.yaml
```

3. Enter the following script in the YAML file and configure the parameters based on your business scenario.

> **Notice**
>
> - The value of the configName parameter must be unique in the Log Service project that you use.
> - If multiple CRDs are associated with the same Logtail configuration, the Logtail configuration is affected when you delete or modify one of the CRDs. After the deletion or modification, the status of the other associated CRDs becomes inconsistent with the status of the Logtail configuration in Log Service.
> - In Sidecar mode, only text logs can be collected. You must set the dockerFile parameter to false.

```
apiVersion: log.alibabacloud.com/v1alpha1      # The default value is used. You do not need to mod
ify this parameter.
kind: AliyunLogConfig                          # The default value is used. You do not need to mod
ify this parameter.
metadata:
  name: simple-stdout-example                  # The name of the resource. The name must be unique
in the current Kubernetes cluster.
spec:
  project: k8s-my-project                      # Optional. The name of the project. The default va
lue is the name of the project that you use to install the Logtail component.
  logstore: k8s-stdout                         # The name of the Logstore. If the Logstore that yo
u specify does not exist, Log Service automatically creates a Logstore.
  machineGroups:                               # The name of the machine group. The name must be t
he same as the value of the ${your_machine_group_user_defined_id} parameter that you configured wh
en you installed Sidecar. This machine group is used to associate Sidecar with the CRD.
  - nginx-log-sidecar
  shardCount: 2                                # Optional. The number of shards. Valid values: 1 t
o 10. Default value: 2.
  lifeCycle: 90                                # Optional. The data retention period of the Logsto
re. Valid values: 1 to 3650. Unit: days. Default value: 90. The value 3650 specifies that log data
is permanently stored in the Logstore.
  logtailConfig:                               # The Logtail configuration.
    inputType: file                            # The type of the data source. In Sidecar mode, you
can use CRDs to collect only text logs. Therefore, you must set the value to file.
    configName: simple-stdout-example          # The name of the Logtail configuration. The name m
ust be the same as the resource name that is specified in metadata.name.
    inputDetail:                               # The detailed settings of the Logtail configuratio
n. For more information, see the following configuration examples.
    ...
```

Parameter	Type	Required	Description
project	string	No	The name of the project. The default value is the name of the project that you use to install the Logtail component.
logstore	string	Yes	The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatically creates a Logstore.
shardCount	int	No	The number of shards. Valid values: 1 to 10. Default value: 2.
lifeCycle	int	No	The data retention period of the Logstore. Valid values: 1 to 3650. Unit: days. Default value: 90. The value 3650 specifies that log data is permanently stored in the Logstore.

Parameter	Type	Required	Description

machineGroups	array	Yes	The name of the machine group. The name must be the same as the value of the *${your_machine_group_user_defined_id}* parameter that you configured when you installed Sidecar. Example: nginx-log-sidecar.
			Log Service creates a machine group to associate Sidecar with the CRD based on the name that you specify.
			Notice
			You must specify a custom identifier for the machine group in the following format:
			`machineGroups:`
			`- nginx-log-sidecar`

logtailConfig	object	Yes	The detailed settings of the Logtail configuration. In most cases, you need to configure only the inputType, configName, and inputDetail parameters.

4. Run the following command to apply the Logtail configuration.

   In this command, the file name is *cube.yaml*. Replace the file name with an actual file name.

   ```
   kubectl apply -f cube.yaml
   ```

   After you create the Logtail configuration, you can view the Logtail configuration in the Log Service console or by using a CRD. For more information, see Manage a Logtail configuration.

Configuration example for a single directory

This section provides an example on how to use a CRD to collect text logs from the nginx-log-demo container in Sidecar mode. The container belongs to a self-managed Kubernetes cluster in a data center. The text logs include NGINX access logs and NGINX error logs and are stored in a single directory. The following list describes the basic information:

- The Log Service project for log collection resides in the China (Hangzhou) region. Logs are collected over the Internet.
- The name of the volume to be mounted is nginx-log and the volume is of the emptyDir type. The nginx-log volume is mounted on the */var/log/nginx* directory of the nginx-log-demo and Logtail containers.
- The path to NGINX access logs is */var/log/nginx/access.log*. The name of the Logstore that is used to store the NGINX access logs is nginx-access.
- The path to NGINX error logs is */var/log/nginx/error.log*. The name of the Logstore that is used to store the NGINX error logs is nginx-error.
- Sidecar configuration example

```
apiVersion: batch/v1
kind: Job
metadata:
  name: nginx-log-sidecar-demo
  namespace: default
spec:
  template:
```

```
    metadata:
      name: nginx-log-sidecar-demo
    spec:
      restartPolicy: Never
      containers:
      - name: nginx-log-demo
        image: registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test:latest
        command: ["/bin/mock_log"]
        args: ["--log-type=nginx", "--stdout=false", "--stderr=true", "--path=/var/log/nginx/access
.log", "--total-count=1000000000", "--logs-per-sec=100"]
        volumeMounts:
        - name: nginx-log
          mountPath: /var/log/nginx
      ##### logtail sidecar container
      - name: logtail
        # more info: https://cr.console.aliyun.com/repository/cn-hangzhou/log-service/logtail/detai
l
        # this images is released for every region
        image: registry.cn-hangzhou.aliyuncs.com/log-service/logtail:latest
        # when recevie sigterm, logtail will delay 10 seconds and then stop
        command:
        - sh
        - -c
        - /usr/local/ilogtail/run_logtail.sh 10
        livenessProbe:
          exec:
            command:
            - /etc/init.d/ilogtaild
            - status
          initialDelaySeconds: 30
          periodSeconds: 30
        env:
          ##### base config
          # user id
          - name: "ALIYUN_LOGTAIL_USER_ID"
            value: "1023****3423"
          # user defined id
          - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
            value: "nginx-log-sidecar"
          # config file path in logtail's container
          - name: "ALIYUN_LOGTAIL_CONFIG"
            value: "/etc/ilogtail/conf/cn-hangzhou-internet/ilogtail_config.json"
          ##### env tags config
          - name: "ALIYUN_LOG_ENV_TAGS"
            value: "_pod_name_|_pod_ip_|_namespace_|_node_name_|_node_ip_"
          - name: "_pod_name_"
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: "_pod_ip_"
            valueFrom:
              fieldRef:
                fieldPath: status.podIP
          - name: "_namespace_"
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
          - name: "_node_name_"
            valueFrom:
              fieldRef:
```

```
            fieldPath: spec.nodeName
        - name: "_node_ip_"
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
      volumeMounts:
      - name: nginx-log
        mountPath: /var/log/nginx
    ##### share this volume
    volumes:
    - name: nginx-log
      emptyDir: {}
```

- CRD configuration example

  Create a Logtail configuration to collect NGINX access logs and another Logtail configuration to collect NGINX error logs.

- Collect NGINX access logs

> **Notice** In Sidecar mode, you must set the dockerFile parameter to false.

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in your Kubernetes cluster.
  name: nginx-log-access-example
spec:
  # The name of the project. The default value is the name of the project that you use to install
Logtail.
  project: k8s-nginx-sidecar-demo
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automa
tically creates a Logstore.
  logstore: nginx-access
  # The name of the machine group. The name must be the same as the value of the ${your_machine_g
roup_user_defined_id} parameter that you configured when you installed Sidecar.
  machineGroups:
  - nginx-log-sidecar
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. In Sidecar mode, you can use CRDs to collect only text logs. T
herefore, you must set the value to file.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that
is specified in metadata.name.
    configName: nginx-log-access-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in full regex mode.
      logType: common_reg_log
      # The log file path.
      logPath: /var/log/nginx
      # The log file name. You can use wildcard characters such as asterisks (*) and question mar
ks (?) when you specify the log file name. Example: log_*.log.
      filePattern: access.log
      # Set the dockerFile parameter to false. This setting is required in Sidecar mode.
      dockerFile: false
      # The regular expression that is used to match the start part in the first line of the log.
If you want to collect single-line logs, set the value to '.*'.
      logBeginRegex: '.*'
      # The regular expression that is used to extract log content. Configure this parameter base
d on your business scenario.
      regex: '(\S+)\s(\S+)\s\S+\s\S+\s"(\S+)\s(\S+)\s+([^"]+)"\s+(\S+)\s(\S+)\s(\d+)\s(\d+)\s(\S+
)\s"([^"]+)"\s.*'
      # The keys that you want to extract from logs.
      key : ["time", "ip", "method", "url", "protocol", "latency", "payload", "status", "response
-size",user-agent"]
```

- Collect NGINX error logs

  > **Notice** In Sidecar mode, you must set the dockerFile parameter to false.

```
# config for error log
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: nginx-log-error-example
spec:
  # The name of the project. The default value is the name of the project that you use to install
Logtail.
  project: k8s-nginx-sidecar-demo
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automa
tically creates a Logstore.
  logstore: nginx-error
  # The name of the machine group. The name must be the same as the value of the ${your_machine_g
roup_user_defined_id} parameter that you configured when you installed Sidecar.
  machineGroups:
  - nginx-log-sidecar
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. In Sidecar mode, you can use CRDs to collect only text logs. T
herefore, you must set the value to file.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that
is specified in metadata.name.
    configName: nginx-log-error-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in full regex mode.
      logType: common_reg_log
      # The log file path.
      logPath: /var/log/nginx
      # The log file name. You can use wildcard characters such as asterisks (*) and question mar
ks (?) when you specify the log file name. Example: log_*.log.
      filePattern: error.log
      # Set the dockerFile parameter to false. This setting is required in Sidecar mode.
      dockerFile: false
```

Configuration example for different directories

This section provides an example on how to use a CRD to collect text logs from the nginx-log-demo container in Sidecar mode. The container belongs to a self-managed Kubernetes cluster in a data center. The text logs include NGINX access logs and are stored in different directories. The following list describes the basic information:

- The Log Service project for log collection resides in the China (Hangzhou) region. Logs are collected over the Internet.
- The names of the volumes to be mounted are nginx-log and nginx-logs and the volumes are of the emptyDir type. The nginx-log volume is mounted on the */var/log/nginx* directory of the nginx-log-demo and Logtail containers. The nginx-logs volume is mounted on the */var/log/nginxs* directory of the nginx-log-demo and Logtail containers.
- One log file path is */var/log/nginx/access.log* and the other log file path is */var/log/nginxs/access.log*.
- The name of the Logstore that is used to store NGINX access logs is nginx-access.
- Sidecar configuration example

```
apiVersion: batch/v1
kind: Job
```

```
metadata:
  name: nginx-log-sidecar-demo
  namespace: default
spec:
  template:
    metadata:
      name: nginx-log-sidecar-demo
    spec:
      restartPolicy: Never
      containers:
      - name: nginx-log-demo
        image: registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test:latest
        command: ["/bin/mock_log"]
        args: ["--log-type=nginx", "--stdout=false", "--stderr=true", "--path=/var/log/nginx/access
.log", "--total-count=1000000000", "--logs-per-sec=100"]
        lifecycle:
        volumeMounts:
        - name: nginx-log
          mountPath: /var/log/nginx
        - name: nginx-logs
          mountPath: /var/log/nginxs
      ##### logtail sidecar container
      - name: logtail
        # more info: https://cr.console.aliyun.com/repository/cn-hangzhou/log-service/logtail/detai
l
        # this images is released for every region
        image: registry.cn-hangzhou.aliyuncs.com/log-service/logtail:latest
        # when recevie sigterm, logtail will delay 10 seconds and then stop
        lifecycle:
        command:
        - sh
        - -c
        - /usr/local/ilogtail/run_logtail.sh 10
        livenessProbe:
          exec:
            command:
            - /etc/init.d/ilogtaild
            - status
          initialDelaySeconds: 30
          periodSeconds: 30
        resources:
          limits:
            memory: 512Mi
          requests:
            cpu: 10m
            memory: 30Mi
        env:
          ##### base config
          # user id
          - name: "ALIYUN_LOGTAIL_USER_ID"
            value: "1023****3423"
          # user defined id
          - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
            value: "nginx-log-sidecar"
          # config file path in logtail's container
          - name: "ALIYUN_LOGTAIL_CONFIG"
            value: "/etc/ilogtail/conf/cn-hangzhou-internet/ilogtail_config.json"
          ##### env tags config
          - name: "ALIYUN_LOG_ENV_TAGS"
            value: "_pod_name_|_pod_ip_|_namespace_|_node_name_|_node_ip_"
```

```
        - name: "_pod_name_"
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: "_pod_ip_"
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: "_namespace_"
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: "_node_name_"
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: "_node_ip_"
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
      volumeMounts:
      - name: nginx-log
        mountPath: /var/log/nginx
      - name: nginx-logs
        mountPath: /var/log/nginxs
    ##### share this volume
    volumes:
    - name: nginx-log
      emptyDir: {}
    - name: nginx-logs
      emptyDir: {}
```

- CRD configuration example

  Create two Logtail configurations to collect NGINX access logs from different directories.

- Collect NGINX access logs from the */var/log/nginx/access.log* directory

  > **Notice** In Sidecar mode, you must set the dockerFile parameter to false.

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: nginx-log-access-example
spec:
  # The name of the project. The default value is the name of the project that you use to install Logtail.
  project: k8s-nginx-sidecar-demo
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automatically creates a Logstore.
  logstore: nginx-access
  # The name of the machine group. The name must be the same as the value of the ${your_machine_group_user_defined_id} parameter that you configured when you installed Sidecar.
  machineGroups:
  - nginx-log-sidecar
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. In Sidecar mode, you can use CRDs to collect only text logs. Therefore, you must set the value to file.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that is specified in metadata.name.
    configName: nginx-log-access-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in full regex mode.
      logType: common_reg_log
      # The log file path.
      logPath: /var/log/nginx
      # The log file name. You can use wildcard characters such as asterisks (*) and question marks (?) when you specify the log file name. Example: log_*.log.
      filePattern: access.log
      # Set the dockerFile parameter to false. This setting is required in Sidecar mode.
      dockerFile: false
      # The regular expression that is used to match the start part in the first line of the log. If you want to collect single-line logs, set the value to '.*'.
      logBeginRegex: '.*'
      # The regular expression that is used to extract log content.
      regex: '(\S+)\s(\S+)\s\S+\s\S+\s"(\S+)\s(\S+)\s+([^"]+)"\s+(\S+)\s(\S+)\s(\d+)\s(\d+)\s(\S+)\s"([^"]+)"\s.*'
      # The keys that you want to extract from logs.
      key : ["time", "ip", "method", "url", "protocol", "latency", "payload", "status", "response-size",user-agent"]
```

- Collect NGINX access logs from the */var/log/nginxs/access.log* directory

> **Notice** In Sidecar mode, you must set the dockerFile parameter to false.

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # The name of the resource. The name must be unique in the current Kubernetes cluster.
  name: nginxs-log-access-example
spec:
  # The name of the project. The default value is the name of the project that you use to install
Logtail.
  project: k8s-nginx-sidecar-demo
  # The name of the Logstore. If the Logstore that you specify does not exist, Log Service automa
tically creates a Logstore.
  logstore: nginxs-access
  # The name of the machine group. The name must be the same as the value of the ${your_machine_g
roup_user_defined_id} parameter that you configured when you installed Sidecar.
  machineGroups:
  - nginx-log-sidecar
  # The Logtail configuration.
  logtailConfig:
    # The type of the data source. In Sidecar mode, you can use CRDs to collect only text logs. T
herefore, you must set the value to file.
    inputType: file
    # The name of the Logtail configuration. The name must be the same as the resource name that
is specified in metadata.name.
    configName: nginxs-log-access-example
    inputDetail:
      # The settings that allow Logtail to collect text logs in full regex mode.
      logType: common_reg_log
      # The log file path.
      logPath: /var/log/nginxs
      # The log file name. You can use wildcard characters such as asterisks (*) and question mar
ks (?) when you specify the log file name. Example: log_*.log.
      filePattern: access.log
      # Set the dockerFile parameter to false. This setting is required in Sidecar mode.
      dockerFile: false
      # The regular expression that is used to match the start part in the first line of the log.
If you want to collect single-line logs, set the value to '.*'.
      logBeginRegex: '.*'
      # The regular expression that is used to extract log content.
      regex: '(\S+)\s(\S+)\s\S+\s\S+\s"(\S+)\s(\S+)\s+([^"]+)"\s+(\S+)\s(\S+)\s(\d+)\s(\d+)\s(\S+
)\s"([^"]+)"\s.*'
      # The keys that you want to extract from logs.
      key : ["time", "ip", "method", "url", "protocol", "latency", "payload", "status", "response
-size",user-agent"]
# config for error log
```

3.1.5.7. Use the Log Service console to collect container text logs in Sidecar mode

This topic describes how to install Sidecar. This topic also describes how to use the Log Service console to create a Logtail configuration that is used to collect container text logs in Sidecar mode.

Prerequisites

The alibaba-log-controller component is installed. For more information, see Collect Kubernetes logs.

Context

In Sidecar mode, the Logtail container shares a log directory with an application container. The application container writes logs to the shared directory. Logtail monitors changes to log files in the shared directory and collects logs. For more information, see Sidecar container with a logging agent and How Pods manage multiple containers.

Step 1: Install Sidecar

1. Log on to your Kubernetes cluster.
2. Create a YAML file.

   In this command, the file name is *sidecar.yaml*. Replace the file name with an actual file name.

   ```
   vim sidecar.yaml
   ```

3. Enter the following script in the YAML file and configure the parameters based on your business scenario.

   Notice Make sure that the time zone you specify for the *TZ* field in the *env* parameter is valid. If the time zones in raw logs and processed logs in a Log Service project are inconsistent, the time that is recorded for the collected logs may be a point in time in the past or in the future. For example, if the Log Service project resides in greater China, you can set the time zone to Asia/Shanghai.

   ```
   apiVersion: batch/v1
   kind: Job
   metadata:
     name: nginx-log-sidecar-demo
     namespace: default
   spec:
     template:
       metadata:
         name: nginx-log-sidecar-demo
       spec:
         restartPolicy: Never
         containers:
         - name: nginx-log-demo
           image: registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test:latest
           command: ["/bin/mock_log"]
           args: ["--log-type=nginx", "--stdout=false", "--stderr=true", "--path=/var/log/nginx/acces
   s.log", "--total-count=1000000000", "--logs-per-sec=100"]
           volumeMounts:
           - name: nginx-log
             mountPath: /var/log/nginx
         ##### logtail sidecar container
         - name: logtail
           # more info: https://cr.console.aliyun.com/repository/cn-hangzhou/log-service/logtail/deta
   il
           # this images is released for every region
           image: registry.cn-hangzhou.aliyuncs.com/log-service/logtail:latest
           # when recevie sigterm, logtail will delay 10 seconds and then stop
           command:
           - sh
           - -c
           - /usr/local/ilogtail/run_logtail.sh 10
           livenessProbe:
             exec:
               command:
               - /etc/init.d/ilogtaild
   ```

```
        - status
      initialDelaySeconds: 30
      periodSeconds: 30
    resources:
      limits:
        memory: 512Mi
      requests:
        cpu: 10m
        memory: 30Mi
    env:
      ##### base config
      # user id
      - name: "ALIYUN_LOGTAIL_USER_ID"
        value: "${your_aliyun_user_id}"
      # user defined id
      - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
        value: "${your_machine_group_user_defined_id}"
      # config file path in logtail's container
      - name: "ALIYUN_LOGTAIL_CONFIG"
        value: "/etc/ilogtail/conf/${your_region_config}/ilogtail_config.json"
      ##### env tags config
      - name: "ALIYUN_LOG_ENV_TAGS"
        value: "_pod_name_|_pod_ip_|_namespace_|_node_name_|_node_ip_"
      - name: "_pod_name_"
        valueFrom:
          fieldRef:
            fieldPath: metadata.name
      - name: "_pod_ip_"
        valueFrom:
          fieldRef:
            fieldPath: status.podIP
      - name: "_namespace_"
        valueFrom:
          fieldRef:
            fieldPath: metadata.namespace
      - name: "_node_name_"
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      - name: "_node_ip_"
        valueFrom:
          fieldRef:
            fieldPath: status.hostIP
    volumeMounts:
    - name: nginx-log
      mountPath: /var/log/nginx
  ##### share this volume
  volumes:
  - name: nginx-log
    emptyDir: {}
```

i. Configure the basic variables in the configuration script. The following table describes the variables.

```
##### base config
          # user id
          - name: "ALIYUN_LOGTAIL_USER_ID"
            value: "${your_aliyun_user_id}"
          # user defined id
          - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
            value: "${your_machine_group_user_defined_id}"
          # config file path in logtail's container
          - name: "ALIYUN_LOGTAIL_CONFIG"
            value: "/etc/ilogtail/conf/${your_region_config}/ilogtail_config.json"
```

Variable	Description
${your_aliyun_user_id}	The ID of your Apsara Stack tenant account. For more information, see Configure a user identifier.
${your_machine_group_user_defined_id}	The custom identifier of your machine group. The identifier must be unique in the region where your project resides. Example: nginx-log-sidecar. For more information, see Create a custom ID-based machine group.
${your_region_config}	The ID of the region where your project resides and the type of the network that your project uses. For more information about regions, see Manage a Logtail configuration.

ii. Specify the mount path in the configuration script.

> **Note** We recommend that you mount containers on a volume of the emptyDir type.

```
        volumeMounts:
        - name: nginx-log
          mountPath: /var/log/nginx
      ##### share this volume
      volumes:
      - name: nginx-log
        emptyDir: {}
```

Parameter	Description
name	The name of the volume. You can specify a name based on your business requirements. **Notice** The value of the name parameter in the volumeMounts node and the value of the name parameter in the volumes node must be the same. This ensures that the Logtail container and the application container are mounted on the same volume.
mountPath	The mount path. You can enter the path of files in which container text logs are recorded.

iii. Specify a waiting period for the Logtail container in the configuration script.

In most cases, the waiting period is 10 seconds. This value specifies that the Logtail container exits 10 seconds after the container receives a stop command. This setting helps prevent incomplete data collection.

```
command:
- sh
- -c
- /usr/local/ilogtail/run_logtail.sh 10
```

4. Run the following command to apply the configurations in the *sidecar.yaml* file.

In this command, the file name is *sidecar.yaml*. Replace the file name with an actual file name.

```
kubectl apply -f sidecar.yaml
```

Step 2: Create a machine group

1. Log on to the Log Service console
2. In the Projects section, click the project that you use to install Logtail components.
3. In the left-side navigation pane, choose **Resources > Machine Groups**.
4. In the **Machine Groups** list, choose > **Create Machine Group**.
5. In the **Create Machine Group** panel, configure the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Name	The name of the machine group. **Notice** After the machine group is created, you cannot change the name of the machine group. Proceed with caution.
Identifier	The identifier of the machine group. Select **Custom ID**.
Topic	The topic of the machine group. The topic is used to differentiate the logs that are generated by different servers.
Custom Identifier	The custom identifier of the machine group. The identifier must be the same as the value of the *${your_machine_group_user_defined_id}* parameter that you configured when you installed Sidecar. Example: nginx-log-sidecar.

Step 3: Create a Logtail configuration

1. Log on to the Log Service console
2. In the **Import Data** section, click **RegEx - Text Log**.

In this example, a Logtail configuration is created to collect text logs in full regex mode. For information about how to collect text logs in other modes, see Collect text logs.

3. Select a project and a Logstore. Then, click **Next**.

Select the project that you use to install Logtail components and the Logstore that you create.

4. Click **Use Existing Machine Groups**.
5. Select a machine group from the **Source Server Groups** section and move the machine group to the **Applied Server Groups** section. Then, click **Next**.

Select the machine group that you created in Step 2: Create a machine group.

> **Notice** If you enable a machine group immediately after you create the machine group, the heartbeat status of the machine group may be **FAIL**. This issue occurs because the machine group is not connected to Log Service. To resolve this issue, you can click **Automatic Retry**. If the issue persists, troubleshoot the issue. For more information, see What can I do if Log Service does not receive heartbeats from a Logtail client?.

6. Create a Logtail configuration and click **Next**.

   You can collect logs in simple mode, NGINX mode, delimiter mode, JSON mode, or full regex mode. For more information, see Collect text logs.

   > **Notice** In Sidecar mode, do not turn on **Docker File**.

7. Preview data, configure indexes, and then click **Next**.

   By default, Log Service enables full-text indexing. You can configure field indexes based on the logs that are collected in manual mode or automatic mode. For more information, see Configure indexes.

   > **Note** If you want to query and analyze logs, you must enable full-text indexing or field indexing. If you enable full-text indexing and field indexing, the system uses only field indexes.

3.1.5.8. Collect logs from standard Docker containers

This topic describes how to deploy a Logtail container and create a Logtail configuration to collect logs from standard Docker containers.

Step 1: Deploy a Logtail container

1. Run the following command to pull the Logtail image:

   ```
   docker pull registry.cn-hangzhou.aliyuncs.com/log-service/logtail
   ```

   Replace *registry.cn-hangzhou.aliyuncs.com* with the actual endpoint. For more information about the endpoints of regions, see the "View the endpoint of a project" section of the Manage a project topic. If your server resides in a virtual private cloud (VPC), you must replace registry with registry-vpc.

2. Start a Logtail container.

   > **Note** Before you configure the parameters, you must complete one of the following configurations. Otherwise, the `container text file busy` error may occur when you delete other containers.
   >
   > - For CentOS 7.4 and later, set fs.may_detach_mounts to 1. For more information, see Bug 1468249, Bug 1441737, and Issue 34538.
   > - Add `--privileged` to the startup parameters to grant Logtail the `privileged` permission. For more information, see Docker run reference.

   Replace the `${your_region_name}`, `${your_aliyun_user_id}`, and `${your_machine_group_user_defined_id}` parameters in the following command with the actual values:

   ```
   docker run -d -v /:/logtail_host:ro -v /var/run:/var/run --env ALIYUN_LOGTAIL_CONFIG=/etc/ilogtail
   /conf/${your_region_name}/ilogtail_config.json --env ALIYUN_LOGTAIL_USER_ID=${your_aliyun_user_id}
   --env ALIYUN_LOGTAIL_USER_DEFINED_ID=${your_machine_group_user_defined_id} registry.cn-hangzhou.al
   iyuncs.com/log-service/logtail
   ```

Parameter	Description
`${your_region_name}`	The ID of the region where your project resides and the type of the network that your project uses.
`${your_aliyun_user_id}`	The ID of your Apsara Stack tenant account. For more information, see Configure a user identifier.
`${your_machine_group_user_defined_id}`	The custom identifier of your machine group. The identifier must be unique in the region where your project resides. For more information, see Create a custom ID-based machine group.

Note

You can customize the startup parameters of the Logtail container only if the following conditions are met:

i. The following environment variables are configured: `ALIYUN_LOGTAIL_USER_DEFINED_ID`, `ALIYUN_LOGTAIL_USER_ID`, and `ALIYUN_LOGTAIL_CONFIG`.

ii. The */var/run* directory of the host is mounted on the */var/run* directory of the Logtail container.

iii. The root directory of the host is mounted on the `/logtail_host` directory of the Logtail container.

iv. If the `The parameter is invalid : uuid=none` error is returned in the */usr/local/ilogtail/ilogtail.LOG* log file, you must create a file named *product_uuid* on the host. Then, you must enter a valid universally unique identifier (UUID) in the file, for example, `169E98C9-ABC0-4A92-B1D2-AA6239C0D261`, and mount the file on the */sys/class/dmi/id/product_uuid* directory of the Logtail container.

Step 2: Create a Logtail configuration

Create a Logtail configuration in the console based on your business requirements.

- To collect Docker text logs, follow the steps that you perform to collect Kubernetes text logs. For more information, see Use the Log Service console to collect container text logs in DaemonSet mode.
- To collect Docker stdout and stderr logs, follow the steps that you perform to collect Kubernetes stdout and stderr. For more information, see Use the Log Service console to collect container stdout and stderr in DaemonSet mode.
- To collect host text logs, follow the steps provided in Collect text logs.

  By default, the root directory of the host is mounted on the */logtail_host* directory of the Logtail container. When you configure the directory to collect logs, you must add the container directory as the prefix to the log path. For example, to collect data from the */home/logs/app_log/* directory of the host, you must set the log path to */logtail_host/home/logs/app_log/*.

When you create a machine group, enter the value of the `ALIYUN_LOGTAIL_USER_DEFINED_ID` parameter in the **Custom Identifier** field. This value is specified in Step 1: Deploy a Logtail container.

Default fields

- Docker stdout and stderr logs

  The following table describes the fields that are uploaded by default for each log.

Log field	Description
time	The point in time when data is uploaded. Example: `2018-02-02T02:18:41.979147844Z`.

Log field	Description
source	The type of the input source. Valid values: stdout and stderr.
_image_name_	The name of the image.
_container_name_	The name of the container.
_container_ip_	The IP address of the container.

- Docker file

  The following table describes the fields that are uploaded by default for each log.

Log field	Description
_image_name_	The name of the image.
_container_name_	The name of the container.
_container_ip_	The IP address of the container.

Other operations

- View the status of Logtail.

  You can run the `docker exec ${logtail_container_id} /etc/init.d/ilogtaild status` command to view the status of Logtail.

- View the version number, IP address, and startup time of Logtail.

  You can run the `docker exec ${logtail_container_id} cat /usr/local/ilogtail/app_info.json` command to view the information of Logtail.

- View the operational logs of Logtail.

  The operational logs of Logtail are stored in the *ilogtail.LOG* file in the */usr/local/ilogtail/* directory. If the log file is rotated, the generated files are compressed and stored as *ilogtail.LOG.x.gz*. Example:

```
docker exec a287de895e40 tail -n 5 /usr/local/ilogtail/ilogtail.LOG
[2018-02-06 08:13:35.721864]    [INFO]    [8]    [build/release64/sls/ilogtail/LogtailPlugin.cpp:10
4]    logtail plugin Resume:start
[2018-02-06 08:13:35.722135]    [INFO]    [8]    [build/release64/sls/ilogtail/LogtailPlugin.cpp:10
6]    logtail plugin Resume:success
[2018-02-06 08:13:35.722149]    [INFO]    [8]    [build/release64/sls/ilogtail/EventDispatcher.cpp:
369]    start add existed check point events, size:0
[2018-02-06 08:13:35.722155]    [INFO]    [8]    [build/release64/sls/ilogtail/EventDispatcher.cpp:
511]    add existed check point events, size:0    cache size:0    event size:0    success count:0
[2018-02-06 08:13:39.725417]    [INFO]    [8]    [build/release64/sls/ilogtail/ConfigManager.cpp:37
76]    check container path update flag:0    size:1
```

  The standard output of the container is irrelevant to this case. Ignore the following standard output:

```
start umount useless mount points, /shm$|/merged$|/mqueue$
umount: /logtail_host/var/lib/docker/overlay2/3fd0043af174cb0273c3c7869500fbe2bdb95d13b1e110172ef57
fe840c82155/merged: must be superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/d5b10aa19399992755de1f85d25009528daa749c1bf8c16edff44
beab6e69718/merged: must be superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/5c3125daddacedec29df72ad0c52fac800cd56c6e880dc4e8a640
b1e16c22dbe/merged: must be superuser to unmount
......
xargs: umount: exited with status 255; aborting
umount done
start logtail
ilogtail is running
logtail status:
ilogtail is running
```

- Restart Logtail.

  To restart Logtail, use the following sample code:

```
docker exec a287de895e40 /etc/init.d/ilogtaild stop
kill process Name: ilogtail pid: 7
kill process Name: ilogtail pid: 8
stop success
docker exec a287de895e40 /etc/init.d/ilogtaild start
ilogtail is running
```

3.1.5.9. Collect Kubernetes events

This topic describes how to use the eventer component to collect events from Kubernetes and send the events to Log Service.

Log Service allows you to collect events from Kubernetes by using kube-eventer. For more information about the source code for Kubernetes event collection, visit GitHub.

Create a Logtail configuration

> **Note** If you use self-managed Kubernetes, you must configure the endpoint, project, logStore, regionId, internal, accessKeyId, and accessKeySecret parameters.

The following example shows the event collection configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: kube-eventer
  name: kube-eventer
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kube-eventer
  template:
    metadata:
      labels:
        app: kube-eventer
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
```

```
    spec:
      dnsPolicy: ClusterFirstWithHostNet
      serviceAccount: kube-eventer
      containers:
        - image: registry.aliyuncs.com/acs/kube-eventer-amd64:v1.1.0-c93a835-aliyun
          name: kube-eventer
          command:
            - "/kube-eventer"
            - "--source=kubernetes:https://kubernetes.default"
            ## .send to sls
            ## --sink=sls:https://{endpoint}?project={project}&logStore=k8s-event&regionId={region-id}
&internal=false&accessKeyId={accessKeyId}&accessKeySecret={accessKeySecret}
            - --sink=sls:https://cn-beijing.log.aliyuncs.com?project=k8s-xxxx&logStore=k8s-event&regio
nId=cn-beijing&internal=false&accessKeyId=xxx&accessKeySecret=xxx
          env:
          # If TZ is assigned, set the TZ value as the time zone
          - name: TZ
            value: "Asia/Shanghai"
          volumeMounts:
            - name: localtime
              mountPath: /etc/localtime
              readOnly: true
            - name: zoneinfo
              mountPath: /usr/share/zoneinfo
              readOnly: true
          resources:
            requests:
              cpu: 10m
              memory: 50Mi
            limits:
              cpu: 500m
              memory: 250Mi
      volumes:
        - name: localtime
          hostPath:
            path: /etc/localtime
        - name: zoneinfo
          hostPath:
            path: /usr/share/zoneinfo
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kube-eventer
rules:
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - get
      - list
      - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kube-eventer
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
  kind: ClusterRole
  name: kube-eventer
subjects:
  - kind: ServiceAccount
    name: kube-eventer
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kube-eventer
  namespace: kube-system
```

Parameter	Type	Required	Description
endpoint	string	Yes	The Log Service endpoint. For more information, see Manage a project.
project	string	Yes	The project in Log Service.
logStore	string	Yes	The Logstore in Log Service.
internal	string	Required for self-managed Kubernetes	If you use self-managed Kubernetes, set the value to false.
regionId	string	Required for self-managed Kubernetes	The ID of the region where the Log Service Logstore resides. For more information, see Manage a project.
accessKeyId	string	Required for self-managed Kubernetes	The AccessKey ID. We recommend that you use the AccessKey ID of a RAM user.
accessKeySecret	string	Required for self-managed Kubernetes	The AccessKey secret. We recommend that you use the AccessKey secret of a RAM user.

Sample log

The following example shows a collected sample log:

```
hostname:  cn-hangzhou.i-***********"
level:  Normal
pod_id:  2a360760-****
pod_name:  logtail-ds-blkkr
event_id:  {
   "metadata":{
      "name":"logtail-ds-blkkr.157b7cc90de7e192",
      "namespace":"kube-system",
      "selfLink":"/api/v1/namespaces/kube-system/events/logtail-ds-blkkr.157b7cc90de7e192",
      "uid":"2aaf75ab-****",
      "resourceVersion":"6129169",
      "creationTimestamp":"2019-01-20T07:08:19Z"
   },
   "involvedObject":{
      "kind":"Pod",
      "namespace":"kube-system",
      "name":"logtail-ds-blkkr",
      "uid":"2a360760-****",
      "apiVersion":"v1",
      "resourceVersion":"6129161",
      "fieldPath":"spec.containers{logtail}"
   },
   "reason":"Started",
   "message":"Started container",
   "source":{
      "component":"kubelet",
      "host":"cn-hangzhou.i-***********"
   },
   "firstTimestamp":"2019-01-20T07:08:19Z",
   "lastTimestamp":"2019-01-20T07:08:19Z",
   "count":1,
   "type":"Normal",
   "eventTime":null,
   "reportingComponent":"",
   "reportingInstance":""
}
```

Log field	Type	Description
hostname	string	The hostname of the server where an event occurs.
level	string	The level of a log. Valid values: Normal and Warning.
pod_id	string	The unique identifier of a pod. This field is available only if the event type is related to the pod.
pod_name	string	The name of a pod. This field is available only if the event type is related to the pod.
eventId	json	The details of an event. The value of this field is a JSON string.

3.1.5.10. Collect container text logs

Logtail can collect and upload container text logs together with container metadata to Log Service.

Features

Logtail can collect and upload container text logs together with container metadata to Log Service. Compared with basic log file collection, Docket file collection by using Logtail has the following features:

- Allows you to specify the log path of a container without the need to manually map the log path of the container to a path on the host.
- Uses labels to specify containers for log collection.
- Uses labels to exclude containers from log collection.
- Uses environment variables to specify containers for log collection.
- Uses environment variables to exclude containers from log collection.
- Supports multi-line logs such as Java Stack logs.
- Supports automatic labeling for Docker container logs.

> **Note**
>
> - The preceding labels are retrieved by running the docker inspect command. These labels are different from the labels that are specified in a Kubernetes cluster.
> - The preceding environment variables are the same as the environment variables that are specified to start containers.

Limits

- Stop policy: If Logtail detects the `die` event on a container that is stopped, Logtail stops collecting logs from the container. If collection latency occurs, some logs that are collected before the container is stopped may be lost.
- Docker storage driver: For Docker containers, only overlay and overlay2 storage drivers are supported. If other storage drivers are used, you must mount the log directory on the on-premises host.
- Logtail running mode: Logtail must run in a container and must be deployed based on Logtail deployment solutions.

Step 1: Deploy and configure Logtail

- Kubernetes

  For more information about how to collect Kubernetes logs, see Collect Kubernetes logs.

- Configure Logtail on other containers

  For more information about the methods used to manage other containers, such as Swarm and Mesos, see Collect standard Docker logs.

Step 2: Create a Logtail configuration for log collection

1. Log on to the Log Service console.
2. In the **Import Data** section, select **Docker File - Container**.
3. Select a destination project and Logstore, and then click **Next**.

   You can also click **Create Now** to create a project and a Logstore.

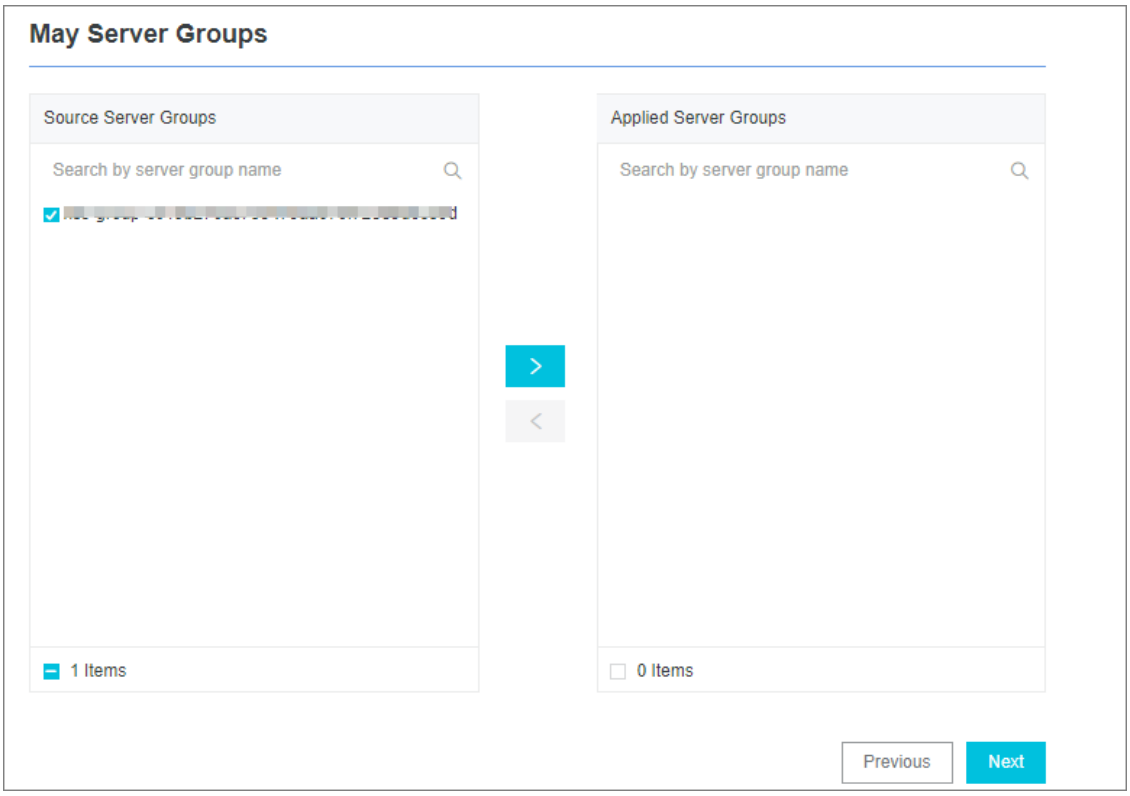
   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**,

and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Create a Logtail configuration.

   The following table describes the parameters of data sources. For information about common parameters, see Configure text log collection.

Parameter	Description
Docker File	Checks whether the file that is collected from the specified data source is a Docker file.

Parameter	Description
Label Whitelist	If you want to configure the label allowlist, you must specify the LabelKey parameter. If the value of the LabelValue parameter is not empty, logs are collected from the containers whose label key-value pairs match the specified key-value pairs. If the value of the LabelValue parameter is empty, logs are collected from the containers whose label keys match the specified keys. Note ◦ Key-value pairs are in the logical OR relation. If a label key-value pair of a container matches one of the specified key-value pairs, the logs of the container are collected. ◦ The labels that are described in this topic refer to the label information in docker inspect.
Label Blacklist	If you want to configure the label denylist, you must specify the LabelKey parameter. If the value of the LabelValue parameter is not empty, logs are not collected from the containers whose label key-value pairs match the specified key-value pairs. If the value of the LabelValue parameter is empty, logs are not collected from the containers whose label keys match the specified keys. Note ◦ Key-value pairs are in the logical OR relation. If a label key-value pair of a container matches one of the specified key-value pairs, the logs of the container are not collected. ◦ The labels that are described in this topic refer to the label information in docker inspect.
Environment Variable Whitelist	If you want to configure the environment variable allowlist, you must specify the EnvKey parameter. If the value of the EnvValue parameter is not empty, logs are collected from the containers whose environment variable key-value pairs match the specified key-value pairs. If the value of the EnvValue parameter is empty, logs are collected from the containers whose environment variable keys match the specified keys. Note ◦ Key-value pairs are in the logical OR relation. If an environment variable key-value pair of a container matches one of the specified key-value pairs, the logs of the container are collected. ◦ The environment variables that are described in this topic refer to the environment information configured in container startup.

Parameter	Description
Environment Variable Blacklist	If you want to configure the environment variable denylist, you must specify the EnvKey parameter. If the value of the EnvValue parameter is not empty, logs are not collected from the containers whose environment variable key-value pairs match the specified key-value pairs. If the value of the EnvValue parameter is empty, logs are not collected from the containers whose environment variable keys match the specified keys. **Note** ○ Key-value pairs are in the logical OR relation. If an environment variable key-value pair of a container matches one of the specified key-value pairs, the logs of the container are not collected. ○ The environment variables that are described in this topic refer to the environment information configured in container startup.

Note

- Labels in an allowlist and a denylist are different from the labels that are defined in Kubernetes. The labels that are described in this topic refer to the label information in docker inspect.
- A namespace and a container name of a Kubernetes cluster can be mapped to a Docker label. The value of the LabelKey parameter for a namespace is `io.kubernetes.pod.namespace`. The value of the LabelKey parameter for a container name is `io.kubernetes.container.name`. For example, the namespace of the pod that you created is backend-prod and the container name is worker-server. In this case, you can set the key-value pair of an allowlist label to `io.kubernetes.pod.namespace : backend-prod` or `io.kubernetes.container.name : worker-server`. Then, you can collect logs from only the worker-server container.
- In a Kubernetes cluster, we recommend that you specify only the `io.kubernetes.pod.namespace` and `io.kubernetes.container.name` labels. You can also specify the Environment Variable Whitelist parameter or the Environment Variable Blacklist parameter based on your business requirements.

7. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

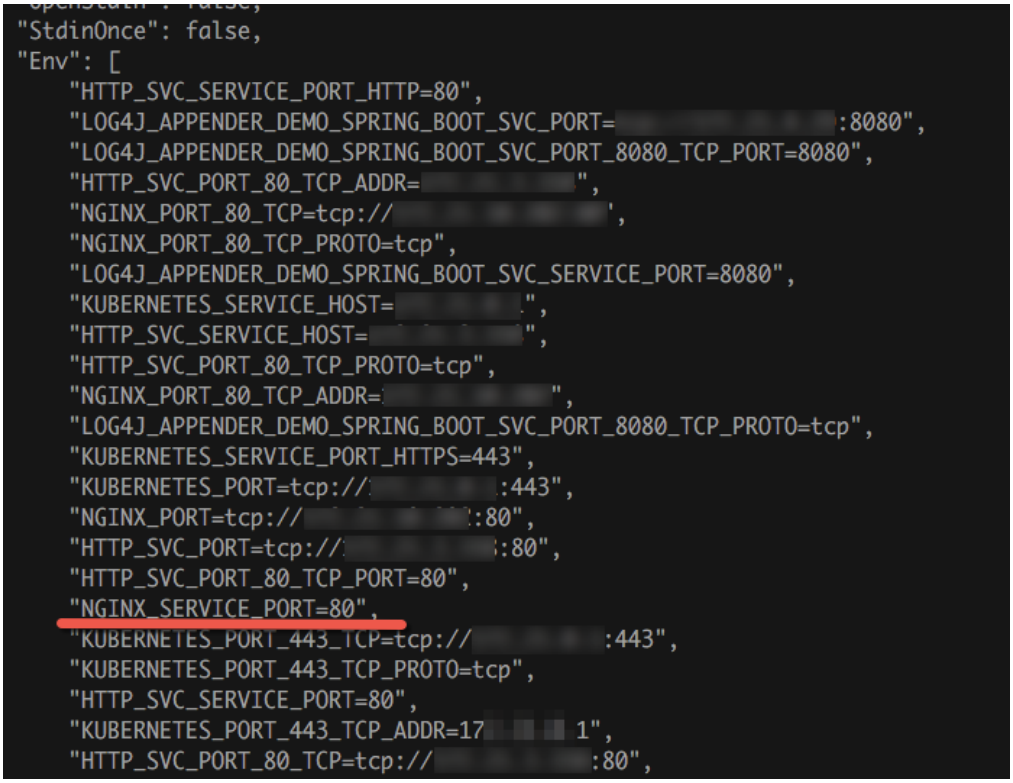
Note

- To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
- If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

Configuration examples

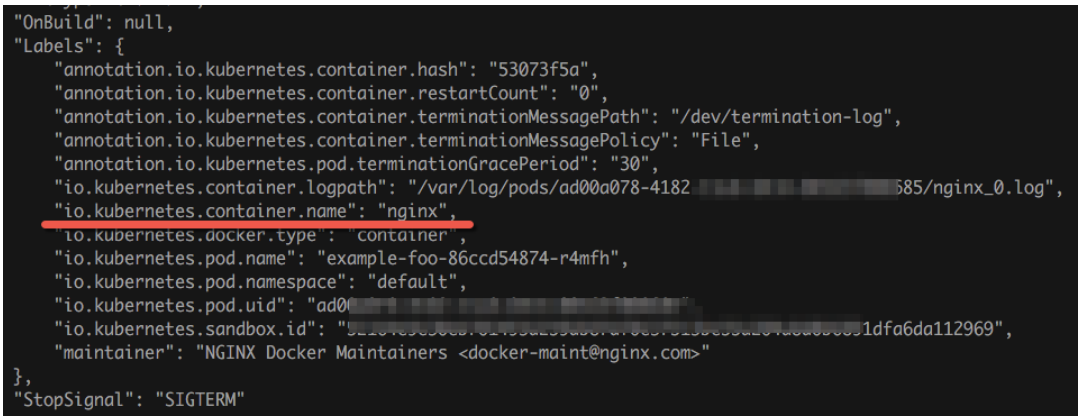
- Configure environment variables

  Collect the logs of the containers whose environment variables include `NGINX_PORT_80_TCP_PORT=80` and exclude `POD_NAMESPACE=kube-system`. The log file path is `/var/log/nginx/access.log` and logs are parsed in simple mode.



- Configure labels

  Collect the logs of the containers whose container labels include `io.kubernetes.container.name=nginx`. The log file path is `/var/log/nginx/access.log` and logs are parsed in simple mode.



Default fields

The following table describes the fields that are uploaded by default for each log of a common Docker container.

Log field	Description
`_image_name_`	The name of the image.
`_container_name_`	The name of the container.
`_container_ip_`	The IP address of the container.

The following table describes the fields that are uploaded by default for each log of a Kubernetes cluster.

Log field	Description
_image_name_	The name of the image.
_container_name_	The name of the container.
_pod_name_	The name of the pod.
namespace	The namespace where the pod resides.
_pod_uid_	The unique identifier of the pod.
_container_ip_	The IP address of the pod.

3.1.5.11. Collect container stdout and stderr logs

Logtail can collect and upload container standard output (stdout) and standard error (stderr) logs together with container metadata to Log Service. This topic describes how to create a Logtail configuration in the Log Service console to collect Kubernetes stdout and stderr logs.

Prerequisites

- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- The alibaba-log-controller Helm package is installed. For more information, see Install Logtail.

Features

Logtail can collect container stdout and stderr logs, and upload the stdout and stderr logs together with container metadata to Log Service. The following features are supported by Logtail to collect container stdout and stderr logs:

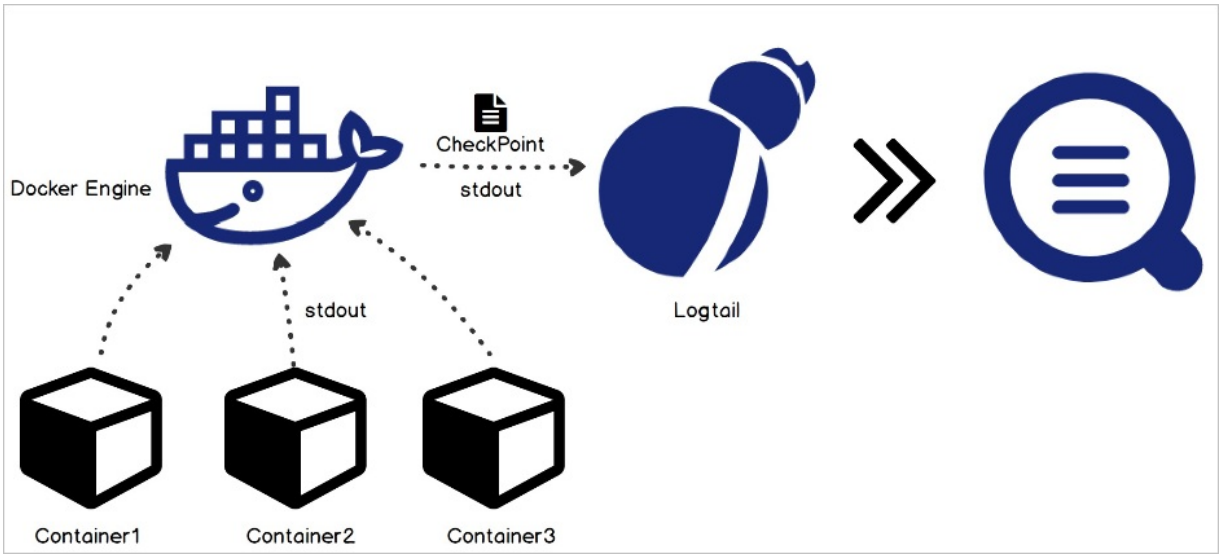
- Collects stdout and stderr logs in real time.
- Uses labels to specify containers for log collection.
- Uses labels to exclude containers from log collection.
- Uses environment variables to specify containers for log collection.
- Uses environment variables to exclude containers from log collection.
- Supports multi-line logs such as Java stack logs.
- Supports automatic labeling for Docker container logs.
- Supports automatic labeling for Kubernetes container logs.

> **Note**
>
> - The preceding labels are retrieved by running the docker inspect command. These labels are different from the labels that are specified in a Kubernetes cluster.
> - The preceding environment variables are the same as the environment variables that are specified to start containers.

Implementation

A Logtail container uses a UNIX domain socket to communicate with the Docker daemon. The Logtail container queries all Docker containers and finds the specified Docker containers based on the specified labels and environment variables. Logtail runs the `docker logs` command to collect the logs of the specified Docker containers.

When Logtail collects the stdout and stderr logs of a Docker container, Logtail periodically stores checkpoints to a checkpoint file. If Logtail is restarted, Logtail collects logs from the last checkpoint.



Limits

- Logtail version: Only Logtail V0.16.0 or later that runs on Linux can be used to collect stdout and stderr logs. For more information, see Install Logtail in Linux.
- Permissions: By default, Logtail uses the `/var/run/docker.sock` socket to access the Docker engine. You must make sure that a UNIX domain socket is available and the Logtail container has permissions to access the Docker engine.
- Multi-line logs: By default, the last multi-line log that is collected by Logtail is cached for 3 seconds. This prevents the multi-line log from being split into multiple logs due to output latency. You can set the cache time by specifying the `BeginLineTimeoutMs` parameter. The value of the `BeginLineTimeoutMs` parameter cannot be less than 1,000 ms. Otherwise, an error may occur.
- Stop policy: If Logtail detects the `die` event on a container that is stopped, Logtail stops collecting stdout and stderr logs from the container. If collection latency occurs, some stdout and stderr logs that are collected before the container is stopped may be lost.
- Docker logging driver: The logging driver collects stdout and stderr logs only in the JSON format from containers that use the Docker engine.
- Context: By default, logs that are collected from different containers by using a Logtail configuration are in the same context. If you want the logs of each container to be in different contexts, create a Logtail configuration for each container.
- Data processing: The collected data is contained in the `content` field. You can process the data by using a common processing method. For more information, see Configure data processing methods.

Create a Logtail configuration

1. Log on to the Log Service console.
2. In the **Import Data** section, select **Docker Standard Output - Container**.
3. Select a destination project and Logstore, and then click **Next**.

   You can also click **Create Now** to create a project and a Logstore.

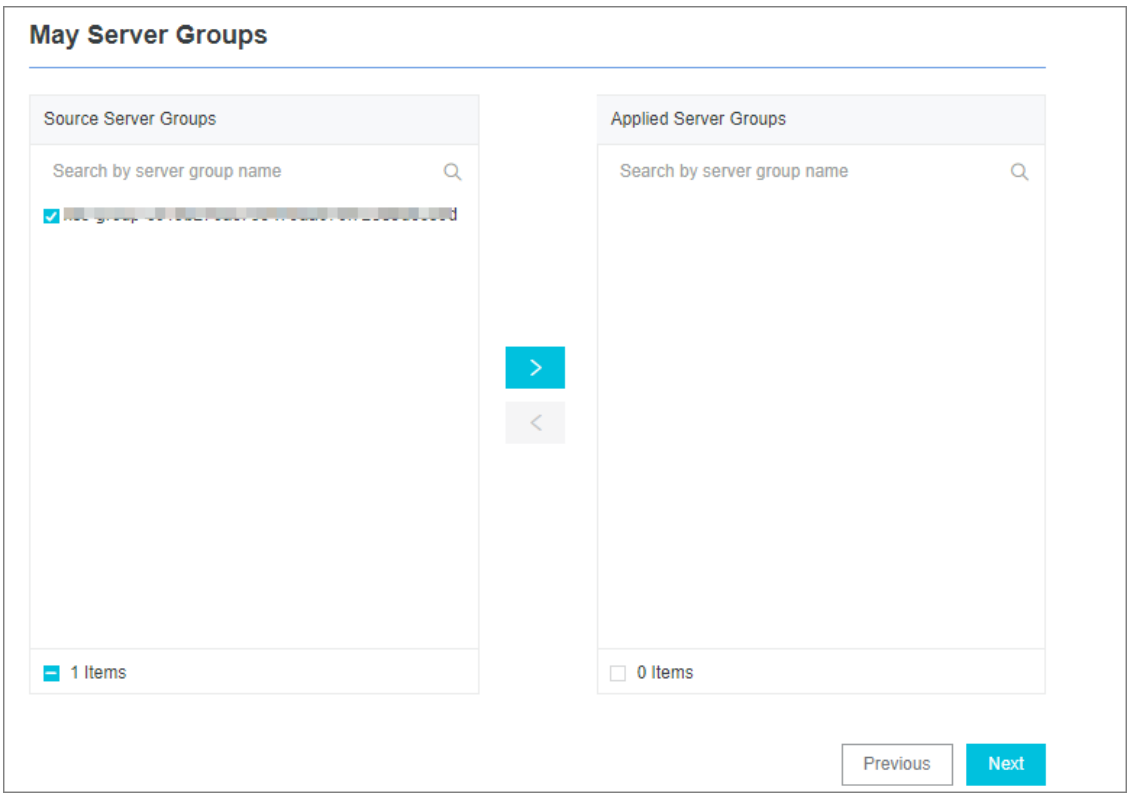
   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.

5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. In the Specify Data Source step, specify the data source and click **Next**.

   Configure the parameters that are used to collect logs in the **Plug-in Config** field. Example:

```
{
 "inputs": [
     {
         "type": "service_docker_stdout",
         "detail": {
             "Stdout": true,
             "Stderr": true,
             "IncludeLabel": {
                 "io.kubernetes.container.name": "nginx"
             },
             "ExcludeLabel": {
                 "io.kubernetes.container.name": "nginx-ingress-controller"
             },
             "IncludeEnv": {
                 "NGINX_SERVICE_PORT": "80"
             },
             "ExcludeEnv": {
                 "POD_NAMESPACE": "kube-system"
             }
         }
     }
 ]
}
```

The type of the input source is `service_docker_stdout`.

Parameter	Type	Required	Description
IncludeLabel	map	Yes	The value of the IncludeLabel parameter is a map. The keys and values of the map are strings. The default value of this parameter is an empty map. This default value indicates that logs from all containers are collected. If the keys are not empty and the values are empty, logs are collected from the containers whose label keys match the specified keys. **Note** ◦ Key-value pairs are in the logical OR relation. If a label key-value pair of a container matches one of the specified key-value pairs, the logs of the container are collected. ◦ By default, the values in the map are strings. Logs are collected from the containers whose names match the values. If you use a regular expression to specify a value, logs are collected from the containers whose names match the regular expression. If you specify a value that starts with a caret (^) and ends with a dollar sign ($), for example, ^(kube-system\|istio-system)$, logs are collected from a container named kube-system and a container named istio-system.

Parameter	Type	Required	Description
ExcludeLabel	map	No	The value of the ExcludeLabel parameter is a map. The keys and values of the map are strings. The default value of this parameter is an empty map and indicates that logs from all containers are collected. If the keys are not empty and the values are empty, logs are not collected from the containers whose label keys match the specified keys. **Note** ◦ Key-value pairs are in the logical OR relation. If a label key-value pair of a container matches one of the specified key-value pairs, the logs of the container are not collected. ◦ By default, the values in the map are strings. Logs are not collected from the containers whose names match the values. If you use a regular expression to specify a value, logs are not collected from the containers whose names match the regular expression. If you specify a value that starts with a caret (^) and ends with a dollar sign ($), for example, ^(kube-systemistio-system)$, logs are not collected from a container named kube-system or a container named istio-system.

Parameter	Type	Required	Description
IncludeEnv	map	No	The value of the IncludeEnv parameter is a map. The keys and values in the map are strings. The default value of this parameter is an empty map and indicates that logs from all containers are collected. If the keys are not empty and the values are empty, logs are collected from the containers whose environment variable keys match the specified keys. **Note** ◦ Key-value pairs are in the logical OR relation. If an environment variable key-value pair of a container matches one of the specified key-value pairs, the logs of the container are collected. ◦ By default, the values in the map are strings. Logs are collected from the containers whose names match the values. If you use a regular expression to specify a value, logs are collected from the containers whose names match the regular expression. If you specify a value that starts with a caret (^) and ends with a dollar sign ($), for example, ^(kube-systemistio-system)$, logs are collected from a container named kube-system and a container named istio-system.

Parameter	Type	Required	Description
ExcludeEnv	map	No	The value of the ExcludeEnv parameter is a map. The keys and values of the map are strings. The default value of this parameter is an empty map and indicates that logs from all containers are collected. If the keys are not empty and the values are empty, logs are not collected from the containers whose environment variable keys match the specified keys. **Note** ○ Key-value pairs are in the logical OR relation. If an environment variable key-value pair of a container matches one of the specified key-value pairs, the logs of the container are not collected. ○ By default, the values in the map are strings. Logs are not collected from the containers whose names match the values. If you use a regular expression to specify a value, logs are not collected from the containers whose names match the regular expression. If you specify a value that starts with a caret (^) and ends with a dollar sign ($), for example, ^(kube-systemistio-system)$, logs are not collected from a container named kube-system or a container named istio-system.
Stdout	bool	No	Default value: true. If you set the value of this parameter to false, stdout logs are not collected.
Stderr	bool	No	Default value: true. If you set the value of this parameter to false, stderr logs are not collected.
BeginLineRegex	string	No	The regular expression that is used to match the start part in the first line of a log. The default value of this parameter is an empty string. If a line matches the specified regular expression, the line is recorded to be the first line of a new log. Otherwise, the line is recorded to be a part of the last log.
BeginLineTimeoutMs	int	No	The timeout period for the specified regular expression to match the start part in the first line of a log. Default value: 3000. Unit: ms. If no new log is generated within 3 seconds, the last log is uploaded.
BeginLineCheckLength	int	No	The size of the start part in the first line of a log that matches the specified regular expression. Default value: 10 × 1,024. Unit: bytes. You can specify this parameter to check whether the start part in the first line of a log matches the regular expression. This improves match efficiency.
MaxLogSize	int	No	The maximum size of a log. Default value: 512 × 1,024. Unit: bytes. If the size of a log exceeds the specified value, the log is uploaded.

> **Note**
>
> - The preceding IncludeLabel and ExcludeLabel parameters are included in the label information that is retrieved by using the `docker inspect` command.
> - A namespace and a container name of a Kubernetes cluster can be mapped to a Docker label. The value of the LabelKey parameter for a namespace is `io.kubernetes.pod.namespace`. The value of the LabelKey parameter for a container name is `io.kubernetes.container.name`. For example, the namespace of the pod that you created is backend-prod and the container name is worker-server. In this case, if you set the key-value pair of an allowlist label to `io.kubernetes.pod.namespace : backend-prod`, the logs of all containers in the pod are collected. If you set the key-value pair of an allowlist label to `io.kubernetes.container.name : worker-server`, the logs of the container are collected.
> - In a Kubernetes cluster, we recommend that you specify only the `io.kubernetes.pod.namespace` and `io.kubernetes.container.name` labels. You can also specify the IncludeEnv or ExcludeEnv parameter based on your business requirements.

7. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

> **Note**
>
> - To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
> - If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

Default fields

- Common Docker containers

  The following table describes the fields that are uploaded by default for each log of a common Docker container.

Log field	Description
`_time_`	The point in time when data is uploaded. Example: `2018-02-02T02:18:41.979147844Z`.
`_source_`	The type of the input source. Valid values: stdout and stderr.
`_image_name_`	The name of the image.
`_container_name_`	The name of the container.
`_container_ip_`	The IP address of the container.

- Kubernetes

  The following table describes the fields that are uploaded by default for each log of a Kubernetes cluster.

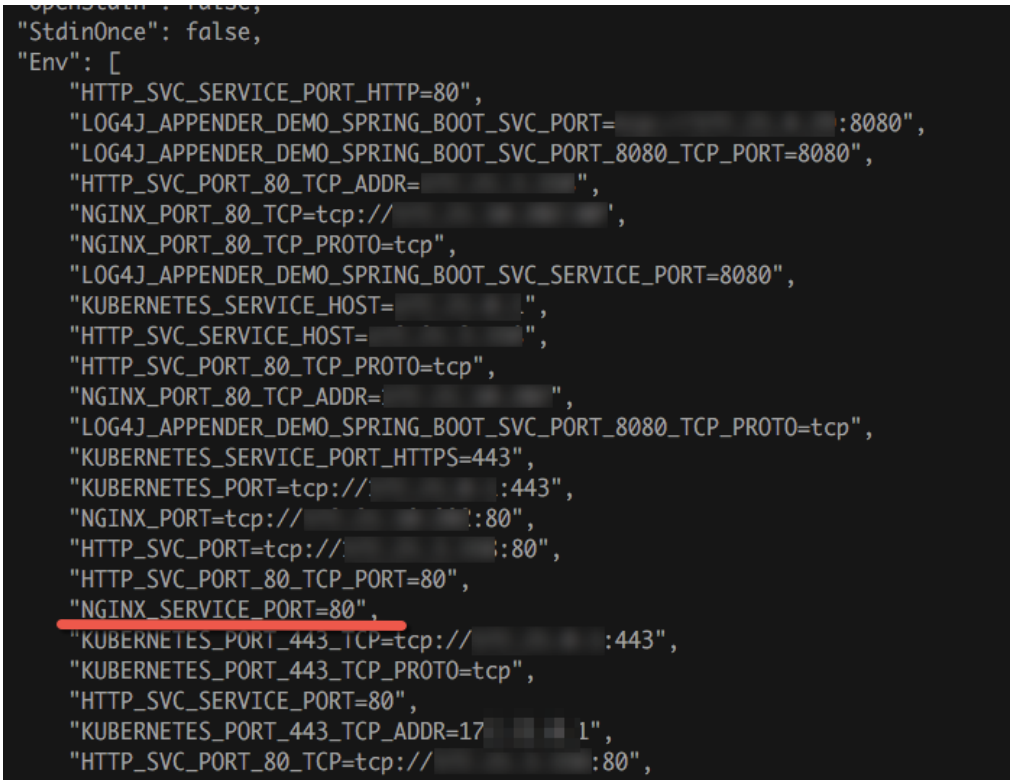
Log field	Description
time	The point in time when data is uploaded. Example: 2018-02-02T02:18:41.979147844Z.
source	The type of the input source. Valid values: stdout and stderr.
_image_name_	The name of the image.
_container_name_	The name of the container.
_pod_name_	The name of the pod.
namespace	The namespace where the pod resides.
_pod_uid_	The unique identifier of the pod.
_container_id_	The IP address of the pod.

Configuration examples of single-line log collection

- Configure environment variables

  Collect the stdout and stderr logs of the containers whose environment variables include NGINX_PORT_80_TCP_PORT=80 and exclude POD_NAMESPACE=kube-system.

  Configuration example of environment variables



  The following script shows the configurations of the environment variables:

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeEnv": {
                    "NGINX_PORT_80_TCP_PORT": "80"
                },
                "ExcludeEnv": {
                    "POD_NAMESPACE": "kube-system"
                }
            }
        }
    ]
}
```

- Configure labels

  Collect the stdout and stderr logs of the containers whose labels include `io.kubernetes.container.name=nginx` and exclude `type=pre`.

  Configuration example of labels

  The following script shows the label configurations:

```
{
    "inputs": [
        {
            "type": "service_docker_stdout",
            "detail": {
                "Stdout": true,
                "Stderr": true,
                "IncludeLabel": {
                    "io.kubernetes.container.name": "nginx"
                },
                "ExcludeLabel": {
                    "type": "pre"
                }
            }
        }
    ]
}
```

Configuration examples of multi-line log collection

Before you can collect Java exception stack logs, you must configure multi-line log collection. The following section describes how to collect stdout and stderr logs of standard Java applications.

- Sample log

```
2018-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoCon
troller : service start
2018-02-03 14:18:41.969 ERROR [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoCon
troller : java.lang.NullPointerException
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193
)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:199)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96)
...
2018-02-03 14:18:41.968  INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoCon
troller : service start done
```

- Log collection configuration

  Collect the logs of the containers whose labels include `app=monitor` and the specified first bytes of a line is of a fixed-format date type. To improve match efficiency, only the first 10 bytes of each line are checked.

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
]
}
```

Data processing examples

Logtail can process the collected Docker standard output. For more information, see Common data processing methods.

- Collect the logs of the containers whose labels include `app=monitor` and the specified first bytes of a line is of a fixed-format data type. To improve match efficiency, only the first 10 bytes of each line are checked. Regular expressions are used to parse logs into the values of the time, level, module, thread, and message. The following script shows the configurations of log collection and data processing:

```
{
"inputs": [
  {
    "detail": {
      "BeginLineCheckLength": 10,
      "BeginLineRegex": "\\d+-\\d+-\\d+.*",
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
],
"processors": [
    {
        "type": "processor_regex",
        "detail": {
            "SourceKey": "content",
            "Regex": "(\\d+-\\d+-\\d+ \\d+:\\d+:\\d+\\.\\d+)\\s+(\\w+)\\s+\\[([^]]+)]\\s+\\[([^]]+)
]\\s+:\\s+([\\s\\S]*)",
            "Keys": [
                "time",
                "level",
                "module",
                "thread",
                "message"
            ],
            "NoKeyError": true,
            "NoMatchError": true,
            "KeepSource": false
        }
    }
]
}
```

The collected log `2018-02-03 14:18:41.968 INFO [spring-cloud-monitor] [nio-8080-exec-4] c.g.s.web.controller.DemoController : service start done` is processed, as shown in the following script:

```
__tag__:__hostname__:logtail-dfgef
_container_name_:monitor
_image_name_:registry.cn-hangzhou.aliyuncs.xxxxxxxxxxxxxx
_namespace_:default
_pod_name_:monitor-6f54bd5d74-rtzc7
_pod_uid_:7f012b72-04c7-11e8-84aa-00163f00c369
_source_:stdout
_time_:2018-02-02T14:18:41.979147844Z
time:2018-02-02 02:18:41.968
level:INFO
module:spring-cloud-monitor
thread:nio-8080-exec-4
class:c.g.s.web.controller.DemoController
message:service start done
```

- Collect the JSON logs of the containers whose labels include `app=monitor`. The following script shows the configurations of log collection and data processing:

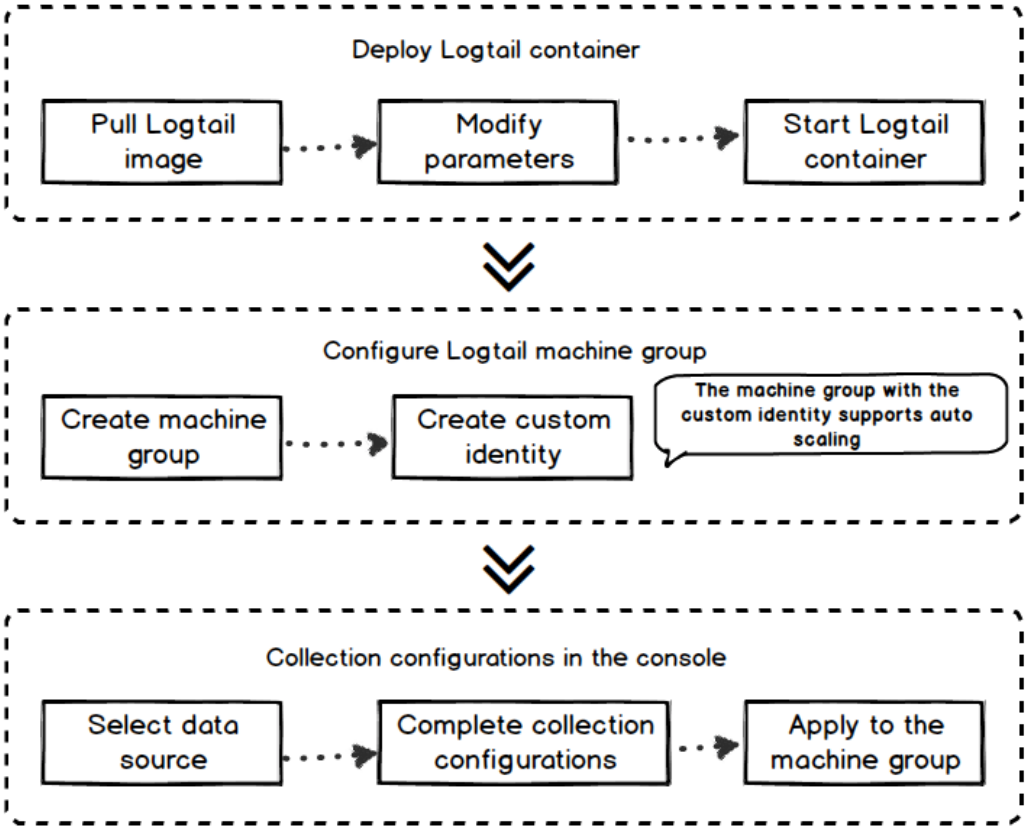
```
{
"inputs": [
  {
    "detail": {
      "IncludeLabel": {
        "app": "monitor"
      }
    },
    "type": "service_docker_stdout"
  }
],
"processors": [
    {
        "type": "processor_json",
        "detail": {
            "SourceKey": "content",
            "NoKeyError":true,
            "KeepSource": false
        }
    }
]
}
```

3.1.5.12. Collect standard Docker logs

This topic describes how to use Logtail to collect standard Docker logs and upload these logs together with the container metadata to Log Service.

Procedure

Procedure



1. Deploy a Logtail container.
2. Configure a Logtail server group.

   Create a server group with a custom ID in the Log Service console. The container cluster can automatically scale up or down based on data traffic.
3. Create a Logtail configuration.

   Create a Logtail configuration in the Log Service console. The Logtail configuration process is completed in the Log Service console. No local configuration is needed.

Deploy a Logtail container

1. Run the following command to pull the Logtail image.

   ```
   docker pull registry.cn-hangzhou.aliyuncs.com/log-service/logtail
   ```
2. Start a Logtail container.

   Set the `${your_region_name}`, `${your_aliyun_user_id}`, and `${your_machine_group_user_defined_id}` parameters in the startup template.

   ```
   docker run -d -v /:/logtail_host:ro -v /var/run:/var/run --env
   ALIYUN_LOGTAIL_CONFIG=/etc/ilogtail/conf/${your_region_name}/ilogtail_config.json
   --env ALIYUN_LOGTAIL_USER_ID=${your_aliyun_user_id} --env
    ALIYUN_LOGTAIL_USER_DEFINED_ID=${your_machine_group_user_defined_id} registry.cn-hangzhou.aliyunc
   s.com/log-service/logtail
   ```

Notice Before you set the parameters, you must complete one of the following configurations. Otherwise, the `container text file busy` error may occur when you delete another container.

- For CentOS 7.4 and later versions, set fs.may_detach_mounts to 1. For more information, see Bug 1468249, Bug 1441737, and Issue 34538.
- Grant the `privileged` permission to Logtail by adding the `--privileged` flag to the startup parameters. For more information, see Docker run reference.

Parameter	Description
`${your_region_name}`	The region of the project. For more information, see View the information of a project.
`${your_aliyun_user_id}`	The user ID. Set this parameter to the ID of your Alibaba Cloud account, which is a string. For information about how to view the ID, see Step 1 in Configure an account ID for a server.
`${your_machine_group_user_defined_id}`	The custom ID of your server group. For information about how to set the custom ID, see Step 1 in Create a machine group based on a custom ID.

After you set the parameters, run the following command to start the Logtail container.

```
docker run -d -v /:/logtail_host:ro -v /var/run:/var/run
--env ALIYUN_LOGTAIL_CONFIG=/etc/ilogtail/conf/cn_hangzhou/ilogtail_config.json --env
 ALIYUN_LOGTAIL_USER_ID=1654218******--env ALIYUN_LOGTAIL_USER_DEFINED_ID=log-docker-demo registry
.cn-hangzhou.aliyuncs.com/log-service/logtail
```

Notice

You can customize the startup parameters of the Logtail container if the following conditions are met:

- The following environment variables exist before you start the Logtail container: `ALIYUN_LOGTAIL_USER_DEFINED_ID`, `ALIYUN_LOGTAIL_USER_ID`, and `ALIYUN_LOGTAIL_CONFIG`.
- The */var/run* directory is mounted on the */var/run* directory of the Logtail container.
- To collect container standard output, container logs, or host files, you must mount the root directory on the `/logtail_host` directory of the Logtail container.
- If an error showing *The parameter is invalid : uuid=none* occurs in the `/usr/local/ilogtail/ilogtail.LOG` Logtail log file, create a file named *product_uuid* on the host. Add a valid UUID such as `169E98C9-ABC0-4A92-B1D2-AA6239C0D261` to the file, and mount the file on the */sys/class/dmi/id/product_uuid* directory of the Logtail container.

Configure a Logtail server group

1. Log on to the Log Service console.
2. Click a project name.
3. In the left-side navigation pane, click the **Server Groups** icon to show the server group list.
4. Click the icon next to Server Groups, and then select **Create Server Group**.

   You can also create a server group when you import data to Log Service.
5. In the dialog box that appears, select **Custom ID** from the Identifier drop-down list. Enter the value of `ALIYUN_LOGTAIL_USER_DEFINED_ID` set in the previous step in the **Custom Identifier** field.

Click OK. One minute later, click the name of the server group in the **Server Groups** list. On the **Server Group Settings** page that appears, you can view the heartbeat status of the Logtail container. For more information, see View the status of a server group.

Create a Logtail configuration

Create a Logtail configuration in the console.

- For more information about Docker logs, see Collect container text logs.
- For more information about Docker standard output, see Collect stdout and stderr logs from containers.
- Host text logs.

  The root directory of a host is mounted on the `/logtail_host` directory of the Logtail container by default. You must add the /logtail_host prefix to the log path. For example, if you want to collect data from the `/home/logs/app_log/` directory of the host, you must set the log path as `/logtail_host/home/logs/app_log/`.

What to do next

- View the status of the Logtail container.

  You can run the `docker exec ${logtail_container_id} /etc/init.d/ilogtaild status` command to view the status of Logtail.

- View the version number, IP address, and startup time of Logtail.

  You can run the `docker exec ${logtail_container_id} cat /usr/local/ilogtail/app_info.json` command to view Logtail information.

- View the operations logs of Logtail.

  The operations logs of Logtail are stored in the `ilogtail.LOG` file in the `/usr/local/ilogtail/` directory. If the log file is rotated and compressed, it is stored as a file named `ilogtail.LOG.x.gz`.

  For example:

```
[root@iZbp17enxc2us3624wexh2Z ilogtail]# docker exec a287de895e40 tail -n 5 /usr/local/ilogtail/ilo
gtail.LOG
[2018-02-06 08:13:35.721864]    [INFO]    [8]    [build/release64/sls/ilogtail/LogtailPlugin.cpp:10
4]    logtail plugin Resume:start
[2018-02-06 08:13:35.722135]    [INFO]    [8]    [build/release64/sls/ilogtail/LogtailPlugin.cpp:10
6]    logtail plugin Resume:success
[2018-02-06 08:13:35.722149]    [INFO]    [8]    [build/release64/sls/ilogtail/EventDispatcher.cpp:
369]    start add existed check point events, size:0
[2018-02-06 08:13:35.722155]    [INFO]    [8]    [build/release64/sls/ilogtail/EventDispatcher.cpp:
511]    add existed check point events, size:0    cache size:0    event size:0    success count:0
[2018-02-06 08:13:39.725417]    [INFO]    [8]    [build/release64/sls/ilogtail/ConfigManager.cpp:37
76]    check container path update flag:0    size:1
```

  Ignore the following standard output:

```
start umount useless mount points, /shm$|/merged$|/mqueue$
umount: /logtail_host/var/lib/docker/overlay2/3fd0043af174cb0273c3c7869500fbe2bdb95d13b1e110172ef57
fe840c82155/merged: must be superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/d5b10aa19399992755de1f85d25009528daa749c1bf8c16edff44
beab6e69718/merged: must be superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/5c3125daddacedec29df72ad0c52fac800cd56c6e880dc4e8a640
b1e16c22dbe/merged: must be superuser to unmount
...
xargs: umount: exited with status 255; aborting
umount done
start logtail
ilogtail is running
logtail status:
ilogtail is running
```

- Restart Logtail.

  To restart Logtail, use the following sample code:

```
[root@iZbp17enxc2us3624wexh2Z ilogtail]# docker exec a287de895e40 /etc/init.d/ilogtaild stop
kill process Name: ilogtail pid: 7
kill process Name: ilogtail pid: 8
stop success
[root@iZbp17enxc2us3624wexh2Z ilogtail]# docker exec a287de895e40 /etc/init.d/ilogtaild start
ilogtail is running
```

3.1.6. Custom plug-ins

Context

Log Service allows you to collect text logs and system logs by using Logtail. Logtail supports connections with multiple data sources, such as HTTP or MySQL query results and MySQL binary logs.

You can collect HTTP request data and upload the processing results to Log Service in real time to check service availability and perform continuous availability monitoring. You can configure MySQL query results as the data source, and then synchronize incremental data based on custom IDs or time. You can also configure an SQL data source to synchronize MySQL binary logs, subscribe to database changes, and query or analyze logs in real time.

> **Note** This feature supports only Logtail V0.16.0 or later that runs on Linux. For more information about Logtail versions and version updates, see Install Logtail in Linux.

Configuration process

1. Configure a collection method for a data source.

   Configure collection methods based on different data sources.

2. Configure a data processing method.

   Logtail provides multiple processing methods for binary logs, MySQL query results, NGINX monitoring data, and HTTP input sources. You can configure multiple processing methods for a single input source. Each input source supports all processing methods. Logtail runs the configured processing methods in sequence.

   For more information, see Configure data processing methods.

3. Apply the configurations to the specified machine group.

   Apply the log collection configurations and processing configurations to the specified machine group. Then, Logtail automatically pulls the configurations and starts to collect logs.

3.1.6.1. Collect MySQL binary logs

This topic describes how to create a Logtail configuration in the Log Service console to collect MySQL binary logs.

Prerequisites

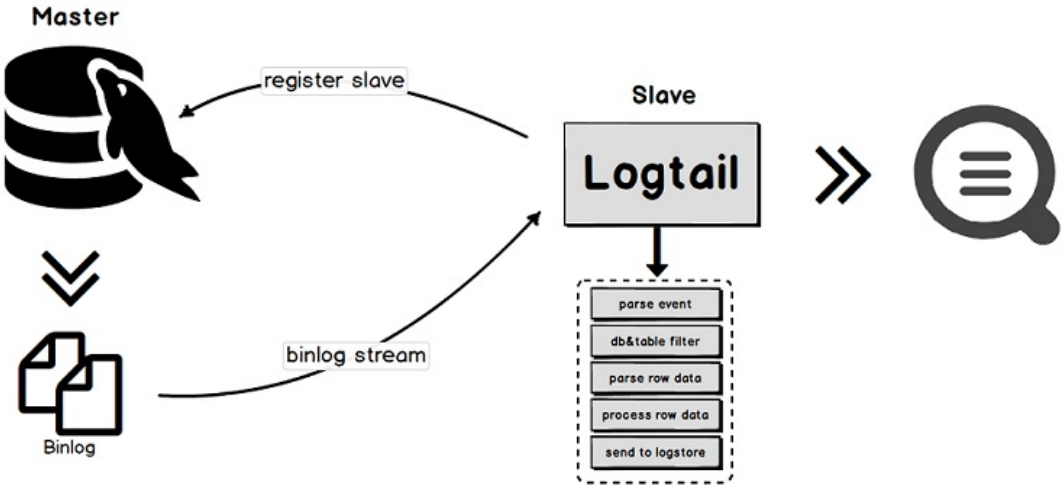
Logtail is installed on the server from which you want to collect MySQL binary logs. For more information, see Install Logtail in Linux or Install Logtail in Windows.

> **Note** Linux servers support Logtail V0.16.0 or later. Windows servers support Logtail V1.0.0.8 or later.

How it works

Logtail acts as a secondary MySQL node to communicate with the primary MySQL node. The following list describes the communication process:

1. Logtail acts as a secondary MySQL node and sends a dump request to the primary MySQL node.
2. After the primary MySQL node receives the dump request, the node sends binary logs to Logtail in real time.
3. Logtail performs operations such as event parsing, event filtering, and data parsing on binary logs. Then, Logtail uploads the parsed data to Log Service.



Features

- Binary logs can be incrementally collected. This way, you can collect data related to the update operations that are performed on your databases. MySQL databases such as ApsaraDB RDS for MySQL are supported.
- Multiple methods are provided to filter data in databases, such as regular expressions.
- You can specify the positions of binary log files.
- Checkpoints are used to synchronize data storage status.

Limits

- Binary logs of MySQL 8.0 or later cannot be collected.
- The binary logging feature must be enabled for your MySQL database, and the binlog_format parameter must be set to ROW for the database. By default, the feature is enabled for an RDS database.

- You can run the following command to check whether the binary logging feature is enabled:

  ```
  show variables like "log_bin";
  ```

  In this example, the following output is returned:

  ```
  +---------------+-------+
  | Variable_name | Value |
  +---------------+-------+
  | log_bin       | ON    |
  +---------------+-------+
  1 row in set (0.02 sec)
  ```

- Run the following command to view the format of binary logs:

  ```
  how variables like "binlog_format";
  ```

  In this example, the following output is returned:

  ```
  +---------------+-------+
  | Variable_name | Value |
  +---------------+-------+
  | binlog_format | ROW   |
  +---------------+-------+
  1 row in set (0.03 sec)
  ```

- The ID of the secondary MySQL node whose role Logtail assumes must be unique on the primary MySQL node.
- Limits on RDS databases:
  - Logtail cannot be installed on a server where an RDS instance resides. You must install Logtail on a server that can communicate with the RDS instance.
  - You cannot collect binary logs from a secondary RDS database. You must configure your primary RDS database to collect binary logs.

Scenarios

The MySQL binary logging feature applies to scenarios in which you need to synchronize large amounts of data and require high performance.

- Query and analyze the incremental data of databases in real time.
- Audit the operations that are performed on databases.
- Use Log Service to query and analyze database updates, visualize query and analysis results, transform data for stream computing, export log data to MaxCompute for offline computing, and export log data to Object Storage Service (OSS) for long-term storage.

Usage notes

We recommend that you increase resource limits on Logtail to accommodate traffic surges and prevent data risks. If the limits are exceeded, Logtail may be forced to restart.

You can modify the related parameters in the */usr/local/ilogtail/ilogtail_config.json* file. For more information, see Set Logtail startup parameters.

The following example shows how to increase the limit on CPU utilization to two cores and the limit on memory usage to 2,048 MB:

```
{
    ...
    "cpu_usage_limit":2,
    "mem_usage_limit":2048,
    ...
}
```

Data reliability

We recommend that you enable the global transaction identifier (GTID) feature on your MySQL server and upgrade Logtail to V0.16.15 or later. This prevents data from being repeatedly collected after a primary/secondary switchover is triggered on your database and ensures data reliability.

- Incomplete data collection: If the network between Logtail and your MySQL server is disconnected for a long period of time, some data may not be collected.

  If the network between Logtail and your primary MySQL node is disconnected, the primary node still generates binary logs and deletes expired binary logs. After the network connection is re-established and your primary MySQL node and Logtail are reconnected, Logtail uses a checkpoint to request binary log data from the primary MySQL node. However, if the network is disconnected for a long period of time, the data that is generated after the checkpoint may be deleted. In this case, the recovery mechanism is triggered. The recovery mechanism identifies the most recent binary log file position from which Logtail resumes collection on the primary MySQL node. The data that is generated between the checkpoint and the most recent binary log file position is not collected. This leads to incomplete data collection.

- Repeated data collection: If a primary/secondary switchover is triggered when the sequence numbers of binary logs are inconsistent between your primary MySQL node and secondary MySQL node, binary logs may be repeatedly collected.

  If you configure primary/secondary synchronization for MySQL, the primary node automatically synchronizes the binary logs to the secondary node. The secondary node stores the logs to local binary log files. If the sequence numbers are inconsistent between the primary and secondary nodes and a primary/secondary switchover is triggered, logs may be repeatedly collected. This issue occurs because the checkpoint mechanism identifies checkpoints based on the names of binary log files and the offsets of the files.

  For example, a data block is in the checkpoint range from `(binlog.100, 4)` to `(binlog.105, 4)` on the primary MySQL node, and in the checkpoint range from `(binlog.1000, 4)` to `(binlog.1005, 4)` on the secondary MySQL node. Logtail has obtained the data from the primary node and updated the local checkpoint to `(binlog.105, 4)`. If a primary/secondary switchover is triggered and no error occurs, Logtail continues to collect binary logs from the new primary node based on the local checkpoint `(binlog.105, 4)`. However, the sequence number of the data that is in the checkpoint range from `(binlog.1000, 4)` to `(binlog.1005, 4)` on the new primary node is greater than the sequence number of the data that is requested by Logtail. The new primary node returns all data in the range to Logtail. This leads to repeated data collection.

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, select **MySQL BinLog - Plug-in**.
3. Select a destination project and Logstore, and then click **Next**.

   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.

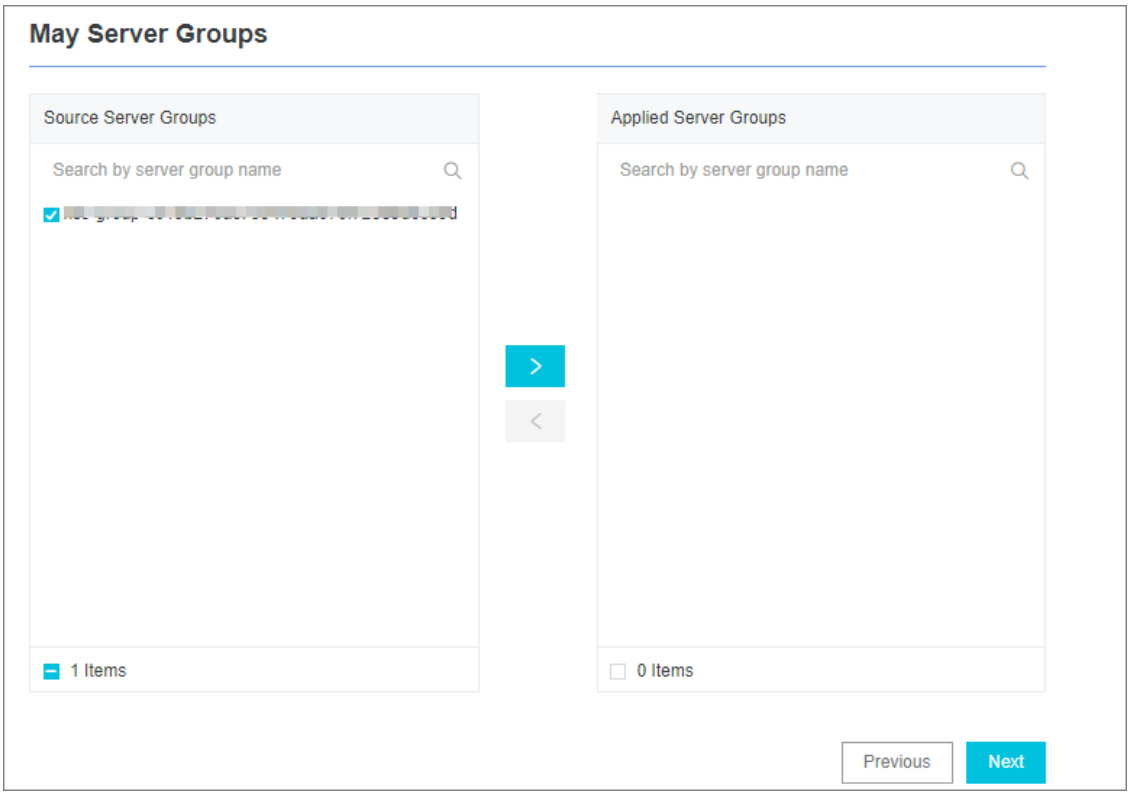
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.

5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. In the **Specify Data Source** step, configure the **Config Name** and **Plug-in Config** parameters.

   A template is provided for the **Plug-in Config** parameter. You can configure the inputs and processors parameters in the template.

   - inputs: specifies the collection configurations. This parameter is required. Configure the inputs parameter based on your data source.

     > **Note** You can configure only one type of data source in the inputs field.

   - processors: specifies the processing method. This parameter is optional. You can configure one or more processing methods in the processors field. For more information, see Configure data processing methods.

```
{
 "inputs": [
     {
         "type": "service_canal",
         "detail": {
             "Host": "************.mysql.rds.aliyuncs.com",
             "Port": 3306,
             "User" : "root",
             "ServerID" : 56321,
             "Password": "*******",
             "IncludeTables": [
                 "user_info\\..*"
             ],
             "ExcludeTables": [
                 ".*\\.\\S+_inner"
             ],
             "TextToString" : true,
             "EnableDDL" : true
         }
     }
 ]
}
```

Parameter	Type	Required	Description
type	string	Yes	The type of the data source. Set the value to service_canal.
Host	string	No	The IP address of the primary MySQL server. Default value: 127.0.0.1.
Port	int	No	The port that is used to access the database of the primary MySQL server. Default value: 3306.

User	string	No	The username of the account that is used to log on to the database. Default value: root.
			Make sure that the user is granted the read permissions on the database and the REPLICATION permission. Example:
			`CREATE USER canal IDENTIFIED BY 'canal';`
			`GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'canal'@'%';`
			`-- GRANT ALL PRIVILEGES ON *.* TO 'canal'@'%' ;`
			`FLUSH PRIVILEGES;`

Parameter	Type	Required	Description
Password	string	No	The password of the account that is used to log on to the database. By default, this parameter is left empty. If you have high requirements for data security, we recommend that you set the username and the password to xxx. After your configurations are synchronized to the on-premises server, find the Password parameter in the */usr/local/ilogtail/user_log_config.json* file and change the value. For more information, see Modify the Logtail configuration on the Logtail server. **Note** If you modify this parameter in the console, the on-premises configuration is overwritten after the modification is synchronized to the on-premises server.
ServerID	int	No	The ID of the secondary MySQL node whose role Logtail assumes. Default value: 125. **Note** The value of the ServerID parameter must be unique for your database. Otherwise, data collection fails.
IncludeTables	String array	Yes	The names of the tables from which data is collected. Each name must include the name of the database to which a table belongs and the name of the table. Example: test_db.test_table. You can specify a regular expression for this parameter. Logtail collects data only from tables whose names match the regular expression specified by the IncludeTables parameter. To collect data from all tables of a database, set the IncludeTables parameter to .*\\..*. **Note** To implement exact match, add ^ to the beginning of a regular expression and $ to the end. Example: ^test_db\\.test_table$.
ExcludeTables	String array	No	The names of the tables from which data is not collected. Each name must include the name of the database to which a table belongs and the name of the table. Example: test_db.test_table. You can specify a regular expression for this parameter. If a table meets one of the conditions that are specified by the ExcludeTables parameter, data from the table is not collected. If you do not configure this parameter, data from all tables is collected. **Note** To implement exact match, add ^ to the beginning of a regular expression and $ to the end. Example: ^test_db\\.test_table$.

Parameter	Type	Required	Description

StartBinName	string	No	The name of the binary log file from which Logtail starts to collect data for the first time. If you do not configure this parameter, Logtail starts to collect data from the current time.
			If you want Logtail to collect data from a specified position of a binary log file, set the StartBinName parameter to the name of the binary log file and set the StartBinlogPos parameter to the offset of the file. For example, you can set the StartBinName parameter to "mysql-bin". 000063" and the StartBinlogPos parameter to 0.
			`show binary logs;`
			Example:
			`+------------------+-----------+`
			`| Log_name | File_size |`
			`+------------------+-----------+`
			`| mysql-bin.000063 | 241 |`
			`| mysql-bin.000064 | 241 |`
			`| mysql-bin.000065 | 241 |`
			`| mysql-bin.000066 | 10778 |`
			`+------------------+-----------+`
			`4 rows in set (0.02 sec)`
			Note
			If you specify a value for the StartBinName parameter, a large volume of traffic is generated when Logtail collects data for the first time.

StartBinlogPos	int	No	The offset of the binary log file from which Logtail starts to collect data for the first time. Default value: 0.
EnableGTID	bool	No	Specifies whether to add GTID. For more information, see GTID. Default value: true. If you set the value to false, no GTIDs are added to the data that is uploaded to Log Service.
EnableInsert	bool	No	Specifies whether to collect the data on INSERT events. Default value: true. If you set the value to false, Logtail does not collect the data on INSERT events.
EnableUpdate	bool	No	Specifies whether to collect the data on UPDATE events. Default value: true. If you set the value to false, Logtail does not collect the data on UPDATE events.
EnableDelete	bool	No	Specifies whether to collect the data on DELETE events. Default value: true. If you set the value to false, Logtail does not collect the data on DELETE events.

Parameter	Type	Required	Description
EnableDDL	bool	No	Specifies whether to collect the data on data definition language (DDL) events. Default value: false. This value indicates that Logtail does not collect the data on DDL events. **Note** If you set the value to true, the IncludeTables and ExcludeTables parameters become unavailable.
Charset	string	No	The encoding format. Default value: utf-8.
TextToString	bool	No	Specifies whether to convert the data of the text type to the string type. Default value: false. This value indicates that the data type is not converted.
PackValues	bool	No	Specifies whether to pack event data in the JSON format. Default value: false. This value indicates that Logtail does not pack event data. If you set the value to true, Logtail packs event data into the data and old_data fields in the JSON format. The old_data field is available only for ROW_UPDATE events. For example, a table contains three columns named c1, c2, and c3. If you set the value to false, the data on ROW_INSERT events contains the c1, c2, and c3 fields. If you set the value to true, Logtail packs all data in the c1, c2, and c3 columns into the data field whose values are in the `{"c1":"...", "c2": "...", "c3": "..."}` format. **Note** This parameter is available only for Logtail V0.16.19 and later.
EnableEvent Meta	bool	No	Specifies whether to collect the metadata of events. Default value: false. This value indicates that Logtail does not collect the metadata of events. The metadata of binary log events includes event_time, event_log_position, event_size, and event_server_id. **Note** This parameter is available only for Logtail V0.16.21 and later.

7. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   Note
   - To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   - If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After the Logtail configuration is delivered to the Logtail server, Logtail immediately collects and sends data to Log Service when changes are made to your database.

> **Note** By default, Logtail collects the incremental data of binary logs.

Modify the Logtail configuration on the Logtail server

If you did not enter real information for parameters such as Host, User, and Password in the **Plug-in Config** field when you created the Logtail configuration, you can modify the parameters after the Logtail configuration is delivered to the Logtail server.

1. Log on to the server where Logtail is installed.
2. Find the service_canal keyword in the */usr/local/ilogtail/user_log_config.json* file and modify parameters such as Host, User, and Password.
3. Run the following command to restart Logtail:

```
sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
```

What's next

After Logtail collects and sends MySQL binary logs to Log Service, you can view the logs in the Log Service console. For example, after you perform the `INSERT`, `UPDATE`, and `DELETE` operations on the `SpecialAlarm` table of the `user_info` database, Logtail collects and sends binary logs to Log Service. The following list describes the schema of the table, the operations that are performed on the table, and the collected logs.

- Table schema

```
CREATE TABLE `SpecialAlarm` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`time` datetime NOT NULL,
`alarmtype` varchar(64) NOT NULL,
`ip` varchar(16) NOT NULL,
`count` int(11) unsigned NOT NULL,
PRIMARY KEY (`id`),
KEY `time` (`time`) USING BTREE,
KEY `alarmtype` (`alarmtype`) USING BTREE
) ENGINE=MyISAM AUTO_INCREMENT=1;
```

- Database operations

  Perform the INSERT, DELETE, and UPDATE operations.

```
insert into specialalarm (`time`, `alarmType`, `ip`, `count`) values(now(), "NO_ALARM", " 203.0.**.
***", 55);
delete from specialalarm where id = 4829235  ;
update specialalarm set ip = " 203.0.***.**" where id = "4829234";
```

  Create an index for `zc.specialalarm`.

```
ALTER TABLE `zc`.`specialalarm`
ADD INDEX `time_index` (`time` ASC);
```

- Collected logs

  You can view the logs that are collected for each operation on the Search & Analysis page of the Logstore that is specified in the Logtail configuration. Examples:

- INSERT statement

```
__source__:  203.0.**.**
__tag__:__hostname__:  iZbp145dd9fccu*****
__topic__:
_db_:  zc
_event_:  row_insert
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:536
_host_:  *********.mysql.rds.aliyuncs.com
_id_:  113
_table_:  specialalarm
alarmtype:  NO_ALARM
count:  55
id:  4829235
ip:  203.0***.***
time:  2017-11-01 12:31:41
```

- DELETE statement

```
__source__:  10.30.**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_delete
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:537
_host_:  *********.mysql.rds.aliyuncs.com
_id_:  114
_table_:  specialalarm
alarmtype:  NO_ALARM
count:  55
id:  4829235
ip:  10.10.**.***
time:  2017-11-01 12:31:41
```

- UPDATE statement

```
__source__:  203.0**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_update
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:538
_host_:  *********.mysql.rds.aliyuncs.com
_id_:  115
_old_alarmtype:  NO_ALARM
_old_count:  55
_old_id:  4829234
_old_ip:  203.0.113.1
_old_time:  2017-10-31 12:04:54
_table_:  specialalarm
alarmtype:  NO_ALARM
count:  55
id:  4829234
ip:  203.0.***.***
time:  2017-10-31 12:04:54
```

- DDL statement

```
__source__:  203.0.**.**
__tag__:__hostname__:  iZbp145dd9fccu****
__topic__:
_db_:  zc
_event_:  row_update
_gtid_:  7d2ea78d-b631-11e7-8afb-00163e0eef52:539
_host_:  *********.mysql.rds.aliyuncs.com
ErrorCode:  0
ExecutionTime:  0
Query:  ALTER TABLE `zc`.`specialalarm`
ADD INDEX `time_index` (`time` ASC)
StatusVars:
```

Field	Description
host	The hostname of the database.
_db_	The name of the database.
table	The name of the table.
event	The type of the event.
_id_	The auto-increment ID. IDs start from 0 and increment by 1 each time data on a binary log event is collected.
gtid	The GTID.
filename	The name of the binary log file.
offset	The offset of the binary log file. The value is updated only when a COMMIT operation is performed.

3.1.6.2. Collect MySQL query results

This topic describes how to create a Logtail configuration in the Log Service console to collect MySQL query results.

Prerequisites

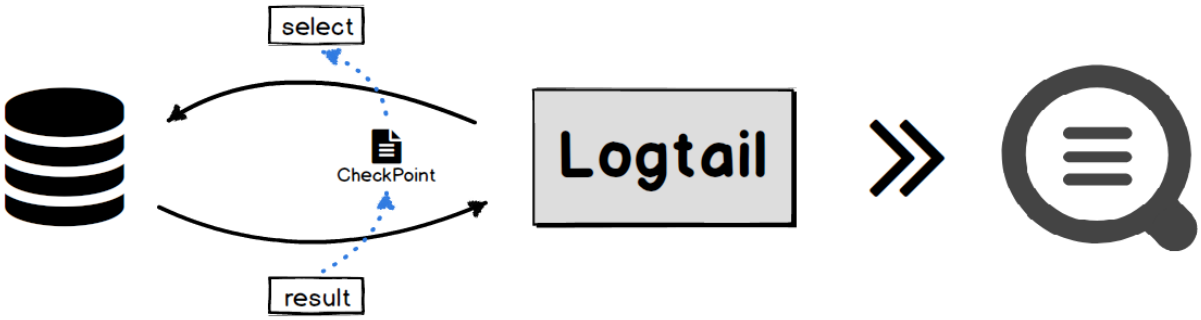
Logtail is installed on the server from which you want to collect MySQL query results. For more information, see Install Logtail in Linux or Install Logtail in Windows.

> **Note** Linux servers support Logtail V0.16.0 or later. Windows servers support Logtail V1.0.0.8 or later.

Implementation

Logtail executes the SELECT statement that is specified in a Logtail configuration on a regular basis, and then uploads the query results to Log Service.

After Logtail obtains query results, Logtail saves the value of the CheckPoint field in the results to the Logtail server. The next time Logtail executes the SELECT statement, Logtail adds the value of the CheckPoint field to the SELECT statement. This way, Logtail can collect incremental data.



Features

- MySQL databases are supported.
- You can configure paged query settings.
- You can specify time zones.
- You can specify timeout periods.
- The values of the CheckPoint field can be saved.
- SSL is supported.
- You can specify the maximum size of data that can be collected at a time.

Scenarios

- Collect incremental data based on marks such as an auto-increment ID or a point in time.
- Synchronize data based on filter conditions.

Procedure

The following procedure describes how to synchronize incremental data from a MySQL database to Log Service. In this procedure, the logtail.VersionOs field is synchronized every 10 seconds and the value of the count parameter in this field is greater than 0. The value of the initial checkpoint is 2017-09-25 11:00:00. Logs are paginated and each page contains 100 logs. The checkpoint of each page is saved. The procedure includes the following steps:

1. Log on to the Log Service console.
2. Select a data source.

   In the **Import Data** section, select **MySQL Query Result - Plug-in**.
3. Select a destination project and Logstore, and then click **Next**.

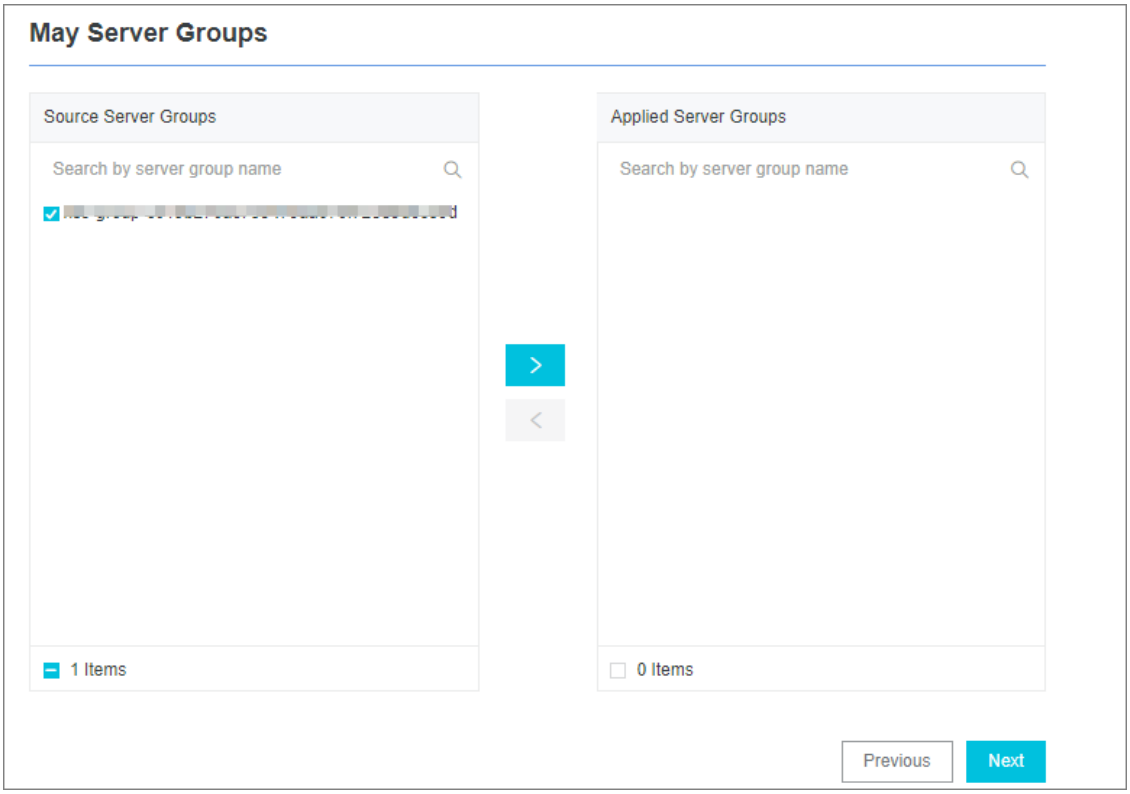
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



Notice If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. In the **Specify Data Source** step, configure the **Config Name** and **Plug-in Config** parameters.
   - In the **Plug-in Config** field, modify the parameter settings in the default configuration template based on your business requirements.
   - `inputs`: specifies the collection configurations. This parameter is required. `processors`: specifies the processing method. This parameter is optional. You must specify statements to collect data based on your data source. For more information, see Configure data processing methods.

Note If you have high requirements for data security, we recommend that you set the username and the password to xxx. After your configurations are synchronized to the on-premises server, find the User and Password parameters in the /usr/local/ilogtail/user_log_config.json file and change the values.

The following example shows the configurations:

```
{
  "inputs": [
    {
      "type": "service_mysql",
      "detail": {
        "Address": "************.mysql.rds.aliyuncs.com",
        "User": "****",
        "Password": "*******",
        "DataBase": "****",
        "Limit": true,
        "PageSize": 100,
        "StateMent": "select * from db.VersionOs where time > ?",
        "CheckPoint": true,
        "CheckPointColumn": "time",
        "CheckPointStart": "2018-01-01 00:00:00",
        "CheckPointSavePerPage": true,
        "CheckPointColumnType": "time",
        "IntervalMs": 60000
      }
    }
  ]
}
```

Parameter	Type	Required	Description
type	string	Yes	The type of the data source. Set the value to service_mysql.
Address	string	No	The address of the MySQL database. Default value: 127.0.0.1:3306.
User	string	No	The username of the account that you use to log on to the MySQL database. Default value: root.
Password	string	No	The password of the account that you use to log on to the MySQL database. By default, this parameter is left empty. If you have high requirements for data security, we recommend that you set the username and the password to xxx. After your configurations are synchronized to the on-premises server, find the User and Password parameters in the */usr/local/ilogtail/user_log_config.json* file and change the values. **Note** If you modify this parameter in the console, the on-premises configuration is overwritten after the modification is synchronized to the on-premises server.
DialTimeOutMs	int	No	The timeout period for connections to the MySQL database. Unit: milliseconds. Default value: 5000.
ReadTimeOutMs	int	No	The timeout period for data reads from the MySQL database. Unit: milliseconds. Default value: 5000.

Parameter	Type	Required	Description
StateMent	string	No	The SQL statement. If you set the CheckPoint parameter to true, you must include the CheckPointColumn parameter in the WHERE clause of the SQL statement that you specified for the StateMent parameter. You must also set the CheckPointColumn parameter to ?. For example, if you set the CheckPointColumn parameter to id, you must specify the value of the StateMent parameter in the `SELECT * from ... where id > ?` format.
Limit	bool	No	Specifies whether to use a LIMIT clause to paginate query results. Default value: false. This value indicates that query results are not paginated. We recommend that you set the Limit parameter to true. If you set the Limit parameter to true, a LIMIT clause is automatically appended to the SQL statement that you specified for the StateMent parameter when Logtail executes the SQL statement.
PageSize	int	No	The maximum number of logs to return on each page. If you set the Limit parameter to true, you must configure this parameter.
MaxSyncSize	int	No	The maximum number of logs that can be synchronized at a time. Default value: 0. This value indicates that no limit is placed on the size of data that can be synchronized at a time.
CheckPoint	bool	No	Specifies whether to use checkpoints during data collection. Default value: false. This value indicates that checkpoints are not used during data collection.
CheckPointColumn	string	No	The name of the checkpoint column. If you set the CheckPoint parameter to true, you must configure this parameter.
CheckPointColumnType	string	No	The type of the checkpoint column. Valid values: int and time. If you set this parameter to int, the values in the checkpoint column are stored as 64-bit integers. If you set this parameter to time, the values in the checkpoint column can be of the date, time, or datetime type that is supported by MySQL. If you set the CheckPoint parameter to true, you must configure this parameter.
CheckPointStart	string	No	The initial value of the checkpoint. If you set the CheckPoint parameter to true, you must configure this parameter.
CheckPointSavePerPage	bool	No	If you set this parameter to true, a checkpoint is saved after each pagination. If you set this parameter to false, a checkpoint is saved after each synchronization.
IntervalMs	int	Yes	The synchronization interval. Unit: milliseconds.

7. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > - To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > - If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

Modify the configurations on the server where Logtail is installed

If you did not enter real information for parameters such as Address, User, and Password in the **Plug-in Config** field when you created the Logtail configuration, you can modify the parameters after the Logtail configuration is delivered to the Logtail server.

1. Log on to the server where Logtail is installed.
2. Find the service_mysql keyword in the */usr/local/ilogtail/user_log_config.json* file and modify parameters such as Address, User, and Password.
3. Run the following command to restart Logtail:

```
sudo /etc/init.d/ilogtaild stop; sudo /etc/init.d/ilogtaild start
```

Example

After Logtail collects and sends MySQL query results to Log Service, you can view the results in the Log Service console. This section shows a sample table schema and a sample log that is collected by Logtail.

- Table schema

```
CREATE TABLE `VersionOs` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT 'id',
  `time` datetime NOT NULL,
  `version` varchar(10) NOT NULL DEFAULT '',
  `os` varchar(10) NOT NULL,
  `count` int(11) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `timeindex` (`time`)
)
```

- Sample log

```
"count":  "4"
"id:  "721097"
"os:  "Windows"
"time:  "2017-08-25 13:00:00"
"version":  "1.3.0"
```

3.1.6.3. Collect syslogs

This topic describes how to create a Logtail configuration in the Log Service console to collect syslogs.

Prerequisites

Logtail is installed on the server from which you want to collect syslogs. For more information, see Install Logtail in Linux or Install Logtail in Windows.

Note Linux servers support Logtail V0.16.13 or later. Windows servers support Logtail V1.0.0.8 or later.

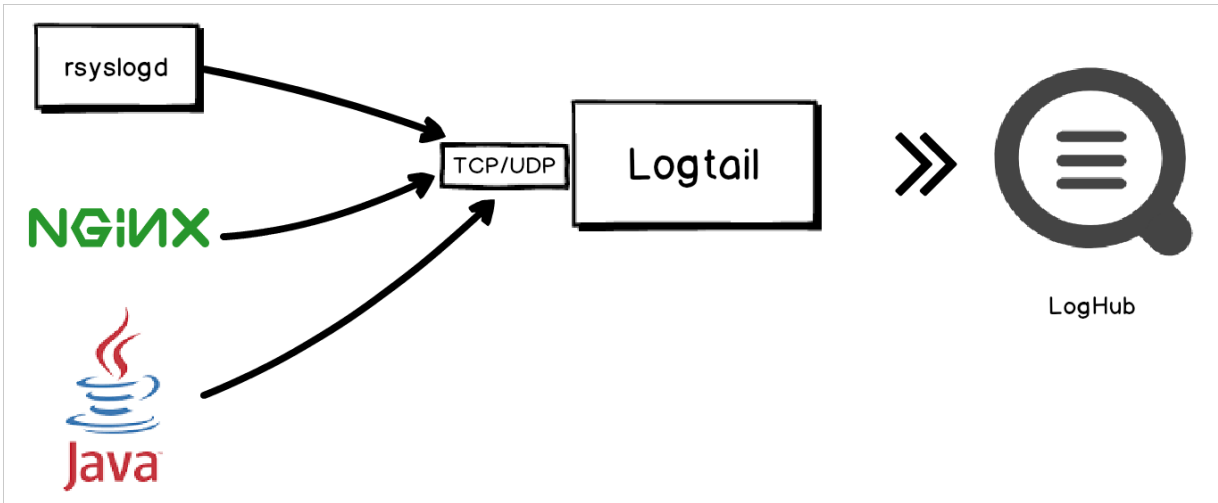
Overview

Linux servers allow you to use syslog agents such as rsyslog to forward on-premises syslogs to the IP address and port of a specified server. After you apply a Logtail configuration to the specified server, the Logtail plug-in specified in the configuration receives the forwarded syslogs over TCP or UDP. The plug-in also parses the syslogs based on the specified syslog protocol, and extracts the facility, tag(program), severity, and content fields from the syslogs. The syslog protocols defined in RFC 3164 and RFC 5424 are supported. For more information, see RFC 5424 and RFC 3164.

You can configure multiple Logtail plug-ins based on your business requirements. For example, you can configure two Logtail plug-ins to listen on 127.0.0.1:9999 over TCP and UDP.

Implementation

After you configure Logtail plug-ins to listen on a specified address and port, Logtail collects and sends data to Log Service. The data includes the system logs that are collected by using rsyslog, the access logs or error logs that are forwarded by NGINX, and the logs that are forwarded by syslog clients.



Configure Logtail plug-ins to collect syslogs

1. Add a forwarding rule for rsyslog.
   i. Modify the */etc/rsyslog.conf* configuration file of rsyslog on the server from which you want to collect syslogs. Add a forwarding rule to the end of the configuration file.

      After the forwarding rule is added, rsyslog forwards syslogs to a specified IP address and port.

      - If Logtail resides on the syslog server, you must specify the IP address 127.0.0.1 and a non-well-known port that is unoccupied in the forwarding rule.
      - If Logtail resides on a different server from the syslog server, you must specify the public IP address of the different server and a non-well-known port that is unoccupied in the forwarding rule.

      The following example shows a forwarding rule, which allows all syslogs to be forwarded to 127.0.0.1:9000 over TCP. For more information about the configuration file, see RSyslog Documentation.

      ```
      *.* @@127.0.0.1:9000
      ```

   ii. Run the following command to restart rsyslog and validate the log forwarding rule:

      ```
      sudo service rsyslog restart
      ```

2. Log on to the Log Service console.

3. In the **Import Data** section, select **Custom Data Plug-in**.
4. Select a destination project and Logstore, and then click **Next**.

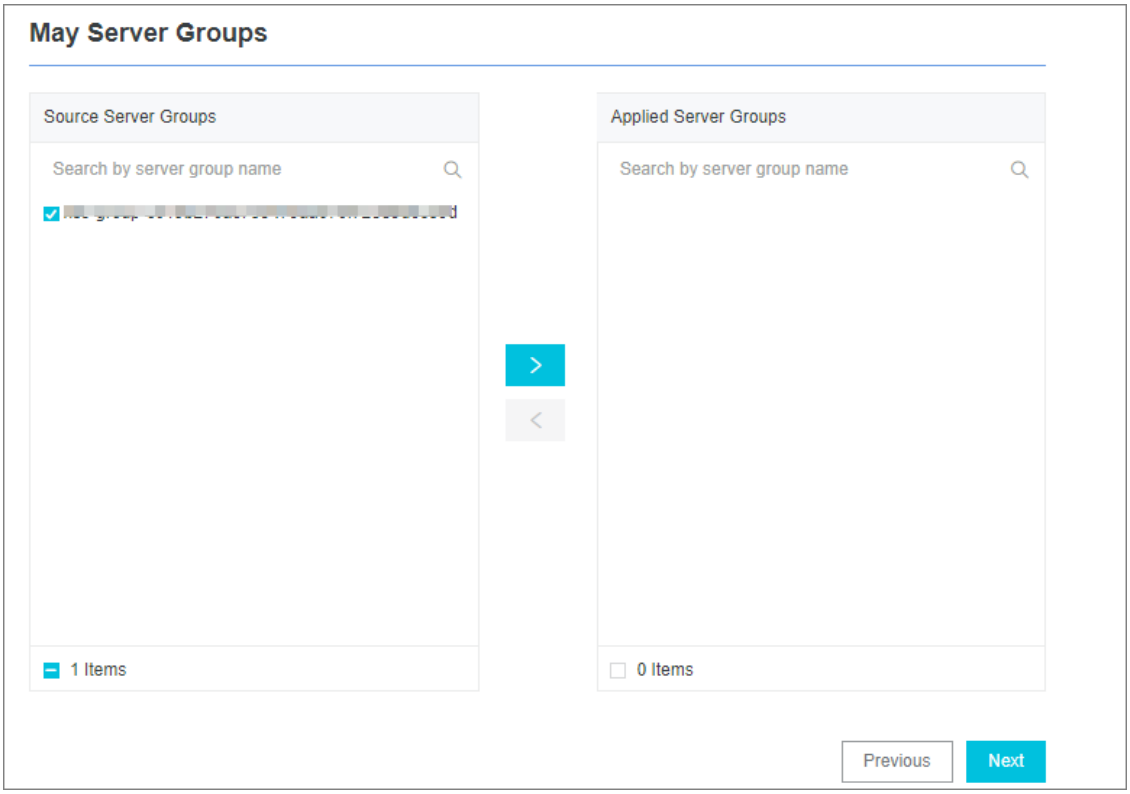
   You can also click **Create Now** to create a project and a Logstore.

   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
5. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

   After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.
6. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

7. In the **Specify Data Source** step, configure the **Config Name** and **Plug-in Config** parameters.
   - inputs: specifies the collection configurations. This parameter is required. Configure the inputs parameter based on your data source.

     > **Note** You can configure only one type of data source in the inputs field.

   - processors: specifies the processing method. This parameter is optional. You can configure one or more processing methods in the processors field. For more information, see Configure data processing methods.

   The following example shows how to configure Logtail plug-ins to listen on 127.0.0.1:9000 over UDP and TCP:

```
{
    "inputs": [
        {
            "type": "service_syslog",
            "detail": {
                "Address": "tcp://127.0.0.1:9000",
                "ParseProtocol": "rfc3164"
            }
        },
        {
            "type": "service_syslog",
            "detail": {
                "Address": "udp://127.0.0.1:9001",
                "ParseProtocol": "rfc3164"
            }
        }
    ]
}
```

Parameter	Type	Required	Description
type	string	Yes	The type of the data source. Set the value to service_syslog.
Address	string	No	The listening protocol, address, and port that are used by a Logtail plug-in. The plug-in listens on and obtains data based on the Logtail configuration. The value of the parameter is in the [tcp/udp]://[*ip*]:[*port*] format. Default value: tcp://127.0.0.1:9999. **Note** ◦ The listening protocol, address, and port that you specify must be the same as those specified in the forwarding rule that is added to the configuration file of rsyslog. ◦ If the Logtail server uses multiple IP addresses to receive data, set the IP address to 0.0.0.0. This address indicates that the plug-in listens on all the IP addresses of the server.
ParseProtocol	string	No	The protocol that is used to parse syslogs. By default, this parameter is empty. If you leave this parameter empty, the system does not parse syslogs. Valid values: ◦ rfc3164: The RFC 3164 protocol is used to parse syslogs. ◦ rfc5424: The RFC 5424 protocol is used to parse syslogs. ◦ auto: The plug-in automatically selects a protocol based on the content of syslogs.

Parameter	Type	Required	Description
IgnoreParseFailure	boolean	No	Specifies whether to perform an operation on a syslog after the syslog fails to be parsed. Default value: true. This value true indicates that the system does not parse the syslog and adds the syslog to the content field. If you set the value to false, the syslog is discarded after it fails to be parsed.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > - To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > - If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

Configure Logtail plug-ins to collect NGINX logs

NGINX servers allow you to forward access logs to specified IP addresses and ports by using the syslog protocol. If you want to deliver all data of a server as syslogs to Log Service, you can create a Logtail configuration to collect the data. The data includes NGINX access logs.

1. Add a forwarding rule for NGINX.
   i. Add a forwarding rule to the *nginx.conf* configuration file on the NGINX server. For more information, see NGINX Beginner's Guide.

      The following sample code provides an example of a forwarding rule:

      ```
      http {
          ...
          # Add this line.
          access_log syslog:server=127.0.0.1:9000,facility=local7,tag=nginx,severity=info combined;
          ...
      }
      ```

   ii. Run the following command to restart the NGINX service and validate the forwarding rule:

      ```
      sudo service nginx restart
      ```

2. Create a Logtail configuration.

   For more information, see Configure Logtail plug-ins to collect syslogs.

What's next

After Logtail collects and sends syslogs to Log Service, you can view the logs in the Log Service console.

Log field	Description
hostname	The hostname. If no hostname is included in the log, the hostname of the current host is obtained.
program	The tag field in the syslog protocol.
priority	The priority field in the syslog protocol.
facility	The facility field in the syslog protocol.
severity	The severity field in the syslog protocol.
unixtimestamp	The timestamp of the log.
content	The content of the log. If the log fails to be parsed, this field contains the content of the raw log.
_ip_	The IP address of the current host.

3.1.6.4. Customize Logtail plug-ins to process data

If you have complex logs that cannot be parsed in basic modes such as full regex, NGINX, and JSON, you can use Logtail plug-ins to parse the logs. You can configure Logtail plug-ins for one or more processing methods. Then, Logtail executes the processing methods in sequence.

Limits

- Performance limits

  If a plug-in is used to process data, Logtail consumes more resources. Most of these resources are CPU resources. You can modify the Logtail parameter settings based on your business requirements. For more information, see Set Logtail startup parameters.

- Limits on text logs

  Log Service allows you to process text logs in basic modes such as full regex, NGINX, or JSON. Log Service also allows you to use Logtail plug-ins to process text logs. However, Logtail plug-ins have the following limits on text logs:

  - If you enable the plug-in processing feature, some advanced features of the specified mode become unavailable. For example, you cannot configure the filter, upload raw logs, specify the system time zone, drop logs that fail to be parsed, or upload incomplete logs in delimiter mode.
  - Plug-ins use the line mode to process text logs. In this mode, file-level metadata such as __tag__:__path__ and __topic__ is stored in each log. If you use Logtail plug-ins to process data, the following limits apply to tag-related features:
    - You cannot use the contextual query and LiveTail features because these features depend on fields such as __tag__:__path__.
    - The name of the __topic__ field is renamed to __log_topic__.
    - Fields such as __tag__:__path__ no longer have original field indexes. You must configure indexes for these fields.

Usage notes

When you configure data processing methods, you must set the key in the configuration file to processors and set the value to an array of JSON objects. Each object of the array contains the details of a processing method.

Each processing method contains the type and detail fields. The type field specifies the type of the processing method and the detail field contains configuration details.

```
"processors" : [
      {
          "type" : "processor_split_char",
          "detail" : {"SourceKey" : "content",
              "SplitSep" : "|",
              "SplitKeys" : ["method", "type", "ip", "time", "req_id", "size", "detail"]
          }
      },
      {
          "type" : "processor_anchor",
          "detail" : "SourceKey" : "detail",
              "Anchors" : [
                  {
                      "Start" : "appKey=",
                      "Stop" : ",env=",
                      "FieldName" : "appKey",
                      "FieldType" : "string"
                  }
              ]
          }
  ]
```

The following table describes the Logtail plug-ins that are available and the operations that you can perform by using these plug-ins.

Logtail plug-in	Description
processor_regex	You can use the processor_regex plug-in to extract the fields that match a specified regular expression. For more information, see Extract log fields by using a regular expression.
processor_anchor	You can use the processor_anchor plug-in to anchor strings and extract fields based on the start and stop keywords that you specify. For more information, see Extract log fields by using start and stop keywords.
processor_split_char	You can use the processor_split_char plug-in to extract fields based on a specified single-character delimiter. For more information, see Extract log fields by using a single-character delimiter.
processor_split_string	You can use the processor_split_string plug-in to extract fields based on a specified multi-character delimiter. For more information, see Extract log fields by using a multi-character delimiter.
processor_split_key_value	You can use the processor_split_key_value plug-in to extract fields based on key-value pairs. For more information, see Extract log fields by splitting key-value pairs.
processor_add_fields	You can use the processor_add_fields plug-in to add fields to a log. For more information, see Add log fields.
processor_drop	You can use the processor_drop plug-in to drop specified fields. For more information, see Drop log fields.
processor_rename	You can use the processor_rename plug-in to rename specified fields. For more information, see Rename log fields.
processor_packjson	You can use the processor_packjson plug-in to encapsulate one or more fields into a field in the JSON format. For more information, see Encapsulate log fields (JSON).

Logtail plug-in	Description
processor_json	You can use the processor_json plug-in to expand JSON fields. For more information, see Expand JSON fields.
processor_filter_regex	You can use the processor_filter_regex plug-in to filter logs. For more information, see Filter logs by using regular expressions.
processor_gotime	You can use the processor_gotime plug-in to extract time information from a field in a time format that is supported by Golangand, and then configure the time information as the log time. For more information, see Extract log time (Go).
processor_strptime	You can use the processor_strptime plug-in to extract time information from a field in a time format that is supported by strptime, and then configure the time information as the log time. For more information, see Extract log time (strptime).
processor_geoip	You can use the processor_geoip plug-in to convert IP addresses in logs to geographical locations. A geographical location includes the following information: country, province, city, longitude, and latitude. For more information, see Convert an IP address to a geographical location.

You can also create a custom method that includes one or more of the preceding methods. For more information, see Custom methods.

Extract log fields by using a regular expression

You can use a regular expression to extract log fields.

The type of the plug-in is `processor_regex`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_regex.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
Regex	String	Yes	The regular expression. Enclose the fields that you want to extract in parentheses `()`.
Keys	String array	Yes	The array of fields that are extracted, for example, ["ip", "time", "method"].
NoKeyError	Boolean	No	Specifies whether to report an error if a field is not matched. Default value: false. This value indicates that no error is reported if a field is not matched.
NoMatchError	Boolean	No	Specifies whether to report an error if the regular expression does not match the value of a specified field. Default value: false. This value indicates that no error is reported if a field is not matched.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: false. This value indicates that the source field is not retained.

Parameter	Type	Required	Description
FullMatch	Boolean	No	Default value: true. This value indicates that exact match is performed when the regular expression specified in the Regex parameter is used to match field values. If you set the value to false, partial match is performed when the regular expression is used to match field values.

- Configuration example

  The following example shows how to extract the value of the content field. Then, you can set the names of the destination fields to ip, time, method, url, request_time, request_length, status, length, ref_url, and browser.

  - Raw log

    ```
    "content" : "203.0.113.10 - - [10/Aug/2017:14:57:51 +0800] \"POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=<yourSignature> HTTP/1.1\" 0.024 18204 200 37 \"-\" \"aliyun-sdk-java"
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
        "type" : "processor_regex",
        "detail" : {"SourceKey" : "content",
             "Regex" : "([\\d\\.]+) \\S+ \\S+ \\[(\\S+) \\S+\\] \"(\\w+) ([^\\\"]*)\" ([\\d\\.]+) (\\d+) (\\d+) (\\d+|-) \"([^\\\"]*)\" \"([^\\\"]*)\" (\\d+)",
             "Keys"   : ["ip", "time", "method", "url", "request_time", "request_length", "status", "length", "ref_url", "browser"],
             "NoKeyError" : true,
             "NoMatchError" : true,
             "KeepSource" : false
        }
    }
    ```

  - Result

    ```
    "ip" : "203.0.113.10"
    "time" : "10/Aug/2017:14:57:51"
    "method" : "POST"
    "url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=<yourSignature>"
    "request_time" : "0.024"
    "request_length" : "18204"
    "status" : "200"
    "length" : "27"
    "ref_url" : "-"
    "browser" : "aliyun-sdk-java"
    ```

Extract log fields by using start and stop keywords

You can use start and stop keywords to anchor strings and extract log fields.

The type of the plug-in is `processor_anchor`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_anchor.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
Anchors	Anchor array	Yes	The list of the parameters that are set to anchor strings.
NoAnchorError	Boolean	No	Specifies whether to report an error if no keyword is found. Default value: false. This value indicates that no error is reported if no keyword is found.
NoKeyError	Boolean	No	Specifies whether to report an error if a field is not matched. Default value: false. This value indicates that no error is reported if a field is not matched.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: false. This value indicates that the source field is not retained.

The following table describes the parameters of the Anchors parameter.

Parameter	Type	Required	Description
Start	String	Yes	The keyword that anchors the start of a substring in a string. If you do not specify the parameter, the start of the string is matched.
Stop	String	Yes	The keyword that anchors the end of a substring in a string. If you do not specify the parameter, the end of the string is matched.
FieldName	String	Yes	The name of the field that you want to extract.
FieldType	String	Yes	The type of the field that you want to extract. Valid values: string and json.
ExpondJson	Boolean	No	Specifies whether to expand a JSON substring that is anchored. Default value: false. This value indicates that a JSON substring that is anchored is not expanded. This parameter is available only if the value of the FieldType parameter is set to json.
ExpondConnecter	String	No	The character that is used to connect expanded keys. Default value: _.
MaxExpondDepth	Int	No	The maximum depth of JSON expansion. Default value: 0. This value indicates that the depth of JSON expansion is unlimited.

- Configuration example

  The following example shows how to extract the value of the content field. Then, you can set the names of the destination fields to time, val_key1, val_key2, val_key3, value_key4_inner1, and value_key4_inner2.

- Raw log

```
"content" : "time:2017.09.12 20:55:36\tjson:{\"key1\" : \"xx\", \"key2\": false, \"key3\":123.456
, \"key4\" : { \"inner1\" : 1, \"inner2\" : false}}"
```

- Logtail plug-in configurations for data processing

```
{
   "type" : "processor_anchor",
   "detail" : {"SourceKey" : "content",
      "Anchors" : [
          {
              "Start" : "time",
              "Stop" : "\t",
              "FieldName" : "time",
              "FieldType" : "string",
              "ExpondJson" : false
          },
          {
              "Start" : "json:",
              "Stop" : "",
              "FieldName" : "val",
              "FieldType" : "json",
              "ExpondJson" : true
          }
      ]
  }
}
```

- Result

```
"time" : "2017.09.12 20:55:36"
"val_key1" : "xx"
"val_key2" : "false"
"val_key3" : "123.456"
"value_key4_inner1" : "1"
"value_key4_inner2" : "false"
```

Extract log fields by using a single-character delimiter

You can use a specified single-character delimiter to extract fields. This processing method allows you to specify a quote to enclose the delimiter.

The type of the plug-in is `processor_split_char`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_split_char.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
SplitSep	String	Yes	The delimiter. The delimiter must be a single character. You can specify a non-printable character as a single-character delimiter, for example, \u0001.
SplitKeys	String array	Yes	The names of the delimited fields, for example, ["ip","time","method"].

Parameter	Type	Required	Description
QuoteFlag	Boolean	No	Specifies whether to use a quote to enclose the specified delimiter. Default value: false. This value indicates that a quote is not used to enclose the specified delimiter.
Quote	String	No	The quote. The quote must be a single character. You can specify a non-printable character as a quote, for example, \u0001. This parameter is available only if the value of QuoteFlag is set to true.
NoKeyError	Boolean	No	Specifies whether to report an error if a field is not matched. Default value: false. This value indicates that no error is reported if a field is not matched.
NoMatchError	Boolean	No	Specifies whether to report an error if a delimiter is not matched. Default value: false. This value indicates that no error is reported if a delimiter is not matched.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: false. This value indicates that the source field is not retained.

- Configuration example

  The following example shows how to use a vertical bar (|) as a delimiter to extract the value of the content field. Then, you can set the names of the destination fields to ip, time, method, url, request_time, request_length, status, length, ref_url, and browser.

  - Raw log

    ```
    "content" : "203.0.113.10|10/Aug/2017:14:57:51 +0800|POST|PutData?
    Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun%202013%2006%3A53%
    3A30%20GMT&Topic=raw&Signature=<yourSignature>|0.024|18204|200|37|-|
    aliyun-sdk-java"
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
       "type" : "processor_split_char",
       "detail" : {"SourceKey" : "content",
          "SplitSep" : "|",
          "SplitKeys" : ["ip", "time", "method", "url", "request_time", "request_length", "status", "
    length", "ref_url", "browser"]
      }
    }
    ```

- Result

```
"ip" : "203.0.113.10"
"time" : "10/Aug/2017:14:57:51 +0800"
"method" : "POST"
"url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun
%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=<yourSignature>"
"request_time" : "0.024"
"request_length" : "18204"
"status" : "200"
"length" : "27"
"ref_url" : "-"
"browser" : "aliyun-sdk-java"
```

Extract log fields by using a multi-character delimiter

You can use a specified multi-character delimiter to extract fields. You cannot specify a quote to enclose the delimiter.

The type of the plug-in is `processor_split_string` .

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_split_string.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
SplitSep	String	Yes	The delimiter. The delimiter contains multiple characters. You can specify non-printable characters in the delimiter, for example, \u0001\u0002.
SplitKeys	String array	Yes	The names of the delimited fields, for example, ["key1","key2"].
PreserveOthers	Boolean	No	Specifies whether to retain excess fields if the number of fields is greater than the number of fields that are specified by the SplitKeys parameter. Default value: false. This value indicates that excess fields are not retained.
ExpandOthers	Boolean	No	Specifies whether to parse excess fields. Default value: false. This value indicates that excess fields are not parsed.
ExpandKeyPrefix	String	No	The name prefix of excess fields. For example, if you specify expand_ for the parameter, the first two excess fields are named expand_1 and expand_2.
NoKeyError	Boolean	No	Specifies whether to report an error if a field is not matched. Default value: false. This value indicates that no error is reported if a field is not matched.

Parameter	Type	Required	Description
NoMatchError	Boolean	No	Specifies whether to report an error if a delimiter is not matched. Default value: false. This value indicates that no error is reported if a delimiter is not matched.
KeepSource	Boolean	No	Specifies whether to retain the source field. This value indicates that the source field is not retained.

- Configuration example

  The following example shows how to use a delimiter (|#|) to extract the value of the content field. Then, you can set the names of the destination fields to ip, time, method, url, request_time, request_length, status, expand_1, expand_2, and expand_3.

  - Raw log

    ```
    "content" : "203.0.113.10|#|10/Aug/2017:14:57:51 +0800|#|POST|#|PutData?
    Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun%202013%2006%3A53%
    3A30%20GMT&Topic=raw&Signature=<yourSignature>|#|0.024|#|18204|#|200|#|27|#|-|#|
    aliyun-sdk-java"
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
       "type" : "processor_split_string",
       "detail" : {"SourceKey" : "content",
          "SplitSep" : "|#|",
          "SplitKeys" : ["ip", "time", "method", "url", "request_time", "request_length", "status"],
          "PreserveOthers" : true,
          "ExpandOthers" : true,
          "ExpandKeyPrefix" : "expand_"
      }
    }
    ```

  - Result

    ```
    "ip" : "203.0.113.10"
    "time" : "10/Aug/2017:14:57:51 +0800"
    "method" : "POST"
    "url" : "/PutData?Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun
    %202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=<yourSignature>"
    "request_time" : "0.024"
    "request_length" : "18204"
    "status" : "200"
    "expand_1" : "27"
    "expand_2" : "-"
    "expand_3" : "aliyun-sdk-java"
    ```

Extract log fields by splitting key-value pairs

You can split key-value pairs to extract log fields.

The type of the plug-in is `processor_split_char`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_split_key_value.

> **Note** Only Logtail V0.16.26 or later supports the plug-in.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
Delimiter	String	No	The delimiter between key-value pairs. Default value: `\t`.
Separator	String	No	The delimiter that is used to separate the key and the value in a single key-value pair. A colon (:) is used by default.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: true. This value indicates that the source field is retained.
ErrIfSourceKeyNotFound	Boolean	No	Specifies whether to trigger an alert if a field is not matched. Default value: true. This value indicates that an alert is triggered if a field is not matched.
DiscardWhenSeparatorNotFound	Boolean	No	Specifies whether to drop the key-value pair if a field is not matched. Default value: false. This value indicates that the key-value pair is not dropped if a field is not matched.
ErrIfSeparatorNotFound	Boolean	No	Specifies whether to trigger an alert if the delimiter specified by the Separator parameter does not exist. Default value: true. This value indicates that an alert is triggered if the specified delimiter does not exist.

- Configuration example

  The following example shows how to split the key-value pairs in the value of the content field. The delimiter that is used to separate key-value pairs is a tab character (`/t`). The delimiter that is used to separate the key and the value in a single key-value pair is a colon (:).

  - Raw log

    ```
    "content": "class:main\tuserid:123456\tmethod:get\tmessage:\"wrong user\""
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
      "processors":[
        {
          "type":"processor_split_key_value",
          "detail": {
            "SourceKey": "content",
            "Delimiter": "\t",
            "Separator": ":",
            "KeepSource": true
          }
        }
      ]
    }
    ```

- Result

```
"content": "class:main\tuserid:123456\tmethod:get\tmessage:\"wrong user\""
"class": "main"
"userid": "123456"
"method": "get"
"message": "\"wrong user\""
```

Convert an IP address to a geographical location

This processing method converts IP addresses in logs to geographical locations. A geographical location includes the following information: country, province, city, longitude, and latitude.

The type of the plug-in is `processor_geoip`.

> **Note**
> - GeoIP databases are not included in the Logtail installation package. You must download and configure a GeoIP database on the server where Logtail is installed. We recommend that you download a database that provides the city information of an IP address.
> - Make sure that the database format is MMDB.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_geoip.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field that you want to convert.
DBPath	String	Yes	The absolute path of the GeoIP database, for example, */user/data/GeoLite2-City_20180102/GeoLite2-City.mmdb*.
NoKeyError	Boolean	No	Specifies whether to report an error if a field is not matched. Default value: false. This value indicates that no error is reported if a field is not matched.
NoMatchError	Boolean	No	Specifies whether to report an error if an IP address is invalid or is not matched in the database. Default value: false. This value indicates that no error is reported if an IP address is invalid or is not matched in the database.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: true. This value indicates that the source field is retained.
Language	String	No	The language of the GeoIP database. Default value: zh-CN. Make sure that your GeoIP database can be displayed in a language that is suitable for your business.

- Configuration example

  The following example shows how to configure the processing method to convert IP addresses in logs to geographical locations.

- Raw log

```
"source_ip" : "203.0.113.10"
```

- Logtail plug-in configurations for data processing

```
{
   "type": "processor_geoip",
    "detail": {
         "SourceKey": "ip",
         "NoKeyError": true,
         "NoMatchError": true,
         "KeepSource": true,
         "DBPath" : "/user/local/data/GeoLite2-City_20180102/GeoLite2-City.mmdb"
    }
}
```

- Result

```
"source_ip_city_" : "**.**.**.**"
"source_ip_province_" : "Zhejiang"
"source_ip_city_" : "Hangzhou"
"source_ip_province_code_" : "ZJ"
"source_ip_country_code_" : "CN"
"source_ip_longitude_" : "120.********"
"source_ip_latitude_" : "30.********"
```

Filter logs by using regular expressions

This method uses regular expressions to filter logs. You can specify conditions in the `Include` and `Exclude` parameters.

The type of the plug-in is `processor_filter_regex`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_filter_regex.

  > **Note** A log is collected only if the log exactly matches the regular expression that is specified in the Include parameter and does not match the regular expression that is specified in the Exclude parameter.

Parameter	Type	Required	Parameters
Include	JSON object that conatins key-value pairs	No	A map that includes key-value pairs. In each key-value pair, the key specifies a field and the value specifies a regular expression that the value of the same field in each log must match. If the values of all fields in a log match the regular expressions that are specified in the Include parameter, the log is collected.
Exclude	JSON object that conatins key-value pairs	No	A map that includes key-value pairs. In each key-value pair, the key specifies a field and the value specifies a regular expression that the value of the same field in each log must match. If the values of all fields in a log match the regular expressions that are specified in the Exclude parameter, the log is not collected.

- Configuration example

  The following example shows how to use regular expressions to filter logs.

  - Raw logs
    - Log 1

      ```
      "ip" : "203.0.113.10"
      "method" : "POST"
      ...
      "browser" : "aliyun-sdk-java"
      ```

    - Log 2

      ```
      "ip" : "203.0.113.20"
      "method" : "POST"
      ...
      "browser" : "chrome"
      ```

    - Log 3

      ```
      "ip" : "198.51.100.10"
      "method" : "POST"
      ...
      "browser" : "ali-sls-ilogtail"
      ```

  - Logtail plug-in configurations for data processing

    ```
    {
       "type" : "processor_filter_regex",
        "detail" : {
             "Include" : {
                "ip" : "203\\..*",
                "method" : "POST"
             },
             "Exclude" : {
                "browser" : "aliyun.*"
             }
        }
    }
    ```

  - Result

Log	Collected	Reason
Log 1	No	The value of the browser parameter matches the regular expression that is specified in the Exclude parameter.
Log 2	Yes	All the filter conditions are met.
Log 3	No	The value of the ip parameter does not match the regular expression that is specified in the Include parameter.

Add log fields

You can use this method to add multiple fields to a log.

The type of the plug-in is `processor_add_fields`.

- Parameters

The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_add_fields.

> **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
Fields	Map	No	The key-value pairs that you want to add. You can specify multiple key-value pairs in the parameter.
IgnoreIfExist	Boolean	No	Specifies whether to retain key-value pairs that have the same key. Default value: false. This value indicates that a key-value pair is not retained if the key is the same as another specified key.

- Configuration example

  The following example shows how to add the aaa2 and aaa3 fields to a log.

  - Raw log

    ```
    "aaa1":"value1"
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
      "processors":[
        {
          "type":"processor_add_fields",
          "detail": {
            "Fields": {
              "aaa2": "value2",
              "aaa3": "value3"
            }
          }
        }
      ]
    }
    ```

  - Result

    ```
    "aaa1":"value1"
    "aaa2":"value2"
    "aaa3":"value3"
    ```

Drop log fields

You can use this method to drop specified fields from a log.

The type of the plug-in is `processor_drop`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_drop.

  > **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
DropKeys	String array	No	The fields that you want to drop. You can drop one or more fields from a log.

- Configuration example

  The following example shows how to drop the aaa1 and aaa2 fields from a log.

  - Raw log

    ```
    "aaa1":"value1"
    "aaa2":"value2"
    "aaa3":"value3"
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
      "processors":[
        {
          "type":"processor_drop",
          "detail": {
            "DropKeys": ["aaa1","aaa2"]
          }
        }
      ]
    }
    ```

  - Result

    ```
    "aaa3":"value3"
    ```

Extract log time (Go)

You can use this method to extract time information from a specified field, and then convert the time format.

The type of the plug-in is `processor_gotime` .

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_gotime.

  > **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
SourceFormat	String	Yes	The format of the time information in the source field.
SourceLocation	Int	Yes	The source time zone. If you do not specify the parameter, the current time zone of the server where Logtail is installed is used.
DestKey	String	Yes	The name of the destination field.
DestFormat	String	Yes	The format of the time information in the destination field.

Parameter	Type	Required	Description
DestLocation	Int	No	The destination time zone. If you do not specify the parameter, the current time zone of the server where Logtail is installed is used.
SetTime	Boolean	No	Specifies whether to configure the time information as the log time. Default value: true. This value indicates that the time information is configured as the log time.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: true. This value indicates that the source field is retained.
NoKeyError	Boolean	No	Specifies whether to report an error if a field is not matched. Default value: true. This value indicates that an error is reported if the source field is not matched.
AlarmIfFail	Boolean	No	Specifies whether to trigger an alert if the time information fails to be extracted. Default value: true. This value indicates that an alert is triggered if the time information fails to be extracted.

- Configuration example

  In this example, the time information `2006-01-02 15:04:05 (UTC+8)` is extracted from the s_key field, converted to `2006/01/02 15:04:05 (UTC+9)`, and then added to the d_key field.

  - Raw log

    ```
    "s_key":"2019-07-05 19:28:01"
    ```

  - Logtail plug-in configurations for data processing

    ```
    {
      "processors":[
        {
          "type":"processor_gotime",
          "detail": {
            "SourceKey": "s_key",
            "SourceFormat":"2006-01-02 15:04:05",
            "SourceLocation":8,
            "DestKey":"d_key",
            "DestFormat":"2006/01/02 15:04:05",
            "DestLocation":9,
            "SetTime": true,
            "KeepSource": true,
            "NoKeyError": true,
            "AlarmIfFail": true
          }
        }
      ]
    }
    ```

- Result

```
"s_key":"2019-07-05 19:28:01"
"d_key":"2019/07/05 20:28:01"
```

Expand JSON fields

You can use this method to expand a JSON field.

The type of the plug-in is `processor_json`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_json.

  > **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
NoKeyError	Boolean	No	Specifies whether to report an error if the source field is not matched. Default value: true. This value indicates that an error is reported if the source field is not matched.
ExpandDepth	Int	No	The depth of JSON expansion. Default value: 0. This value indicates that the depth of JSON expansion is unlimited. If the value is n, the depth of JSON expansion is n.
ExpandConnector	String	No	The character that is used to connect expanded keys. You can leave this parameter empty. Default value: _.
Prefix	String	No	The prefix that is added to expanded keys. You can leave this parameter empty.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: true. This value indicates that the source field is retained.
UseSourceKeyAsPrefix	Boolean	No	Specifies whether to add the name of the source field as a prefix to all expanded keys. Default value: false. This value indicates that the name of the source field is not added.

- Configuration example

  The following example shows how to expand the JSON field s_key, and then add j and the name of the source field s_key as a prefix to the expanded keys.

  - Raw log

```
"s_key":"{\"k1\":{\"k2\":{\"k3\":{\"k4\":{\"k51\":\"51\",\"k52\":\"52\"},\"k41\":\"41\"}}})"
```

- Logtail plug-in configurations for data processing

```
{
  "processors":[
    {
      "type":"processor_json",
      "detail": {
        "SourceKey": "s_key",
        "NoKeyError":true,
        "ExpandDepth":0,
        "ExpandConnector":"-",
        "Prefix":"j",
        "KeepSource": false,
        "UseSourceKeyAsPrefix": true
      }
    }
  ]
}
```

- Result

```
"s_key":"{\"k1\":{\"k2\":{\"k3\":{\"k4\":{\"k51\":\"51\",\"k52\":\"52\"},\"k41\":\"41\"}}})"
"js_key-k1-k2-k3-k4-k51":"51"
"js_key-k1-k2-k3-k4-k52":"52"
"js_key-k1-k2-k3-k41":"41"
```

Encapsulate log fields (JSON)

You can use this method to encapsulate one or more fields into a field in the JSON format.

The type of the plug-in is `processor_packjson` .

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_packjson.

  > **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
SourceKeys	String array	Yes	The field that you want to encapsulate. The field is in the string array format.
DestKey	String	No	The destination field in the JSON format.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: true. This value indicates that the source field is retained.
AlarmIfIncomplete	Boolean	No	Specifies whether to trigger an alert if the source field does not exist. Default value: true. This value indicates that an alert is triggered if the source field does not exist.

- Configuration example

  The following example shows how to encapsulate the a and b fields into the d_key field.

- Raw log

```
"a":"1"
"b":"2"
```

- Logtail plug-in configurations for data processing

```
{
  "processors":[
    {
      "type":"processor_packjson",
      "detail": {
        "SourceKeys": ["a","b"],
        "DestKey":"d_key",
        "KeepSource":true,
        "AlarmIfEmpty":true
      }
    }
  ]
}
```

- Result

```
"a":"1"
"b":"2"
"d_key":"{\"a\":\"1\",\"b\":\"2\"}"
```

Rename log fields

You can use this method to rename multiple fields.

The type of the plug-in is `processor_rename`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_rename.

  > **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
NoKeyError	Boolean	Yes	Specifies whether to report an error if a field that you want to rename is not matched. Default value: false. This value indicates that no error is reported if a field that you want to rename is not matched.
SourceKeys	String array	Yes	The source fields that you want to rename.
DestKeys	String array	Yes	The fields that are renamed.

- Configuration example

  The following example shows how to rename the aaa1 field to bbb1 and the aaa2 field to bbb2.

  - Raw log

```
"aaa1":"value1"
"aaa2":"value2"
"aaa3":"value3"
```

- Logtail plug-in configurations for data processing

```
{
  "processors":[
    {
      "type":"processor_rename",
      "detail": {
        "SourceKeys": ["aaa1","aaa2"],
        "DestKeys": ["bbb1","bbb2"],
        "NoKeyError": true
      }
    }
  ]
}
```

- Result

```
"bbb1":"value1"
"bbb2":"value2"
"aaa3":"value3"
```

Extract log time (strptime)

You can use this method to extract time information from a field, and then configure the time information as the log time.

The type of the plug-in is `processor_strptime`.

- Parameters

  The following table describes the parameters that you can specify in the detail parameter if you set the type parameter to processor_strptime.

  > **Note** Only Logtail V0.16.28 or later supports the plug-in.

Parameter	Type	Required	Description
SourceKey	String	Yes	The name of the source field.
Format	String	Yes	The format of the time information in the source field.
AdjustUTCOffset	Boolean	No	Specifies whether to modify the time zone. Default value: false. This value indicates that the time zone is not modified.
UTCOffset	Int	No	The offset that is used to modify the time zone. For example, the value 28800 indicates that the time zone is modified to UTC+8.
AlarmIfFail	Boolean	No	Specifies whether to trigger an alert if the time information fails to be extracted. Default value: true. This value indicates that an alert is triggered if the time information fails to be extracted.
KeepSource	Boolean	No	Specifies whether to retain the source field. Default value: true. This value indicates that the source field is retained.

- Configuration examples

The following examples show how to parse the value of the log_time field into the `%Y/%m/%d %H:%M:%S` format. The current time zone of the server where Logtail is installed is used.

- Example 1: The time zone is UTC+8.
  - Raw log

```
"log_time":"2016/01/02 12:59:59"
```

  - Logtail plug-in configurations for data processing

```
{
  "processors":[
    {
      "type":"processor_strptime",
      "detail": {
        "SourceKey": "log_time",
        "Format": "%Y/%m/%d %H:%M:%S"
      }
    }
  ]
}
```

  - Result

```
"log_time":"2016/01/02 12:59:59"
Log.Time = 1451710799
```

- Example 2: The time zone is UTC+7.
  - Raw log

```
"log_time":"2016/01/02 12:59:59"
```

  - Logtail plug-in configurations for data processing

```
{
  "processors":[
    {
      "type":"processor_strptime",
      "detail": {
        "SourceKey": "log_time",
        "Format": "%Y/%m/%d %H:%M:%S",
        "AdjustUTCOffset": true,
        "UTCOffset": 25200
      }
    }
  ]
}
```

  - Result

```
"log_time":"2016/01/02 12:59:59"
Log.Time = 1451714399
```

Custom methods

You can use multiple processing methods to process logs. The following example shows how to use a single-character delimiter to split a log into several fields and then specify anchor points to extract content from the `detail` field.

- Raw log

```
"content" :
"ACCESS|QAS|203.0.113.10|1508729889935|52460dbed4d540b88a973cf5452b1447|1238|appKey=ba,env=pub,requ
estTime=1508729889913,latency=22ms,
request={appKey:ba,optional:{\\domains\\:\\daily\\,\\version\\:\\v2\\},rawQuery:{\\query\\:\\The ro
ute to Location A\\,\\domain\\:\\Navigation\\,\\intent\\:\\navigate\\,\\slots\\:\\to_geo:level3=Loc
ation A\\,\\location\\:\\Location B\\},
requestId:52460dbed4d540b88a973cf5452b1447},
response={answers:[],status:SUCCESS}|"
```

- Logtail plug-in configurations for data processing

```
"processors" : [
      {
          "type" : "processor_split_char",
          "detail" : {"SourceKey" : "content",
              "SplitSep" : "|",
              "SplitKeys" : ["method", "type", "ip", "time", "req_id", "size", "detail"]
          }
      },
      {
          "type" : "processor_anchor",
          "detail" : "SourceKey" : "detail",
              "Anchors" : [
                  {
                          "Start" : "appKey=",
                      "Stop" : ",env=",
                      "FieldName" : "appKey",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",env",
                      "Stop" : ",requestTime=",
                      "FieldName" : "env",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",requestTime=",
                      "Stop" : ",latency",
                      "FieldName" : "requestTime",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",latency=",
                      "Stop" : ",request=",
                      "FieldName" : "latency",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",request=",
                      "Stop" : ",response=",
                      "FieldName" : "request",
                      "FieldType" : "string"
                  },
                  {
                      "Start" : ",response=",
                      "Stop" : "",
                      "FieldName" : "response",
                      "FieldType" : "json"
                  }
              ]
          }
      }
  ]
```

- Result

```
"method" : "ACCESS"
"type" : "QAS"
"ip" : "203.0.113.10"
"time" : "1508729889935"
"req_id" : "52460dbed4d540b88a973cf5452b1447"
"size" : "1238"
"appKey" : "ba"
"env" : "pub"
"requestTime" : "1508729889913"
"latency" : "22ms"
"request" : "{appKey:nui-banma,optional:{\\domains\\:\\daily-faq\\,\\version\\:\\v2\\},rawQuery:{\\
query\\:\\\345\216\273\344\271\220\345\261\261\347\232\204\350\267\257\347\272\277\\,\\domain\\:\\\
345\257\274\350\210\252\\,\\intent\\:\\navigate\\,\\slots\\:\\to_geo:level3=\344\271\220\345\261\26
1\\,\\location\\:\\\345\214\227\344\272\254\\},requestId:52460dbed4d540b88a973cf5452b1447}"
"response_answers" : "[]"
"response_status" : "SUCCESS"
```

3.1.7. Limits

This topic describes the limits of Logtail. These limits apply when you collect files, manage resources, and resolve errors.

Limits on file collection

Item	Description
File encoding	Log files can be encoded in UTF-8 and GBK. To improve processing performance, we recommend that you encode log files in UTF-8. If log files are encoded in other formats, errors such as garbled characters and data loss may occur.
Log file size	Unlimited.
Log file rotation	Supported. Both `.log*` and `.log` are supported for file names.
Log collection behavior when log parsing is blocked	When log parsing is blocked, Logtail keeps the log file descriptor (FD) open. If log file rotation occurs multiple times during the blocking period, Logtail attempts to parse new log files in sequence. If the number of new log files that are not parsed exceeds 20, Logtail does not process the excess log files.
Symbolic link	Monitored directories can be symbolic links.
Size of a single log	The maximum size of a single log is 512 KB. If a regular expression is used to split a multi-line log to match the start part in the first line of the log, the maximum size of each log after splitting is still 512 KB. If the size of a log exceeds 512 KB, the log is forcibly split into multiple parts and collected. For example, if the size of a log is 1,025 KB, the log is split into three parts of the following sizes: 512 KB, 512 KB, and 1 KB. Then, the log parts are collected in sequence.
Regular expression	Perl-based regular expressions can be used.
Multiple Logtail configurations for the same log file	Not supported. We recommend that you collect and store log files to one Logstore, and then configure multiple subscriptions. If this feature is required, configure symbolic links for log files to bypass this limit.
File opening behavior	When Logtail collects data from a log file, Logtail keeps the log file open. If the log file is not updated for more than 5 minutes and log rotation does not occur, Logtail closes the log file.

Item	Description
First log collection behavior	Logtail collects data only from incremental log files. If the size of a log file exceeds 1 MB the first time an update to the log file is detected, Logtail collects data from the last 1 MB. If the log file size does not exceed 1 MB, Logtail collects data from the beginning of the log file. If the log file is not updated after the Logtail configuration is delivered, Logtail does not collect data from the log file.
Non-standard text logs	If a log contains `'\0'` in multiple lines, the log is truncated at the first `'\0'`.

Limits on checkpoints

Item	Description
Checkpoint timeout period	If a log file is not updated for more than 30 days, the checkpoint of the log file is deleted.
Checkpoint storage policy	Checkpoints are saved every 15 minutes and are automatically saved when you exit Logtail.
Checkpoint storage path	By default, checkpoints are stored in the `/tmp/logtail_checkpoint` directory. You can modify the values of the related parameters. For more information, see Set Logtail startup parameters.

Limits on configurations

Item	Description
Configuration update	A custom configuration update requires approximately 30 seconds to take effect.
Dynamic loading of Logtail configurations	Supported. The update of a Logtail configuration does not affect other Logtail configurations.
Number of Logtail configurations	Unlimited. However, we recommend that you create a maximum of 100 Logtail configurations on a server.
Multi-tenant isolation	Logtail configurations for different tenants are isolated.

Limits on resources and performance metrics

Item	Description
Throughput for log processing	The default transmission speed of raw logs is limited to 2 MB/s. Log data is uploaded after it is encoded and compressed. The compression ratio ranges from 5:1 to 10:1. If the transmission speed exceeds the limit, log data may be lost. You can modify the values of the related parameters. For more information, see Set Logtail startup parameters.
Maximum processing speed	Single-core processing speed: The maximum processing speed is 100 MB/s for logs in simple mode, 40 MB/s for logs in delimiter mode, and 30 MB/s for logs in JSON mode. By default, the maximum processing speed is 20 MB/s for logs in full regex mode based on the complexity of regular expressions. If multiple processing threads are enabled, the performance can be improved by 1.5 to 3 times.

Item	Description
Number of monitored directories	Logtail limits the depth of monitored directories to reduce the consumption of your resources. If the upper limit is reached, Logtail stops monitoring additional directories or log files. Logtail can monitor a maximum of 3,000 directories, including subdirectories.
Number of monitored files	By default, you can use a Logtail configuration on each server to monitor a maximum of 10,000 files. By default, a Logtail client on each server can monitor a maximum of 100,000 files. Excessive files are not monitored. If the upper limit is reached, you can perform the following operations: • Improve the depth of the monitored directory in each Logtail configuration. • Increase the value of the mem_usage_limit parameter to raise the threshold of memory resources that are available for Logtail. For more information, see Set Logtail startup parameters. You can raise the threshold to a maximum of 2 GB. This way, the maximum number of files that can be monitored by using each Logtail configuration is increased to 100,000, and the maximum number of files that the Logtail client on each server can monitor is increased to 1,000,000.
Default resources	By default, Logtail occupies a maximum of 40% of the CPU and 256 MB of memory. If logs are generated at a high speed, you can modify the values of the related parameters. For more information, see Set Logtail startup parameters.
Processing policy of threshold-crossing resources	If the resources that are occupied by Logtail exceed the upper limit and this issue lasts for 5 minutes or more, Logtail is forcibly restarted. The restart may cause data loss or duplication.

Limits on error handling

Item	Description
Network error handling	If a network error occurs, Logtail automatically retries and adjusts the retry interval.
Processing policy of threshold-crossing resources	If the data transmission speed exceeds the quota of the Logstore, Logtail restricts the log collection speed and retries the log collection.
Maximum retry period before timeout	If data fails to be transmitted and the issue lasts for more than six consecutive hours, Logtail discards the data.
Status self-check	Logtail restarts if an exception occurs, for example, an application unexpectedly exits or the resource usage exceeds the quota.

Other limits

Item	Description
Log collection latency	A latency of less than 1 second exists between the point in time when a log is written to a disk and the point in time when Logtail collects the log. However, if the log collection speed is restricted, the latency increases.
Log upload policy	Before Logtail uploads logs, Logtail aggregates the logs in the same file. The log upload starts if the number of logs exceeds 2,000, the total size of logs exceeds 2 MB, or the log collection duration exceeds 3 seconds.

3.2. Other collection methods

3.2.1. Use the web tracking feature to collect logs

Log Service provides the web tracking feature that you can use to collect logs from the HTML, HTML5, iOS, and Android platforms. You can also customize dimensions and metrics to collect logs. This topic describes how to use the web tracking feature to collect logs.

Context

You can use the web tracking feature to collect user information from browsers, iOS apps, or Android apps. The information includes:

- Browsers, operating systems, and resolutions that are used by users.
- User browsing behavior, such as the number of clicks and purchases on a website.
- The amount of time that users spend on an app and whether users are active users.

Usage notes

- After you enable the web tracking feature for a Logstore, the write permissions on the Logstore are granted to anonymous users from the Internet. This may generate dirty data.
- The HTTP body of each GET request cannot exceed 16 KB.
- You can use the POST method to call the PutLogs API operation and write a maximum of 3 MB or 4,096 log entries to Log Service.

Step 1: Enable the web tracking feature

You can use the Log Service console or an SDK to enable the web tracking feature.

- Enable the web tracking feature in the Log Service console.
  i. Log on to the Log Service console.
  ii. In the **Projects** section, click the project in which you want to enable the web tracking feature for a Logstore.
  iii. Find the Logstore for which you want to enable the web tracking feature and choose > **Modify**.
  iv. In the upper-right corner of the **Logstore Attributes** page, click **Modify**.
  v. Turn on **WebTracking** and click **Save**.
- Use an SDK to enable the web tracking feature.

  The following script shows how to use Log Service SDK for Java to enable the web tracking feature:

```
import com.aliyun.openservices.log.Client;
import com.aliyun.openservices.log.common.LogStore;
import com.aliyun.openservices.log.exception.LogException;
public class WebTracking {
  static private String accessId = "your accesskey id";
  static private String accessKey = "your accesskey";
  static private String project = "your project";
  static private String host = "log service data address";
  static private String logStore = "your logstore";
  static private Client client = new Client(host, accessId, accessKey);
  public static void main(String[] args) {
      try {
          // Enable the web tracking feature for an existing Logstore.
          LogStore logSt = client.GetLogStore(project, logStore).GetLogStore();
          client.UpdateLogStore(project, new LogStore(logStore, logSt.GetTtl(), logSt.GetShardCount
(), true));
          // Disable the web tracking feature.
          //client.UpdateLogStore(project, new LogStore(logStore, logSt.GetTtl(), logSt.GetShardCou
nt(), false));
          // Create a Logstore for which you want to enable the web tracking feature.
          //client.UpdateLogStore(project, new LogStore(logStore, 1, 1, true));
      }
      catch (LogException e){
          e.printStackTrace();
      }
  }
}
```

Step 2: Collect logs

After you enable the web tracking feature for a Logstore, you can upload logs to a Logstore by using the following methods:

- Use SDK for JavaScript to upload logs.
  i. Install the dependency.

  ```
  npm install --save js-sls-logger
  ```

  ii. Import the application module.

  ```
  import SlsWebLogger from 'js-sls-logger'
  ```

  iii. Set the opts parameter. The following table describes the parameters.

  ```
  const opts = {
    host: 'cn-qingdao-env12-d01.sls-pub.cloud.env12.shuguang.com',
    project: 'my_project_name',
    logstore: 'my_logstore_name',
    time: 10,
    count: 10,
  }
  ```

Parameter	Required	Description
host	Yes	The endpoint of the region where Log Service resides. In this example, the endpoint of the China (Hangzhou) region is used. Replace the value of the parameter with the actual endpoint. For more information, see the **Obtain an endpoint** topic in *Log Service Developer Guide*.

Parameter	Required	Description
project	Yes	The name of the project.
logstore	Yes	The name of the Logstore.
time	No	The time interval at which logs are sent. Default value: 10. Unit: seconds.
count	No	The number of logs that are sent. Default value: 10.

iv. Create SlsWebLogger.

```
const logger = new SlsWebLogger(opts)
```

v. Upload logs.

```
logger.send({
    customer: 'zhangsan',
    product: 'iphone 12',
    price: 7998
})
```

- Use the GET method to upload logs.

  Run the following command to upload logs. Replace the values of the parameters based on your business requirements. The following table describes the parameters.

```
curl --request GET 'http://${project}.${host}/logstores/${logstore}/track?APIVersion=0.6.0&key1=val1&key2=val2'
```

Parameter	Required	Description
${project}	Yes	The name of the project.
${host}	Yes	The endpoint of the region where Log Service resides. For more information, see the **Obtain an endpoint** topic in *Log Service Developer Guide*.
${logstore}	Yes	The name of the Logstore.
APIVersion=0.6.0	Yes	A reserved parameter.
__topic__=yourtopic	No	The topic of the log that you want to upload.
key1=val1&key2=val2	Yes	The key-value pairs that you want to upload to Log Service. Make sure that the data size is less than 16 KB.

- Use HTML <img> tags to upload logs.

```
<img src='http://${project}.${host}/logstores/${logstore}/track.gif?APIVersion=0.6.0&key1=val1&key2=val2'/>
<img src='http://${project}.${host}/logstores/${logstore}/track_ua.gif?APIVersion=0.6.0&key1=val1&key2=val2'/>
```

  The *track_ua.gif* file contains custom parameters that you want to upload to Log Service. If you use this method to upload logs, Log Service records the custom parameters and the User-Agent and Referer HTTP headers as log fields.

> **Note** To collect the Referer HTTPS header, make sure that the URL in the preceding <img> tag uses the HTTPS protocol.

- Use the POST method to upload logs.

  You can send an HTTP POST request to upload a large amount of data. For more information, see the "PutWebtacking" topic of **API Reference** in *Log Service Developer Guide*.

3.2.2. Use SDKs to collect logs

3.2.2.1. Producer Library

The Aliyun LOG Java Producer supports Java applications that run in big data processing scenarios with high concurrency. The library is easy to use and highly customizable.

For more information about the related GitHub project, visit Aliyun LOG Java Producer.

3.2.2.2. Log4j Appender

This topic describes Alibaba Cloud Log4j Appender.

Log4j is an open source project of Apache. Log4j allows you to specify the output destination and format of logs. You can also specify the severity level of each log for fine-grained control on log generation. Log4j consists of the following three components:

- Loggers

  The severity levels of logs are classified into ERROR, WARN, INFO, and DEBUG in descending order.

- Appenders

  An appender specifies that logs are sent to the Log Service console or files.

- Layouts

  A layout specifies the output format of logs.

You can use Alibaba Cloud Log4j Appender to send logs to Log Service. For more information about Alibaba Cloud Log4j Appender, visit Log4j Appender.

3.2.2.3. Logback Appender

This topic describes how to write logs to Log Service by using Aliyun Log Logback Appender.

Logback is an open source project that is developed by the founder of Log4j. Logback allows you to write logs to multiple destinations. These destinations include the Log Service console, files, graphical user interface (GUI) components, socket servers, NT kernel loggers, and UNIX syslog daemons. You can specify the output format of each log. You can also specify the severity level of each log for fine-grained control on log generation.

The following example shows the format of a log that is written to Log Service by using Aliyun Log Logback Appender:

```
level: ERROR
location: com.aliyun.openservices.log.logback.example.LogbackAppenderExample.main(LogbackAppenderExamp
le.java:18)
message: error log
throwable: java.lang.RuntimeException: xxx
thread: main
time: 2018-01-02T03:15+0000
log: 2018-01-02 11:15:29,682 ERROR [main] com.aliyun.openservices.log.logback.example.LogbackAppenderE
xample: error log
__source__: xxx
__topic__: yyy
```

For more information about Aliyun Log Logback Appender, see Logback Appender.

3.2.2.4. Golang Producer Library

The Aliyun LOG Go Producer Library supports Go applications that run in big data processing scenarios with high concurrency. The library is easy to use and highly customizable. You can use the library to create producers that allow you to resend failed logs. Before Go applications send log data to Log Service, you can use these producers to compress the log data. This improves write performance.

For more information about the related GitHub project, visit Aliyun Log Go Producer.

3.2.2.5. Python logging

This topic describes how to use the Python logging module to collect log data.

Configurations

For more information about the configurations that are related to the Python logging module, see Logging configuration.

The Python logging module allows you to use code or a configuration file to configure logging. The following example shows how to use the `logging.conf` configuration file to configure logging.

```
[loggers]
keys=root,sls
[handlers]
keys=consoleHandler, slsHandler
[formatters]
keys=simpleFormatter, rawFormatter
[logger_root]
level=DEBUG
handlers=consoleHandler
[logger_sls]
level=INFO
handlers=consoleHandler, slsHandler
qualname=sls
propagate=0
[handler_consoleHandler]
class=StreamHandler
level=DEBUG
formatter=simpleFormatter
args=(sys.stdout,)
[handler_slsHandler]
class=aliyun.log.QueuedLogHandler
level=INFO
formatter=rawFormatter
args=(os.environ.get('ALIYUN_LOG_SAMPLE_ENDPOINT', ''), os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSID', '
'), os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSKEY', ''), os.environ.get('ALIYUN_LOG_SAMPLE_TMP_PROJECT',
''), "logstore")
[formatter_simpleFormatter]
format=%(asctime)s - %(name)s - %(levelname)s - %(message)s
[formatter_rawFormatter]
format=%(message)s
```

Two handlers named `root` and `sls` are created. The `sls` handler is an object of the `aliyun.log.QueuedLogHandler` class. The following script shows the parameters that you can specify for the sls handler. For more information, see Parameters.

```
args=(os.environ.get('ALIYUN_LOG_SAMPLE_ENDPOINT', ''), os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSID', '
'), os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSKEY', ''), os.environ.get('ALIYUN_LOG_SAMPLE_TMP_PROJECT',
''), "logstore")
```

Note In this case, the `os.environ` function is used to obtain configurations from environment variables. You can also specify values for these parameters based on your business requirements.

Upload logs

If you want to upload logs to Log Service, you can use the configuration file.

```
import logging
import logging.config
# Configurations
logging.config.fileConfig('logging.conf')
logger = logging.getLogger('sls')
# Use the logger
logger.info("test1")
try:
    1/0
except ZeroDivisionError as ex:
    logger.exception(ex)
```

Then, logs are automatically uploaded to Log Service. If you want to use the query and analysis feature, you must enable the indexing feature for the related Logstore.

Configure indexes for a Logstore

Enable the indexing feature for the Logstore that receives logs and configure indexes for specific fields. We recommend that you use the Log Service command-line interface (CLI) to configure indexes. For more information, see python_logging_handler_index.json.

```
aliyunlog log update_index --project_name="project1" --logstore_name="logstore1" --index_detail="file:
///Users/user1/loghandler_index.json"
```

Specify log fields that you want to collect

The following table describes the log fields that you can collect.

Field	Description
message	The content of a log.
record_name	The name of a handler. In the preceding example, `sls` is used.
level	The severity level of a log, such as INFO and ERROR.
file_path	The full path of a configuration file.
func_name	The name of a function.
line_no	The number of a log line.
module	The name of a module where the function resides.
thread_id	The ID of the thread that runs the function.
thread_name	The name of the thread that runs the function.
process_id	The ID of the process that runs the function.
process_name	The name of the process that runs the function.

You can specify log fields that you want to collect based on the `fields` parameter of a class. For more information, see aliyun.log.LogFields.

The following example shows how to modify the preceding configuration file and collect several fields, such as module and func_name.

```
[handler_slsHandler]
class=aliyun.log.QueuedLogHandler
level=INFO
formatter=rawFormatter
args=('cn-beijing.log.aliyuncs.com', 'ak_id', 'ak_key', 'project1', "logstore1", 'mytopic', ['level',
'func_name', 'module', 'line_no']  )
```

Note

- The message field is collected regardless of your configurations.
- If you want to add a prefix and suffix to the names of these fields, use the `buildin_fields_prefix` and `buildin_fields_suffix` parameters. Example: `__level__`.

Use a JSON text to configure logging

If you want to create flexible logging configurations, you can use a JSON text.

```
#encoding: utf8
import logging, logging.config, os
# Configurations
conf = {'version': 1,
        'formatters': {'rawformatter': {'class': 'logging.Formatter',
                                        'format': '%(message)s'}
                       },
        'handlers': {'sls_handler': {'()':
                                     'aliyun.log.QueuedLogHandler',
                                     'level': 'INFO',
                                     'formatter': 'rawformatter',
                                     # custom args:
                                     'end_point': os.environ.get('ALIYUN_LOG_SAMPLE_ENDPOINT', ''),
                                     'access_key_id': os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSID', '')
,
                                     'access_key': os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSKEY', ''),
                                     'project': 'project1',
                                     'log_store': "logstore1"
                                     }
                     },
        'loggers': {'sls': {'handlers': ['sls_handler', ],
                                    'level': 'INFO',
                                    'propagate': False}
                    }
        }
logging.config.dictConfig(conf)
# Use the logger
logger = logging.getLogger('sls')
logger.info("Hello world")
```

Note If you want to instantiate an object of the `aliyun.log.QueuedLogHandler` class, pass named parameters to the constructor. For more information, see Parameters.

3.2.3. Collect common logs

3.2.3.1. Collect Log4j logs

Log Service allows you to use LogHub Log4j Appender or Logtail to collect Log4j logs.

Log format

Log4j is an open source project of Apache. Log4j allows you to specify the output destination and format of logs. You can also specify the severity level of logs. The severity levels of logs are classified into ERROR, WARN, INFO, and DEBUG in descending order. The output destination specifies whether logs are sent to the console or files. The output format specifies the format of logs. The following example shows the default configurations of Log4j:

```
<Configuration status="WARN">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss:SSS zzz} [%t] %-5level %logger{36} - %msg%n"/>
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="com.foo.Bar" level="trace">
      <AppenderRef ref="Console"/>
    </Logger>
    <Root level="error">
      <AppenderRef ref="Console"/>
    </Root>
  </Loggers>
</Configuration>
```

The following example shows a sample log:

```
2013-12-25 19:57:06,954 [10.10.10.10] WARN impl.PermanentTairDaoImpl - Fail to Read Permanent Tair,key
:e:470217319319741_1,result:com.example.tair.Result@172e3ebc[rc=code=-1, msg=connection error or timeo
ut,value=,flag=0]
```

Regular expression that matches each IP address that indicates the start of a line:

```
\d+-\d+-\d+\s.*
```

Regular expression that is used to extract log information:

```
(\d+-\d+-\d+\s\d+:\d+:\d+,\d+)\s\[([^\]]*)\]\s(\S+)\s+(\S+)\s-\s(.*)
```

Time conversion format:

```
%Y-%m-%d %H:%M:%S
```

The following table lists the extraction results of the sample log.

Key	Value
time	2013-12-25 19:57:06,954
ip	203.0.113.2
level	WARN
class	impl.PermanentTairDaoImpl

Key	Value
message	Fail to Read Permanent Tair,key:e:470217319319741_1,result:com.example.tair.Result@172e3ebc[rc=code=-1, msg=connection error or timeout,value=,flag=0]

Use LogHub Log4j Appender to collect Log4j logs

For more information about how to collect Log4j logs by using LogHub Log4j Appender, see Log4j Appender.

Use Logtail to collect Log4j logs

The procedure when you use Logtail to collect Log4j logs is similar to that when you use Logtail to collect Python logs. Configure Logtail based on the actual network deployment and your business requirements. For more information, see Python logs.

3.2.3.2. Collect Python logs

Log Service allows you to use the Python logging module to collect Python logs. This topic describes how to use Logtail to collect Python logs.

Context

The Python logging module provides a general logging system, which can be used by third-party modules or applications. The logging module defines multiple log severity levels and logging methods. The logging module consists of the following components: loggers, handlers, filters, and formatters.

To collect Python logs, we recommend that you use logging handlers. For more information, see the following topics:

- Use logging handlers to automatically upload Python logs
- Use logging handlers to automatically upload and parse logs in the key-value format
- Use logging handlers to automatically parse logs in the JSON format

Log format

Formatters specify the output format of logs. The fields in the configurations of a formatter are in the %(key)s format.

```
import logging
import logging.handlers
LOG_FILE = 'tst.log'
handler = logging.handlers.RotatingFileHandler(LOG_FILE, maxBytes = 1024*1024, backupCount = 5) # Crea
te a handler object.
fmt = '%(asctime)s - %(filename)s:%(lineno)s - %(levelno)s %(levelname)s %(pathname)s %(module)s %(fun
cName)s %(created)f %(thread)d %(threadName)s %(process)d %(name)s - %(message)s' // Define the output
format of logs.
formatter = logging.Formatter(fmt)   # Create a formatter object.
handler.setFormatter(formatter)      # Add the formatter to the handler.
logger = logging.getLogger('tst')    # Retrieve a logger that is named tst.
logger.addHandler(handler)           # Add the handler to the logger.
logger.setLevel(logging.DEBUG)
logger.info('first info message')
logger.debug('first debug message')
```

The following table describes the fields in the formatter configurations.

Field	Description
%(name)s	The name of the logger that generates a log.
%(levelno)s	The severity level of a log in the numeric format.
%(levelname)s	The severity level of a log in the text format. Valid values: DEBUG, INFO, WARNING, ERROR, and CRITICAL.
%(pathname)s	The full path name of the source file where the logging call is initiated.
%(filename)s	The name of the source file.
%(module)s	The name of the module where the logging call is initiated.
%(funcName)s	The name of the function from which the logging call is initiated.
%(lineno)d	The line number in the source file where the logging call is initiated.
%(created)f	The time when a log is created. The value is a UNIX timestamp. It is the number of seconds that have elapsed since 00:00:00 UTC, Thursday, January 1, 1970.
%(relativeCreated)d	The difference between the time when a log is created and the time when the logging module is loaded. Unit: milliseconds.
%(asctime)s	The time when a log is created. Example: 2003-07-08 16:49:45,896. The digits after the comma (,) indicate the millisecond portion of the time.
%(msecs)d	The millisecond portion of the time when a log is created.
%(thread)d	The ID of the thread.
%(threadName)s	The name of the thread.
%(process)d	The ID of the process.
%(message)s	The log content.

The following example shows sample logs:

```
2015-03-04 23:21:59,682 - log_test.py:16 - tst - first info message
2015-03-04 23:21:59,682 - log_test.py:17 - tst - first debug message
```

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, select **RegEx - Text Log**.
3. Select a destination project and Logstore, and then click **Next**.

   You can also click **Create Now** to create a project and a Logstore.

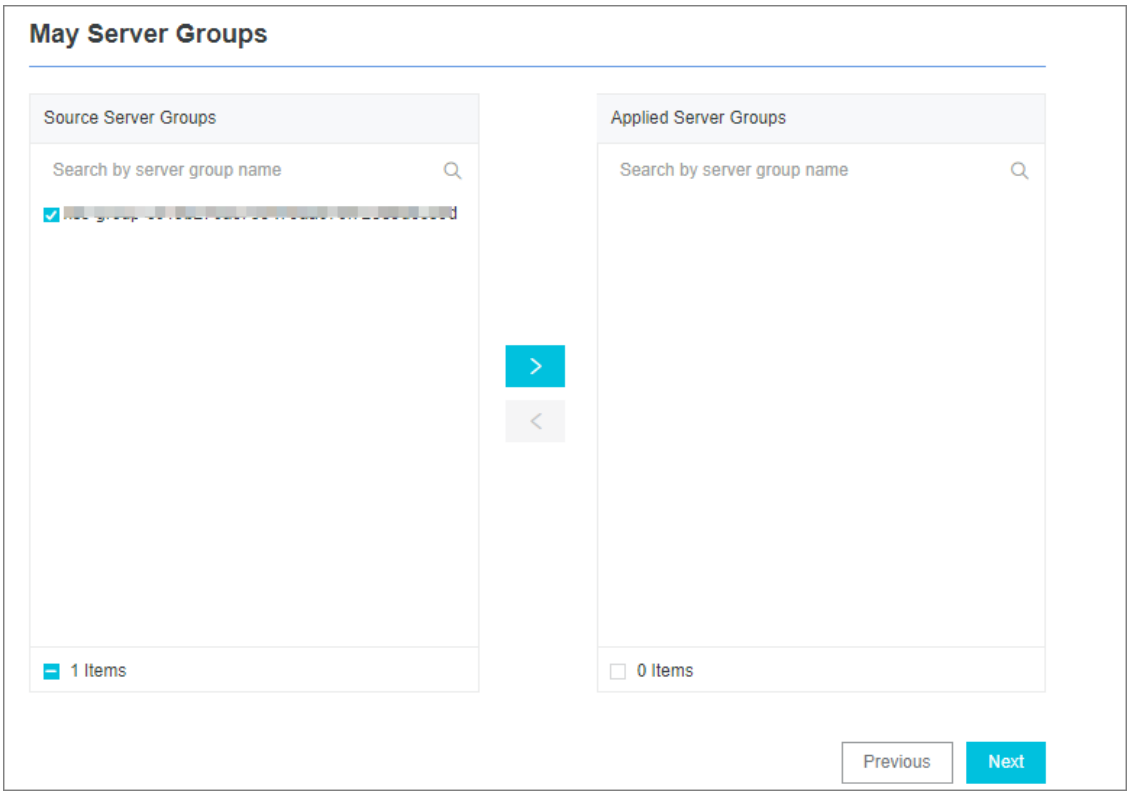
   If you enter the configuration procedure by clicking the plus sign (+) next to **Data Import** under a Logstore on the Logstores tab, the system skips this step.
4. Create a machine group and click **Next**.

   Before you can create a machine group, you must install Logtail.

   Install Logtail as prompted. For more information, see Install Logtail in Linux or Install Logtail in Windows.

After you install Logtail, click **Complete Installation** to create a machine group. If a machine group is created, click **Use Existing Machine Groups** to select the machine group.

5. Select a machine group, move the machine group from **Source Machine Groups** to **Applied Server Groups**, and then click **Next**.



> **Notice** If you want to apply a machine group immediately after it is created, the heartbeat status of the machine group may be **FAIL**. This issue occurs because no servers in the machine group are connected to Log Service. In this case, you can click **Automatic Retry**.

6. Configure the parameters in the Logtail Config step.
   i. Configure the **Config Name** and **Log Path** parameters and set the Mode parameter to **Full Regex Mode**.
   ii. Turn on **Singleline**.
   iii. Enter a sample log in the **Log Sample** field.
   iv. Turn on **Extract Field**.
   v. Specify a regular expression in the **RegEx** field.
   - Automatically generate a regular expression.

     In the **Log Sample** field, select the content that you want to extract and click **Generate Regular Expression**. A regular expression is automatically generated.
   - Manually enter a regular expression

     Click **Manual**. In the RegEx field, enter a regular expression. Then, click **Validate** to check whether the regular expression can be used to parse logs or extract content from logs.

vi. Verify the result in the **Extracted Content** field.

View the extraction results of log fields and specify keys for the extracted fields.

Specify an informative name for each log field in the extraction results. For example, you can use time as the name for a time field. If you do not use the system time, you must specify the name of a time field in the Value field and time in the Key field.

7. (Optional)Specify **Advanced Options** and click **Next**.

Set the parameters in the Advanced Options section based on your business requirements. We recommend that you do not modify the settings. The following table describes the parameters in the Advanced Options section.

Parameter	Description
Enable Plug-in Processing	Specifies whether to enable plug-in processing. If you turn on this switch, you can use the plug-in of Logtail to process text logs. **Note** If you turn on **Enable Plug-in Processing**, specific parameters such as Upload Raw Log, Timezone, Drop Failed to Parse Logs, Filter Configuration, and Incomplete Entry Upload (Delimiter mode) become unavailable.
Upload Raw Log	If you turn on **Upload Raw Log**, each raw log is uploaded to Log Service as a value of the __raw__ field together with the parsed log.
Topic Generation Mode	The topic generation mode. ◦ **Null - Do not generate topic**: This mode is selected by default. In this mode, the topic field is set to an empty string. You do not need to enter a topic to query logs. ◦ **Machine Group Topic Attributes**: This mode is used to differentiate logs that are generated by different servers. ◦ **File Path RegEx**: In this mode, you must specify a regular expression in the **Custom RegEx** field. The part of a log path that matches the regular expression is used as the topic. This mode is used to differentiate logs that are generated by different users or instances.
Custom RegEx	If you set the Topic Generation Mode parameter to **File Path RegEx**, you must enter a custom regular expression.
Log File Encoding	The encoding format of log files. Valid values: ◦ utf8: UTF-8 encoding format ◦ gbk: GBK encoding format
Timezone	The time zone where logs are collected. Valid values: ◦ System Timezone: This option is selected by default. It indicates that the time zone where logs are collected is the same as the time zone to which the server belongs. ◦ Custom: If you select this value, you must select a time zone.

Parameter	Description
Timeout	The timeout period of log files. If a log file is not updated within the specified period, Logtail reckons the file to be timed out. Valid values: ○ Never: All log files are continuously monitored and never time out. ○ 30 Minute Timeout: If a log file is not updated within 30 minutes, Logtail considers the log file to be timed out and no longer monitors the file. If you select **30 Minute Timeout**, you must specify the **Maximum Timeout Directory Depth** parameter. Valid values: 1 to 3.
Filter Configuration	Only logs that **meet all filter conditions** are collected. Examples: ○ Collect logs that meet specified conditions: If you set **Key** to **level** and **Regex** to **WARNING|ERROR**, only WARNING-level and ERROR-level logs are collected. ○ Filter out logs that do not meet specified conditions. ■ If you set **Key** to **level** and **Regex** to **^(?!.*(INFO|DEBUG)).***, INFO-level or DEBUG-level logs are not collected. ■ If you set **Key** to **url** and **Regex** to **.*^(?!.*(healthcheck)).***, logs whose URLs contain healthcheck are not collected. For example, the log is not collected if the value of the Key field is url and the value of the Value field is /inner/healthcheck/jiankong.html.

8. Configure indexes in the Configure Query and Analysis step. Click **Next**.

   By default, Log Service enables Full Text Index to query and analyze logs. You can manually or automatically configure Field Search. For more information, see Enable the index feature and configure indexes for a Logstore.

   > **Note**
   > ○ To query and analyze logs, you must enable Full Text Index or Field Search. If you enable both Full Text Index and Field Search, the settings of Field Search prevail.
   > ○ If the data type of the index is long or double, the Case-Sensitive switch and the Delimiter field are unavailable.

After you complete the settings, you can start to collect Python logs.

3.2.3.3. Collect Node.js logs

Node.js logs are displayed in the Log Service console by default. This affects your data collection and troubleshooting efficiency. Log4js is a tool used to manage Node.js logs. You can use Log4js to send Node.js logs to files and customize the log format. Log4js allows you to collect and consolidate data in an efficient manner.

The following code shows how to configure Log4js to send logs to a file:

```
var log4js = require('log4js');
log4js.configure({
  appenders: [
    {
      type: 'file', // Output to a file
      filename: 'logs/access.log',
      maxLogSize: 1024,
      backups:3,
      category: 'normal'
    }
  ]
});
var logger = log4js.getLogger('normal');
logger.setLevel('INFO');
logger.info("this is a info msg");
logger.error("this is a err msg");
```

Log format

After you use Log4js to write logs to text files, the logs are displayed in the following format:

```
[2016-02-24 17:42:38.946] [INFO] normal - this is a info msg
[2016-02-24 17:42:38.951] [ERROR] normal - this is a err msg
```

Log4js classifies log severities into the following six levels in ascending order: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.

Use Logtail to collect Node.js logs

The procedure when you configure Logtail to collect Node.js logs is similar to that when you configure Logtail to collect Python logs. For more information, see Python logs. Set related parameters based on the actual network deployment and your business requirements.

The regular expression that is automatically generated is based on the sample log and may not apply to other logs. Therefore, you must modify the regular expression based on your business requirements before you use it. You can use the following sample Node.js logs to configure regular expressions for your logs.

Sample Node.js logs and regular expressions:

- Example 1
  - Sample log

    ```
    [2016-02-24 17:42:38.946] [INFO] normal - this is a info msg
    ```

  - Regular expression

    ```
    \[([^]]+)]\s\[([^\]]+)]\s(\w+)\s-(.*)
    ```

  - Extracted fields

    `time`, `level`, `loggerName`, and `message`

- Example 2
  - Sample log

    ```
    [2016-01-31 12:02:25.844] [INFO] access - 42.120.73.203 - - "GET /user/projects/ali_sls_log?ignor
    eError=true HTTP/1.1" 304 - "http://
    aliyun.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3) AppleWebKit/537.36 (KHTML, like Gec
    ko) Chrome/48.0.2564.97 Safari/537.36"
    ```

- Regular expression

```
\[([^]]+)]\s\[(\w+)]\s(\w+)\s-\s(\S+)\s-\s-\s"([^"]+)"\s(\d+)[^"]+("[^"]+)"\s"([^"]+).*
```

- Extracted fields

`time`, `level`, `loggerName`, `ip`, `request`, `status`, `referer`, and `user_agent`

3.2.3.4. Collect WordPress logs

This topic describes the format of WordPress logs and extraction results of a sample log.

Log format

Sample log:

```
172.64.0.2 - - [07/Jan/2016:21:06:39 +0800] "GET /wp-admin/js/password-strength-meter.min.js?ver=4.4 H
TTP/1.0" 200 776 "http://wordpress.c4a1a0aecdb1943169555231dcc4adfb7.cn-hangzhou.alicontainer.com/wp-a
dmin/install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/47.0.2526.106 Safari/537.36"
```

Configure Logtail to collect WordPress logs

If you use Logtail to collect WordPress logs, you must configure the following settings:

- Regular expression that matches each IP address that indicates the start of a line

```
\d+\.\d+\.\d+\.\d+\s-\s.*
```

- Regular expression that is used to extract log information

```
(\S+) - - \[([^\]]*)] "(\S+) ([^"]+)" (\S+) (\S+) "([^"]+)" "([^"]+)"
```

- Time conversion format

```
%d/%b/%Y:%H:%M:%S
```

- The following table lists the extraction results of the sample log.

Key	Value
ip	10.10.10.1
time	07/Jan/2016:21:06:39 +0800
method	GET
url	/wp-admin/js/password-strength-meter.min.js?ver=4.4 HTTP/1.0
status	200
length	776
ref	http://wordpress.c4a1a0aecdb1943169555231dcc4adfb7.cn-hangzhou.alicontainer.com/wp-admin/install.php
user-agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36

3.2.3.5. Collect Unity3D logs

This topic describes how to use the web tracking feature of Log Service to collect Unity3D logs.

Context

Unity3D is a cross-platform game engine that is developed by Unity Technologies. You can use the engine to create 3D video games, VR buildings, real-time 3D animation, and other interactive content.

In this example, Unity Debug.Log is used to describe how to collect Unity3D logs.

Procedure

1. Enable the web tracking feature.

   For more information, see WebTracking.

2. Create a Unity3D logging handler.

   In the Unity editor, create a C# file named *LogOutputHandler.cs*, add the following code to the file, and then modify the following variables:

   - project: the name of the Log Service project.
   - logstore: the name of the Logstore.
   - serviceAddr: the endpoint of the Log Service project. For more information, see **Obtain an endpoint** in *Log Service Developer Guide*.

```
using UnityEngine;
using System.Collections;
public class LogOutputHandler : MonoBehaviour
{
    //Register the HandleLog function on scene start to fire on debug.log events
    public void OnEnable()
    {
        Application.logMessageReceived += HandleLog;
    }
    //Remove callback when object goes out of scope
    public void OnDisable()
    {
        Application.logMessageReceived -= HandleLog;
    }
    string project = "your project name";
    string logstore = "your logstore name";
    string serviceAddr = "http address of your log service project";
    //Capture debug.log output, send logs to Loggly
    public void HandleLog(string logString, string stackTrace, LogType type)
    {
        string parameters = "";
        parameters += "Level=" + WWW.EscapeURL(type.ToString());
        parameters += "&";
        parameters += "Message=" + WWW.EscapeURL(logString);
        parameters += "&";
        parameters += "Stack_Trace=" + WWW.EscapeURL(stackTrace);
        parameters += "&";
        //Add any User, Game, or Device MetaData that would be useful to finding issues later
        parameters += "Device_Model=" + WWW.EscapeURL(SystemInfo.deviceModel);
        string url = "http://" + project + "." + serviceAddr + "/logstores/" + logstore + "/track?
APIVersion=0.6.0&" + parameters;
        StartCoroutine(SendData(url));
    }
    public IEnumerator SendData(string url)
    {
        WWW sendLog = new WWW(url);
        yield return sendLog;
    }
}
```

You can use the preceding code to asynchronously send logs to Log Service. You can also specify other fields in the code to collect the fields.

3. Generate Unity3D logs.

   Create a file named *LogglyTest.cs* and add the following code to the file:

```
using UnityEngine;
using System.Collections;
public class LogglyTest : MonoBehaviour {
    void Start () {
        Debug.Log ("Hello world");
    }
}
```

4. View logs in the Log Service console.

   After you run the Unity3D application, logs are generated and sent to Log Service. You can view the logs in the Log Service console.

4.Query and analysis

4.1. Log search overview

Log Service allows you to search for 1 billion to hundreds of billions of rows of log data within seconds. This topic describes the syntax and limits of the log search feature and provides examples.

Syntax

Each query statement consists of a search statement and an analytic statement. The search statement and the analytic statement are separated by a vertical bar (|). For more information about the query statement, see Search syntax.

> Note

- A search statement can be executed alone. However, an analytic statement must be executed together with a search statement. The log analysis feature is based on search results or all data in a Logstore.
- If you need to search for tens of billions of rows of data, you can repeatedly execute a search statement up to 10 times to obtain the complete result.
- Syntax

```
Search statement|Analytic statement
```

Statement	Description
Search statement	A search statement specifies one or more search conditions and returns the logs that meet the specified conditions. A search statement can be a keyword, a value, a value range, a space character, or an asterisk (*). If you specify a space character or an asterisk (*) as the search statement, no conditions are specified and all logs are returned. For more information, see Search syntax.
Analytic statement	An analytic statement is used to aggregate or analyze all log data or the log data that meets the specified search conditions in a Logstore. For more information, see Log analysis overview.

- Example

```
* | SELECT status, count(*) AS PV GROUP BY status
```

Limits

- Each project supports a maximum of 1,000 concurrent search statements at the same time.

  For example, 1,000 users can concurrently search for data in all Logstores of a project at the same time.
- You can specify a maximum of 30 keywords for each search statement.
- The maximum size of a field value is 10 KB. If the size of a field value exceeds 10 KB, the excess content is not queried.
- The returned logs are displayed on multiple pages. Each page displays a maximum of 100 search results.
- Log Service performs the DOM operation only on the first 10,000 characters of a log.
- If you perform a fuzzy search, Log Service searches for 100 words that meet the specified conditions. Logs that contain one or more of the 100 words and meet the search conditions are returned.

Search methods

> **Notice** Before you search for logs, you must make sure that logs are collected and indexes are configured for the fields. Indexes are used in a storage structure to sort one or more columns of log data. For more information, see Enable the indexing feature and configure indexes for a Logstore.

- Use the Log Service console

  Log on to the Log Service console. On the Search & Analysis page of a Logstore, specify a time range and execute a search statement. For more information, see Query logs and Search syntax.

- Call API operations

  Call the GetLogs and GetHistograms operations to search for log data. For more information, see the GetLogs and GetHistograms topics of API Reference in *Developer Guide*.

4.2. Log analysis overview

Log Service provides the log analysis feature. This feature allows you to search for log data and use SQL functions to analyze the data. This topic describes the syntax and limits of the analytic statements. This topic also provides the SQL functions that you can call when you use the log analysis feature.

> **Note** If you want to use the log analysis feature, you must turn on **Enable Analytics** when you configure indexes for log fields. For more information, see Enable the indexing feature and configure indexes for a Logstore. If you turn on **Enable Analytics**, you can analyze log data within seconds without additional costs.

Syntax

Each query statement consists of a search statement and an analytic statement. The search statement and the analytic statement are separated by a vertical bar (|). You can execute a search statement alone. However, you must execute an analytic statement together with a search statement. You can use the log analysis feature to analyze data that meets specified search conditions in a Logstore. You can also use the feature to analyze all data in a Logstore.

> **Note**

- You do not need to specify a FROM or WHERE clause in an analytic statement. By default, all data of the current Logstore is analyzed.
- You do not need to add a semicolon (;) at the end of an analytic statement to end the statement.
- Analytic statements are case-insensitive.
- Syntax

```
Search statement|Analytic statement
```

Statement	Description
Search statement	A search statement specifies one or more search conditions. A search statement can be a keyword, a value, a value range, a space character, or an asterisk (*). If you specify a space character or an asterisk (*) as the search statement, no conditions are specified and all logs are returned. For more information, see Search syntax.
Analytic statement	An analytic statement is used to aggregate or analyze all log data or the log data that meets the specified search conditions in a Logstore.

- Example

```
* | SELECT status, count(*) AS PV GROUP BY status
```

Limits

- Each project supports a maximum of 15 concurrent analytic statements at the same time.

  For example, 15 users can concurrently execute analytic statements in all Logstores of a project at the same time.
- You can analyze only the data that is written to Log Service after the log analysis feature is enabled.
- By default, an analytic statement returns a maximum of 100 rows of data.

  If you want to view more data, use a LIMIT clause. For more information, see LIMIT syntax.
- The maximum size of a field value is 16 KB. If the size of a field value exceeds 16 KB, the excess content is not analyzed.
- The maximum timeout period for an analytic statement is 55 seconds.
- Each shard supports only 1 GB of data for an analytic statement.
- The value of a double-type field can contain a maximum of 52 digits after the decimal point.

  If the number of digits after the decimal point is greater than 52, the accuracy of the field value is compromised.

SQL functions and syntax

This section lists the SQL functions and syntax that Log Service supports.

- The following aggregate functions are available for SELECT statements:
  - General aggregate functions
  - Security check functions
  - Map functions
  - Approximate functions
  - Mathematical statistics functions
  - Mathematical calculation functions
  - String functions
  - Date and time functions
  - URL functions
  - Regular expression functions
  - JSON functions
  - Type conversion functions
  - IP functions
  - Array functions
  - Binary string functions
  - Bitwise operations
  - Interval-valued comparison and periodicity-valued comparison functions
  - Comparison functions and operators
  - Lambda functions
  - Logical functions
  - Geospatial functions
  - Geography functions
  - Machine learning syntax and functions
- GROUP BY syntax

- Window functions
- HAVING syntax
- ORDER BY syntax
- LIMIT syntax
- Syntax for CASE statements and if() functions
- UNNEST function
- Field aliases
- Nested subqueries

4.3. Configure indexes

Indexes are used in a storage structure to sort one or more columns of log data. You can query and analyze log data only after you configure indexes. Query and analysis results vary based on index configurations. Therefore, you must configure indexes based on your business requirements.

Prerequisites

Logs are collected. For more information, see Data collection.

Index types

The following table describes the index types that are supported by Log Service.

Index type	Description
Full-text index	Log Service splits an entire log into multiple words based on specified delimiters to create indexes. In a search statement, the field names (keys) and field values (values) are both plain text. For example, the search statement `error` returns the logs that contain the keyword `error`.
Field index	After you configure field indexes, you can specify field names and field values in the key:value format to search for logs. For example, the search statement `level:error` returns the logs in which the value of the `level` field contains `error`. If you want to use the analysis feature, you must configure field indexes and turn on Enable Analytics for the required fields. The analysis feature does not generate index traffic or occupy storage space.

Notice

- The indexing feature is applicable only to the log data that is written to the current Logstore after you configure indexes.
- If you configure both full-text indexes and field indexes, the configurations of the field indexes take precedence.

Configure full-text indexes

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to query and analyze logs.
3. Choose **Log Storage > Logstores**. On the Logstores tab, click the Logstore in which logs are stored.
4. On the Search & Analysis page of the Logstore, choose **Index Attributes > Attributes**.
   If the indexing feature is not enabled, click **Enable**.
5. Configure indexes in the Search & Analysis panel.

Parameter	Description
LogReduce	If you turn on **LogReduce**, Log Service automatically clusters text logs that have the same pattern during log collection. This way, you can obtain the overall information of the logs. For more information, see LogReduce.
Full Text Index	If you turn on **Full Text Index**, the full-text indexing feature is enabled.
Case Sensitive	Specifies whether searches are case-sensitive. ◦ If you turn on **Case Sensitive**, searches are case-sensitive. For example, if a log contains `internalError`, you can search for the log only by using the keyword `internalError`. ◦ If you turn off **Case Sensitive**, searches are not case-sensitive. For example, if a log contains `internalError`, you can search for the log by using the keyword `INTERNALERROR` or `internalerror`.
Include Chinese	Specifies whether to distinguish between Chinese content and English content in searches. ◦ If you turn on **Include Chinese** and a log contains Chinese characters, the Chinese content is split based on the Chinese grammar. The English content is split based on specified delimiters. **Notice** When the Chinese content is split, the data write speed is reduced. Proceed with caution. ◦ If you turn off **Include Chinese**, all content is split based on specified delimiters.
Delimiter	The delimiters that are used to split the content of a log into multiple words. The following delimiters are supported: `, '";=()[]{}?@&<>/:\n\t\r`. `\n` indicates a line feed, `\t` indicates a tab character, and `\r` indicates a carriage return. For example, the content of a log is `/url/pic/abc.gif`. ◦ If you do not specify a delimiter, the log is processed as a single word `/url/pic/abc.gif`. You can search for the log only by using the keyword `/url/pic/abc.gif`. You can also perform a fuzzy search by using the keyword `/url/pic/*`. ◦ If you set the delimiter to a forward slash (/), the content of the log is split into the following three words: `url`, `pic`, and `abc.gif`. You can search for the log by using the keyword `url`, `abc.gif`, or `/url/pic/abc.gif`. You can also perform a fuzzy search by using the keyword `pi*`. ◦ If you set the delimiter to a forward slash (/) and a period (.), the content of the log is split into the following four words: `url`, `pic`, `abc`, and `gif`.

6. Click **OK**.

   The index configurations take effect within 1 minute.

Configure field indexes

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to query and analyze logs.
3. Choose **Log Storage > Logstores**. On the Logstores tab, click the Logstore in which logs are stored.
4. On the Search & Analysis page of the Logstore, choose **Index Attributes > Attributes**.

   If the indexing feature is not enabled, click **Enable**.

5. Configure indexes in the **Search & Analysis** panel.

Parameter	Description
Key Name	The name of the log field. Example: client_ip. **Note** ◦ When you configure an index for a tag field, you must specify the value of the **Key Name** parameter in the __tag__:KEY format. For example, you can set this parameter to __tag__:__receive_time__. Different tag fields are supported. For example, a tag field can indicate a public IP address or a UNIX timestamp. ◦ When you configure an index for a tag field, you must set the **Type** parameter for each tag field to **text**. Numeric data types are not supported.
Type	The data type of the log field value. Valid values: text, long, double, and json. For more information, see Data types. **Note** If a field is of the long or double type, you cannot set the **Case Sensitive**, **Include Chinese**, or **Delimiter** parameter.
Alias	The alias of the field. Example: ip. An alias is used only in analytic statements. You must use the original field name in search statements. For more information, see Field aliases.
Case Sensitive	Specifies whether searches are case-sensitive. ◦ If you turn on **Case Sensitive**, searches are case-sensitive. For example, if a log contains `internalError`, you can search for the log only by using the keyword `internalError`. ◦ If you turn off **Case Sensitive**, searches are not case-sensitive. For example, if a log contains `internalError`, you can search for the log by using the keyword `INTERNALERROR` or `internalerror`.
Delimiter	The delimiters that are used to split the content of a log into multiple words. The following delimiters are supported: `, '";=()[]{}?@&<>/:\n\t\r`. `\n` indicates a line feed, `\t` indicates a tab character, and `\r` indicates a carriage return. For example, the content of a log is `/url/pic/abc.gif`.
Include Chinese	Specifies whether to distinguish between Chinese content and English content in searches. ◦ If you turn on **Include Chinese** and a log contains Chinese characters, the Chinese content is split based on the Chinese grammar. The English content is split based on specified delimiters. **Notice** When the Chinese content is split, the data write speed is reduced. Proceed with caution. ◦ If you turn off **Include Chinese**, all content is split based on specified delimiters.
Enable Analytics	To use the analysis feature, you must turn on **Enable Analytics**.

Parameter	Description

6. Click **OK**.

   The index configurations take effect within 1 minute.

4.4. Query and analyze logs

After you enable the indexing feature and configure indexes for a Logstore, you can query and analyze the logs that are stored in the Logstore in real time.

Prerequisites

- Logs are collected and stored in a Logstore. For more information, see Data collection.
- The indexing feature is enabled and indexes are configured. For more information, see Enable the indexing feature and configure indexes for a Logstore.

Query and analyze logs

1. Log on to the Log Service console.
2. In the Projects section, click the project to which the Logstore belongs.
3. Choose **Log Storage > Logstores**. On the Logstores tab, click the Logstore where logs are stored.
4. Enter a query statement in the search box.

   A query statement consists of a search statement and an analytic statement in the Search statement|Analytic statement format. For more information, see Search syntax and SQL syntax and functions.
5. Click **15 Minutes(Relative)** to specify a time range.

   You can select a relative time or a time frame. You can also specify a custom time range.

   > **Note** The query results may contain logs that are generated 1 minute earlier or later than the specified time range.
6. Click **Search & Analyze** to view the query and analysis results.

Manage query and analysis results

You can view the query and analysis results in a log distribution histogram, on the Raw Logs tab, or in a chart that is displayed on the Graph tab. You can also configure alerts and saved searches.

> **Note** By default, only 100 rows of data are returned after you execute a query statement. You can use a LIMIT clause to change the number of returned rows. For more information, see LIMIT syntax.

- Log distribution histogram

  The log distribution histogram displays the distribution of query and analysis results in different time ranges.

  - If you move the pointer over a green rectangle, you can view the time range that is represented by the rectangle and the number of logs that are obtained within the time range.
  - If you click the green rectangle, you can view a more fine-grained log distribution. You can also view the query and analysis results on the **Raw Logs** tab.
- Raw Logs tab

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

- Quick analysis: You can use this feature to analyze the distribution of a specific field within a specific period of time. For more information, see Quick analysis.
- Contextual query: If you click the icon of a log and select **Context View**, you can view the context of the log. For more information, see Contextual query.

  > **Note** The contextual query feature supports only the log data that is collected by Logtail.

- LiveTail: If you click the icon of a log on the **Raw Log** tab, you can monitor logs in real time and extract important information. For more information, see LiveTail.

  > **Note** LiveTail can monitor and extract only the log data that is collected by Logtail.

- Log download: To download logs, click the download icon, select a method, and then click **OK**. For more information, see Export logs.
- Column settings: You can click the icon and select **Column Settings** to specify the columns that you want to display in the table. The column names are field names, and the column content is used as field values.

  > **Note** To view the log content on the tab, select **Content**.

- JSON configurations: You can click the icon and select **JSON Configurations** to specify the levels of JSON data.
- Tag configurations: On the **Raw Data** tab, you can click the icon and select **Tag Configurations** to hide fields that are less important.

- Charts

  If you turn on Enable Analytics when you configure indexes for fields and use query statements to query logs, you can view the analysis results on the **Graph** tab.

  - Log Service provides multiple chart types, such as tables, line charts, and bar charts. You can select a chart type to display analysis results. For more information, see Chart overview.
  - Log Service allows you to create dashboards to perform real-time data analysis. You can click **Add to New Dashboard** to save query statements as charts to a dashboard. For more information, see Dashboard overview.
  - Drill-down analysis allows you to view more details of analysis results. You can configure the drill-down parameters and add a chart to the dashboard. Then, you can click the values in the chart to view the analysis results in multiple dimensions. For more information, see Drill-down analysis.

- LogReduce tab

  On the **LogReduce** tab, you can click **Enable LogReduce** to cluster similar logs. For more information, see LogReduce.

- Alerts

  On the Search & Analysis page, click **Save as Alert** to create an alert monitoring rule for query results. For more information, see Alert overview.

- Saved searches

  On the Search & Analysis page, you can click **Save Search** to save a query statement as a saved search. For more information, see Saved search.

4.5. Download logs

This topic describes how to download logs from Log Service to an on-premises host.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project in which you want to download logs.
3. Click the icon next to the name of the Logstore whose logs you want to download and select **Search & Analysis**.
4. In the upper-right corner of the **Raw Logs** tab, click the icon.
5. In the **Log Download** dialog box, select a method that you can use to download logs and click **OK**.
   - **Download Log in Current Page**: Download the logs that are displayed on the current page to a file in the comma-separated values (CSV) format.
   - **Download All Logs Using Command Line Tool**: Download all logs as prompted.

4.6. Index data type

4.6.1. Overview

When you configure indexes, you can set the data type of a field to text, long, double, or JSON. This topic describes the index data types that are supported by Log Service.

Data types

The following table describes the supported data types.

Query type	Index data type	Description	Example
Basic query	text	The text type. You can use keywords and fuzzy matches to query logs.	`uri:"login*" and method:"post"`
	long	The numeric type. You can specify numeric ranges to query indexes of this type.	`status in [200, 500]`
	double	The floating-point type.	`price>28.95`
Combined query	JSON	Indicates that the index is a JSON field that supports nested queries. By default, the data type of the field is text. You can configure indexes of the text, long, and double types for the b elements at layer a in the a.b path format.	`level0.key>29.95 and level0.key2:"action"`
	text	Creates indexes for all fields in a log except the time field. The data type of the indexes is text.	`error and "login fail"`

4.6.2. Text type

This topic describes how to query text data.

Usage notes

Similar to search engines, Log Service queries text data based on terms. Therefore, you must set the Delimiter and Case Sensitive parameters when you configure indexes.

- Case Sensitive switch

  You can specify whether searches are case-sensitive. For example, you want to query a log entry that contains `internalError`.

  - If you turn off Case Sensitive, searches are case-insensitive, and you can find the log entry by using the `INTERNALERROR` or `internalerror` keyword.
  - If you turn on Case Sensitive, searches are case-sensitive, and you can find the log entry only by using the `internalError` keyword.

- Delimiter parameter

  You can use delimiters to split the content of a log entry into multiple words. For example, you want to query a log entry that contains the following content:

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

  - If you do not specify a delimiter, the entire string is processed as a single word in the `/url/pic/abc.gif` format. In this case, you can find the log entry by using the entire string as a keyword for exact match or by using the `/url/pic/*` keyword for fuzzy match.
  - If you set the delimiter to a forward slash (/), the content is divided into the following three words: `url`, `pic`, and `abc.gif`. You can find the log entry by using one of the three words. You can also use part of each word to search for the log entry in fuzzy match mode.

    For example, you can find the log entry by using the `url`, `abc.gif`, or `pi*` keyword. You can also find the log entry by using the `/url/pic/abc.gif` keyword. The /url/pic/abc.gif keyword is split into the following search conditions: `url and pic and abc.gif`.
  - If you set the delimiter to a forward slash (/) and a period (.), the content is split into the following four words: `url`, `pic`, `abc`, and `gif`.

  > **Note** You can specify appropriate delimiters to extend query ranges.

- Full Text Index switch

  By default, after you turn on Full Text Index, the data type of all fields, except the time field, is set to text. You do not need to specify keys. For example, you want to query a log entry that consists of the following four fields:

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

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

Note

- Prefixes are not required for full-text indexes. If you use error as a keyword, the level and message field values that contain error match the keyword.
- You must specify delimiters for full-text indexes. For example, if you specify a comma (,) as a delimiter, the `status:200` string is processed as a single word. If you specify a colon (:) as a delimiter, the string is split into the following two words: `status` and `200`.
- Numbers are processed as text data. For example, you can find the log entry by using the keyword 200. The time field is not processed as text data.
- If the query statement is a key, for example, `status`, the log entry is matched.

4.6.3. Numeric type

When you configure indexes, you can set the data type of a field to a numeric type. Then, you can query the value of the field by value range.

Usage notes

You can query the value of a field by using a numeric range only after you set the data type of the field to long or double.

- If the value of a log field is an integer, we recommend that you set the data type of the field to long when you configure indexes.
- If the value of a log field is a floating-point number, we recommend that you set the data type of the field to double when you configure indexes.

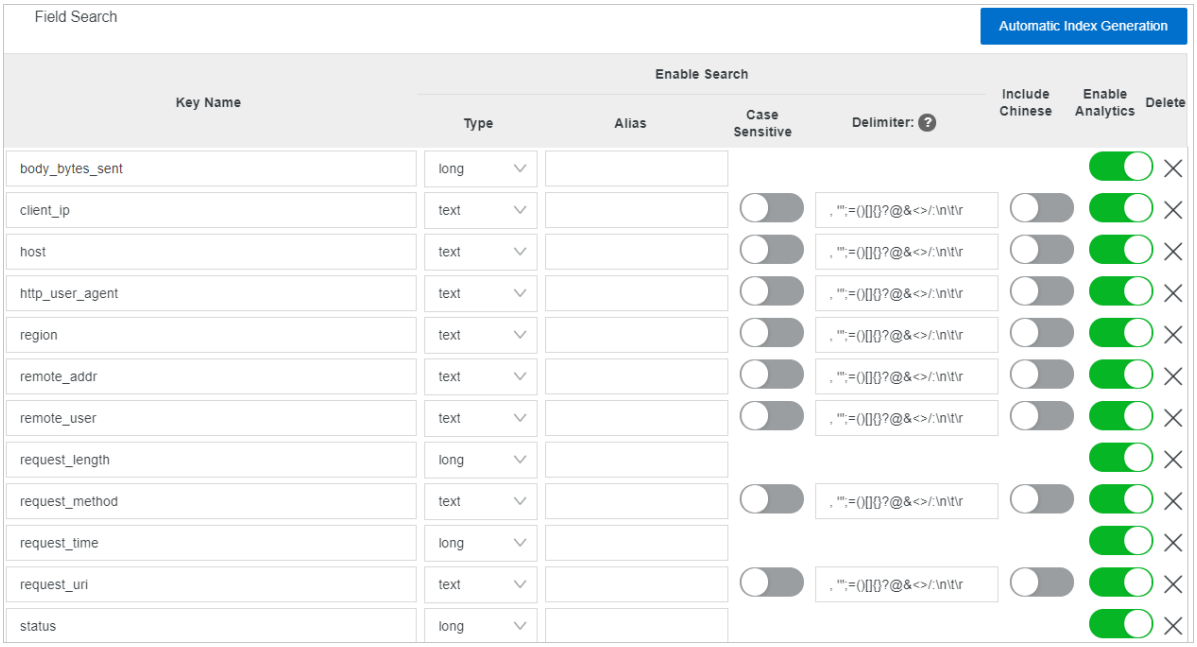
Notice

- If you set the data type of a field to long but the value of the field is a floating-point number, you cannot query the value of the field.
- If you set the data type of a field to long or double but the value of the field is a string, you cannot query the value of the field.
- If you set the data type of a field to long or double, you cannot use asterisks (*) or question marks (?) to query the value of the field in fuzzy match mode.
- If the value of a field is an invalid numeric value, you can query data by using the **not key > -1000000** search statement. The not key > -1000000 search statement returns the log entries in which a field value is an invalid numeric value. -100000 can be replaced by a valid value that is less than or equal to the smallest valid value of the field in your log entries.

Sample search statements

- Sample log entry

- Index configurations



- Query statements
  - To query the log entries in which the request duration is greater than 60 seconds, execute the following search statement:

    ```
    request_time > 60
    ```

  - To query the log entries in which the request duration is greater than or equal to 60 seconds and less than 200 seconds, execute one of the following search statements:

    ```
    request_time in [60 200)
    ```

    ```
    request_time >= 60 and request_time < 200
    ```

  - To query the log entries in which the response status code is 200, execute the following search statement:

    ```
    status = 200
    ```

4.6.4. JSON type

If the value of a field is in the JSON format, you can set the data type of the field to JSON when you configure indexes. This topic describes how to set the data type of a field to JSON and provides some examples.

Usage notes

- You can set the data type of a field in JSON objects to long, double, or text based on the field value, and turn on Enable Analytics to enable the analysis feature. After you turn on Enable Analytics, Log Service allows you to query and analyze fields in JSON objects.
- For partially valid JSON-formatted data, only the valid parts can be parsed in Log Service.

  The following example shows an incomplete JSON log entry. Log Service can parse the conctent.remote_addr, content.request.request_length, and content.request.request_method fields.

```
content: {
     remote_addr:"192.0.2.0"
     request: {
             request_length:"73"
             request_method:"GE
```

> **Notice**
>
> - Log Service allows you to configure indexes for leaf nodes in JSON objects. However, you cannot configure indexes for child nodes that contain leaf nodes.
> - You cannot configure indexes for fields whose values are JSON arrays or configure indexes for the fields in a JSON array.
> - If the value of a field is of the Boolean type, you can set the data type of the field to text when you configure indexes.
> - The format of a query statement in Log Service is `Search statement | Analytic statement`. In an analytic statement, you must enclose a field name by using double quotation marks ("") and enclose a string by using single quotation marks ('').

Examples

The following table lists the keys included in the sample log entry. The data type of the `message` field is JSON.

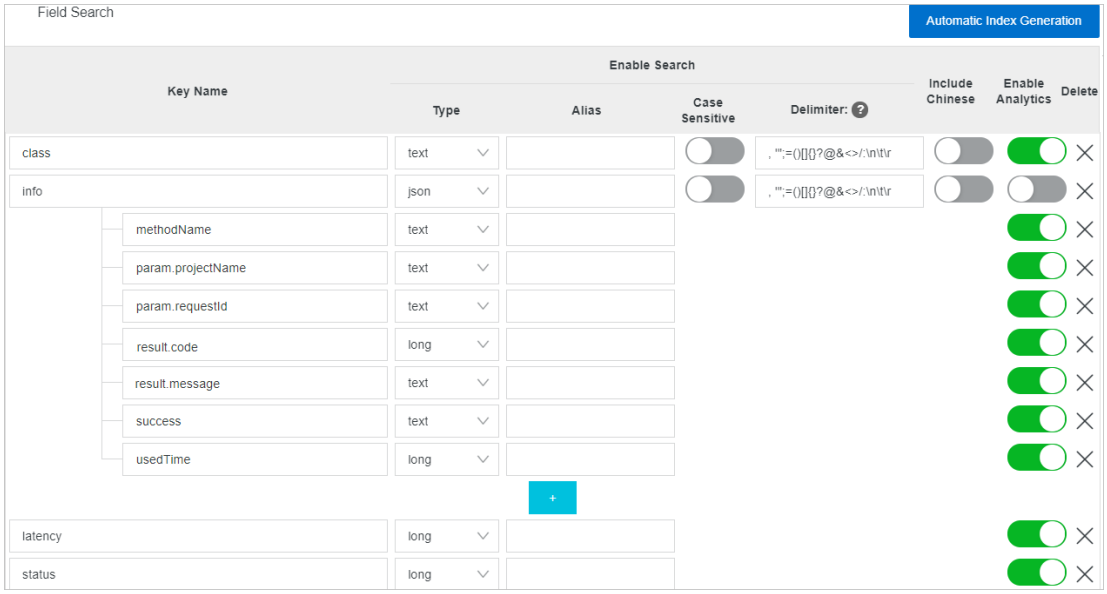
Serial number	Key	Type
0	time	N/A
1	class	text
2	status	long
3	latency	double
4	message	json

Sample log entry:

```
0. time:2018-01-01 12:00:00
1. class:central-log
2. status:200
3. latency:68.75
4. message:
  {
      "methodName": "getProjectInfo",
      "success": true,
      "remoteAddress": "203.0.113.10:11111",
      "usedTime": 48,
      "param": {
              "projectName": "ali-log-test-project",
              "requestId": "d3f0c96a-51b0-4166-a850-f4175dde7323"
      },
      "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 an example on how to configure indexes.

Index configurations



The following settings are configured in the preceding figure:

- ① specifies that Log Service can query data of the string and Boolean types in JSON fields.
- ② specifies that Log Service can query data of the long type.
- ③ enables SQL analysis for specified fields.
- Query log data of the string and Boolean types.

> **Note**
> - You do not need to configure JSON fields.
> - JSON maps and arrays are automatically expanded and can contain multiple layers. You must separate multiple layers with periods (.).

Query statement:

```
message.traceInfo.requestId : 92.137_1518139699935_5599
message.param.projectName : ali-log-test-project
message.success : true
message.result.data.ProjectStatus : Normal
```

- Query log data of the double and long types.

  > **Note** You must configure each JSON field. A JSON field cannot be contained in an array.

  Query statement:

  ```
  message.usedTime > 40
  ```

- Use SQL statements to analyze fields.

  > **Note**
  > - You must configure each JSON field. A JSON field cannot be contained in an array.
  > - You must enclose a field name by using double quotation marks ("") or specify an alias for the field.

  Query statement:

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

4.7. Query syntax and functions

4.7.1. Search syntax

This topic describes how to use the search syntax that is provided by Log Service to specify search conditions. You can efficiently query logs based on the search syntax.

Search types

A search statement specifies one or more search conditions and returns the logs that meet the specified conditions. Searches are classified by indexing method into full-text searches and field-specific searches, or classified by precision into exact searches and fuzzy searches.

> **Note**

- If you configure both full-text indexes and field indexes, the configurations of the field indexes take precedence.
- Before you can specify a numeric range to query logs based on a field, you must set the data type of the field to double or long. If you do not set the data type of a field to double or long, or the syntax of the numeric range is invalid, Log Service performs a full-text search and the search result that is returned may be different from the expected result. For example, if you execute the `owner_id>100` search statement and the data type of the owner_id field is not double or long, logs that contain owner_id, > (non-delimiter), and 100 are returned.
- If you change the data type of a field from text to double or long, you can use only the equal sign (=) to query the logs that are collected before the change.

- Full-text searches and field-specific searches

Search type	Description	Example
Full-text search	After you configure full-text indexes, Log Service splits a log into multiple words by using the delimiters that you specify. You can specify keywords and rules in a search statement to query logs. The keywords can be field names or field values.	`PUT and cn-shanghai` : returns the logs that contain the keywords PUT and cn-shanghai.
Field-specific search	After you configure field indexes, you can query logs. To query logs, specify field names and field values in the key:value format. You can perform basic searches or combined searches based on the data types of the fields in the field indexes. For more information, see Data types.	`request_time>60 and request_method:Ge*` : returns the logs in which the value of the request_time field is greater than 60 and the value of the request_method field starts with Ge.

- Exact searches and fuzzy searches

Search type	Description	Example
Exact search	Complete strings are used for queries.	◦ `host:www.yl.mock.com` : returns the logs in which the value of the host field is www.yl.mock.com. ◦ `PUT` : returns the logs that contain the keyword PUT.

Search type	Description	Example
Fuzzy search	You can add an asterisk (*) or a question mark (?) as a wildcard in the middle or at the end of a keyword to perform a fuzzy search. Each keyword must be 1 to 64 characters in length. If a keyword contains a wildcard, Log Service searches all logs and obtains up to 100 strings that match the keyword. Then, Log Service returns the logs that contain one or more of these strings. The more accurate a keyword is, the more accurate the search results are. **Note** ◦ A keyword cannot start with an asterisk (*) or a question mark (?). ◦ The long and double data types do not support asterisks (*) or question marks (?) in fuzzy searches. You can specify a numeric range when you perform a fuzzy search. Example: status in [200 299]. A fuzzy search is performed based on samples by using the following mechanism: ◦ If you enable the field indexing feature and specify a field to query logs, Log Service randomly obtains samples from the indexed data of the field and returns part of the search results. ◦ If you enable the full-text indexing feature and do not specify a field to query logs, Log Service randomly obtains samples from the full-text indexed data and returns part of the search results.	◦ `addr*`: searches for 100 strings that start with addr from logs, and returns the logs that contain one or more of these strings. ◦ `host:www.yl*`: searches for 100 strings that start with www.yl from the value of the host field. Then, Log Service returns the logs that contain one or more of these strings.

Operators

The following table describes the operators that are supported by search statements.

Note

- The in operator is case-sensitive. Other operators are not case-sensitive.
- Log Service supports the following operators: **sort**, **asc**, **desc**, **group by**, **avg**, **sum**, **min**, **max**, and **limit**. If you want to use the preceding operators as keywords, you must enclose the operators in double quotation marks ("").
- The following list shows the priorities of the operators in descending order:
  i. Colons (:)
  ii. Double quotation marks ("")
  iii. Parentheses ()
  iv. and

v. not

vi. or

Operator	Description
and	The and operator. Example: `request_method:GET and status:200`. If no syntax keyword exists among multiple keywords, the keywords are joined by using the and operator by default. For example, `GET 200 cn-shanghai` is equivalent to `GET and 200 and cn-shanghai`.
or	The or operator. Example: `request_method:GET or status:200`.
not	The not operator. Examples: `request_method:GET not status:200` and `not status:200`.
( )	This operator is used to increase the priority of the search conditions that are enclosed in parentheses (). Example: `(request_method:GET or request_method:POST) and status:200`.
:	This operator is used for field-specific searches based on the key:value format. Example: `request_method:GET`. If a field name or field value contains reserved characters such as spaces and colons (:), enclose the field name or field value in double quotation marks (""). Example: `"file info":apsara`.
""	This operator is used to enclose a syntax keyword. If a syntax keyword is enclosed in double quotation marks (""), the keyword is converted to an ordinary character. For example, `"and"` returns the logs that contain and. In this case, and is not an operator. In a field-specific search, the strings that are enclosed in double quotation marks ("") are considered as a whole string.
\	The escape character. This character is used to escape double quotation marks (""). Double quotation marks ("") can indicate themselves only after they are escaped. For example, if the content of a log is `instance_id:nginx"01"`, you can execute the `instance_id:nginx\"01\"` statement to search for the log.
*	The wildcard character. This character is used to match zero, one, or multiple characters. Example: `host:www.yl.mo*k.com`. **Note** Log Service searches all logs and obtains up to 100 strings that meet the specified conditions. Then, Log Service returns the logs that contain one or more of the 100 strings and meet the search conditions.
?	The wildcard character. This character is used to match a single character. Example: `host:www.yl.mo?k.com`.
>	This operator is used to query the logs in which the value of a specified field is greater than a specified numeric value. Example: `request_time>100`.
>=	This operator is used to query the logs in which the value of a specified field is greater than or equal to a specified numeric value. Example: `request_time>=100`.
<	This operator is used to query the logs in which the value of a specified field is smaller than a specified numeric value. Example: `request_time<100`.

Operator	Description
<=	This operator is used to query the logs in which the value of a specified field is smaller than or equal to a specified numeric value. Example: `request_time<=100`.
=	This operator is used to query the logs in which the value of a specified field is equal to a specified numeric value. Equal signs (=) and colons (:) have the same effect on fields of the double or long data type. For example, `request_time=100` is equivalent to `request_time:100`.
in	This operator is used to query the logs in which the value of a specified field is within a specified numeric range. Brackets [] indicate a closed interval, and parentheses () indicate an open interval. A space character is used to separate two numbers in a numeric range. Examples: `request_time in [100 200]` and `request_time in (100 200]`. **Note** The characters of in must be in lowercase.
__source__	This operator is used to query the logs of a specified log source. Wildcard characters are supported. Example: `__source__:192.0.2.*`. **Notice** The __source__ field is a reserved field in Log Service. This field can be abbreviated to source. If you configure a custom field in the source format, the custom field conflicts with the reserved source field in Log Service. If you want to search for the custom field, you must use Source or SOURCE in a search statement.
__tag__	This operator is used to query logs based on metadata. Example: `__tag__:__receive_time__:1609837139`.
__topic__	This operator is used to query the logs of a specified log topic. Example: `__topic__:nginx_access_log`.

Examples of search statements

Expected search result	Search statement
Logs that contain successful GET requests (status codes: 200 to 299)	`request_method:GET and status in [200 299]`
Logs that contain GET requests that are not sent from the China (Shanghai) region	`request_method:GET not region:cn-shanghai`
Logs that contain GET requests or POST requests	`request_method:GET or request_method:POST`
Logs that do not contain GET requests	`not request_method:GET`
Logs that contain successful GET requests or successful POST requests	`(request_method:GET or request_method:POST) and status in [200 299]`

Expected search result	Search statement
Logs that contain failed GET requests or failed POST requests	`(request_method:GET or request_method:POST) not status in [200 299]`
Logs that contain successful GET requests (status codes: 200 to 299) and in which the request duration is less than 60 seconds	`request_method:GET and status in [200 299] not request_time>=60`
Logs in which the request duration is equal to 60 seconds	`request_time:60` `request_time=60`
Logs in which the request duration is greater than or equal to 60 seconds and is less than 200 seconds	`request_time>=60 and request_time<200` `request_time in [60 200)`
Logs in which the value of the http_user_agent field contains Firefox	`http_user_agent:Firefox`
Logs in which the value of the http_user_agent field contains Linux and Chrome	`http_user_agent:"Linux Chrome"` `http_user_agent:Linux and http_user_agent:Chrome`
Logs that contain and	`"and"` In this search statement, and is a common string but not an operator.
Logs in which the value of the http_user_agent field contains Firefox or Chrome	`http_user_agent:Firefox or http_user_agent:Chrome`
Logs in which the value of the file info field contains apsara	`"file info":apsara`
Logs that contain strings that start with cn	`cn*`
Logs in which the value of the region field starts with cn	`region:cn*`
Logs in which the value of the region field contains cn*	`region:"cn*"`

Expected search result	Search statement
Logs in which the value of the region field ends with hai	Not supported.
Logs that contain strings that start with mo, end with la, and contain one character between mo and la	`mo?la`
Logs that contain strings that start with mo, end with la, and contain zero, one, or more characters between mo and la	`mo*la`
Logs that contain strings that start with Moz and strings that start with Sa	`Moz* and Sa*`
Logs whose topic is HTTPS or HTTP	`__topic__:HTTPS or __topic__:HTTP`
Logs that are collected from the 192.0.2.1 host	`__tag__:__client_ip__:192.0.2.1` `__tag__:__client_ip__` indicates the IP address of the host from which logs are collected.
Logs in which the remote_user field is not empty	`not remote_user:""`
Logs in which the remote_user field is empty	`remote_user:""`
Logs that do not contain the remote_user field	`not remote_user:*`
Logs in which the value of the remote_user field is null	`not request_uri:"null"`
Logs that contain the remote_user field	`remote_user:*`
Logs in which the value of the request_uri field is /request/path-2	`request_uri:/request/path-2`
Logs in which the value of the city field is not Shanghai	`not city:Shanghai`
Logs in which the value of the path field starts with /learn but does not contain /learn/level	`path:/learn* not path:/learn/level`

4.7.2. LiveTail

This topic describes how to use LiveTail to monitor and analyze logs.

Prerequisites

Logs are collected by Logtail. For more information, see Use Logtail to collect logs.

> **Note** LiveTail can monitor and extract only the log data that is collected by Logtail.

Context

In online O&M scenarios, you may need to monitor log data in real time and extract key information from the latest log data to identify causes of errors. If you use a traditional O&M method, you must run the **tail -f** command on each server to query log data. If you want to narrow the scope of the command output, you must run the **grep** or **grep -v** command to filter the log data by keyword. LiveTail that is provided in the Log Service console allows you to monitor and analyze online log data in real time. This helps you reduce your O&M workloads.

Benefits

- Logs are monitored in real time and filtered by keyword.
- Logs are collected and indexed based on the configurations of log collection.
- Log fields are delimited. This allows you to query contextual logs that contain delimiters.
- A log file can be found based on a log in the log file. This allows you to monitor the log file in real time without the need to log on to a server.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. Click the icon next to the name of the Logstore in which logs are stored, and then select **Search & Analysis**.
4. On the **Raw Logs** tab, click the icon of a log.
5. In the **LiveTail** section, view logs.

   After LiveTail is started, log data collected by Logtail is displayed on the page in real time. By default, the latest log data is displayed at the bottom of the list. You can view the latest log data without the need to scroll down. Up to 1,000 logs can be displayed on the page. If more than 1,000 logs are collected, the page is automatically refreshed to show the latest 1,000 logs.
6. If anomalies are detected in log data, click **Stop**.

   After you stop LiveTail, logs in the log monitoring list are no longer updated. You can analyze and fix errors that are found when you monitor logs.

More operations

Operation	Description
Highlight strings	You can enter one or more strings in the **Highlight** field. The specified strings are highlighted in the LiveTail section.
Filter logs by string	You can enter one or more strings in the **Filter By** field. The LiveTail section displays only the logs that contain the specified strings.

Operation	Description
Filter logs by field	You can select one or more fields from the **Filter by Field** drop-down list. The LiveTail section does not display the logs that contain the specified fields.
Stop LiveTail	You can click **Stop** to stop LiveTail. After you stop LiveTail, logs in the log monitoring list are no longer updated. You can analyze and fix errors that are found when you monitor logs.

4.7.3. LogReduce

This topic describes how to use the LogReduce feature of Log Service. You can enable the feature, view log clustering results and raw logs, and compare the number of clustered logs in different time periods.

Context

The LogReduce feature allows you to cluster similar logs and extract patterns from the logs. The feature can cluster text logs in multiple formats. You can use the feature to perform O&M operations in DevOps scenarios. For example, you can use the feature to identify errors, detect anomalies, and roll back versions. You can also use the feature to detect intrusions in security scenarios. You can save log clustering results as charts to a dashboard and view the clustered data in real time.

Benefits

- You can cluster logs in multiple formats, such as Log4j logs, JSON-formatted logs, and single-line logs.
- You can cluster hundreds of millions of logs within seconds.
- You can cluster logs in a variety of modes.
- You can retrieve raw logs based on pattern signatures.
- You can compare patterns that are extracted in different time periods.
- You can adjust the precision of log clustering based on your business requirements.

Enable the LogReduce feature

By default, the LogReduce feature is disabled.

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. Click the icon next to the name of the Logstore that you want to manage, and then select **Search & Analysis**.
4. Enable the LogReduce feature.
   i. Choose **Index Attributes > Attributes**.

      If the indexing feature is not enabled, click **Enable**.
   ii. In the **Search & Analysis** panel, turn on **LogReduce**.
   iii. (Optional)Configure an allowlist or denylist to filter fields.

      You can filter logs based on keywords. Logs that are filtered based on keywords are automatically clustered.
   iv. Click **OK**.

View raw logs and the log clustering results

1. On the Search & Analysis page, enter a search statement in the search box, specify a time range, and then click

Search & Analyze.

> **Note** You can use only search statements to filter logs. You cannot use analytic statements to filter logs because the LogReduce feature cannot cluster analysis results.

2. Click the **LogReduce** tab to view the log clustering results.

   On the **LogReduce** tab, you can view the filtered log clustering results.

Parameter	Description
Number	The ordinal number of the log cluster.
Count	The number of logs for the pattern in the specified query time range.
Pattern	The log pattern. Each log cluster has one or more sub-patterns.

   - Move the pointer over a number in the **Count** column to view the sub-patterns of the log cluster. You can also view the percentage of each sub-pattern in the log cluster. Click the plus sign (+) next to a number in the Count column to expand the sub-pattern list.
   - Click a number in the **Count** column. You are redirected to the **Raw Logs** tab. On this tab, you can view the raw logs of the pattern.

Change the precision of log clustering

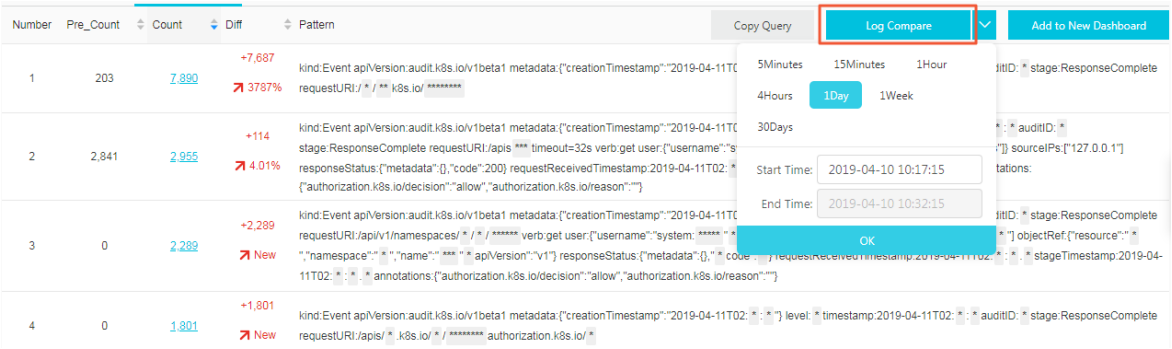
On the **LogReduce** tab, you can adjust the **Pattern Count** slider to change the precision of log clustering.

- If you adjust the slider toward **Many**, you can obtain a more precise log clustering result that has more detailed patterns.
- If you adjust the slider toward **Little**, you can obtain a less precise log clustering result that has less detailed patterns.

Compare the number of logs that are clustered in different time periods

1. On the **LogReduce** tab, click **Log Compare**.
2. Specify a time range and click **OK**.

   For example, if you set the time range to 15 minutes when you query logs and specify **1Day** for **Log Compare**, the start time and end time of log comparison are automatically displayed. The time ranges for comparison are the previous 15 minutes and the 15 minutes on the previous day.



Parameter	Description
Number	The ordinal number of the log cluster.
Pre_Count	The number of logs for the pattern in the time range that is specified by **Log Compare**.

Parameter	Description
Count	The number of logs for the pattern in the time range that is specified for the query.
Diff	The difference between the number of logs in the Pre_Count column and the number of logs in the Count column, and the growth rate.
Pattern	The log pattern.

Examples of query statements

You can use query statements to obtain log clustering results.

- Obtain log clustering results.
  - Query statement

```
* | select a.pattern, a.count,a.signature, a.origin_signatures from (select log_reduce(3) as a fr
om log) limit 1000
```

> **Note** When you view log clustering results, you can click **Copy Query** to obtain the query statement of the log clustering results.

  - Parameter settings

    Modify the parameter settings in log_reduce(precision) of the query statement. The precision parameter specifies the precision of log clustering. A smaller value indicates a higher precision and more patterns. Valid values: 1 to 16. Default value: 3.

  - Returned fields

    You can view log clustering details on the **Graph** tab.

Parameter	Description
pattern	The log pattern.
count	The number of logs for the pattern in the time range that is specified for the query.
signature	The signature of the log pattern.
origin_signatures	The secondary signature of the log pattern. You can use the secondary signature to retrieve the raw logs.

- Compare the number of logs that are clustered in different time periods.
  - Query statement

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

> **Note** When you use **Log Compare** to compare log clustering results in different time periods, you can click **Copy Query** to obtain the query statement of the log clustering results.

- Parameter settings

  Modify the parameter settings in compare_log_reduce(precision, compare_interval) of the query statement.

  - The precision parameter specifies the precision of log clustering. A smaller value indicates a higher precision and more patterns. Valid values: 1 to 16. Default value: 3.
  - The compare_interval parameter specifies the time difference between the two time ranges for comparison. The value is a positive integer. Unit: seconds.

- Returned fields

Parameter	Description
pattern	The log pattern.
count_compare	The number of logs for the pattern in the previous time range that is specified for comparison.
count	The number of logs for the pattern in the time range that is specified for the query.
diff	The difference between the numbers of logs in the count and count_compare columns.
signature	The signature of the log pattern.

Disable the LogReduce feature

If you no longer need to use the LogReduce feature, you can disable the feature.

1. On the Search and Analysis page of the Logstore for which you want to disable this feature, choose **Index Attributes > Attributes**.
2. Turn off **LogReduce**.
3. Click **OK**.

4.7.4. Contextual query

This topic describes the contextual query feature provided in the Log Service console. You can use this feature to query the full context of the log file from which specified logs are obtained.

Prerequisites

- Logs are collected by Logtail. For more information, see Use Logtail to collect logs.

  > **Note** The contextual query feature is supported only for log data that is collected by Logtail.

- The indexing feature is enabled and indexes are configured. For more information, see Enable the indexing feature and configure indexes for a Logstore.

Context

To perform a contextual query, you must specify a source server, a source file, and a log whose context you want to query. You can obtain the logs that precede or follow the specified logcollected from the log file of the server. This helps you identify and resolve errors.

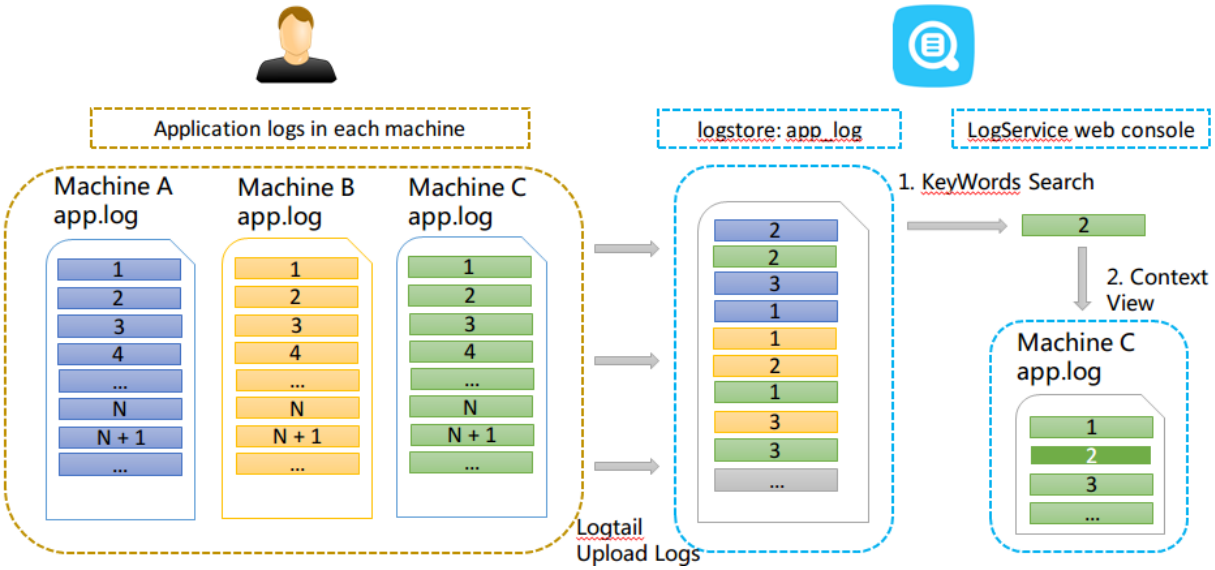
Scenario

For example, a transaction on an online-to-offline (O2O) takeout website is recorded in an application log file on a server. You must perform the following steps to complete a transaction: logon to the website, browse products, select a product, add the product to the shopping cart, place an order, pay for the order, deduct the order amount, and generate the order.

If the order fails, the O&M engineers must identify the cause of the failure at the earliest opportunity. In traditional contextual queries, the O&M engineers must be authorized by an administrator before they can log on to each server on which the O2O application is deployed. After the authorization is complete, the O&M engineers can search application logs files by order ID to identify the cause of the failure.

In Log Service, the O&M engineers can perform the following steps to locate the cause of the failure:

1. Install Logtail on the server. Create a machine group and a Logtail configuration in the Log Service console. Then, enable Logtail to upload incremental logs to Log Service.
2. On the Search & Analysis page of the Log Service console, specify a time range and find the log that records the failure based on the order ID.
3. After you find the log, scroll up until other related logs are found, for example, a log that records a credit card payment failure.



Note The contextual query feature does not support syslog logs.

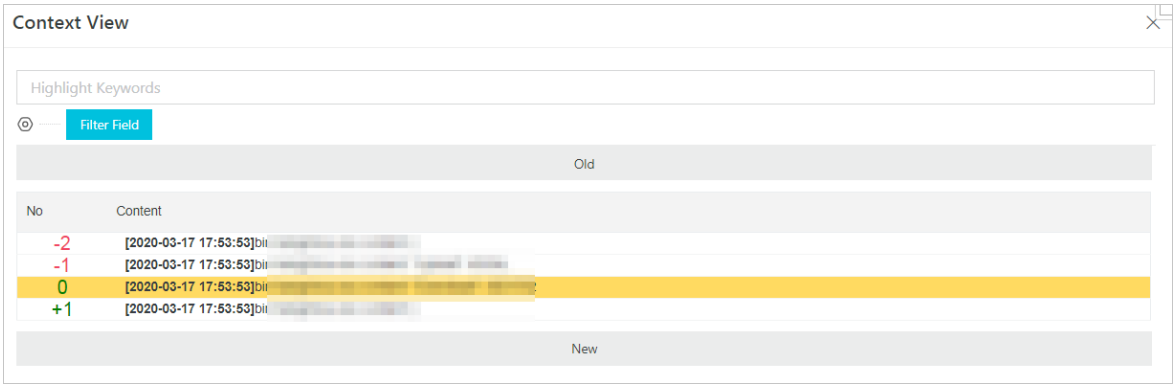
Benefits

- You can identify the causes of failures without the need to modify applications or log file formats.
- You can query the context of a log from a log file that is collected from a server in the Log Service console. You do not need to log on to the server to query the context.
- You can specify a time range to find suspicious logs before you perform a contextual query in the Log Service console. This improves troubleshooting efficiency.
- You do not need to worry about data loss that is caused by insufficient server storage or log file rotation. You can view historical log data in the Log Service console at any time.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to perform contextual queries.
3. Click the icon next to the name of the Logstore in which you want to perform contextual queries, and then select **Search & Analysis**.

4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Raw Logs** tab, find the log whose context you want to query and click the icon.
6. On the page that appears, scroll up and down to view the contextual logs.
   - To scroll up, click **Old**.
   - To scroll down, click **New**.
   - To highlight specific strings, enter the strings in the **Highlight** field. The specified strings are highlighted in the Context View section.
   - To filter logs by string, enter strings in the **Filter By** field. The Context View section displays only the logs that contain the specified strings.



4.7.5. Saved search

If you need to frequently view the result of a query statement, you can save the query statement as a saved search. Log Service provides the saved search feature to save the required data query and analysis operations. You can use a saved search to quickly perform query and analysis operations.

Prerequisites

The indexing feature is enabled and indexes are configured. For more information, see Enable the indexing feature and configure indexes for a Logstore.

Context

If you need to frequently view the result of a query statement, you can save the query statement as a saved search. Then, you can click the name of the saved search on the left side of the Search & Analysis page to execute the query statement and view the result.

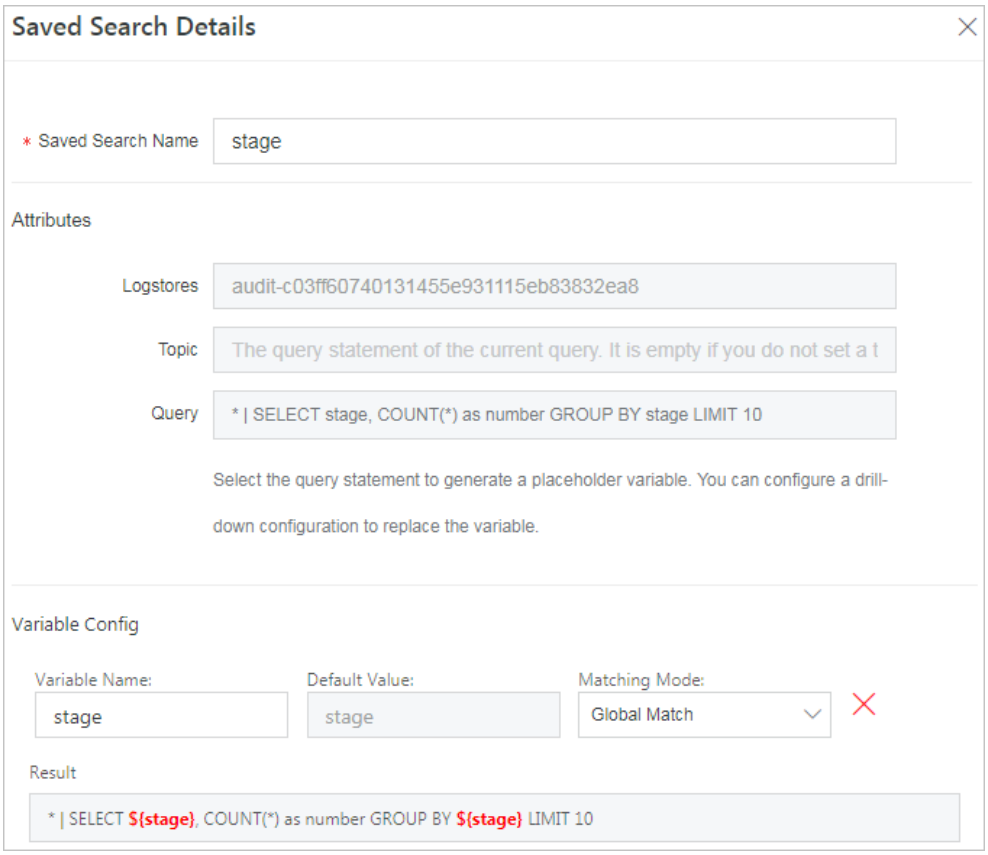
You can also use the saved search in alert rules. Log Service periodically executes the query statement of the saved search and sends alert notifications if the query result meets the preset condition.

Create a saved search

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. Click the icon next to the name of the Logstore that you want to manage, and then select **Search & Analysis**.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.

   A query statement consists of a search statement and an analytic statement in the Search statement|Analytic statement format. For more information, see Search syntax and SQL syntax and functions.
5. In the upper-right corner of the page, click **Save Search**.

6. In the **Saved Search Details** panel, configure the parameters. The following table describes the parameters.



Parameter	Description
Saved Search Name	The name of the saved search. The name must be 3 to 63 characters in length.
Variable Config	Select the required content of the query statement in the **Query** field and click **Generate Variable** to generate a placeholder variable. ○ **Variable Name**: the name of the placeholder variable. ○ **Default Value**: the content that you select from the **Query** field. ○ **Matching Mode**: the match mode. You can use the match mode to replace the default value by triggering a drill-down event. Valid values: Global Match and Exact Match. For example, you set **Event Action** to **Open Saved Search** for a chart when you configure drill-down analysis for the chart, and specify the saved search. The **Variable** of the chart is the same as the **Variable Name** of the saved search. When you click the chart value, you are redirected to the saved search. The **Default Value** of the placeholder variable is replaced by the chart value that triggers the drill-down event. Then, the new query statement is executed. For more information, see Drill-down analysis. **Note** Before you can set **Event Action** to **Open Saved Search**, you must create a saved search and configure variables.

7. Click **OK**.

After you create a saved search, click the icon next to the search box on the Search & Analysis page of the

Logstore, and click the name of the saved search to quickly perform query and analysis operations.

Modify a saved search

1. In the left-side navigation pane, choose **Resources > Saved Search**.
2. In the Saved Search list, click the saved search that you want to modify.
3. Enter a query statement and click **Search & Analyze**.

   A query statement consists of a search statement and an analytic statement in the Search statement|Analytic statement format. For more information, see Search syntax and SQL syntax and functions.
4. In the upper-right corner of the page, click **Modify Saved Search**.
5. In the **Saved Search Details** panel, modify the settings and click **OK**.

Obtain the ID of a saved search

After you create a saved search, you can use the ID of the saved search to embed the saved search page to a self-managed web page.

1. In the left-side navigation pane, choose **Resources > Saved Search**.
2. In the Saved Search list, click the saved search whose ID you want to obtain.
3. Obtain the ID of the saved search in the URL.

4.7.6. Quick analysis

Log Service provides the quick analysis feature that allows you to analyze the distribution of a field within a specified time range in an efficient manner.

Prerequisites

Indexes are configured and the analysis feature is enabled for specified fields. For more information, see Enable the indexing feature and configure indexes for a Logstore.

Features

- Allows you to analyze the first 100,000 log entries that are returned for a query.

  > **Note** When you perform quick analysis on log entries within a specified time range, Log Service collects the first 100,000 log entries. If you use a saved search to query all data in a Logstore, you must delete the Limit 100000 clause.

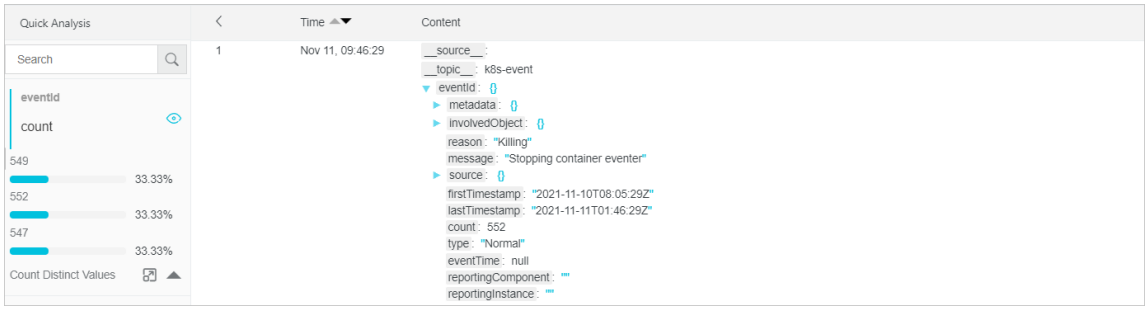
- Groups fields of the text type and provides statistics about the top 10 groups.
- Generates **approx_distinct** statements for fields of the text type.
- Supports histogram-based statistics about the approximate distribution of fields of the long and double type.

  Histogram-based statistics group sampling data and calculate the average value of each group.
- Searches for the maximum, minimum, average, or sum of fields of the long and double type.
- Generates a query statement based on quick analysis.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to perform quick analysis.
3. Click the name of the Logstore in which you want to perform quick analysis.
4. On the **Raw Logs** tab, click the field that you want to analyze in the **Quick Analysis** column.
5. Click the icon to perform quick analysis based on the current query time range and view the distribution of

the field in logs.



Text type

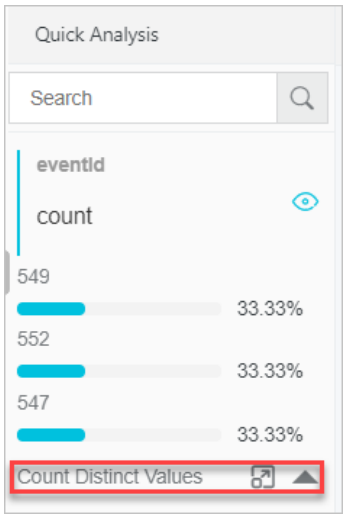
- Group and analyze log data by field of the text type.

  Click the Eye icon next to a field of the **text** type to group the first 100,000 log entries and obtain the percentages of the top 10 groups.

  The following example shows a 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
  ```

  The following figure shows the result that is returned after log entries are grouped by the `request_method` field. In this example, `GET` requests account for the majority of all returned requests.



- Calculate the number of unique values of a field.

  Under the specified field in the **Quick Analysis** column, click **Count Distinct Values** to calculate the number of unique values of the `${keyName}` field.

- Automatically copy the query statement that is used to group and analyze data to the Search & Analyze search box.

  Click the icon next to **Count Distinct Values**. The query statement that is used to group and analyze data is automatically copied to the Search & Analyze search box. You can modify the query statement.

Long and double types

- Display approximate distribution by using histograms.

A large number of values exist for fields of the `long and double` types. Therefore, the preceding grouping and analysis method is not suitable for these fields. You can use the following query statement to divide field values into 10 buckets and display the approximate distribution of the values in a histogram:

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

The following figure shows the approximate distribution of the values in the `request_time` field. In this example, the largest percentage of request time is distributed at approximately 0.059 seconds.

- Perform quick analysis by using the `Max`, `Min`, `Avg`, and `Sum` clauses.

  You can click `Max` under a field to search for the maximum value, `Min` to search for the minimum value, `Avg` to calculate the average value, and `Sum` to calculate the sum of the values.

- Automatically copy the query statement that is used to calculate the approximate distribution to the Search & Analyze search box.

  Click the icon next to `Sum`. The query statement that is used to calculate the approximate distribution is automatically copied to the Search & Analyze search box. You can modify the query statement.

4.8. Analysis grammar

4.8.1. General aggregate functions

An aggregate function is used to calculate a set of values and return a single value. This topic describes the syntax of aggregate functions. This topic also provides examples on how to use the functions.

Function	Description	Example
arbitrary(KEY)	Returns a random, non-null value from a specified column.	The following query statement returns an arbitrary value from the request_method column: `* | SELECT arbitrary(request_method) AS request_method`
avg(KEY)	Calculates the arithmetic mean of the values in a specified column.	The following query statement returns the projects whose average latency is greater than 1,000 microseconds. You can execute the statement to analyze the write latency of the projects. `method: PostLogstoreLogs | SELECT avg(latency) AS avg_latency, Project GROUP BY Project HAVING avg_latency > 1000`
checksum(KEY)	Calculates the checksum of a specified column and returns a result that is encoded in Base64.	The following query statement calculates the checksum of the request_method column: `* | SELECT checksum(request_method)` The returned result is `D2UmTL3octI=`.

Function	Description	Example
count(*)	Calculates the number of log entries.	The following query statement calculates the number of page views (PVs): `* | SELECT count(*) AS PV`
count(KEY)	Calculates the number of the log entries that contain a specified field. If the field value of a log entry is null, the log entry is not counted.	The following query statement calculates the number of the log entries that contain the request_method field: `* | SELECT count(request_method)`
count(1)	Calculates the number of log entries. This function is equivalent to `count(*)`.	The following query statement calculates the number of PVs: `* | SELECT count(1) AS PV`
count_if(KEY)	Calculates the number of log entries that meet a specified condition.	The following query statement calculates the number of requests for the value of the url field. The value ends with abc. `* | SELECT count_if(url like '%abc')`
geometric_mean(KEY)	Calculates the geometric mean of the values in a specified column.	The following query statement calculates the geometric mean of request durations: `* | SELECT geometric_mean(request_time)`
max_by(KEY_01,KEY_02)	Returns the value of KEY_01 that is associated with KEY_02 whose value is the maximum value.	The following query statement returns the point in time when the highest consumption occurs: `* | SELECT max_by(UsageEndTime, PretaxAmount) as time`
max_by(KEY_01,KEY_02,n)	Returns the n values of KEY_01 that is associated with KEY_02 whose values are the first n maximum values. The returned result is a JSON array.	The following query statement returns the three request methods that have the longest request durations: `* | SELECT max_by(request_method,request_time,3)` The returned result is `["GET","PUT","DELETE"]`.
min_by(KEY_01,KEY_02)	Returns the value of KEY_01 that is associated with KEY_02 whose value is the minimum value.	The following query statement returns the request method whose request duration is the shortest: `* | SELECT min_by(request_method,request_time)`

Function	Description	Example
min_by(KEY_01,KEY_02,n)	Returns the n values of KEY_01 that is associated with KEY_02 whose values are the first n minimum values. The returned result is a JSON array.	The following query statement returns the three request methods that have the shortest request durations: `* | SELECT min_by(request_method,request_time,3)` The returned result is `["GET","PUT","DELETE"]`.
max(KEY)	Queries the maximum value of a specified column.	The following query statement queries the longest request duration: `* | SELECT max(request_time)`
min(KEY)	Queries the minimum value of a specified column.	The following query statement queries the shortest request duration: `* | SELECT min(request_time)`
sum(KEY)	Calculates the total value of a specified column.	The following statement calculates the total size of daily NGINX traffic: `* | select date_trunc('day',__time__) AS time, sum(body_bytes_sent) AS body_bytes_sent GROUP BY time ORDER BY time`
bitwise_and_agg(KEY)	Returns the result of the bitwise AND operation for the values of a specified column. The returned result is in the two's complement format.	The following query statement performs a bitwise AND operation on all values of the request_time column: `* | SELECT bitwise_and_agg(request_time)`
bitwise_or_agg(KEY)	Returns the result of the bitwise OR operation for values of a specified column. The returned result is in the two's complement format.	The following query statement performs a bitwise OR operation on all values of the request_time column: `* | SELECT bitwise_or_agg(request_time)`

4.8.2. Security check functions

Security check functions in Log Services are designed based on the globally shared WhiteHat Security asset library. This topic describes security check functions that you can use to check whether an IP address, domain name, or URL in logs is secure.

Scenarios

- O&M personnel of enterprises and institutions in Internet, gaming, information, and other industries that require robust O&M services can use security check functions to identify suspicious requests or attacks. They can also use the functions to implement in-depth analysis and defend against potential attacks.
- O&M personnel of enterprises and institutions in banking, securities, e-commerce, and other industries that require strong protection for internal assets can use security check functions to identify requests to suspicious websites and downloads initiated by trojans. Then the O&M personnel can take immediate actions to prevent potential losses.

Features

- Reliability: built upon the globally shared WhiteHat Security asset library that is updated in a timely manner.
- Efficiency: capable of screening millions of IP addresses, domain names, and URLs within seconds.
- Ease of use: supports the analysis of network logs by using the security_check_ip, security_check_domain, and security_check_url functions.
- Flexibility: supports interactive queries, report creation, and alert configurations and subsequent actions.

Functions

Function	Description	Example
security_check_ip	Checks whether an IP address is secure. • The value 1 indicates that the specified IP address is suspicious. • The value 0 indicates that the specified IP address is secure.	`select security_check_ip(real_client_ip)`
security_check_domain	Checks whether a domain name is secure. • The value 1 indicates that the specified domain name is suspicious. • The value 0 indicates that the specified domain name is secure.	`select security_check_domain(site)`
security_check_url	Checks whether a URL is secure. • The value 1 indicates that the specified URL is suspicious. • The value 0 indicates that the specified URL is secure.	`select security_check_domain(concat(host, url))`

Examples

- Check external suspicious requests and generate reports

  For example, an e-commerce enterprise collects logs from its NGINX servers and wants to scan suspicious client IP addresses. To do this, the enterprise can pass the ClientIP field in logs that are collected from the NGINX servers to the `security_check_ip` function and filter out IP addresses associated with the returned value 1. Then the enterprise can query the countries where the IP addresses are located and ISPs to which the IP addresses belong.

  SQL statement for this scenario:

  ```
  * | select ClientIP, ip_to_country(ClientIP) as country, ip_to_provider(ClientIP) as provider, count(1) as PV where security_check_ip(ClientIP) = 1 group by ClientIP order by PV desc
  ```

  Display the ISPs and countries in a map.

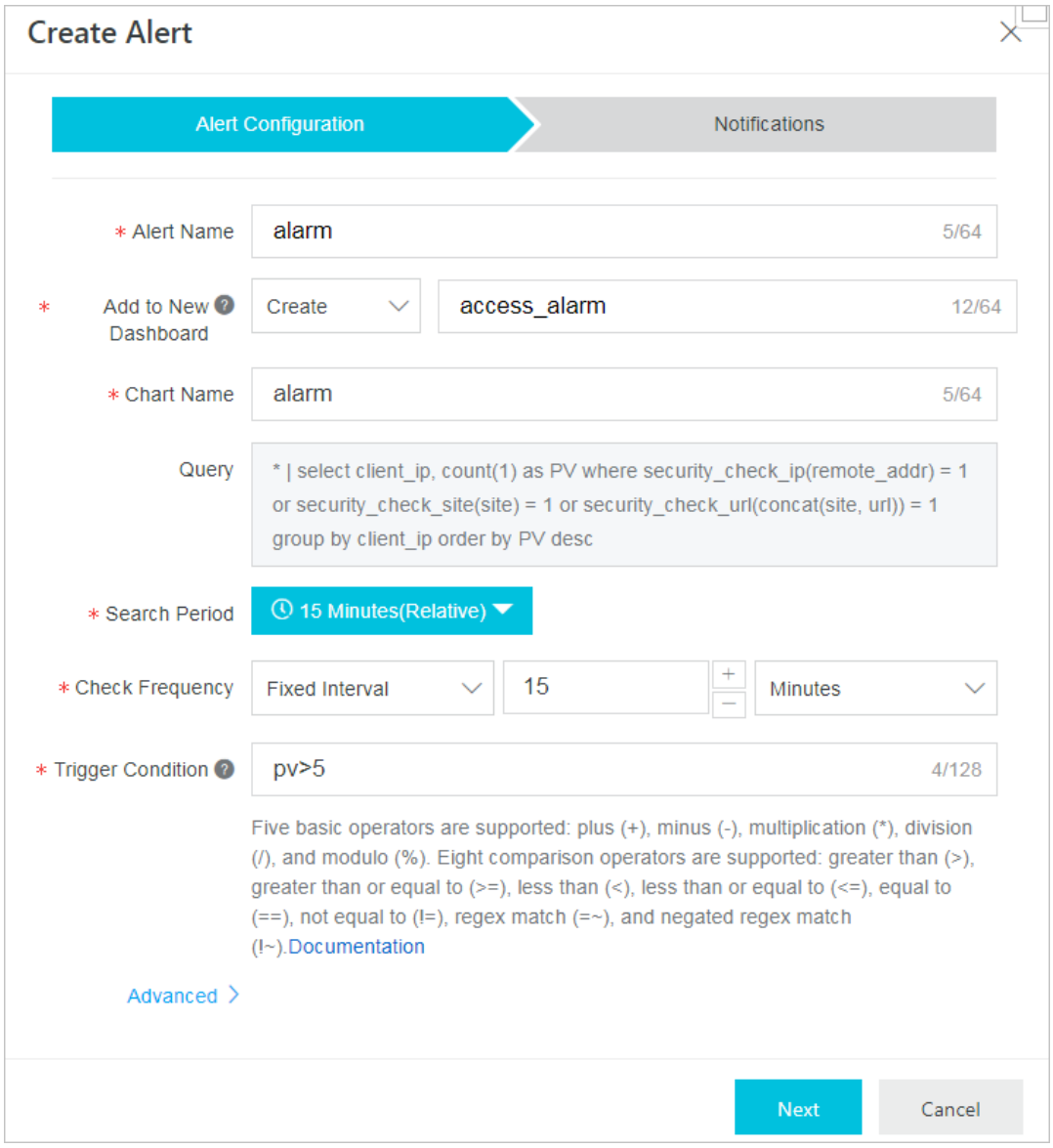
client_ip	country	provider	PV
180 3	CN		3
103	CN		3
180 7	CN		1

- Check internal suspicious requests and send alerts

  For example, a securities operator collects logs of its internal devices that access the Internet through gateways. To check requests to suspicious websites, the operator can run the following statement:

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

  The operator can save this statement as a saved search and configure an alert. An alert is triggered when a client frequently accesses suspicious websites. The statement in the alert can be configured to run every five minutes to check if a client has frequently (more than five times) accessed suspicious websites in the past one hour. The following figure shows the configurations of an alert.



4.8.3. Map functions and operators

This topic describes the syntax of map functions and operators. This topic also provides examples on how to use the functions and operators.

The following table describes the map functions and operators that are supported by Log Service.

> **Notice** If you want to use strings in analytic statements, you must enclose the strings in single quotation marks (''). Strings that are not enclosed or are enclosed in double quotation marks ("") indicate field names or column names. For example, 'status' indicates the status string, and status or "status" indicates the status log field.

Function	Syntax	Description
Subscript operator	[*x*]	Returns the value of a key from a map.
cardinality function	cardinality(*x*)	Returns the size of a map.
element_at function	element_at(*x*, *key*)	Returns the value of a key from a map.
histogram function	histogram(*x*)	Groups query and analysis results and returns data in the JSON format.
histogram_u function	histogram_u(*x*)	Groups query and analysis results and returns data in multiple rows and multiple columns.
map function	map()	Returns an empty map.
	map(*x*, *y*)	Returns a map that is created by using two arrays.
map_agg function	map_agg(*x*, *y*)	Returns a map that is created by using *x* and *y*. *x* is a key in the map. *y* is the value of the key in the map. If *y* has multiple values, a random value is extracted as the value of the key.
map_concat function	map_concat(*x*, *y*...)	Returns the union of multiple maps.
map_filter function	map_filter(*x*, *lambda_expression*)	Filters elements in a map based on a lambda expression.
map_keys function	map_keys(*x*)	Returns an array that consists of all keys in a map.
map_values function	map_values(*x*)	Returns an array that consists of all values in a map.
multimap_agg function	multimap_agg(*x*, *y*)	Returns a multimap that is created by using *x* and *y*. *x* is a key in the multimap. *y* is the value of the key in the multimap. The value is of the array type. If *y* has multiple values, all the values are extracted as the values of the key.

Subscript operator

The subscript operator is used to return the value of a key from a map.

- Syntax

```
[x]
```

- Parameters

Parameter	Description
x	The value of this parameter is of the varchar type.

- Return value type

  An arbitrary data type.

- Example

  In a log that is transformed by a data transformation job, the value of the etl_context field is of the map type. You can use the subscript operator to obtain the value of the project key from the value of the etl_context field.

  - Sample field

```
etl_context: {
 project:"datalab-148****6461-cn-chengdu"
 logstore:"internal-etl-log"
 consumer_group:"etl-83****4d1965"
 consumer:"etl-b2d40ed****c8d6-291294"
 shard_id:"0" }
```

  - Query statement

```
* | SELECT try_cast(json_parse(etl_context) AS map(varchar, varchar))['project']
```

  - Query and analysis result

cardinality function

The cardinality function is used to return the size of a map.

- Syntax

```
cardinality(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the map type.

- Return value type

  The bigint type.

- Example

  Use the histogram function to obtain the number of requests for each request method. Then, use the cardinality function to obtain the number of request methods.

- Query statement

```
* |
SELECT
  histogram(request_method) AS request_method,
  cardinality(histogram(request_method)) AS "kinds"
```

- Query and analysis result

request_method	kinds
{"DELETE":5,"POST":7,"GET":41,"PUT":4}	4

element_at function

The element_at function is used to return the value of a key from a map.

- Syntax

```
element_at(x, key)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the map type.
key	The value of this parameter is a key in the specified map.

- Return value type

  An arbitrary data type.

- Example

  Use the histogram function to obtain the number of requests for each request method. Then, use the element_at function to obtain the value of the DELETE field.

  - Query statement

```
* |
SELECT
  histogram(request_method) AS request_method,
  element_at(histogram(request_method),'DELETE') AS "count"
```

  - Query and analysis result

request_method	count
{"HEAD":9,"DELETE":140,"POST":319,"GET":1298,"PUT":337}	140

histogram function

The histogram function is used to group query and analysis results and return data in the JSON format. This function is equivalent to `* | SELECT count(*) GROUP BY x`.

- Syntax

```
histogram(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of an arbitrary data type.

- Return value type

  The map type.

- Example

  Use the histogram function to obtain the number of requests for each request method.

  - Query statement

    ```
    * | SELECT histogram(request_method) AS request_method
    ```

  - Query and analysis result

histogram_u function

The histogram_u function is used to group query and analysis results and return data in multiple rows and multiple columns.

- Syntax

  ```
  histogram_u(x)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of an arbitrary data type.

- Return value type

  The bigint type.

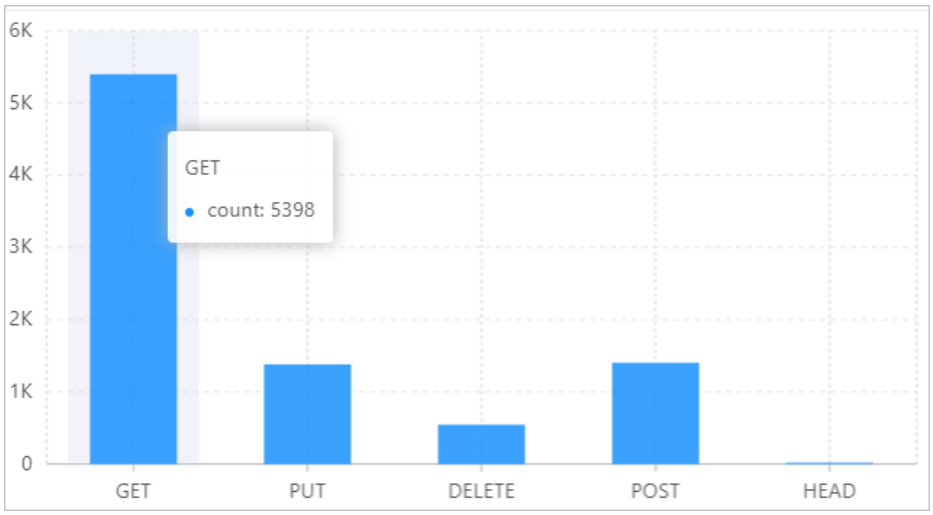
- Example

  Use the histogram_u function to obtain the number of requests for each request method and then display the number on a column chart.

  - Query statement

    ```
    *|SELECT  histogram_u(request_method) as request_method
    ```

- Query and analysis result



map function

The map function is used to return an empty map or return a map that is created by using two arrays.

- Syntax
  - The following syntax of the map function is used to return an empty map:

    ```
    map()
    ```

  - The following syntax of the map function is used to return a map that is created by using two arrays:

    ```
    map(x,y)
    ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.

- Return value type

  The map type.

- Examples

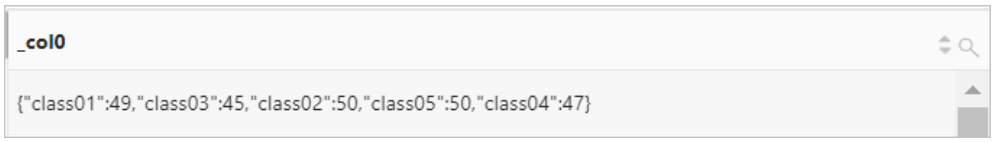
- Example 1: The class field specifies classes. The number field specifies the number of students in the classes. The values of the two fields are of the array type. Use the map function to create a map based on the values of the two fields. In the returned result, each class is mapped to the number of students in the class.
  - Sample fields

    ```
    class:["class01","class02","class03","class04","class05"]
    number:[49,50,45,47,50]
    ```

  - Query statement

    ```
    * | SELECT map(try_cast(json_parse(class) AS array(varchar)) ,try_cast(json_parse(number) AS ar
    ray(bigint)))
    ```

  - Query and analysis result

- Example 2: Return an empty map.
  - Query statement

    ```
    *| SELECT map()
    ```

  - Query and analysis result

map_agg function

The map_agg function is used to return a map that is created by using *x* and *y*. *x* is a key in the map. *y* is the value of the key in the map. If *y* has multiple values, a random value is extracted as the value of the key.

- Syntax

  ```
  map_agg(x, y)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of an arbitrary data type.
y	The value of this parameter is of an arbitrary data type.

- Return value type

  The map type.

- Example

  Extract the values of the request_method and request_time fields and then use the extracted values to create a map. The value of the request_method field is a key in the map. The value of the request_time field is the value of the key in the map.

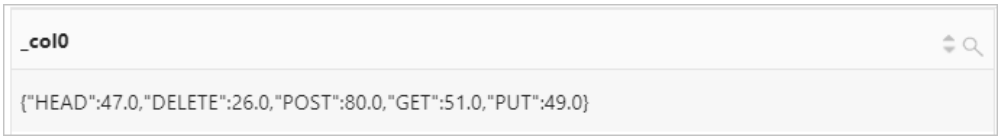
  - Sample fields

    ```
    request_method:POST
    request_time:80
    ```

- Query statement

```
* | SELECT map_agg(request_method,request_time)
```

- Query and analysis result



map_concat function

The map_concat function is used to return the union of multiple maps.

- Syntax

```
map_concat(x, y)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the map type.
y	The value of this parameter is of the map type.

- Return value type

  The map type.

- Example

  In a log that is transformed by a data transformation job, the values of the etl_context and progress fields are of the map type. You can use the map_concat function to obtain the union of the field values.

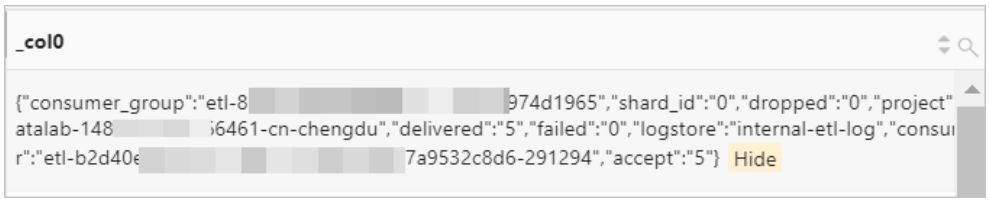
  - Sample fields

```
etl_context: {
 project:"datalab-148****6461-cn-chengdu"
 logstore:"internal-etl-log"
 consumer_group:"etl-83****4d1965"
 consumer:"etl-b2d40ed****c8d6-291294"
 shard_id:"0" }
progress: {
 accept:3
 dropped:0
 delivered:3
 failed:0 }
```

  - Query statement

```
* |
SELECT
  map_concat(
    cast (
      json_parse(etl_context) AS map(varchar, varchar)
    ),
    cast (json_parse(progress) AS map(varchar, varchar))
  )
```

- Query and analysis result



Example

map_filter function

The map_filter function is used to filter elements in a map based on a lambda expression.

- Syntax

```
map_filter(x, lambda_expression)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the map type.
lambda_expression_expression	The lambda expression. For more information, see Lambda functions.

- Return value type

  The map type.

- Example

  Create a map that does not contain null values from two arrays by using the lambda expression `(k, v) -> v is not null`.

  - Query statement

    ```
    * | SELECT map_filter(map(array[10, 20, 30], array['a', NULL, 'c']), (k, v) -> v is not null)
    ```

  - Query and analysis result

map_keys function

The map_keys function is used to return an array that consists of all keys in a map.

- Syntax

```
map_keys(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the map type.

- Return value type

  The array type.

- Example

  In a log that is transformed by a data transformation job, the value of the etl_context field is of the map type. You can use the map_keys function to obtain all keys from the value of the etl_context field.

  - Sample field

    ```
    etl_context: {
     project:"datalab-148****6461-cn-chengdu"
     logstore:"internal-etl-log"
     consumer_group:"etl-83****4d1965"
     consumer:"etl-b2d40ed****c8d6-291294"
     shard_id:"0" }
    ```

  - Query statement

    ```
    * | SELECT map_keys(try_cast(json_parse(etl_context) AS map(varchar, varchar)))
    ```

  - Query and analysis result

map_values function

The map_values function is used to return an array that consists of all values in a map.

- Syntax

  ```
  map_values(x)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the map type.

- Return value type

  The array type.

- Example

  In a log that is transformed by a data transformation job, the value of the etl_context field is of the map type. You can use the map_values function to obtain the values of all keys from the value of the etl_context field.

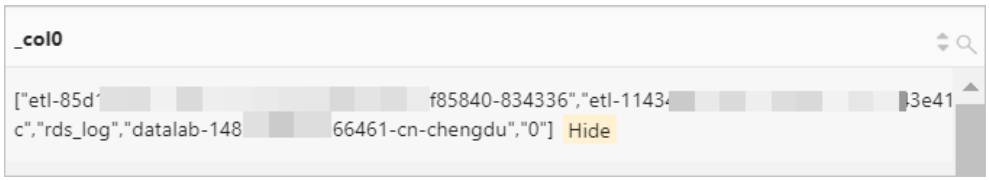
  - Sample field

    ```
    etl_context: {
     project:"datalab-148****6461-cn-chengdu"
     logstore:"internal-etl-log"
     consumer_group:"etl-83****4d1965"
     consumer:"etl-b2d40ed****c8d6-291294"
     shard_id:"0" }
    ```

  - Query statement

    ```
    * | SELECT map_values(try_cast(json_parse(etl_context) AS map(varchar, varchar)))
    ```

- Query and analysis result



multimap_agg function

The multimap_agg function is used to return a multimap that is created by using *x* and *y*. *x* is a key in the multimap. *y* is the value of the key in the multimap. The value is of the array type. If *y* has multiple values, all the values are extracted as the values of the key.

- Syntax

```
multimap_agg(x, y)
```

- Parameters

Parameter	Description
x	The value of this parameter is of an arbitrary data type.
y	The value of this parameter is of an arbitrary data type.

- Return value type

  The map type.

- Example

  Extract all values of the request_method and request_time fields and then use the extracted values to create a multimap. The value of the request_method field is a key in the multimap. The value of the request_time field is the value of the key in the multimap. The value of the key is of the array type.

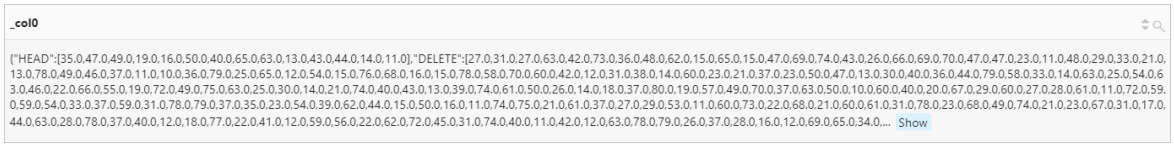
  - Sample field

```
request_method:POST
request_time:80
```

  - Query statement

```
* | SELECT multimap_agg(request_method,request_time)
```

  - Query and analysis result



4.8.4. Approximate functions

This topic describes the syntax of approximate functions. This topic also provides examples on how to use the functions.

Function	Description	Example
`approx_distinct(x)`	Estimates the number of unique values in the x field.	None

Function	Description	Example
`approx_percentile(x,percentage)`	Sorts the values of the x field in ascending order and returns the value that is approximately at the percentage position.	`approx_percentile(x,0.5)`: returns the value that is approximately at the 50% position in the x field.
`approx_percentile(x,percentages)`	This function is similar to `approx_percentile(x,percentage)`. You can specify multiple percentages to return the values at the specified percentage positions.	`approx_percentile(x,array[0.1,0.2])`
`numeric_histogram(buckets,Value)`	Distributes all values of the *Value* field to multiple buckets. The *buckets* parameter specifies the number of buckets. The key of each bucket and the number of values in a bucket are returned. This function is equivalent to `select count group by`. **Note** The returned result is of the JSON type.	`method:POST | select numeric_histogram(10,latency)`: distributes the values of the latency field for POST requests to 10 buckets and calculates the number of latency field values in each bucket.
`numeric_histogram_u(buckets,Value)`	Distributes all values of the *Value* field to multiple buckets. The *buckets* parameter specifies the number of buckets. The key of each bucket and the number of values in a bucket are returned. This function is equivalent to `select count group by`. **Note** The returned result is a table that includes multiple rows and columns.	`method:POST | select numeric_histogram(10,latency)`: distributes the values of the latency field for POST requests to 10 buckets and calculates the number of latency field values in each bucket.

Note Buckets are evenly divided by aggregation degree. The returned result for each bucket includes the average value of the bucket and the number of values in the bucket.

4.8.5. Mathematical statistics functions

This topic describes the syntax of mathematical statistics functions. This topic also provides examples on how to use the functions.

Syntax

Function	Description
corr(*key1*, *key2*)	Calculates the correlation coefficient between two specific columns. The return value is in the range of [0,1].

Function	Description
covar_pop(*key1*, *key2*)	Calculates the population covariance of two specific columns.
covar_samp(*key1*, *key2*)	Calculates the sample covariance of two specific columns.
regr_intercept(*key1*, *key2*)	Returns the linear regression intercept of input values. *key1* is the dependent value and *key2* is the independent value.
regr_slope(*key1*, *key2*)	Returns the linear regression slope of input values. *key1* is the dependent value and *key2* is the independent value.
stddev(*key*)	Calculates the sample standard deviation of the *key* column. This function is equivalent to the stddev_samp function.
stddev_samp(*key*)	Calculates the sample standard deviation of the *key* column.
stddev_pop(*key*)	Returns the population standard deviation of the *key* column.
variance(*key*)	Calculates the sample variance of the *key* column. This function is equivalent to the var_samp function.
var_samp(*key*)	Calculates the sample variance of the *key* column.
var_pop(*key*)	Calculates the population variance of the *key* column.

Examples

- Example 1: Calculate the correlation coefficient of two specific columns.
  - Query statement

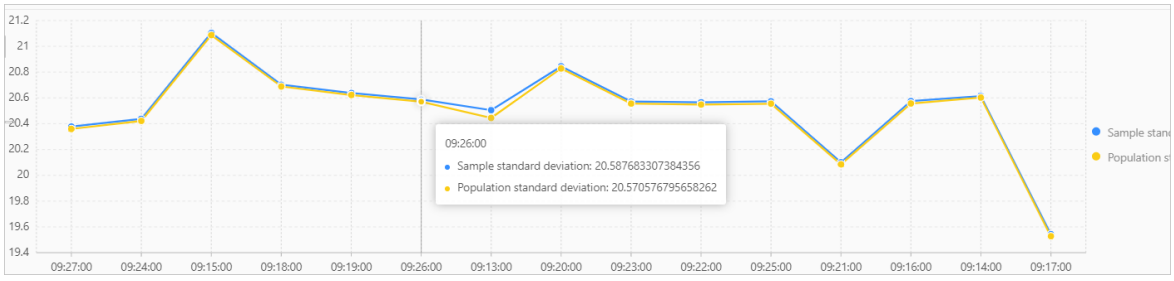
```
* | SELECT corr(request_length, request_time)
```

  - Query and analysis result

- Example 2: Query the sample standard deviation and population standard deviation of pre-tax income.
  - Query statement

```
* | SELECT stddev(PretaxGrossAmount) as "sample standard deviation", stddev_pop(PretaxGrossAmount
) as "population standard deviation", time_series(__time__, '1m', '%H:% I:%s', '0') AS time GROUP
BY time
```

  - Query and analysis result



4.8.6. Mathematical calculation functions

This topic describes the syntax of mathematical calculation functions. This topic also provides examples on how to use the functions.

Syntax

> **Note**
>
> - Mathematical calculation functions support the following operators: `+-*/%`.
> - In the following functions, *x* and *y* can be numbers, log fields, or expressions whose calculation result is a number.

Function	Description
abs(x)	Calculates the absolute value of a number.
cbrt(x)	Calculates the cube root of a number.
sqrt(x)	Calculates the square root of a number.
cosine_similarity(x,y)	Calculates the cosine similarity between x and y.
degrees(x)	Converts radians to degrees.
radians(x)	Converts degrees to radians.
e()	Returns Euler's number.
exp(x)	Returns Euler's number raised to the power of a number.
ln(x)	Calculates the natural logarithm of a number.
log2(x)	Calculates the base-2 logarithm of a number.
log10(x)	Calculates the base-10 logarithm of a number.
log(x,b)	Calculates the base-b logarithm of a number.
pi()	Returns the value of π to 14 decimal places.
pow(x,b)	Calculates the value of a number raised to the power of b.
rand()	Returns a random number.
random(0,n)	Returns a random number that is greater than or equal to 0 and less than n.
round(x)	Returns a number rounded to the nearest integer.
round(x, N)	Returns a number rounded to N decimal places.
floor(x)	Returns a number rounded down to the nearest integer. For example, when you execute the `* | SELECT floor(2.5)` statement, 2.0 is returned.
ceiling(x)	Returns a number rounded up to the nearest integer. For example, when you execute the `* | SELECT ceiling(2.5)` statement, 3.0 is returned.

Function	Description
from_base(varchar, bigint)	Converts a string to a base-encoded number.
to_base(x, radix)	Converts a number to a base-encoded string.
truncate(x)	Truncates the fractional part of a number.
acos(x)	Calculates the arc cosine of a number.
asin(x)	Calculates the arc sine of a number.
atan(x)	Calculates the arc tangent of a number.
atan2(y,x)	Calculates the arc tangent of the quotient of a number divided by another number.
cos(x)	Calculates the cosine of a number.
sin(x)	Calculates the sine of a number.
cosh(x)	Calculates the hyperbolic cosine of a number.
tan(x)	Calculates the tangent of a number.
tanh(x)	Calculates the hyperbolic tangent of a number.
infinity()	Returns a positive infinity value.
is_nan(x)	Checks whether a value is a non-numeric value.

Example

Compare the number of page views (PVs) of today with the number of PVs of the previous day, and show the comparison result as a percentage.

- Query statement

```
* | SELECT diff [1] AS today, round((diff [3] -1.0) * 100, 2) AS growth FROM (SELECT compare(pv, 86400) as diff FROM (SELECT COUNT(*) as pv FROM website_log))
```

- Query and analysis result

today	growth
1564075.0	-22.11

4.8.7. String functions

This topic describes the syntax of string functions. This topic also provides examples on how to use the functions.

Syntax

Note

- If you want to use strings in analytic statements, you must enclose the strings in single quotation marks (''). Strings that are not enclosed or enclosed in double quotation marks ("") indicate field names or column names. For example, 'status' indicates the status string, and status or "status" indicates the status log field.
- The key parameter in the following table indicates a log field.

Function	Description
chr(*number*)	Returns the characters that match the ASCII value of a specified parameter.
codepoint(*key*)	Converts a field of the ASCII type to a field value of the bigint type.
length(*key*)	Calculates the length of a string. The return value is of the integer type.
lower(*key*)	Converts the characters in a string to lowercase letters. The return value is of the varchar type in lowercase letters.
upper(*key*)	Converts the characters in a string to lowercase letters. The return value is of the varchar type in uppercase letters.
lpad(*key*, *length*, *lpad_string*)	Pads a string to a specified length from the left with a specified substring. The value of the *length* parameter is an integer that specifies the length of the result string. • If the length of the string is less than the value of the *length* parameter, the string is padded by the specified substring from the left. • If the length of the string is greater than the value of the *length* parameter, the function returns only the first *length* characters in the string. The return value is of the varchar type.
rpad(*key*, *length*,*rpad_string*)	Pads a string to a specified length from the right with a specified substring. The value of the *length* parameter is an integer that specifies the length of the result string. • If the length of the string is less than the value of the *length* parameter, the string is padded by the specified substring from the right. • If the length of the string is greater than the value of the *length* parameter, the function returns only the first *length* characters in the string. The return value is of the varchar type.
trim(*key*)	Deletes space characters from the start and the end of a string. The return value is of the varchar type.
ltrim(*key*)	Deletes space characters from the start of a string. The return value is of the varchar type.
rtrim(*key*)	Deletes space characters from the end of a string. The return value is of the varchar type.

Function	Description
replace(*key*,*substring*,*replace*)	Replaces matched characters in a string with specified characters. The return value is of the varchar type.
replace(*key*,*substring*)	Deletes matched characters from a string. The return value is of the varchar type.
reverse(*key*)	Reverses the characters in a string.
split(*key*,*delimeter*,*N*)	Splits a string with a specified delimiter and returns a set of N substrings. The return value is an array.
split_part(*key*,*delimeter*,*part*)	Splits a string with a specified delimiter and returns the substring at a specified position. The value of the *part* parameter is an integer that is greater than 0. The return value is of the varchar type.
split_to_map(*key*, *delimiter01*, *delimiter02*)	Splits a string with the first specified delimiter, and then splits the string with the second specified delimiter. The return value is a map.
position(*substring* IN *key*)	Returns the position of a specified substring in a string. The return value is of the integer type. The value starts from 1.
strpos(*key*, *substring*)	Returns the position of a specified substring in a string. This function is equivalent to the position(*substring* IN *key*) function. The return value is of the integer type. The value starts from 1.
substr(*key*, *start*)	Returns the substring at a specified position in a string. The *start* parameter specifies the position of the substring to be extracted. The value of the start parameter starts from 1. The return value is of the varchar type.
substr(*key*, *start*, *length*)	Returns the substring at a specified position in a string and specifies the length of the substring. The *start* parameter specifies the position of the substring to be extracted. The value of the start parameter starts from 1. The *length* parameter specifies the length of the substring. The return value is of the varchar type.
concat(*key01*,*key02*,*key03*)	Concatenates multiple strings into one string. The return value is of the integer type. The value starts from 1.

Function	Description
levenshtein_distance(*key01*, *key02*)	Returns the minimum edit distance between two strings.
hamming_distance (*string1*,*string2*)	Returns the Hamming distance between two strings.

Examples

Sample log:

```
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_4; en-us) AppleWebKit/528.4+ (KHTML, li
ke Gecko) Version/4.0dp1 Safari/526.11.2
request_uri:/request/path-1/file-9?0457349059345
scheme:https
server_protocol:HTTP/2.0
region:cn-shanghai
time: upstream_response_time:"80", request_time:"40"
```

- Use a question mark (?) to split the value of the request_uri field and return the first substring. The returned substring indicates a file path. Then, calculate the number of requests that correspond to each path.

  ```
  * | SELECT count(*) AS PV, split_part(request_uri, '?', 1) AS Path GROUP BY Path ORDER BY pv DESC L
  IMIT 3
  ```

PV	Path
4355	/request/path-1/file-5
4328	/request/path-1/file-1
4296	/request/path-2/file-3

- Extract the first four characters (HTTP) from the value of the server_protocol field and calculate the number of requests that use the HTTP protocol.

  ```
  * | SELECT substr(server_protocol,1,4) AS protocol, count(*) AS count GROUP BY server_protocol
  ```

protocol	count
HTTP	9078

- Use commas (,) and colons (:) to split the value of the time field and return a value of the map type.

  ```
  * | SELECT split_to_map(time,',',':')
  ```

_col0
{"request_time":"\"40\"","upstream_response_time":"\"80\""}
{"request_time":"\"40\"","upstream_response_time":"\"80\""}

- Check whether the value of the http_user_agent field starts with the letter M.

  ```
  * | SELECT substr(http_user_agent, 1, 1)=chr(77)
  ```

- Return the position of the letter H in the value of the server_protocol field.

```
* | SELECT strpos(server_protocol,'H')
```

- Use a forward slash (/) to split the value of the server_protocol field into two substrings and return an array of the substrings.

```
* | SELECT split(server_protocol,'/',2)
```

- Replace cn in the region field with China.

```
* | select replace(region,'cn','China')
```

4.8.8. Date and time functions

Log Service provides the following types of date and time functions that you can use to analyze log data: date function, time function, truncation function, interval function, and time series padding function. You can use the functions to convert the date and time formats of log data. You can also use the functions to group and aggregate log data.

> Note
>
> - The timestamp of a log in Log Service is accurate to seconds. Therefore, you can specify the precision of the time format only to seconds.
> - You need to specify the time format only for the time in a time string. Other parameters such as the time zone are not required.
> - Each log in Log Service contains the reserved __time__ field. The value of the field is a UNIX timestamp. For example, 1592374067 indicates 2020-06-17 14:07:47.

Date functions

Function	Description	Example
current_date	Returns the current date. • Return value format: `YYYY-MM-DD`. Example: 2021-01-12. • Return value type: date.	`* | select current_date`
current_time	Returns the current time. • Return value format: `HH:MM:SS.Ms Time zone`. Example: 01:14:51.967 Asia/Shanghai. • Return value type: time.	`* | select current_time`
current_timestamp	Returns the current date and time. • Return value format: `YYYY-MM-DD HH:MM:SS.Ms Time zone`. Example: 2021-01-12 17:16:09.035 Asia/Shanghai. • Return value type: timestamp.	`* | select current_timestamp`
current_timezone()	Returns the current time zone. Return value type: varchar. Example: Asia/Shanghai.	`* | select current_timezone()`
localtime	Returns the local time. • Return value format: `HH:MM:SS.Ms` • Return value type: time.	`* | select localtime`
localtimestamp	Returns the local date and time. • Return value format: `YYYY-MM-DD HH:MM:SS.Ms` • Return value type: timestamp.	`* | select localtimestamp`
now()	Returns the current date and time. This function is equivalent to the current_timestamp function. • Return value format: `YYYY-MM-DD HH:MM:SS.Ms Time zone`. • Return value type: timestamp.	`* | select now()`
from_iso8601_timestamp(*ISO8601*)	Converts an ISO 8601-formatted datetime expression to a timestamp expression that contains a time zone. • Return value format: `YYYY-MM-DD HH:MM:SS.Ms Time zone`. • Return value type: timestamp.	`* | select from_iso8601_timestamp('2020-05-03T17:30:08')`
from_iso8601_date(*ISO8601*)	Converts an ISO 8601-formatted date expression to a date expression. • Return value format: `YYYY-MM-DD` • Return value type: date.	`* | select from_iso8601_date('2020-05-03')`

Function	Description	Example
from_unixtime(*UNIX timestamp*)	Converts a UNIX timestamp to a timestamp expression. • Return value format: `YYYY-MM-DD HH:MM:SS.Ms` • Return value type: timestamp.	`* | select from_unixtime(1494985275)`
from_unixtime(*UNIX timestamp*,*time zone*)	Converts a UNIX timestamp to a timestamp expression that contains a time zone. • Return value format: `YYYY-MM-DD HH:MM:SS.Ms Time zone`. • Return value type: timestamp.	`* | select from_unixtime(1494985275,'Asia/Shanghai')`
to_unixtime(*timestamp*)	Converts a timestamp expression to a UNIX timestamp. Return value type: long. Example: 1494985500.848.	`*| select to_unixtime('2017-05-17 09:45:00.848 Asia/Shanghai')`

Time functions

Function	Description	Example
date_format(*timestamp*,*format*)	Converts a timestamp expression to a datetime format string.	`* | select date_format(date_parse('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%S'), '%Y-%m-%d')`
date_parse(*string*,*format*)	Represents a datetime format string, and then converts the datetime format string to a timestamp expression. The following table describes the formats.	`* | select date_format(date_parse(time,'%Y-%m-%d %H:%i:%S'), '%Y-%m-%d')`

Formats

format	Description
%a	The abbreviated name of the day of the week. Examples: Sun and Sat.
%b	The abbreviated name of the month. Examples: Jan and Dec.
%c	The month. The value is of the numeric type. Valid values: 1 to 12.
%D	The day of the month. Examples: 0th, 1st, 2nd, and 3rd.
%d	The day of the month. The value is in the decimal format. Valid values: 01 to 31.
%e	The day of the month. The value is in the decimal format. Valid values: 1 to 31.
%H	The hour. The 24-hour clock is used.

format	Description
%h	The hour. The 12-hour clock is used.
%I	The hour. The 12-hour clock is used.
%i	The minute. The value is of the numeric type. Valid values: 00 to 59.
%j	The day of the year. Valid values: 001 to 366.
%k	The hour. Valid values: 0 to 23.
%l	The hour. Valid values: 1 to 12.
%M	The full month name. Examples: January and December.
%m	The month. The value is of the numeric type. Valid values: 01 to 12.
%p	The abbreviation that indicates the morning or afternoon of the day. Valid values: AM and PM.
%r	The time. The 12-hour clock is used. The time is in the `hh:mm:ss AM/PM` format.
%S	The second. Valid values: 00 to 59.
%s	The second. Valid values: 00 to 59.
%T	The time. The 24-hour clock is used. The time is in the `hh:mm:ss` format.
%V	The week number of the year. Sunday is the first day of each week. Valid values: 01 to 53.
%v	The week number of the year. Monday is the first day of each week. Valid values: 01 to 53.
%W	The full name of the day of the week. Examples: Sunday and Saturday.
%w	The day of the week. The value 0 indicates Sunday.
%Y	The four-digit year. Example: 2020.
%y	The two-digit year. Example: 20.
%%	The escape character of the percent sign (%).

Truncation function

The date_trunc() function truncates a datetime expression based on the specified part of a time. You can use a truncation function to truncate a time by second, minute, hour, day, month, or year. This function is suitable for time-based statistics.

- Syntax

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

- Parameters

The value of the x parameter can be a datetime expression, for example, 2021-01-12 03:04:05.000 or 1610350836. The value of the x parameter can be a time field, for example, __time__. The valid values of the unit parameter are second, minute, hour, day, week, month, quarter, and year. The following table describes examples of this parameter.

Example	Result	Description
* | select date_trunc('second', 2021-01-12 03:04:05.000)	2021-01-12 03:04:05.000	None.
* | select date_trunc('minute', 2021-01-12 03:04:05.000)	2021-01-12 03:04:00.000	None.
* | select date_trunc('hour', 2021-01-12 03:04:05.000)	2021-01-12 03:00:00.000	None.
* | select date_trunc('day', 2021-01-12 03:04:05.000)	2021-01-12 00:00:00.000	Returns 00:00:00.000 of the specified date.
* | select date_trunc('week', 2021-01-12 03:04:05.000)	2021-01-11 00:00:00.000	Returns 00:00:00.000 of the Monday of the specified week.
* | select date_trunc('month', 2021-01-12 03:04:05.000)	2021-01-01 00:00:00.000	Returns 00:00:00.000 of the first day of the specified month.
* | select date_trunc('quarter', 2021-01-11 03:04:05.000)	2021-01-01 00:00:00.000	Returns 00:00:00.000 of the first day of the specified quarter.
* | select date_trunc('year', 2021-01-11 03:04:05.000)	2021-01-01 00:00:00.000	Returns 00:00:00.000 of the first day of the specified year.

- Query and analysis examples

  To truncate the average request durations by minute, and group and sort the average request durations by time, execute the following query statement:

```
* | select  date_trunc('minute' ,  __time__)  as time,
       truncate (avg(request_time) ) as avg_time ,
       current_date as date
       group by  time
       order by time  desc
       limit 100
```

  You can use the date_trunc('unit', *x*) function to truncate a time only by second, minute, hour, day, week, month, quarter, or year. To truncate a time based on specified intervals such as 5 minutes, you must use a GROUP BY clause based on the modulus method.

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

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

Interval functions

You can use interval functions to perform interval-related calculations. For example, you can add or subtract an interval based on a date, or calculate the interval between two dates.

Function	Description	Example

Function	Description	Example
date_add(*unit*, *N*,*timstamp*)	Adds N units to a `timestamp`. To subtract an interval, set the value of *N* to a negative value.	`*| select date_add('day', -7, '2018-08-09 00:00:00')` Indicates seven days before August 9, 2018 (2018-08-02 00:00:00.000).
date_diff(*unit*, *timestamp1*, *timestamp2*)	Returns the time difference between two time expressions. For example, you can calculate the difference between `timestamp1` and `timestamp2` by unit.	`*| select date_diff('day', '2018-08-02 00:00:00', '2018-08-09 00:00:00')`

The following table describes the valid values of the unit parameter.

unit	Description
millisecond	milliseconds
second	seconds
minute	minutes
hour	hours
day	days
week	weeks
month	months
quarter	quarters
year	years

Time series padding function

You can use the time_series() function to add the missing data when you query in the time window.

Notice You must use the time_series() function together with GROUP BY and ORDER BY clauses. You cannot use the DESC keyword in an ORDER BY clause to sort data returned in descending order.

- Syntax

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

- Parameters

Parameter	Description
time_column	The sequence of time (KEY), for example, __time__. The value of this parameter can be a long datetime or timestamp expression.

Parameter	Description
window	The size of the window. Valid units: s (seconds), m (minutes), h (hours), and d (days). Examples: 2h, 5m, and 3d.
format	The format in which you want the function to return the value.
padding_data	The content that you want to add. Valid values: ○ 0: adds 0. ○ null: adds null. ○ last: adds the value of the previous point in time. ○ next: adds the value of the next point in time. ○ avg: adds the average of the value of the previous point in time and the value of the next point in time.

- Example

  To add missing data by 2 hours, execute the following query statement:

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

time	num
2021-07-20 00:00:00	11602
2021-07-20 02:00:00	63089
2021-07-20 04:00:00	36583
2021-07-20 06:00:00	11135
2021-07-20 08:00:00	62746
2021-07-20 10:00:00	18314

4.8.9. URL functions

This topic describes the syntax of URL functions. This topic also provides examples on how to use the functions.

URL functions extract fields from standard URLs. The following example shows the format of a URL:

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

The following table describes common URL functions.

Function	Description	Example	
		Query statement	Query result
`url_extract_fragment(url)`	Extracts the fragment from a URL. The return value is of the varchar type.	`*| select url_extract_fragment('https://sls.console.aliyun.com/#/project/dashboard-demo/categoryList')`	`/project/dashboard-demo/categoryList`

Function	Description	Example	
		Query statement	Query result
`url_extract_host(url)`	Extracts the host from a URL. The return value is of the varchar type.	`*|select url_extract_host('http://www.aliyun.com/product/sls')`	`www.aliyun.com`
`url_extract_parameter(url, name)`	Extracts the value of a specified parameter in the query string from a URL. The return value is of the varchar type.	`*|select url_extract_parameter('http://www.aliyun.com/product/sls?userid=testuser','userid')`	`testuser`
`url_extract_path(url)`	Extracts the path from a URL. The return value is of the varchar type.	`*|select url_extract_path('http://www.aliyun.com/product/sls?userid=testuser')`	`/product/sls`
`url_extract_port(url)`	Extracts the port number from a URL. The return value is of the bigint type.	`*|select url_extract_port('http://www.aliyun.com:80/product/sls?userid=testuser')`	`80`
`url_extract_protocol(url)`	Extracts the protocol from a URL. The return value is of the varchar type.	`*|select url_extract_protocol('http://www.aliyun.com:80/product/sls?userid=testuser')`	`http`
`url_extract_query(url)`	Extracts the query string from a URL. The return value is of the varchar type.	`*|select url_extract_query('http://www.aliyun.com:80/product/sls?userid=testuser')`	`userid=testuser`
`url_encode(value)`	Encodes a URL.	`*|select url_encode('http://www.aliyun.com:80/product/sls?userid=testuser')`	`http%3a%2f%2fwww.aliyun.com%3a80%2fproduct%2fsls%3fuserid%3dtestuser`
`url_decode(value)`	Decodes a URL.	`*|select url_decode('http%3a%2f%2fwww.aliyun.com%3a80%2fproduct%2fsls%3fuserid%3dtestuser')`	`http://www.aliyun.com:80/product/sls?userid=testuser`

4.8.10. Regular expression functions

This topic describes the available regular expression functions. You can use these functions when you query and analyze data in Log Service.

A regular expression function parses a string and returns the required substrings.

The following table lists common regular expression functions.

Function	Description	Example
`regexp_extract_all(string, pattern)`	Returns an array where each element is a substring that matches the regular expression. These substrings derive from the specified string.	The returned result of `*| SELECT regexp_extract_all('5a 67b 890m', '\d+')` is `['5','67','890']`. The returned result of `*|SELECT regexp_extract_all('5a 67a 890m', '(\d+)a')` is `['5a','67a']`.
`regexp_extract_all(string, pattern, group)`	Returns an array where each element is a part of a substring that matches the regular expression. This part is the content in the group parameter value of the `()` of a substring that derives from the specified string.	The returned result of `*| SELECT regexp_extract_all('5a 67a 890m', '(\d+)a',1)` is `['5','67']`.
`regexp_extract(string, pattern)`	Returns the first substring of the specified string that matches the regular expression.	The returned result of `*|SELECT regexp_extract('5a 67b 890m', '\d+')` is `'5'`.
`regexp_extract(string, pattern,group)`	Returns a part of the first substring that matches the regular expression. This part is the content in the group parameter value of the `()` of the substring that derives from the specified string.	The returned result of `*|SELECT regexp_extract('5a 67b 890m', '(\d+)([a-z]+)',2)` is `'a'`.
`regexp_like(string, pattern)`	Returns a Boolean value. If the string and its substrings cannot match the regular expression, the value False is returned.	The returned result of `*|SELECT regexp_like('5a 67b 890m', '\d+m')` is True.
`regexp_replace(string, pattern, replacement)`	Replaces the substrings of the specified string that match the regular expression with the value of the replacement parameter.	The returned result of `*|SELECT regexp_replace('5a 67b 890m', '\d+','a')` is `'aa ab am'`.
`regexp_replace(string, pattern)`	Removes the substrings of the specified string that match the regular expression. This function is equivalent to `regexp_replace(string,pat term,'')`.	The returned result of `*|SELECT regexp_replace('5a 67b 890m', '\d+')` is `'a b m'`.
`regexp_split(string, pattern)`	Returns an array where each element is a substring of the specified string that is split based on the regular expression.	The returned result of `*|SELECT regexp_split('5a 67b 890m', '\d+')` is `['a','b','m']`.

4.8.11. JSON functions

This topic describes the syntax of JSON functions. This topic also provides examples on how to use the functions.

Note

- If a string fails to be parsed into JSON data, null is returned.
- In analytic statements of Log Service, a JSON array that is enclosed in single quotation marks ('') indicates a string.
- If the value of a log field is of the JSON type and needs to be expanded to multiple rows, we recommend that you use UNNEST clauses. For more information, see UNNEST function.

json_parse() function

The json_parse() function is used to convert a string to JSON data. The returned result is of the JSON type.

- Syntax

```
json_parse(string)
```

- Example

  Convert the [1, 2, 3] string to the [1,2,3] JSON array.

```
 * | SELECT json_parse('[1, 2, 3]')
```

  The returned result is [1,2,3].

json_format()

The json_format() function is used to convert JSON data to a string. The returned result is a string.

- Syntax

```
json_format(json)
```

- Example

  Convert the [1,2,3] JSON array to the [1, 2, 3] string.

```
* | SELECT json_format(json_parse('[1, 2, 3]'))
```

  The returned result is [1,2,3].

json_array_contains()

The json_array_contains() function is used to check whether a JSON array or a JSON string contains a specified value. The returned result is true or false.

- Syntax

```
json_array_contains(json , value)
```

- Examples
  - Check whether the [1, 2, 3] JSON array contains 2.

```
* | SELECT json_array_contains(json_parse('[1, 2, 3]'), 2)
```

    The returned result is true.

  - Check whether the [1, 2, 3] JSON string contains 2.

```
* | SELECT json_array_contains('[1, 2, 3]', 2)
```

    The returned result is true.

json_array_get()

The json_array_get() function is used to extract the element that corresponds to the subscript of a JSON array.

- Syntax

```
json_array_get(json_array, index)
```

- Example

  Extract the element that corresponds to the subscript 0 of the ["status", "request_time", "request_method"] JSON array.

```
* | SELECT json_array_get('["status", "request_time", "request_method"]', 0)
```

  The returned result is status.

json_array_length()

The json_array_length() function is used to calculate the number of elements in a JSON array.

- Syntax

```
json_array_length(json array)
```

- Example

  Calculate the number of the elements in the ["status", "request_time", "request_method"] JSON array.

```
* | SELECT json_array_length('["status", "request_time", "request_method"]')
```

  The returned result is 3.

json_extract()

The json_extract() function is used to extract the value of a specified field from a JSON object. The returned result is of the JSON type.

> **Note** If the JSON data is invalid when you use the json_extract() function, an error message appears. We recommend that you use the json_extract_scalar() function.

- Syntax

```
json_extract(json, json_path)
```

  The format of json_path is `$.store.book[0].title`.

- Examples
  - Extract the value of the status field from the content field. The content field is a JSON object.

```
* | SELECT json_extract(content, '$.status')
```

    The returned result is the value of the status field, for example, **"200"**.

  - Expand the value of the request_time field and use row to represent the expanded rows. The value of the request_time field is a JSON array. Then, extract and calculate the sum of the values of the status field from the rows.

```
* | select sum(cast (json_extract_scalar(row, '$.status') as bigint) ) from  log, unnest(cast(jso
n_parse(request_time)  as array(json) ) as  t(row)
```

    The returned result is the sum result.

json_extract_scalar()

The json_extract_scalar() function is used to extract the value of a specified field from a JSON object. The returned result is a string.

- Syntax

```
json_extract_scalar(json, json_path)
```

  The format of json_path is `$.store.book[0].title`.

- Examples
  - Extract the value of the status field from the content field. The content field is a JSON object.

```
* | SELECT json_extract_scalar(content, '$.status')
```

    The returned result is the value of the status field, for example, **"200"**.
  - Extract the value of the status field from the content field. The content field is a JSON object. Then, convert the value to the bigint type and calculate the sum.

```
* | select sum( cast (json_extract_scalar(content, '$.status') as bigint) )
```

    The returned result is the sum result.

json_size()

The json_size() function is used to calculate the number of elements in a JSON object or JSON array.

- Syntax

```
json_size(json,json_path)
```

- Example

  Calculate the number of elements in the status field.

```
* | SELECT json_size('{"status":[1, 2, 3]}','$.status')
```

  The returned result is 3.

4.8.12. Type conversion functions

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

You can use the index attribute feature of Log Service to set the data type of a field to LONG, DOUBLE, TEXT, or JSON. You can also query 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 an index into the data type that you use in a query.

Syntax

> **Note** We recommend that you use the try_cast() function if a log contains dirty data. Otherwise, a query may fail due to the dirty data.

- Convert the data type of a column of values or a specific value into the specified type in a query. If the data type of a value fails to be converted, the query is terminated.

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

- Convert the data type of a column of values or a specific value into the specified type in a query. If the data type of a 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 a field whose value data type is to be converted.
value	The field value whose data type is to be converted into the specified type.

Example

- To convert the numeric value 123 to a value of the VARCHAR type, use the following statement:

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

- To convert the data type of the uid field values to the VARCHAR type, use the following statement:

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

4.8.13. IP functions

IP functions can be used to identify whether an IP address is an internal or external IP address. IP functions can also be used to identify the country, state, and city to which an IP address belongs. This topic describes the syntax of IP functions and provides examples on how to use the functions.

> **Note** The KEY parameter in the following functions indicates a log field, for example, client_ip. The value of this parameter is an IP address.

Function	Description	Example
ip_to_domain(KEY)	Checks whether an IP address is an internal IP address or an external IP address. The returned result is **intranet** or **internet**. • **intranet**: an internal IP address. • **internet**: an external IP address.	`* | SELECT ip_to_domain(client_ip)`
ip_to_country(KEY)	Identifies the country or the region to which an IP address belongs. The returned result is the Chinese name of a country or a region.	`* | SELECT ip_to_country(client_ip)`
ip_to_country(KEY,'en')	Identifies the country or the region to which an IP address belongs. The returned result is the code of a country or a region.	`* | SELECT ip_to_country(client_ip,'en')`
ip_to_country_code(KEY)	Identifies the country or the region to which an IP address belongs. The returned result is the code of a country or a region.	`* | SELECT ip_to_country_code(client_ip)`

Function	Description	Example
ip_to_province(KEY)	Identifies the state to which an IP address belongs. The returned result is the Chinese name of a state.	`* | SELECT ip_to_province(client_ip)`
ip_to_province(KEY,'en')	Identifies the state to which an IP address belongs. The returned result is the administrative region code of a state.	`* | SELECT ip_to_province(client_ip,'en')`
ip_to_city(KEY)	Identifies the city to which an IP address belongs. The returned result is the Chinese name of a city.	`* | SELECT ip_to_city(client_ip)`
ip_to_city(KEY,'en')	Identifies the city to which an IP address belongs. The returned result is the administrative region code of a city.	`* | SELECT ip_to_city(client_ip,'en')`
ip_to_geo(KEY)	Identifies the longitude and latitude of the location to which an IP address belongs. The returned result is in the `latitude,longitude` format. For information about geohash functions, see Geography functions.	`* | SELECT ip_to_geo(client_ip)`
ip_to_city_geo(KEY)	Identifies the longitude and latitude of the city to which an IP address belongs. This function returns the longitude and latitude of a city. Each city has only one set of coordinates. The returned result is in the `latitude,longitude` format.	`* | SELECT ip_to_city_geo(client_ip)`
ip_to_provider(KEY)	Identifies the Internet service provider (ISP) of an IP address. The returned result is the name of an ISP.	`* | SELECT ip_to_provider(client_ip)`

4.8.14. GROUP BY clause

GROUP BY clauses are used together with aggregate functions to group analysis results based on one or more columns.

Syntax

```
* | SELECT column name, aggregate function GROUP BY [ column name | alias | serial number ]
```

> **Note** If you use a GROUP BY clause in an SQL statement, you can perform aggregate calculations on only a column that is specified in the GROUP BY clause or a random column that is not a non-GROUP BY column. For example, `* | SELECT status, request_time, COUNT(*) AS PV GROUP BY status` is an invalid query statement because request_time is not a GROUP BY column.

A GROUP BY clause can be used to group data by column name, alias, and serial number. The following table describes the related parameters.

Parameter	Description
Column name	The name of a log field or the return column of an aggregate function. You can group data by log field name (key) or the result of an aggregate function. A GROUP BY clause supports single column or multiple columns.
Alias	Data is grouped by the alias of a log field name or the return column alias of an aggregate function. You can specify the alias of a log field in the Field Search section of the Search & Analysis panel. For more information, see Field aliases.

Serial number	The serial number of a column in a SELECT statement. The number starts from 1.
	For example, the serial number of the status column is 1. In this case, the following two statements are equivalent:
	`* | SELECT status, count(*) AS PV GROUP BY status`
	`* | SELECT status, count(*) AS PV GROUP BY 1`

Aggregate function	A GROUP BY clause can be used together with aggregate functions such as MIN, MAX, AVG, SUM, and COUNT. For more information, see General aggregate functions.

Examples

- To calculate the number of access requests of different HTTP status codes, you can execute the following query statement:

```
* | SELECT status, count(*) AS PV GROUP BY status
```

- To calculate the number of page views (PVs) by 1 hour, you can execute the following query statement:

```
* | SELECT count(*) AS PV , date_trunc('hour', __time__) AS time GROUP BY time ORDER BY time limit 1000
```

The __time__ field is a reserved field in Log Service. This field indicates the time column. time is the alias of date_trunc('hour', __time__). For more information about the date_trunc() function, see Truncation function.

> **Note**
> - The clause limit 1000 indicates that a maximum of 1,000 rows of data can be returned. If you do not use a LIMIT clause, you can obtain a maximum of 100 rows of data by default.
> - If you turn on Enable Analytics for a log field in the Search & Analysis panel, the analysis feature is automatically enabled for the __time__ field.

- To calculate the number of PVs by 5 minutes, you can execute the following query statement.

The date_trunc() function can only collect statistics at a specified interval. If you want to perform statistical analysis by custom time, you can group data based on the modulus method. In this example, %300 in the following statement indicates that the modulo and truncation operations are performed every 5 minutes.

```
* | SELECT count(*) AS PV, __time__ - __time__%300 AS time GROUP BY time limit 1000
```

- To extract a column that is not grouped by using a GROUP BY clause, you can execute the following query statement.

  If you use a GROUP BY clause in an SQL statement, you can perform aggregate calculations on only a column that is specified in the GROUP BY clause or a random column that is not a non-GROUP BY column. For example, `* | SELECT status, request_time, COUNT(*) AS PV GROUP BY status` is an invalid query statement because request_time is not a GROUP BY column.

  ```
  * | SELECT status, arbitrary(request_time), count(*) AS PV GROUP BY status
  ```

4.8.15. Window functions

This topic describes the syntax of window functions.

Window functions are used to perform calculations across rows of a log. 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 fill 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
```

`rank() OVER (PARTITION BY KEY2 ORDER BY value DESC)` indicates that PARTITION BY is first used to partition by KEY2 and sort by the value if KEY2 is the same, and then the rank() function is used to aggregate data.

Special aggregate functions used in windows

Function	Description
rank()	Returns the rank of a value within a group of values. The rank is one plus the number of preceding rows that are not peers of the current row.
row_number()	Returns a unique, sequential number for each row.
first_value(x)	Returns the first value in the window. In most cases, the function is used to sort all values in a window and then return the maximum value.
last_value(x)	Returns the last value in the window. In most cases, the function is used to sort all values in a window and then return the minimum value.
nth_value(x, offset)	Returns the value at the specified offset from the beginning of the window.
lead(x,offset,defaut_value)	Returns the value in offset rows that follow the current row in the window. If the target row does not exist, the default_value is returned.

Function	Description
lag(x,offset,defaut_value)	Returns the value in offset rows that precede the current row in the window. If the target row does not exist, the default_value is returned.

Examples

- To rank the salaries of employees in their departments, execute the following query statement:

```
* | select department, persionId, sallary , rank() over(PARTITION BY department order by sallary de
sc) as sallary_rank  order by department,sallary_rank
```

Query and analysis result

department	persionId	sallary	sallary_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

- To calculate the percentages of salaries of employees in their departments, execute the following query statement:

```
* | select department, persionId, sallary *1.0 / sum(sallary) over(PARTITION BY department ) as sal
lary_percentage
```

Query and analysis result

department	persionId	sallary	sallary_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

- To calculate the daily UV increase over the previous day, execute the following query statement:

```
* | 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
)
```

Query and analysis result

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

4.8.16. HAVING clause

This topic describes the syntax of HAVING clauses.

The query and analysis feature of Log Service supports the standard SQL HAVING clause. A HAVING clause is used together with a GROUP BY clause to filter GROUP BY results.

The following example shows the syntax of a HAVING clause:

```
method :PostLogstoreLogs |select avg(latency),projectName group by projectName  having avg(latency) >
100
```

Difference between HAVING and WHERE clauses

A HAVING clause is used to filter the aggregation and calculation results after you use a GROUP BY clause. A WHERE clause is used to filter the raw data during aggregation.

Example

To calculate the average rainfall of each province in which the temperature is higher than 10°C, and return only the provinces in which the average rainfall is greater than 100 ml, execute the following query statement:

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

4.8.17. ORDER BY clause

This topic describes the syntax of ORDER BY clauses.

An ORDER BY clause is used to sort query results based on only one column.

- Syntax

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

- Example

```
method :PostLogstoreLogs |select avg(latency) as avg_latency,projectName group by projectName
HAVING avg(latency) > 5700000
order by avg_latency desc
```

4.8.18. LIMIT syntax

The LIMIT clause is used to limit the number of returned rows.

Syntax formats

Log Service supports the following LIMIT syntax formats:

- Reads only the first N rows:

```
limit N
```

- Reads N rows starting from the S-th row:

```
limit S , N
```

> **Note**
>
> - If you use the LIMIT clause to paginate results, the final results rather than the intermediate results of the SQL query are obtained.
> - You cannot apply the LIMIT clause to subqueries. For example, the following statement is not supported:
>
>   ```
>   * | select count(1) from ( select distinct(url) from limit 0,1000)
>   ```
>
> - If you use the LIMIT clause for pagination, the offset value cannot exceed 1,000,000. For example, in the `limit S , N` clause, the sum of S and N cannot exceed 1,000,000, and the value of N cannot exceed 10,000.

Example

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

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

- To obtain a total of 1,000 results from row 0 to row 999, run the following statement.

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

- To obtain a total of 1,000 results from row 1,000 to row 1,999, run the following statement:

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

4.8.19. Conditional expressions

This topic describes the syntax of conditional expressions and provides examples on how to use conditional expressions.

CASE WHEN statement

CASE WHEN statements are used to classify data.

- Syntax

```
CASE WHEN condition1 THEN result1
     [WHEN condition2 THEN result2]
     [ELSE result3]
END
```

- Examples
  - Extract browser information from the value of the http_user_agent field, classify the information into Chrome, Safari, and unknown types, and then calculate the number of page views (PVs) for the three types.
    - Query statement

```
* | SELECT CASE
 WHEN http_user_agent like '%Chrome%' then 'Chrome'
 WHEN http_user_agent like '%Safari%' then 'Safari'
 ELSE 'unknown'
 END AS http_user_agent,
    count(*) AS pv
    GROUP BY http_user_agent
```

    - Query and analysis result

http_user_agent	pv
Chrome	5563
Safari	1842
unknown	1666

  - Query the distribution of requests that are sent at different points in time.
    - Query statement

```
* | SELECT
 CASE
 WHEN request_time < 10 then 't10'
 WHEN request_time < 100 then 't100'
 WHEN request_time < 1000 then 't1000'
 WHEN request_time < 10000 then 't10000'
 ELSE 'large' END
 AS request_time,
 count(*) AS pv
 GROUP BY request_time
```

    - Query and analysis result

request_time	pv
t100	1563542
large	533

if() function

The if() function is used to classify data. This function is similar to CASE WHEN statements.

- Syntax
  - If the *condition* is true, the *true_value* column is returned. Otherwise, null is returned.

```
if(condition, true_value)
```

- If the *condition* is true, the *true_value* column is returned. Otherwise, the *false_value* column is returned.

  ```
  if(condition, true_value, false_value)
  ```

- Example

  Calculate the ratio of requests whose status code is 200 to all requests.

  - Query statement

    ```
    * | SELECT sum(if(status =200,1,0))*1.0 / count(*) AS status_200_percentage
    ```

  - Query and analysis result

status_200_percentage
0.8846858366766299

coalesce() function

The coalesce() function is used to return the first non-null value in multiple columns.

- Syntax

  ```
  coalesce(expression1, expression2, expression3, expression4)
  ```

- Example

  Calculate the ratio of the expenses of yesterday to the expenses of the same day last month.

  - Query statement

    ```
    * | SELECT compare("expenses of yesterday", 604800) AS diff FROM (SELECT coalesce(sum(PretaxAmoun
    t), 0) AS "expenses of yesterday" FROM website_log)
    ```

  - Query and analysis result

diff
[6514393413.0,19578267596.0,0.33273594719539659]

    - The value 6514393413.0 indicates the expenses of yesterday.
    - The value 19578267596.0 indicates the expenses of the same day last month.
    - The value 0.33273594719539659 indicates the ratio of the expenses of yesterday to the expenses of the same day last month.

nullif() function

The nullif() function is used to check whether the values of two columns are the same. If the values are the same, null is returned. Otherwise, the value of expression1 is returned.

- Syntax

  ```
  nullif(expression1, expression2)
  ```

- Example

  Check whether the values of the client_ip and host fields are the same.

  - Query statement

    ```
    * | SELECT nullif(client_ip,host)
    ```

- Query and analysis result

  If the values of the client_ip and host fields are different, the value of the client_ip field is returned.

try() function

The try() function is used to capture errors to ensure that Log Service can continue to query and analyze data.

- Syntax

```
try(expression)
```

- Example

  If an error occurs when the regexp_extract function is invoked, the try() function captures the error and Log Service continues to query and analyze data. The query and analysis result is returned.

  - Query statement

```
* | SELECT try(regexp_extract(request_uri, '.*\/(file.*)', 1)) AS file, count(*) AS count GROUP B
Y file
```

  - Query and analysis result

file	count
file-5	851
file-7	928
file-3	837
file-4	863

4.8.20. Nested subquery

This topic describes how to use nested subqueries when you query logs.

You can use nested queries to perform more complicated queries.

You must specify a FROM clause in the SQL statement of each nested query. However, this rule does not apply to non-nested queries. You must specify the `from log` keyword in each SQL statement to read raw data from logs.

Example:

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

4.8.21. Array functions and operators

This topic describes the syntax of array functions and operators. This topic also provides examples on how to use the functions and operators.

The following table describes the array functions and operators that are supported by Log Service.

> **Notice** If you want to use strings in analytic statements, you must enclose the strings in single quotation marks (''). Strings that are not enclosed or are enclosed in double quotation marks ("") indicate field names or column names. For example, 'status' indicates the status string, and status or "status" indicates the status log field.

Function	Syntax	Description
Subscript operator	[*x*]	Returns the element whose index is *x* from an array. This operator is equivalent to the element_at function.
array_agg function	array_agg(*x*)	Returns an array that consists of all values in *x*.
array_distinct function	array_distinct(*x*)	Removes duplicate elements from an array.
array_except function	array_except(*x*, *y*)	Returns the difference between two arrays.
array_intersect function	array_intersect(*x*, *y*)	Returns the intersection of two arrays.
array_join function	array_join(*x*, *delimiter*)	Concatenates the elements in an array into a string by using a specified delimiter. If the array contains a null element, the null element is ignored. **Notice** The array_join function can return a maximum of 1 KB of data. If the size of the returned data exceeds 1 KB, the excess data is truncated.
	array_join(*x*, *delimiter*, *null_replacement*)	Concatenates the elements in an array into a string by using a specified delimiter. If the array contains a null element, the null element is replaced by the value of the *null_replacement* parameter. **Notice** The array_join function can return a maximum of 1 KB of data. If the size of the returned data exceeds 1 KB, the excess data is truncated.
array_max function	array_max(*x*)	Returns the maximum value in an array.
array_min function	array_min(*x*)	Returns the minimum value in an array.

Function	Syntax	Description
array_position function	array_position(*x*, *element*)	Returns the index of a specified element in an array. The index starts from 1. If the specified element does not exist, the function returns 0.
array_remove function	array_remove(*x*, *element*)	Removes a specified element from an array.
array_sort function	array_sort(*x*)	Sorts the elements in an array in ascending order. If the array contains a null element, the null element is placed at the end.
array_transpose function	array_transpose(*x*)	Transposes a matrix and returns a new two-dimensional array that consists of the elements in the matrix. The elements are located by using the same indexes.
array_union function	array_union(*x*, *y*)	Returns the union of two arrays.
cardinality function	cardinality(*x*)	Returns the number of elements in an array.
concat function	concat(*x*, *y*...)	Concatenates multiple arrays into one array.
contains function	contains(*x*, *element*)	Checks whether an array contains a specified element. If the array contains the specified element, the function returns true.
element_at function	element_at(*x*, *y*)	Returns the element whose index is *y* from an array.
filter function	filter(*x*, *lambda_expression*)	Filters elements in an array based on a lambda expression and returns elements that match the lambda expression.
flatten function	flatten(*x*)	Transforms a two-dimensional array into a one-dimensional array.
reduce function	reduce(*x*, *lambda_expression*)	Returns the sum of the elements in an array based on a lambda expression.
reverse function	reverse(*x*)	Reverses the elements in an array.
sequence function	sequence(*x*, *y*)	Returns an array of elements within a specified range. The elements are consecutive and incremental. The default incremental step is 1.
	sequence(*x*, *y*, *step*)	Returns an array of elements within a specified range. The elements are consecutive and incremental. The incremental step is a custom value.
shuffle function	shuffle(*x*)	Shuffles the elements in an array.
slice function	slice(*x*, *start*, *length*)	Returns a subset of an array.

Function	Syntax	Description
transform function	transform(*x*, *lambda_expression*)	Transforms each element in an array by using a lambda expression.
zip function	zip(*x*, *y*...)	Merges multiple arrays into a two-dimensional array. Elements that have the same index in the input arrays form a new array in the two-dimensional array.
zip_with function	zip_with(*x*, *y*, *lambda_expression*)	Merges two arrays into a single array by using a lambda expression.

Subscript operator

The subscript operator is used to return the element whose index is *x* from an array. This operator is equivalent to the element_at function.

- Syntax

```
[x]
```

- Parameters

Parameter	Description
x	The index of an element in an array. The index starts from 1. The value of this parameter is of the bigint type.

- Return value type

  The data type of the specified element.

- Example

  Obtain the first element from the value of the number field.

  - Sample field

    ```
    number:[49,50,45,47,50]
    ```

  - Query statement

    ```
    * | SELECT cast(json_parse(number) as array(bigint)) [1]
    ```

  - Query and analysis result

array_agg function

The array_agg function is used to return an array that consists of all values in *x*.

- Syntax

```
array_agg (x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of an arbitrary data type.

- Return value type

  The array type.

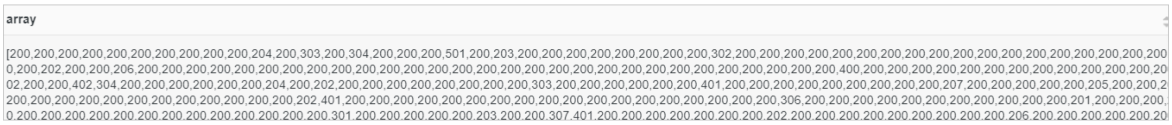
- Example

  Obtain an array that consists of all values in the status field.

  - Query statement

    ```
    * | SELECT array_agg(status) AS array
    ```

  - Query and analysis result

array_distinct function

The array_distinct function is used to remove duplicate elements from an array.

- Syntax

  ```
  array_distinct(x)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Remove duplicate elements from the value of the number field.

  - Sample field

    ```
    number:[49,50,45,47,50]
    ```

  - Query statement

    ```
    *| SELECT array_distinct(cast(json_parse(number) as array(bigint)))
    ```

  - Query and analysis result

array_except function

The array_except function is used to return the difference between two arrays.

- Syntax

  ```
  array_except(x, y)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Obtain the difference between the [1,2,3,4,5] and [1,3,5,7] arrays.

  - Query statement

    ```
    * | SELECT array_except(array[1,2,3,4,5],array[1,3,5,7])
    ```

  - Query and analysis result

array_intersect function

The array_intersect function is used to return the intersection of two arrays.

- Syntax

  ```
  array_intersect(x, y)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Obtain the intersection of the [1,2,3,4,5] and [1,3,5,7] arrays.

  - Query statement

    ```
    * | SELECT array_intersect(array[1,2,3,4,5],array[1,3,5,7])
    ```

  - Query and analysis result

array_join function

The array_join function is used to concatenate the elements of an array into a string by using a specified delimiter.

- Syntax
  - The following syntax of the array_join function is used to concatenate the elements of an array into a string by using a specified delimiter. If the array contains a null element, the null element is ignored.

    ```
    array_join(x, delimiter)
    ```

  - The following syntax of the array_join function is used to concatenate the elements of an array into a string by using a specified delimiter. If the array contains a null element, the null element is replaced by the value of the *null_replacement* parameter.

    ```
    array_join(x, delimiter,null_replacement)
    ```

- Parameters

Parameter	Description
x	The value of this parameter is of an arbitrary array type.
delimiter	The delimiter that is used to connect elements. You can specify a string for this parameter.
null_replacement	The string that is used to replace a null element.

- Return value type

  The varchar type.

- Example

  Concatenate the elements of the [null, 'Log','Service'] array into a string by using space characters and replace the null element with Alicloud.

  - Query statement

    ```
    * | SELECT array_join(array[null,'Log','Service'],' ','Alicloud')
    ```

  - Query and analysis result

array_max function

The array_max function is used to return the maximum value in an array.

- Syntax

  ```
  array_max(x)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type. **Notice** If an array contains a null element, the function returns null.

- Return value type

The data type of elements in the parameter value.

- Example

  Obtain the maximum value in an array.

  - Sample field

    ```
    number:[49,50,45,47,50]
    ```

  - Query statement

    ```
    *| SELECT array_max(try_cast(json_parse(number) as array(bigint))) AS max_number
    ```

  - Query and analysis result

max_number
50

array_min function

The array_min function is used to return the minimum value in an array.

- Syntax

  ```
  array_min(x)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type. **Notice** If an array contains a null element, the function returns null.

- Return value type

  The data type of elements in the parameter value.

- Example

  Obtain the minimum value in an array.

  - Sample field

    ```
    number:[49,50,45,47,50]
    ```

  - Query statement

    ```
    *| SELECT array_min(try_cast(json_parse(number) as array(bigint))) AS min_number
    ```

  - Query and analysis result

min_number
45

array_position function

The array_position function is used to return the index of a specified element in an array. The index starts from 1. If the specified element does not exist, the function returns 0.

- Syntax

```
array_position(x, element)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
element	The value of this parameter is the element whose index you want to obtain. **Note** If the element is null, the function returns null.

- Return value type

  The bigint type.

- Example

  Obtain the index of 45 from the [49,45,47] array.

  - Query statement

```
* | SELECT array_position(array[49,45,47],45)
```

  - Query and analysis result

_col0
2

array_remove function

The array_remove function is used to remove a specified element from an array.

- Syntax

```
array_remove(x, element)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
element	The value of this parameter is the element that you want to remove. **Note** If the element is null, the function returns null.

- Return value type

  The array type.

- Example

  Remove 45 from the [49,45,47] array.

  - Query statement

```
* | SELECT array_remove(array[49,45,47],45)
```

- Query and analysis result

array_sort function

The array_sort function is used to sort the elements in an array in ascending order. If the array contains a null element, the null element is placed at the end.

- Syntax

```
array_sort(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Sort the elements in the ['b', 'd', null, 'c', 'a'] array in ascending order.

  - Query statement

```
* | SELECT array_sort(array['b','d',null,'c','a'])
```

  - Query and analysis result

array_transpose function

The array_transpose function is used to transpose a matrix and return a new two-dimensional array that consists of the elements in the matrix. The elements are located by using the same indexes.

- Syntax

```
array_transpose(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array(double) type.

- Return value type

  The array(double) type.

- Example

  Create a two-dimensional array from elements that are located by using the same indexes in a different two-dimensional array. For example, in the [0,1,2,3], [10,19,18,17], and [0,9,8,7] arrays, 0, 10, and 9 are all located by using the index 1. This way, the new array [0.0,10.0,9.0] is formed.

- Query statement

```
* | SELECT array_transpose(array[array[0,1,2,3],array[10,19,18,17],array[9,8,7]])
```

- Query and analysis result

array_union function

The array_union function is used to return the union of two arrays.

- Syntax

```
array_union(x, y)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Obtain the union of the [1,2,3,4,5] and [1,3,5,7] arrays.

  - Query statement

```
* | SELECT array_union(array[1,2,3,4,5],array[1,3,5,7])
```

  - Query and analysis result

cardinality function

The cardinality function is used to return the number of elements in an array.

- Syntax

```
cardinality(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

- Return value type

  The bigint type.

- Example

Obtain the number of elements in the value of the number field.

- Sample field

```
number:[49,50,45,47,50]
```

- Query statement

```
*| SELECT cardinality(cast(json_parse(number) as array(bigint)))
```

- Query and analysis result

concat function

The concat function is used to concatenate multiple arrays into one array.

- Syntax

```
concat(x, y…)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Concatenate the ['red','blue'] and ['yellow','green'] arrays into one array.

  - Query statement

```
* | SELECT concat(array['red','blue'],array['yellow','green'])
```

  - Query and analysis result

contains function

The contains function is used to check whether an array contains a specified element. If the array contains the specified element, the function returns true.

- Syntax

```
contains(x, element)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

Parameter	Description
element	The value of this parameter is the element that you want to check.

- Return value type

  The Boolean type.

- Example

  Check whether the value of the region field contains cn-beijing.

  - Sample field

    ```
    region:["cn-hangzhou","cn-shanghai","cn-beijing"]
    ```

  - Query statement

    ```
    *| SELECT contains(cast(json_parse(region) as array(varchar)),'cn-beijing')
    ```

  - Query and analysis result

element_at function

The element_at function is used to return the element whose index is *y* from an array.

- Syntax

  ```
  element_at(x, y)
  ```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The index of an element in an array. The index starts from 1. The value of this parameter is of the bigint type.

- Return value type

  An arbitrary data type.

- Example

  Obtain the second element from the value of the number field.

  - Sample field

    ```
    number:[49,50,45,47,50]
    ```

  - Query statement

    ```
    * |
    SELECT
      element_at(cast(json_parse(number) AS array(varchar)), 2)
    ```

- Query and analysis result

filter function

The filter function is used to filter elements in an array based on a lambda expression and return elements that match the lambda expression.

- Syntax

```
filter(x, lambda_expression)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
lambda_expression	The lambda expression. For more information, see Lambda functions.

- Return value type

  The array type.

- Example

  Obtain the elements that are greater than 0 from the [5,-6,null,7] array by using the lambda expression `x -> x > 0`.

  - Query statement

    ```
    * | SELECT filter(array[5,-6,null,7],x -> x > 0)
    ```

  - Query and analysis result

flatten function

The flatten function is used to transform a two-dimensional array into a one-dimensional array.

- Syntax

```
flatten(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Transform the two-dimensional array [array[1,2,3,4],array[5,2,2,4] into a one-dimensional array.

- Query statement

```
* | SELECT flatten(array[array[1,2,3,4],array[5,2,2,4]])
```

- Query and analysis result

reduce function

The reduce function is used to return the sum of the elements in an array based on a lambda expression.

- Syntax

```
reduce(x, lambda_expression)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
lambda_expression	The lambda expression. For more information, see Lambda functions.

- Return value type

  The bigint type.

- Example

  Obtain the sum of the elements in the [5,20,50] array.

  - Query statement

```
* | SELECT reduce(array[5,20,50],0,(s, x) -> s + x, s -> s)
```

  - Query and analysis result

reverse function

The reverse function is used to reverse the elements in an array.

- Syntax

```
reverse(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

Reverse the elements in the [1,2,3,4,5] array.

- Query statement

```
* | SELECT reverse(array[1,2,3,4,5])
```

- Query and analysis result

sequence function

The sequence function is used to return an array of elements within a specified range. The elements are consecutive and incremental.

- Syntax
  - The following syntax of the sequence function uses the default incremental step. The default incremental step is 1.

```
sequence(x, y)
```

  - The following syntax of the sequence function uses a custom incremental step:

```
sequence(x, y, step)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the bigint or timestamp type. UNIX timestamps and date and time expressions are supported.
y	The value of this parameter is of the bigint or timestamp type. UNIX timestamps and date and time expressions are supported.
step	The incremental step. If the values of the x and y parameters are date and time expressions, the value of the *step* parameter is in one of the following formats: `interval ' n' year to month`: The incremental step is n years. `interval 'n' day to second`: The incremental step is n days.

- Return value type

  The array type.

- Examples
  - Example 1: Obtain the even numbers within the range from 0 to 10.
    - Query statement

```
* | SELECT sequence(0,10,2)
```

    - Query and analysis result

- Example 2: Obtain the dates within the range from 2017-10-23 to 2021-08-12 at the incremental step of 1 year.
  - Query statement

```
ww* | SELECT  sequence(from_unixtime(1508737026),from_unixtime(1628734085),interval '1' year to month )
```

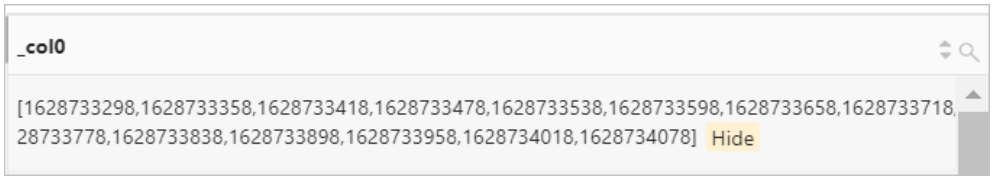
  - Query and analysis result

```
["2017-10-23 13:37:06.000","2018-10-23 13:37:06.000","2019-10-23 13:37:06.000","2020-10-23 13:37:0
00"]
```

- Example 3: Obtain the UNIX timestamps within the range from 1628733298 to 1628734085 at the incremental step of 60 seconds.
  - Query statement

```
* | SELECT  sequence(1628733298,1628734085,60)
```

  - Query and analysis result



shuffle function

The shuffle function is used to shuffle the elements in an array.

- Syntax

```
shuffle(x)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Shuffle the elements in the [1,2,3,4,5] array.

  - Query statement

```
*| SELECT shuffle(array[1,2,3,4,5])
```

  - Query and analysis result

slice function

The slice function is used to return a subset of an array.

- Syntax

```
slice(x, start, length)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
start	The index at which Log Service starts to extract elements. ○ If the value of the *start* parameter is negative, Log Service starts to extract elements from the end of the array. ○ If the value of the *start* parameter is a positive number, Log Service starts to extract elements from the beginning of the array.
length	The number of elements that you want to include in the subset.

- Return value type

  The array type.

- Example

  Obtain a subset of the [1,2,4,5,6,7,7] array from the third element with two elements.

  ○ Query statement

  ```
  * | SELECT slice(array[1,2,4,5,6,7,7],3,2)
  ```

  ○ Query and analysis result

transform function

The transform function is used to transform each element in an array by using a lambda expression.

- Syntax

```
transform(x, lambda_expression)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
lambda_expression	The lambda expression. For more information, see Lambda functions.

- Return value type

  The array type.

- Example

  Add 1 to each element in the [5,6] array and return a new array.

- Query statement

```
* | SELECT transform(array[5,6],x -> x + 1)
```

- Query and analysis result

zip function

The zip function is used to merge multiple arrays into a two-dimensional array. Elements that have the same index in the input arrays form a new array in the two-dimensional array.

- Syntax

```
zip(x, y...)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.

- Return value type

  The array type.

- Example

  Merge the [1, 2,3], ['1b', null, '3b'], and [1, 2,3] arrays into a two-dimensional array.

  - Query statement

```
* | SELECT zip(array[1,2,3], array['1b',null,'3b'],array[1,2,3])
```

  - Query and analysis result

zip_with function

The zip_with function is used to merge two arrays into a single array by using a lambda expression.

- Syntax

```
zip_with(x, y, lambda_expression)
```

- Parameters

Parameter	Description
x	The value of this parameter is of the array type.
y	The value of this parameter is of the array type.
lambda_expression	The lambda expression. For more information, see Lambda functions.

- Return value type

  The array type.

- Example

  Use the lambda expression `(x, y) -> x + y` to add the elements in the [1, 2] and [3, 4] arrays and return a new array.

  - Query statement

    ```
    SELECT zip_with(array[1,2], array[3,4],(x,y) -> x + y)
    ```

  - Query and analysis result

4.8.22. Binary string functions

This topic describes the syntax of binary string functions. This topic also provides examples on how to use the functions.

Varbinary data is different from varchar data.

Function	Description
Concatenation operator (`||`)	The result of `a || b` is `ab` .
length(binary)	Returns the length of a binary string in bytes. The return value is of the bigint type.
concat(binary1, ..., binaryN)	Concatenates binary strings. This function is equivalent to ‖. The return value is of the varbinary type.
to_base64(binary)	Converts a binary string to a Base64 string. The return value is of the varchar type.
from_base64(string)	Converts a Base64 string to a binary string. The return value is of the varbinary type.
to_base64url(binary)	Converts a string to a URL-safe Base64 string. The return value is of the varchar type.
from_base64url(string)	Converts a URL-safe Base64 string to a binary string. The return value is of the varbinary type.
to_hex(binary)	Converts a binary string to a hexadecimal string. The return value is of the varchar type.
from_hex(string)	Converts a hexadecimal string to a binary string. The return value is of the varbinary type.
to_big_endian_64(bigint)	Converts a number to a binary string in big endian mode. The return value is of the varbinary type.
from_big_endian_64(binary)	Converts a binary string in big endian mode to a number. The return value is of the bigint type.

Function	Description
md5(binary)	Calculates the MD5 value of a binary string. The return value is of the varbinary type.
sha1(binary)	Calculates the SHA1 value of a binary string. The return value is of the varbinary type.
sha256(binary)	Calculates the SHA256 hash value of a binary string. The return value is of the varbinary type.
sha512(binary)	Calculate the SHA512 value of a binary string. The return value is of the varbinary type.
xxhash64(binary)	Calculates the xxhash64 value of a binary string. The return value is of the varbinary type.

4.8.23. Bitwise functions

This topic describes the syntax of bitwise functions. This topic also provides examples on how to use the functions.

Function	Description	Example
bit_count(x, bits)	Counts the number of 1s in x in two's complement. The x variable is a signed integer that includes the specified number of bits. The return value is of the bigint type.	• `SELECT bit_count(9, 64)` returns 2. • `SELECT bit_count(9, 8)` returns 2. • `SELECT bit_count(-7, 64)` returns 62. • `SELECT bit_count(-7, 8)` returns 6.
bitwise_and(x, y)	Returns the bitwise AND of x and y in two's complement. The return value is of the bigint type.	None
bitwise_not(x)	Returns the bitwise NOT of x in two's complement. The return value is of the bigint type.	None
bitwise_or(x, y)	Returns the bitwise OR of x and y in two's complement. The return value is of the bigint type.	None
bitwise_xor(x, y)	Returns the bitwise XOR of x and y in two's complement. The return value is of the bigint type.	None

4.8.24. Interval-valued comparison and periodicity-valued comparison functions

Log Service supports interval-valued comparison and periodicity-valued comparison functions. You can use these functions to query and analyze log data.

compare() function

The compare() function is used to compare the calculation result of the current time period with the calculation result of a time period N seconds before. You can use this function to perform an interval-valued comparison or periodicity-valued comparison on data.

- Syntax

```
compare(column name,N)
```

> **Note** The compare() function can be used to compare the calculation results of multiple periods of time, for example, compare(*column name*,N1,*N2*,*N3*).

  - column name: the name of the specified column. The value of this parameter must be of the double type or long type.
  - N: the time window. Unit: seconds. Example: 3600 (1 hour), 86400 (one day), or 604800 (one week).

- Response

  The returned result is a JSON array in the following format: [the current value, the value before N seconds, the ratio of the current value to the value of N seconds before, the UNIX timestamp before N seconds]. Example: [1176.0,1180.0,0.9966101694915255,1611504000.0].

- Examples

- Calculate the ratio of the page views (PVs) of the current hour to the PVs of the same time period the day before.

  Set the time range to **1 Hour(Time Frame)** and execute the following query statement. 86400 indicates the current time minus 86400 seconds (one day). log indicates the Logstore name.

  ```
  * | SELECT compare(PV, 86400) FROM (SELECT count(*) AS PV FROM log)
  ```

  The following figure shows the returned result.

_col0
[3337.0,3522.0,0.947473026689381]

  - **3337.0** indicates the PVs of the current 1 hour, for example, Dec 25, 2020, 14:00:00 ~ Dec 25, 2020, 15:00:00.
  - **3522.0** indicates the PVs of the same time period the day before, for example, Dec 24, 2020, 14:00:00 ~ Dec 24, 2020, 15:00:00.
  - **0.947473026689381** indicates the ratio of the PVs of the current hour to the PVs of the same time period the day before.

  To display the analysis result in multiple columns, you can execute the following query statement:

  ```
  * | SELECT diff[1] AS today, diff[2] AS yesterday, diff[3] AS ratio FROM (SELECT compare(PV,86400
  ) AS diff FROM (SELECT count(*) AS PV FROM log))
  ```

  The following figure shows the returned result.

today	yesterday	ratio
3337.0	3522.0	0.947473026689381

  Note

  To compare the data of a specified year or week with the data of the previous year or week, you can use the query statements in the following examples:

  - For example, if you want to calculate the ratio of the PVs of November 2020 to the PVs of November 2019, you can set the time range to **Nov 1, 2020, 00:00~Dec 1, 2020, 00:00**, and execute the following query statement:

    ```
    * | SELECT compare(PV, 31622400) FROM (SELECT count(*) AS PV FROM log)
    ```

  - For example, if you want to calculate the ratio of the PVs of a Tuesday to the PVs of the previous Tuesday, you can set the time range to **Jan 18, 2021, 00:00~Jan 19, 2021, 00:00**, and execute the following query statement:

    ```
    * | SELECT compare(PV, 604800) FROM (SELECT count(*) AS PV FROM log)
    ```

- Calculate the ratio of the PVs of each hour of the current day to the PVs of the same time period the day before and two days before.

  Set the time range to **Today(Time Frame)** and execute the following query statement. 86400 indicates the current time minus 86400 seconds (one day). 172800 indicates the current time minus 172800 seconds (two days). log indicates the Logstore name. date_format(from_unixtime(__time__), '%H:00') indicates the format of the returned time.

  ```
  * | SELECT time, compare(PV, 86400,172800) as diff from (SELECT count(*) as PV, date_format(from_unixtime(__time__), '%H:00') as time from log GROUP BY time) GROUP BY time ORDER BY time
  ```

  The following figure shows the returned result.

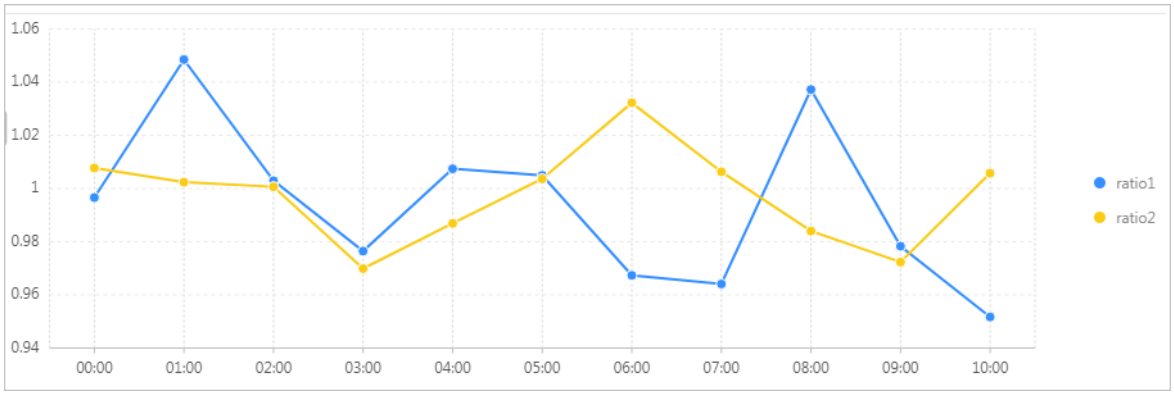
time	diff
00:00	[1176.0,1180.0,1167.0,0.9966101694915255,1.0077120822622108]
01:00	[10077.0,9611.0,10053.0,1.0484861096660077,1.0023873470605789]
02:00	[26921.0,26842.0,26903.0,1.002943148796662,1.0006690703638999]

  - **1176.0** indicates the PVs of the current time period, for example, Dec 25, 2020, 00:00 ~ Dec 25, 2020, 01:00.
  - **1180** indicates the PVs of the same time period the day before, for example, Dec 24, 2020, 00:00 ~ Dec 24, 2020, 01:00.
  - **1167.0** indicates the PVs of the same time period two days before, for example, Dec 23, 2020, 00:00:00 ~ Dec 23, 2020, 01:00:00.
  - **0.9966101694915255** indicates the ratio of the PVs of the current time period to the PVs of the same time period the day before.
  - **1.0077120822622108** indicates the ratio of the PVs of the current period to the PVs of the same period two days before.

  To display the analysis result in multiple columns, you can execute the following query statement:

  ```
  * | SELECT time, diff[1] AS day1, diff[2] AS day2, diff[3] AS day3, diff[4] AS ratio1,  diff[5] AS ratio2 FROM (SELECT time, compare(PV, 86400,172800) as diff from (SELECT count(*) as PV, date_format(from_unixtime(__time__), '%H:00') as time from log GROUP BY time) GROUP BY time ORDER BY time)
  ```

  The following figure shows the returned result.

- Calculate the ratio of the PVs of December to the PVs of November in the same year.

  Set the time range to **This Month(Time Frame)** and execute the following query statement. 2592000 indicates the current time minus 2592000 seconds (one month). log indicates the Logstore name. date_trunc('month', __time__) indicates that the date_trunc function is used to truncate a point in time by month.

  ```
  *| SELECT time, compare(PV, 2592000) AS diff from (SELECT count(*) AS PV, date_trunc('month', __time__) AS time from log GROUP BY time) GROUP BY time ORDER BY time
  ```

  The following figure shows the returned result.

time	diff
2021-01-01 00:00:00.000	[11958378.0,448571.0,26.658829928818404]

ts_compare() function

The ts_compare() function is used to compare the calculation result of the current time period with the calculation result of a time period N seconds before. You can use this function to perform an interval-valued comparison or periodicity-valued comparison on data. The analysis results of the ts_compare() function must be grouped by the time column by using GROUP BY clauses.

- Syntax

  ```
  ts_compare(column name,N)
  ```

  > **Note** The ts_compare() function can be used to compare the calculation results of multiple periods of time, for example, ts_compare(*column name*,N1,*N2*,*N3*).

  - column name: the name of the specified column. The value of this parameter must be of the double type or long type.
  - N: the time window. Unit: seconds. Example: 3600 (1 hour), 86400 (one day), or 604800 (one week).

- Response

  The returned result is a JSON array in the following format: [the current value, the value before N seconds, the ratio of the current value to the value of N seconds before, the UNIX timestamp before N seconds]. Example: [1176.0,1180.0,0.9966101694915255,1611504000.0].

- Example

  Calculate the ratio of the PVs of every hour today to the PVs of the previous hour.

  Set the time range to **Today(Relative)** and execute the following query statement. 3600 indicates the current time minus 3600 seconds (1 hour). log indicates the Logstore name. date_trunc('hour',__time__) indicates that the date_trunc function is used to truncate the time by hour.

  ```
  * | SELECT time, ts_compare(PV, 3600) AS data FROM(SELECT date_trunc('hour',__time__ ) AS time, count(*) AS PV from log GROUP BY time ORDER BY time ) GROUP BY time
  ```

  The following figure shows the returned result.

time	data
2021-01-27 00:00:00.000	[1160.0,10034.0,0.11560693641618497,1611673200.0]
2021-01-27 01:00:00.000	[10177.0,1160.0,8.773275862068966,1611676800.0]
2021-01-27 02:00:00.000	[26804.0,10177.0,2.6337820575808195,1611680400.0]

4.8.25. Comparison functions and operators

This topic describes the comparison functions and operators in Log Service. You can use these functions and operators to query and analyze log data.

A comparison function compares the values of two parameters. The values can be one of the arbitrary comparable data types, such as integer, bigint, double, and text.

Comparison operators

A comparison operator is used to compare two values. If the statement is true, TRUE is returned. Otherwise, FALSE is returned.

Operator	Description
<	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

The BETWEEN operator is used to check whether a value falls in a specified closed interval.

- If the value falls in the specified closed interval, TRUE is returned. Otherwise, FALSE is returned.

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

  The preceding statement is equivalent to `SELECT 3 >= 2 AND 3 <= 6;`.
- The BETWEEN operator can be specified after the NOT operator to check whether a value falls out of a specified closed interval.

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

  The preceding statement is equivalent to `SELECT 3 < 2 OR 3 > 6;`.
- If one of the three values is NULL, the result is NULL.

IS NULL and IS NOT NULL

The IS NULL and IS NOT NULL operators are used to check whether a value is NULL.

IS DISTINCT FROM and IS NOT DISTINCT FROM

The IS DISTINCT FROM and IS NOT DISTINCT FROM operators are similar to the EQUAL TO and NOT EQUAL TO operators. The difference is that the IS DISTINCT FROM and IS NOT DISTINCT FROM operators can be used to check whether a NULL value exists.

Examples:

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

You can use the DISTINCT operator to compare parameter values under multiple conditions. The following table describes the conditions.

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

The GREATEST operator is used to obtain the maximum value from multiple columns. The LEAST operator is used to obtain the minimum value from multiple columns.

Example:

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

Quantified comparison predicates: ALL, ANY, and SOME

The ALL, ANY, and SOME quantifiers are used to check whether a parameter value meets specified conditions.

- ALL is used to check whether a parameter value meets all conditions. If the statement is true, TRUE is returned. Otherwise, FALSE is returned.
- ANY is used to check whether a parameter value meets one of the specified conditions. If the statement is true, TRUE is returned. Otherwise, FALSE is returned.
- SOME is used to check whether a parameter value meets one of the specified conditions. SOME is equivalent to ANY.
- ALL, ANY, and SOME must be specified after comparison operators.

You can use ALL and ANY to compare values under multiple conditions. The following table describes the conditions.

Expression	Description
A = ALL (...)	Returns TRUE if A matches all values.
A <> ALL (...)	Returns TRUE if A does not match all values.
A < ALL (...)	Returns TRUE if A is smaller than the smallest value.
A = ANY (...)	Returns TRUE if A is equal to a value. This statement is equivalent to A IN (...).
A <> ANY (...)	Returns TRUE if A does not match a value.
A < ANY (.....)	Returns TRUE if A is smaller than the largest value.

Examples:

```
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
```

4.8.26. Lambda expressions

Log Service allows you to define a lambda expression in an analytic statement and pass the expression to a specified function. This topic describes the syntax of lambda expressions. This topic also provides examples on how to use the expressions.

Syntax

You must use lambda expressions together with functions, such as Array functions, Array functions, Array functions, Array functions, and Map functions. Syntax:

```
parameter -> expression
```

Parameter	Description
parameter	The identifier that is used to pass parameters.

expression	The lambda expression, which can include most MySQL expressions. Examples:
	`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)`

Examples

- Example 1: x -> x is not null

  This lambda expression is used to return the non-null elements in the [5, null, 7, null] array.

  - Query statement

    ```
    * | SELECT filter(array[5, null, 7, null], x -> x is not null)
    ```

  - Query and analysis result

- Example 2: 0, (s, x) -> s + x, s -> s

  This lambda expression is used to return the sum of the elements in the [5,20,50] array.

  - Query statement

    ```
    * | SELECT reduce(array[5, 20, 50], 0, (s, x) -> s + x, s -> s)
    ```

  - Query and analysis result

- Example 3: (k,v) -> v > 10

  This lambda expression is used to create a map from two arrays. The values of keys in the map are greater than 10.

- Query statement

```
* | SELECT map_filter(map(array['class01', 'class02', 'class03'], array[11, 10, 9]), (k,v) -> v > 10)
```

- Query and analysis result

- Example 4: (x, y) -> (y, x)

  This lambda expression is used to transpose the elements in two arrays and return a new two-dimensional array that is created from the elements in the two arrays. The elements are located by using the same indexes.

  - Query statement

```
* | SELECT zip_with(array[1, 3, 5], array['a', 'b', 'c'], (x, y) -> (y, x))
```

  - Query and analysis result

- Example 5: x -> coalesce(x, 0) + 1

  This lambda expression is used to add 1 to each element in the [5, NULL, 6] array and return the result. The null element in the array is converted to 0 before 1 is added.

  - Query statement

```
* | SELECT transform(array[5, NULL, 6], x -> coalesce(x, 0) + 1)
```

  - Query and analysis result

- Additional examples

```
* | SELECT filter(array[], x -> true)
* | SELECT map_filter(map(array[],array[]), (k, v) -> true)
* | 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))
* | SELECT reduce(array[2147483647, 1], cast(0 AS bigint), (s, x) -> s + x, s -> s)
* | SELECT reduce(array[5, 20, null, 50], 0, (s, x) -> s + x, s -> s)
* | SELECT transform(array[array[1, null, 2], array[3, null]], a -> filter(a, x -> x is not null))
* | SELECT zip_with(array['a', 'b', 'c'], array['d', 'e', 'f'], (x, y) -> concat(x, y))
```

4.8.27. Logical functions

This topic describes the available logical functions in Log Service. You can use these functions to query and analyze log data.

Logical operators

Operator	Description	Example
AND	The result is TRUE if both values are TRUE.	a AND b
OR	The result is TRUE if either value is TRUE.	a OR b
NOT	The result is TRUE if the value is FALSE.	NOT a

Effect of NULL on logical operators

The following tables list the truth values when the values of a and b are TRUE, FALSE, and NULL, respectively.

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

Truth table 2

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

4.8.28. Column aliases

This topic describes how to specify an alias for a column and provides some examples.

A column name in an SQL statement can contain only letters, digits, and underscores (_). The column name must start with a letter.

When you configure log collection, you may specify a column name that does not conform to the SQL standard, for example, User-Agent. In this case, you must specify an alias for the column in the Search & Analysis panel in which you can configure index attributes. The alias is used only if you execute an SQL statement to query and analyze logs. The original name of each column is stored. Therefore, you must search for columns by original name.

If the original name of a column is long, you can specify an alias for the column in an SQL statement.

Sample aliases

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

4.8.29. JOIN queries on a Logstore and a MySQL database

Log Service allows you to use the JOIN syntax to query data from a Logstore and a MySQL database. The query results are saved to the database.

Prerequisites

An external store is created. For more information, see Associate Log Service with a MySQL database.

Context

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project that you want to manage.
3. Choose **Log Storage > Logstores**. On the Logstores tab, click the Logstore that you want to manage.
4. Execute a query statement.

   Log Service supports the following JOIN syntax:

   ```
   [ INNER ] JOIN
   LEFT [ OUTER ] JOIN
   RIGHT [ OUTER ] JOIN
   FULL [ OUTER ] JOIN
   ```

   The following sample code provides an example of a JOIN query. For more information, see JOIN syntax.

   ```
   method:postlogstorelogs | select count(1) , histogram(logstore) from log  l join join_meta m on l.
   projectid = cast( m.ikey as varchar)
   ```

   > **Notice**
   > - You can use the JOIN syntax only on a Logstore and a small table in a MySQL database. A small table contains less than 20 MB of data.
   > - In a query statement, the name of the Logstore must precede the join keyword, and the name of the external store must follow the join keyword.
   > - You must specify the name of the external store in a query statement. When the system executes the statement, the system replaces the name with the name of the database and the name of the table. Do not enter only the table name.

5. Save the query results to the MySQL database.

Log Service allows you to insert the query results into the database by using an INSERT statement. The following sample code provides an example of an INSERT statement:

```
method:postlogstorelogs | insert into method_output  select cast(method as varchar(65535)),count(1) from log group by method
```

Sample Python script

```
# encoding: utf-8
from __future__ import print_function
from aliyun.log import *
from aliyun.log.util import base64_encodestring
from random import randint
import time
import os
from datetime import datetime
    endpoint = os.environ.get('ALIYUN_LOG_SAMPLE_ENDPOINT', 'cn-chengdu.log.aliyuncs.com')
    accessKeyId = os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSID', '')
    accessKey = os.environ.get('ALIYUN_LOG_SAMPLE_ACCESSKEY', '')
    logstore = os.environ.get('ALIYUN_LOG_SAMPLE_LOGSTORE', '')
    project = "ali-yunlei-chengdu"
    client = LogClient(endpoint, accessKeyId, accessKey, token)
    # Create an external store.
    res = client.create_external_store(project,ExternalStoreConfig("rds_store","region","rds-vpc","vpcid","Instance ID","Instance IP address","Instance port","Username","Password","Database name","Table name"));
    res.log_print()
    # Retrieve the details of the external store.
    res = client.get_external_store(project,"rds_store");
    res.log_print()
    res = client.list_external_store(project,"");
    res.log_print();
    # Perform a JOIN query.
    req = GetLogsRequest(project,logstore,From,To,"","select count(1) from  "+ logstore +"  s join  meta m on  s.projectid = cast(m.ikey as varchar)");
    res = client.get_logs(req)
    res.log_print();
     # Save the query results to the MySQL database.
    req = GetLogsRequest(project,logstore,From,To,"," insert into rds_store select count(1) from  "+ logstore );
    res = client.get_logs(req)
    res.log_print();
```

4.8.30. Geospatial functions

This topic describes the available geospatial functions in Log Service. You can use these functions to query and analyze log data.

Concept of geometry

Geospatial functions support geometries in the well-known text (WKT) format.

Geometry formats

Geometry	WKT format
Point	`POINT (0 0)`

Geometry	WKT format
LineString	`LINESTRING (0 0, 1 1, 1 2)`
Polygon	`POLYGON ((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1))`
MultiPoint	`MULTIPOINT (0 0, 1 2)`
MultiLineString	`MULTILINESTRING ((0 0, 1 1, 1 2), (2 3, 3 2, 5 4))`
MultiPolygon	`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,`
GeometryCollection	`GEOMETRYCOLLECTION (POINT(2 3), LINESTRING (2 3, 3 4))`

Constructors

Constructor description

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

Operations

Function	Description
ST_Boundary(Geometry) → Geometry	Returns the closure of the combinatorial boundary of a geometry.
ST_Buffer(Geometry, distance) → Geometry	Returns the geometry that represents all points whose distance from the specified geometry is shorter than or equal to the specified distance.
ST_Difference(Geometry, Geometry) → Geometry	Returns the geometry value that represents the point set difference of the specified geometries.
ST_Envelope(Geometry) → Geometry	Returns the bounding rectangular polygon of a geometry.
ST_ExteriorRing(Geometry) → Geometry	Returns a line string that represents 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.

Function	Description
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 points of the second geometry are on the boundary of the first geometry.
ST_Crosses(Geometry, Geometry) → boolean	Returns True if the specified geometries share some, but not all, interior points in common.
ST_Disjoint(Geometry, Geometry) → boolean	Returns True if the specified geometries do not spatially intersect.
ST_Equals(Geometry, Geometry) → boolean	Returns True if the specified geometries represent the same geometry.
ST_Intersects(Geometry, Geometry) → boolean	Returns True if the specified geometries spatially intersect in two dimensions.
ST_Overlaps(Geometry, Geometry) → boolean	Returns True if the specified geometries share space in 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	Returns True if the specified 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 points in common at the boundaries.

Accessors

Function	Description
ST_Area(Geometry) → double	Returns the two-dimensional Euclidean area of a geometry.
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 a geometry.
ST_Dimension(Geometry) → bigint	Returns the inherent dimension of a geometry object, which must be less than or equal to the coordinate dimension.

Function	Description
ST_Distance(Geometry, Geometry) → double	Returns the minimum two-dimensional Cartesian distance (based on spatial ref) between two geometries in projected units.
ST_IsClosed(Geometry) → boolean	Returns True if the start and end points of the linestring are coincident.
ST_IsEmpty(Geometry) → boolean	Returns True if the specified geometry is an empty geometry, such as geometry collection, polygon, and point.
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 LineString or multi-LineString by using Euclidean measurement on a two-dimensional plane (based on spatial ref) in projected units.
ST_XMax(Geometry) → double	Returns the X maximum of the bounding box of the geometry.
ST_YMax(Geometry) → double	Returns the Y maximum of the bounding box of the geometry.
T_XMin(Geometry) → double	Returns the X minimum of the bounding box of the geometry.
ST_YMin(Geometry) → double	Returns the Y minimum of the bounding box of the geometry.
ST_StartPoint(Geometry) → point	Returns the first point of a geometry LineString instance.
ST_EndPoint(Geometry) → point	Returns the last point of a geometry LineString instance.
ST_X(Point) → double	Returns the X coordinate of a point.
ST_Y(Point) → double	Returns the Y coordinate of a point.
ST_NumPoints(Geometry) → bigint	Returns the number of points in a geometry.
ST_NumInteriorRing(Geometry) → bigint	Returns the cardinality of the collection of interior rings of a polygon.

4.8.31. Geography functions

This topic describes the syntax of geography functions and provides some examples.

IP functions identify the country, province, city, Internet service provider (ISP), and longitude and latitude of a specific IP address. For more information, see IP functions.

Function	Description	Example

Function	Description	Example
geohash(string)	Returns the geohash value of a specified geographical coordinate. The geographical coordinate is represented by a string in the format of "<latitude>, <longitude>". The return value is a string. Example: `geohash('34.1,120.6')`.	`* | select geohash('34.1,120.6')= 'wwjcbrdnzs'`
geohash(lat,lon)	Returns the geohash value of a specified geographical coordinate. The geographical coordinate is represented by two separate parameters that indicate the latitude and longitude. The return value is a string.	`* | select geohash(34.1,120.6)= 'wwjcbrdnzs'`

4.8.32. JOIN clause

You can use JOIN clauses in SQL statements to join multiple tables by using fields that are shared by the tables. In Log Service, you can join one or more Logstores. You can also join Logstores with ApsaraDB RDS instances. This topic describes how to join different Logstores.

Procedure

1. Download the latest version of the Log Service SDK for Python.
2. Call the GetProjectLogs operation to query logs.

Sample SDK

```
#!/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.getlogsrequest import GetProjectLogsRequest
from aliyun.log.putlogsrequest import PutLogsRequest
from aliyun.log.listtopicsrequest import ListTopicsRequest
from aliyun.log.listlogstoresrequest import ListLogstoresRequest
from aliyun.log.gethistogramsrequest import GetHistogramsRequest
from aliyun.log.index_config import *
from aliyun.log.logtail_config_detail import *
from aliyun.log.machine_group_detail import *
from aliyun.log.acl_config import *
if __name__=='__main__':
    token = None
    endpoint = "http://cn-hangzhou.log.aliyuncs.com"
    accessKeyId  = '******'
    accessKey = '******'
    client = LogClient(endpoint, accessKeyId, accessKey,token)
    logstore = "meta"
    # In the query statement, specify two Logstores, the query time ranges of both Logstores, and the
key to join the Logstores.
    req = GetProjectLogsRequest(project,"select count(1) from sls_operation_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 00:0
0:00' and m.__date__ <'2018-04-24 00:00:00' and s.projectid = cast(m.ikey as varchar)");
    res = client.get_project_logs(req)
    res.log_print();
    exit(0)
```

Note For more information about the syntax and usage examples of JOIN clauses, see JOIN clause.

4.8.33. UNNEST clause

This topic describes the syntax of UNNEST clauses.

Scenario

The value of a column in log data is stored as a primitive data type, such as string or number. In some cases, the value of a column may be of a complex data type, such as array, map, or JSON. When you query and analyze logs that contain fields whose values are of the preceding types, you can use an UNNEST clause to expand the field values into multiple rows for analysis.

Example:

```
__source__: 1.1.1.1__tag__:__hostname__: vm-req-170103232316569850-tianchi111932.tc__topic__: TestTopi
c_4array_column: [1,2,3]double_column: 1.23map_column: {"a":1,"b":2}text_column: Product
```

The value of the `array_column` field is an array. To obtain the sum of all elements in the value of the `array_column` field, you must traverse all elements of each array.

Syntax of UNNEST clauses

Syntax	Description
`unnest(array) as table_alias(column_name)`	Expands an array into multiple rows. `column_name` specifies the column name of the rows.
`unnest(map) as table(key_name, value_name)`	Expands a map into multiple rows. `key_name` specifies the column name of the keys and `value_name` specifies the column name of the values.

Note An UNNEST clause is applicable only to arrays or maps. If you want to expand a string, you must convert the string to JSON data. Then, you can use the `cast(json_parse(array_column) as array(bigint))` syntax to convert the JSON data to an array or a map.

Traverse the elements of an array

Use an UNNEST clause to expand an array into multiple rows. The rows are stored in a table named t. The column name of the rows is referenced as a.

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

When the elements in an array are traversed, you can also use other SQL syntax to query and analyze data. Examples:

- Calculate the sum of the elements in an array:

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

- Use a GROUP BY clause to group the elements in an array by column name:

  ```
  * | select a, count(1) from log, unnest(cast(json_parse(array_column) as array(bigint))) as t(a) gr
  oup by a
  ```

Traverse the elements of a map

- Traverse the elements of a map:

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

- Use a GROUP BY clause to group the elements in a map by key:

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

Visualize the results of the histogram and numeric_histogram functions

- histogram

  The histogram function is similar to the count group by syntax. For more information, see Map functions.

  The histogram function returns JSON data that cannot be visualized. The following example shows a query statement:

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

  To visualize the logs that contain the method field, you can use an UNNEST clause to expand the JSON data that is returned by the histogram function into multiple rows. The following example shows a query statement:

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

- numeric_histogram

  The numeric_histogram function is used to compute the approximate histogram of a specified field based on the number of histogram columns specified by the bucket parameter. This function is equivalent to the GROUP BY clause that is used to group data by numeric value column. For more information, see Approximate functions.

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

  To visualize the result of the numeric_histogram function, execute the following query statement:

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

4.9. Machine learning syntax and functions

4.9.1. Overview

Log Service provides the machine learning feature that supports multiple algorithms and calling methods. You can use SELECT statements and machine learning functions to call machine learning algorithms and analyze the characteristics of one or more fields within a specific period of time.

Log Service offers various time series analysis algorithms. You can call these algorithms to solve problems that are related to time series data. For example, you can predict time series, detect time series anomalies, decompose time series, and cluster multiple time series. In addition, the algorithms are compatible with standard SQL functions. This simplifies the usage of the algorithms and improves the efficiency of troubleshooting.

Features

- Supports various smooth operations on single-time series data.
- Supports algorithms that are used for the prediction, anomaly detection, change point detection, inflection point detection, and multi-period estimation of single-time series data.
- Supports decomposition operations on single-time series data.
- Supports various clustering algorithms for multi-time series data.
- Supports multi-field pattern mining based on the sequence of numeric data or text.

Limits

When you use the machine learning feature of Log Service, take note of the following limits:

- The specified time series data must be sampled based on the same interval.
- The specified time series data cannot contain data that is repeatedly sampled from the same point in time.
- The processing capacity cannot exceed the maximum capacity. The following table describes the limits.

Item	Description
Capacity of the time-series data processing	Data can be collected from a maximum of 150,000 consecutive points in time. If the data volume exceeds the processing capacity, you must aggregate the data or reduce the sampling amount.
Capacity of the density-based clustering algorithm	A maximum of 5,000 time series curves can be clustered at a time. Each curve cannot contain more than 1,440 points in time.
Capacity of the hierarchical clustering algorithm	A maximum of 2,000 time series curves can be clustered at a time. Each curve cannot contain more than 1,440 points in time.

Machine learning functions

Type	Function	Description
Smooth functions	ts_smooth_simple	Uses the Holt-Winters forecasting algorithm to filter time series data.
	ts_smooth_fir	Uses a finite impulse response (FIR) filter to filter time series data.
	ts_smooth_iir	Uses an infinite impulse response (IIR) filter to filter time series data.
Multi-period estimation functions	ts_period_detect	Estimates time series data by period.
Change point detection functions	ts_cp_detect	Detects the intervals in which data has different statistical features. The interval endpoints are change points.
	ts_breakout_detect	Detects the points in time at which data dramatically changes.
Maximum value detection function	ts_find_peaks	Detects the local maximum value of time series data in a specified window.
Prediction and anomaly detection functions	ts_predicate_simple	Uses default parameters to model time series data, predict time series data, and detect anomalies.
	ts_predicate_ar	Uses an autoregressive (AR) model to model time series data, predict time series data, and detect anomalies.
	ts_predicate_arma	Uses an autoregressive moving average (ARMA) model to model time series data, predict time series data, and detect anomalies.
	ts_predicate_arima	Uses an autoregressive integrated moving average (ARIMA) model to model time series data, predict time series data, and detect anomalies.
	ts_regression_predict	Predicts the trend for a single periodic time series.
	ts_anomaly_filter	Filters the anomalies that are detected from multiple time series curves based on the custom anomaly mode. The anomalies are detected during the anomaly detection. This function helps you find abnormal curves at the earliest opportunity.
Time series decomposition function	ts_decompose	Uses the Seasonal and Trend decomposition using Loess (STL) algorithm to decompose time series data.
Time series clustering functions	ts_density_cluster	Uses a density-based clustering method to cluster multiple time series.
	ts_hierarchical_cluster	Uses a hierarchical clustering method to cluster multiple time series.
	ts_similar_instance	Queries time series curves that are similar to a specified time series curve.

Type	Function	Description
Frequent pattern statistics function	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 function	pattern_diff	Identifies the pattern that causes differences between two collections in specified conditions.
Root cause analysis function	rca_kpi_search	Analyzes the subdimension attributes that cause the anomalies of a monitoring metric.
Correlation analysis functions	ts_association_analysis	Identifies the metrics that are correlated to a specified metric among multiple observed metrics in the system.
	ts_similar	Identifies the metrics that are correlated to specified time series data among multiple observed metrics in the system.
Kernel density estimation function	kernel_density_estimation	Uses the smooth peak function to fit the observed data points. In this way, the function simulates the real probability distribution curve.

4.9.2. Smooth functions

This topic describes the smooth functions that you can use to smooth and filter specified time series curves. Filtering is the first step to discover the shapes of time series curves.

Functions

Function	Description
ts_smooth_simple	Uses the Holt-Winters forecasting algorithm to filter time series data. This function is the default smooth function.
ts_smooth_fir	Uses a finite impulse response (FIR) filter to filter time series data.
ts_smooth_iir	Uses an infinite impulse response (IIR) filter to filter time series data.

ts_smooth_simple

- Syntax

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

- The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None

- Example

- Query statement

```
* | 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 )
```

- Query result

- The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The UNIX timestamp of the data. Unit: seconds.
Vertical axis	src	The unfiltered data.
	filter	The filtered data.

ts_smooth_fir

- Syntax
  - If you cannot determine filter parameters, use the built-in window parameters in the following statement:

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

  - If you can determine filter parameters, you can specify the parameters as needed in the following statement:

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

- The following table describes the parameters in the function.

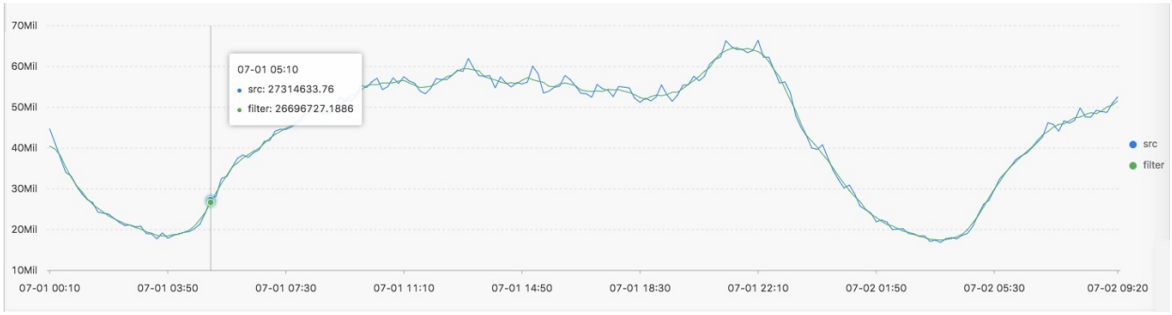
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
winType	The type of the window that you want to use to filter data.	Valid values: ○ rectangle: rectangle window ○ hanning: hanning window ○ hamming: hamming window ○ blackman: blackman window **Note** We recommend that you set the winType parameter to rectangle for better display effects.

Parameter	Description	Value
winSize	The length of the filter window.	The value is of the long type. Valid values: 2 to 15.
array[]	The parameter that you want to use for FIR filtering.	The value is an array and the sum of the elements in the array is 1. Example: array[0.2, 0.4, 0.3, 0.1].

- Example 1
  - Query statement

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

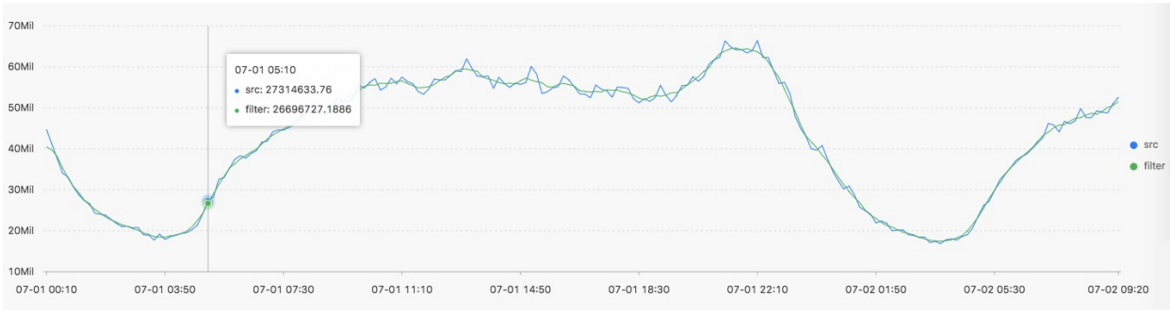
  - Query result



- Example 2
  - Query statement

```
* | select ts_smooth_fir(stamp, value, array[0.2, 0.4, 0.3, 0.1]) from ( select __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp )
```

  - Query result



- The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The UNIX timestamp of the data. Unit: seconds.
Vertical axis	src	The unfiltered data.
	filter	The filtered data.

ts_smooth_iir

- Syntax

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

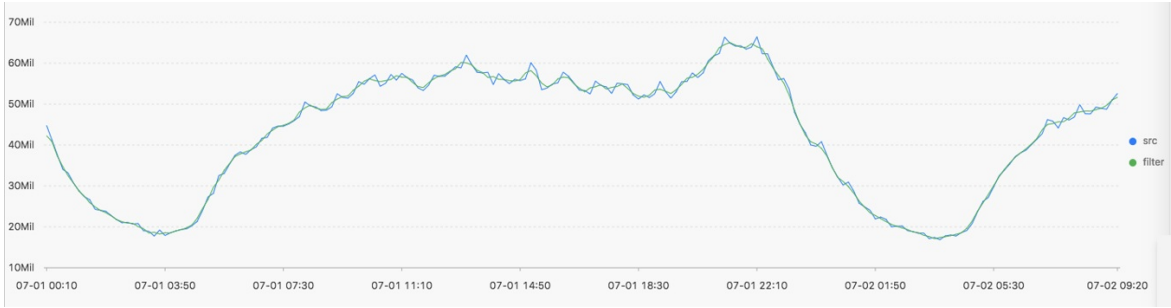
- The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
array[]	The parameter that you want to use for IIR filtering in terms of x_i.	The value is an array and the sum of the elements in the array is 1. The length of the array ranges from 2 to 15. Example: array[0.2, 0.4, 0.3, 0.1].
array[]	The parameter that you want to use for IIR filtering in terms of y_{i-1}.	The value is an array and the sum of the elements in the array is 1. The length of the array ranges from 2 to 15. Example: array[0.2, 0.4, 0.3, 0.1].

- Example
  - Query statement

```
* | select ts_smooth_iir(stamp, value, array[0.2, 0.4, 0.3, 0.1], array[0.4, 0.3, 0.3]) from ( se
lect __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp )
```

  - Query result



- The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The UNIX timestamp of the data. Unit: seconds.

Display item		Description
Vertical axis	src	The unfiltered data.
	filter	The filtered data.

4.9.3. Multi-period estimation functions

This topic describes multi-period estimation functions that you can use to estimate the periodicity of time series data distributed in different time intervals. This topic also describes how to extract the periodicity by using a series of operations such as Fourier transform (FT).

Functions

Function	Description
ts_period_detect	Estimates the periodicity of time series data that is distributed in different time intervals.
ts_period_classify	Uses FT to calculate the periodicity of specified time series curves. This function can be used to identify periodic curves.

ts_period_detect

Syntax

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

The following table describes the parameters in the function.

Parameter	Description
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis. Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.
minPeriod	The ratio of the minimum length of the estimated period to the total length of the time series data. The value of this parameter is of the float type. Valid values: (0,1].
maxPeriod	The ratio of the maximum length of the estimated period to the total length of the time series data. The value of this parameter is of the float type. Valid values: (0,1]. **Note** The value of the *maxPeriod* parameter must be greater than the value of the *minPeriod* parameter.

Example

- Query statement

```
* | select ts_period_detect(stamp, value, 0.2, 1.0) from ( select __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp )
```

- Query result

The following table describes the display items.

Display item	Description
period_id	An array with a length of 1. The element in the array indicates the sequence number of the period. The array [0] indicates the original time series curve.
time_series	The sequence of timestamps.
data_series	The sequence of data at each timestamp. • If the value of period_id is 0, the function returns the original time series data. • If the value of period_id is not 0, the function returns filtered time series data.

ts_period_classify

Syntax

```
select ts_period_classify(stamp,value,instanceName)
```

The following table describes the parameters in the function.

Parameter	Description
stamp	The time sequence. Points in time are sorted in ascending order along the horizontal axis. Each point in time is a UNIX timestamp. Unit: seconds.
value	The sequence of numeric data at a specific point in time.
instanceName	The name of the time series curve.

Example

- Query statement

```
* and h : nu2h05202.nu8 | select ts_period_classify(stamp, value, name) from log
```

- Query result

line_name	prob	type
asg-2zxjqn6zf5ewg188pg5	1.0	-1.0
asg-bp1j8snc92p6v5pptgpj	0.07203669207039314	0.0
asg-wz99hse7u4ubopo5dt9o	0.0	0.0
asg-bp18oqni0gq96vy85te4	0.05590892692207093	0.0

The following table describes the display items.

Display item	Description
line_name	An array with a length of 1. The element in the array indicates the sequence number of the period. The array [0] indicates the original time series curve.
prob	The ratio of the number of values within the primary period to the total number of values on the time series curve. Valid values: [0, 1]. You can set the value to 0.15 for testing.
type	The type of the curve. Valid values: • -1: The time series curve has a length of less than 64 points. • -2: The time series curve has a failure rate of higher than 20%. • 0: The time series curve is periodic.

4.9.4. Change point detection functions

This topic describes the change point detection functions that you can use to detect the change points in time series data.

Change point detection functions can detect the following two kinds of change points:

- Changes in statistical features within a specific period of time
- Anomalies in time series data

Functions

Function	Description
ts_cp_detect	Detects the intervals in which data has different statistical features. The interval endpoints are change points.
ts_breakout_detect	Detects the points in time at which data dramatically changes.

ts_cp_detect

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None

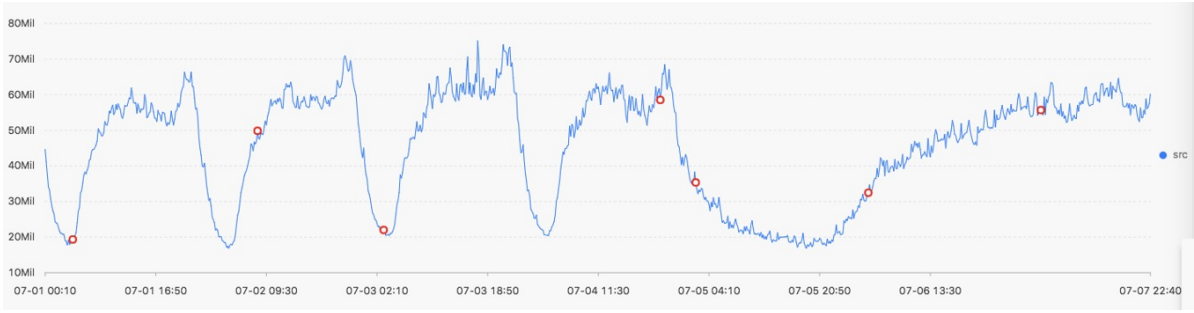
Parameter	Description	Value
minSize	The minimum length of time series data within a consecutive interval.	The minimum length is 3. The maximum length cannot exceed one tenth of the length of the specified time series data. Default value: 10.

Example

- Query statement

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

- Query result



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The timestamp of the data. Unit: seconds. Example: 1537071480.
Vertical axis	src	The unfiltered data. Example: 1956092.7647745228.
	prob	The probability that a point in time is a change point. Valid values: 0 to 1.

ts_breakout_detect

Syntax

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

The following table describes the parameters in the function.

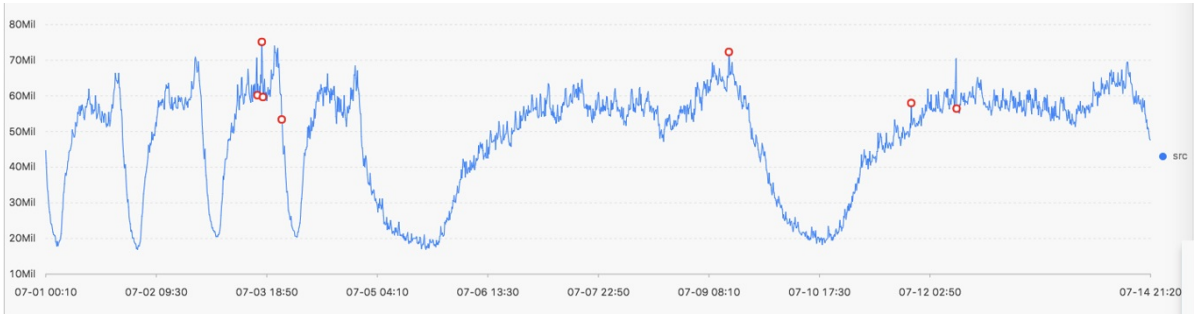
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
winSize	The minimum length of time series data within a consecutive interval.	The minimum length is 3. The maximum length cannot exceed one tenth of the length of the specified time series data. Default value: 10.

Example

- Query statement

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

- Query result



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The timestamp of the data. Unit: seconds. Example: 1537071480.
Vertical axis	src	The unfiltered data. Example: 1956092.7647745228.
	prob	The probability that a point in time is a change point. Valid values: 0 to 1.

4.9.5. Maximum value detection function

This topic describes the maximum value detection function that you can use to detect the local maximum value of time series data in a specified window.

ts_find_peaks

Syntax

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

The following table describes the parameters in the function.

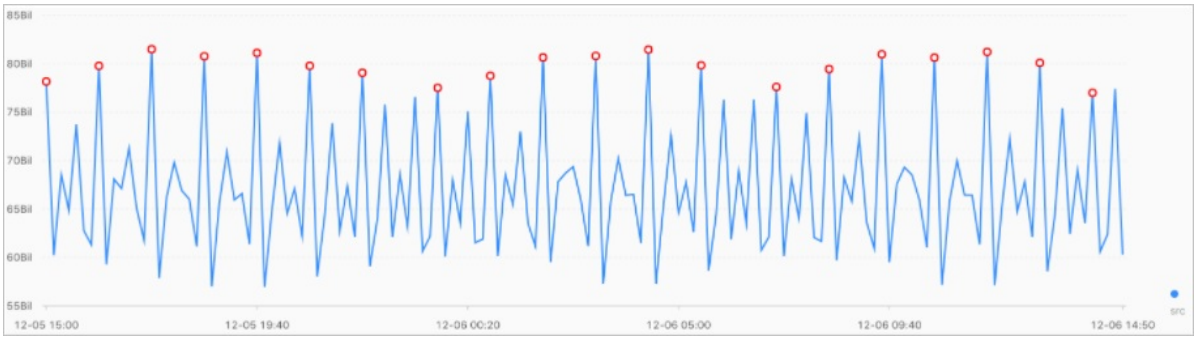
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
winSize	The minimum length of the detection window.	The value of this parameter is of the long type and ranges from 1 to the length of time series data. We recommend that you set this parameter to one tenth of the actual data length.

Example

- Query statement

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

- Query result



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The timestamp of the data. Unit: seconds. Example: 1537071480.
Vertical axis	src	The unfiltered data. Example: 1956092.7647745228.
	peak_flag	Indicates whether the numeric value at a point in time is the maximum value. Valid values: • 1.0: The numeric value at the point in time is the maximum value. • 0.0: The numeric value at the point in time is not the maximum value.

4.9.6. Prediction and anomaly detection functions

Prediction and anomaly detection functions predict the trend of time series curves and identify the Ksigma and quantiles of the errors between a predicted curve and an actual curve. You can use the functions to detect anomalies.

- Introduction to Log Service machine learning (01): time series statistics modeling
- Introduction to Log Service machine learning (03): time series anomaly detection modeling
- Introduction to Log Service machine learning (05): time series prediction
- Best practices for Log Service machine learning: time series anomaly detection and alerting

Functions

Function	Description
ts_predicate_simple	Uses default parameters to model time series data, predict time series data, and detect anomalies.
ts_predicate_ar	Uses an autoregressive (AR) model to model time series data, predict time series data, and detect anomalies.
ts_predicate_arma	Uses an autoregressive moving average (ARMA) model to model time series data, predict time series data, and detect anomalies.

Function	Description
ts_predicate_arima	Uses an autoregressive integrated moving average (ARIMA) model to model time series data, predict time series data, and detect anomalies.
ts_regression_predict	Predicts the trend for a single periodic time series. Scenario: You can use this function to predict metering data, network traffic, financial data, and different business data that follows certain rules.
ts_anomaly_filter	Filters the anomalies that are detected from multiple time series curves based on the custom anomaly mode. The anomalies are detected during the anomaly detection. This function helps you find abnormal curves at the earliest opportunity.

ts_predicate_simple

Syntax

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

The following table describes the parameters in the function.

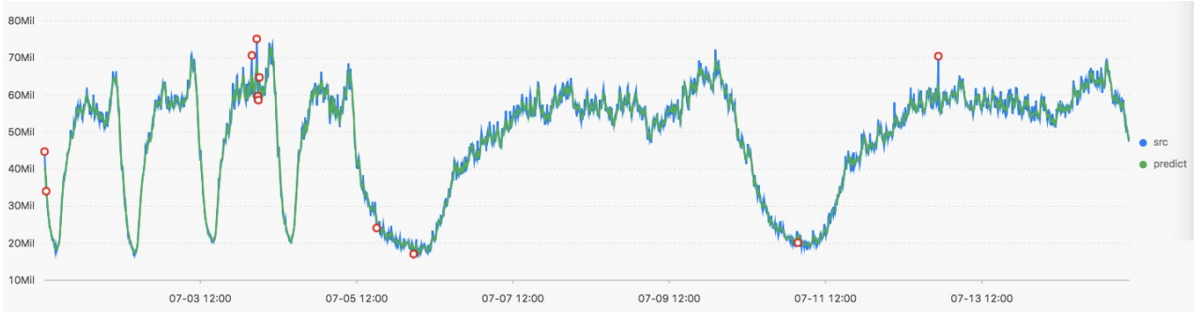
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
nPred	The number of points for prediction.	The value is of the long type. This value must be equal to or greater than 1.
isSmooth	Specifies whether to filter the raw data. • true: The raw data is filtered. • false: The raw data is not filtered.	The value is of the Boolean type. Default value: true.

Example

- Query statement

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

- Query result



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The UNIX timestamp of the data. Unit: seconds.
Vertical axis	src	The raw data.
	predict	The predicted data.
	upper	The upper limit of the prediction. The confidence level is 0.85. This value cannot be modified.
	lower	The lower limit of the prediction. The confidence level is 0.85. This value cannot be modified.
	anomaly_prob	The probability that the point is an anomaly. Valid values: 0 to 1.

ts_predicate_ar

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
p	The order of the AR model.	The value is of the long type. Valid values: 2 to 8.
nPred	The number of points for prediction.	The value is of the long type. Valid values: 1 to 5*p*.
isSmooth	Specifies whether to filter the raw data. • true: The raw data is filtered. • false: The raw data is not filtered.	The value is of the Boolean type. Default value: true.

Query statement

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

Note The result is similar to the result that is returned by the ts_predicate_simple function. For more information, see ts_predicate_simple.

ts_predicate_arma

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
p	The order of the AR model.	The value is of the long type. Valid values: 2 to 100.
q	The order of the ARMA model.	The value is of the long type. Valid values: 2 to 8.
nPred	The number of points for prediction.	The value is of the long type. Valid values: 1 to 5*p*.
isSmooth	Specifies whether to filter the raw data. • true: The raw data is filtered. • false: The raw data is not filtered.	The value is of the Boolean type. Default value: true.

Query statement

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

Note The result is similar to the result that is returned by the ts_predicate_simple function. For more information, see ts_predicate_simple.

ts_predicate_arima

Syntax

```
select ts_predicate_arima(x, y, p, d, q, nPred, isSmooth)
```

The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
p	The order of the AR model.	The value is of the long type. Valid values: 2 to 8.
d	The order of the ARIMA model.	The value is of the long type. Valid values: 1 to 3.
q	The order of the ARMA model.	The value is of the long type. Valid values: 2 to 8.

Parameter	Description	Value
nPred	The number of points for prediction.	The value is of the long type. Valid values: 1 to 5*p*.
isSmooth	Specifies whether to filter the raw data. • true: The raw data is filtered. • false: The raw data is not filtered.	The value is of the Boolean type. Default value: true.

Query statement

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

Note The result is similar to the result that is returned by the ts_predicate_simple function. For more information, see ts_predicate_simple.

ts_regression_predict

Syntax

```
select ts_regression_predict(x, y, nPred, algotype, processType)
```

The following table describes the parameters in the function.

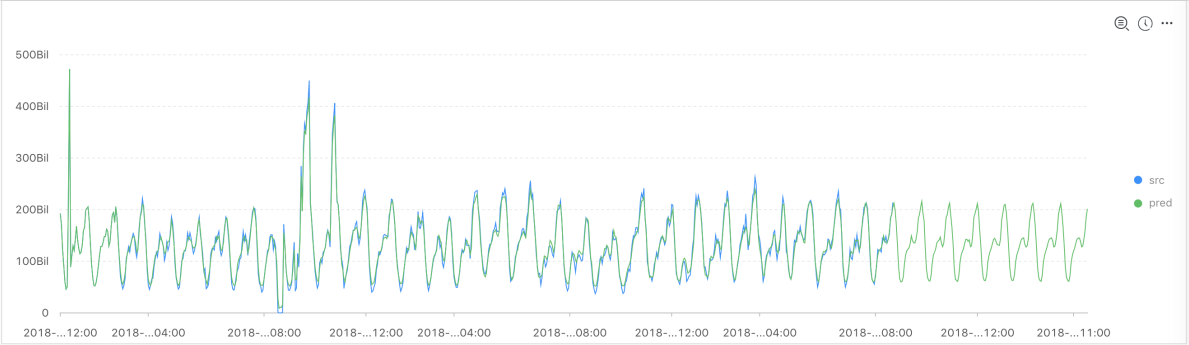
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
nPred	The number of points for prediction.	The value is of the long type. Valid values: 1 to 500.
algotype	The type of the algorithm used for prediction. Valid values: • origin: uses the Gradient Boosted Regression Tree (GBRT) algorithm for prediction. • forest: uses the GBRT algorithm for prediction based on the trend component decomposed by Seasonal and Trend decomposition using Loess (STL), and then uses the additive model to calculate the sum of 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 calculate the sum of the decomposed components and obtains the predicted data.	None
processType	Specifies whether to preprocess the data. Valid values: • 0: No additional data preprocessing is performed. • 1: Abnormal data is removed before prediction.	None

Example

- Query statement

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

- Query result



The following table describes the display items.

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

ts_anomaly_filter

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
lineName	The name of each curve. The value is of the varchar type.	None
ts	The time sequence of the curve, which indicates the time of the current curve.	The value of this parameter is an array of points in time of the double type. The points in time are sorted in ascending order.
ds	The actual value sequence of the curve.	The value of this parameter is an array of data points of the double type. The length of the value is the same as the length of the value of the ts parameter.
preds	The predicted value sequence of the curve.	The value of this parameter is an array of data points of the double type. The length of the value is the same as the length of the value of the ts parameter.

Parameter	Description	Value
probs	The sequence of anomaly detection results of the curve.	The value of this parameter is an array of data points of the double type. The length of the value is the same as the length of the value of the ts parameter.
nWatch	The number of the actual values that are recently observed on the curve. The value is of the long type. This value must be less than the number of points in time on the curve.	The value is of the long type.
anomalyType	The type of anomaly that you want to filter. Valid values: • 0: all anomalies • 1: positive anomalies • -1: negative anomalies	The value is of the long type.

Example

- Query statement

```
* | 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 lowe
rs, res[6] as probs
     from (
         select name, array_transpose(ts_predicate_ar(stamp, value, 10)) as res
         from (
           select name, stamp, value from log where name like '%asg-%') group by name)) );
```

- Query result

```
| name                     | ts                                                                | ds          | p
reds     | probs       |
| ------------------------ | ----------------------------------------------------------------- | ----------- | -
-------- | ----------- |
| asg-bp1hylzdi2wx7civ0ivk | [1.5513696E9, 1.5513732E9, 1.5513768E9, 1.5513804E9] | [1,2,3,NaN] | [
1,2,3,4] | [0,0,1,NaN] |
```

4.9.7. Time series decomposition function

This topic describes the time series decomposition function that you can use to decompose time series curves and show the trend and periodicity of curves.

ts_decompose

Syntax

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

The following table describes the parameters in the function.

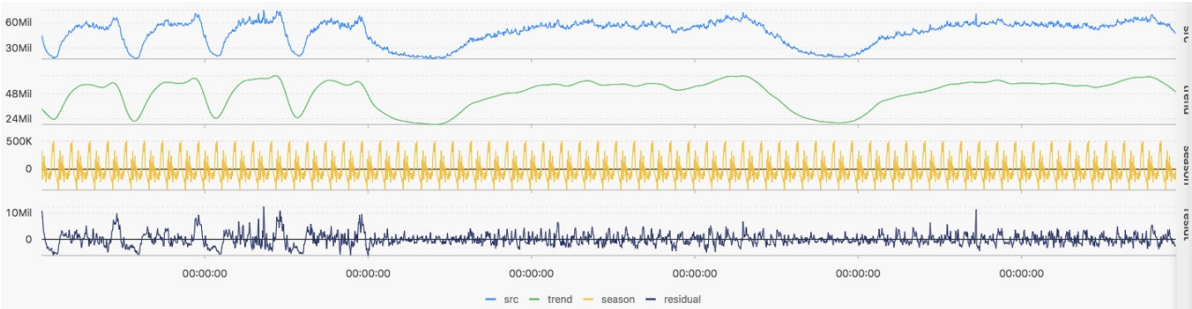
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None

Example

- Query statement

```
* | select ts_decompose(stamp, value) from (select __time__ - __time__ % 60 as stamp, avg(v) as val
ue from log GROUP BY stamp order by stamp)
```

- Query result



The following table describes the display items.

Display item		Description
Horizontal axis	unixtime	The UNIX timestamp of the data. Unit: seconds.
Vertical axis	src	The raw data.
	trend	The decomposed data that indicates the trend of the time series data.
	season	The decomposed data that indicates the periodicity of the time series data.
	residual	The residual data that is decomposed from the time series data.

4.9.8. Time series clustering functions

You can use time series clustering functions to cluster data of multiple time series and obtain different curve shapes. Then, you can use the data to find the cluster center and identify curves with shapes that are different from other curve shapes in the cluster.

Functions

Function	Description
ts_density_cluster	Uses a density-based clustering method to cluster multiple time series.

Function	Description
ts_hierarchical_cluster	Uses a hierarchical clustering method to cluster multiple time series.
ts_similar_instance	Queries time series curves that are similar to a specified time series curve.

ts_density_cluster

Syntax

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

The following table describes the parameters in the function.

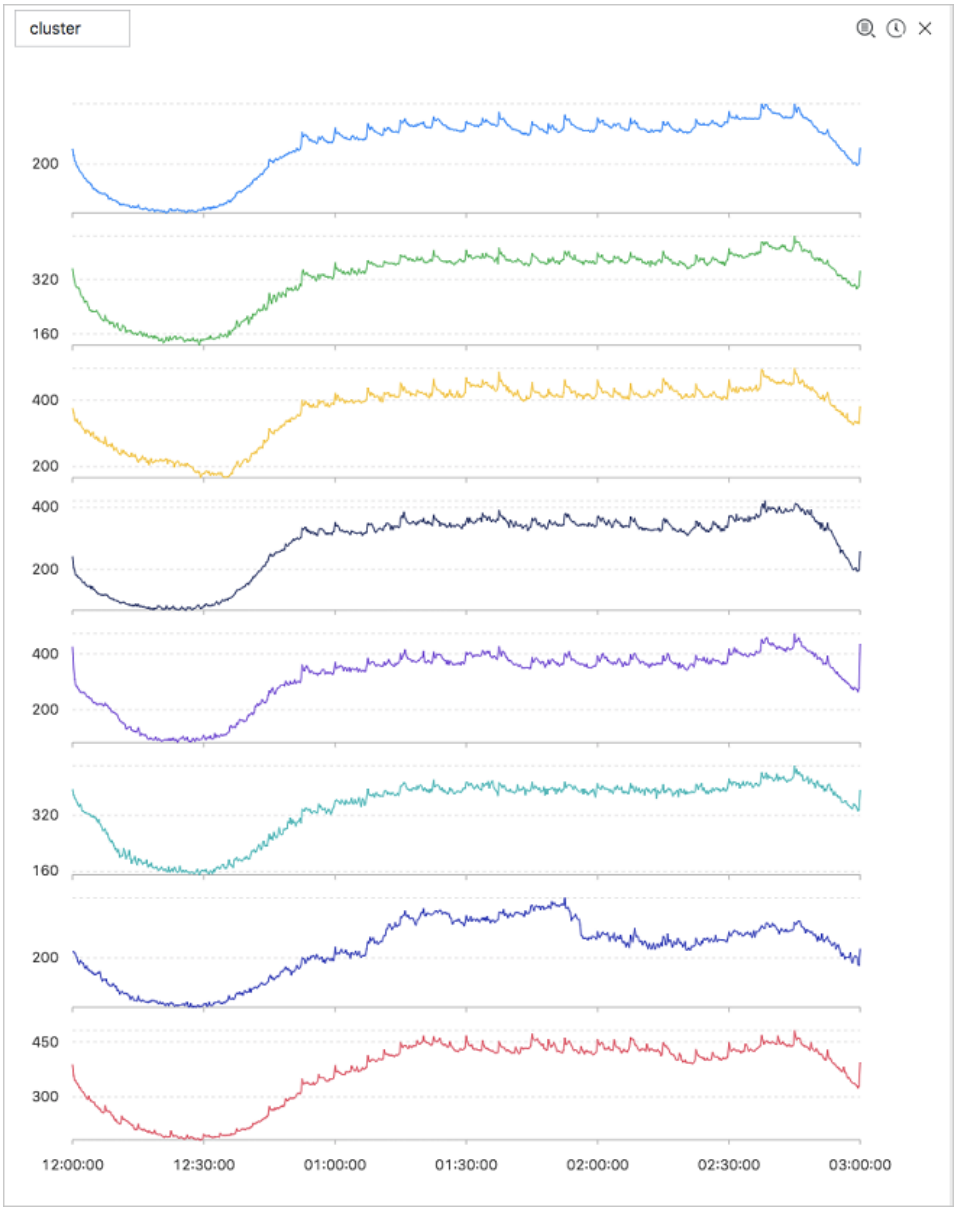
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
z	The name of the curve that corresponds to the data at a specified point in time.	The value of this parameter is a string. Example: machine01.cpu_usr.

Example

- Query statement

```
* and (h: "machine_01" OR h: "machine_02" OR h : "machine_03") | select ts_density_cluster(stamp, m
etric_value,metric_name ) from ( select __time__ - __time__ % 600 as stamp, avg(v) as metric_value,
h as metric_name from log GROUP BY stamp, metric_name order BY metric_name, stamp )
```

- Query 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 a cluster center.
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 instances that are included in the cluster center.
sim_instance	The name of an instance in the cluster.

ts_hierarchical_cluster

Syntax

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

The following table describes the parameters in the function.

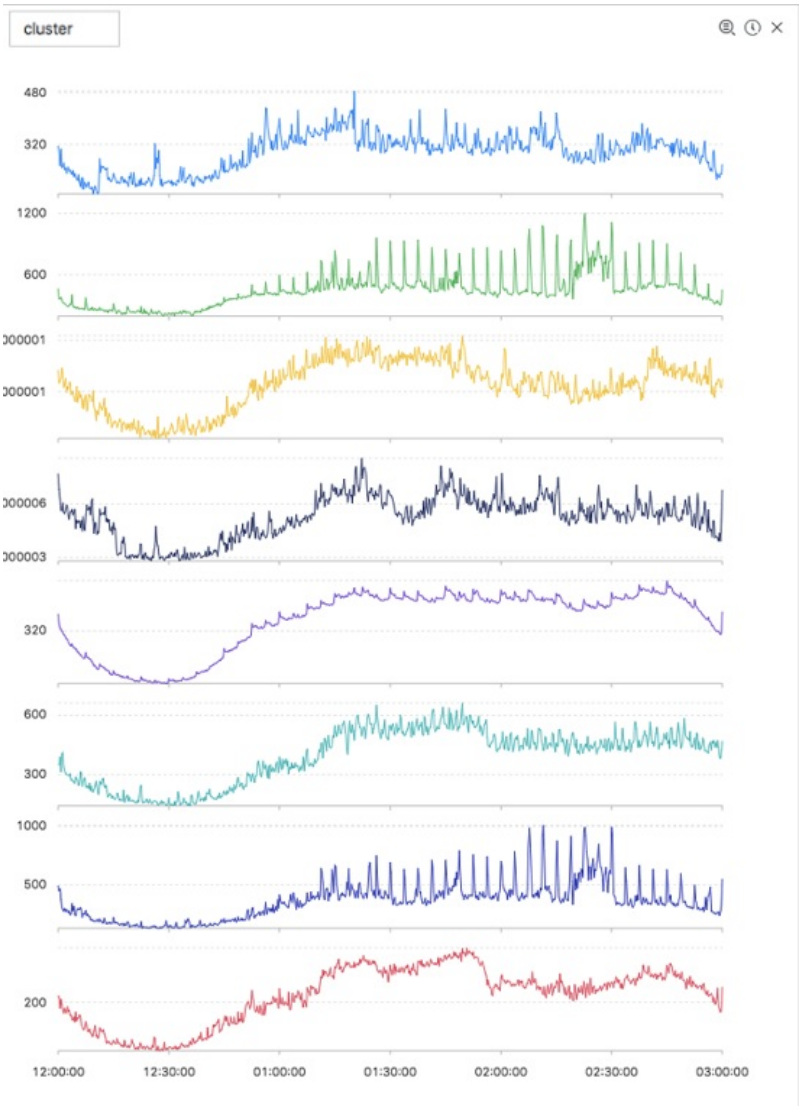
Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
z	The name of the curve that corresponds to the data at a specified point in time.	The value of this parameter is a string. Example: machine01.cpu_usr.

Example

- Query statement

```
* and (h: "machine_01" OR h: "machine_02" OR h : "machine_03") | select ts_hierarchical_cluster(sta
mp, metric_value, metric_name) from ( select __time__ - __time__ % 600 as stamp, avg(v) as metric_v
alue, h as metric_name from log GROUP BY stamp, metric_name order BY metric_name, stamp )
```

- Query 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 a cluster center.
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 instances that are included in the cluster center.
sim_instance	The name of an instance in the cluster.

ts_similar_instance

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
x	The time sequence. Points in time are sorted in ascending order along the horizontal axis.	Each point in time is a UNIX timestamp. Unit: seconds.
y	The sequence of numeric data at a specific point in time.	None
z	The name of the curve that corresponds to the data at a specified point in time.	The value of this parameter is a string. Example: machine01.cpu_usr.
instance_name	The name of a specified curve that you want to query.	The value is of this parameter is a string. Example: machine01.cpu_usr. **Notice** You must specify an existing curve.
topK	The maximum number of curves that are similar to the specified curve can be returned.	None
metricType	`{'shape', 'manhattan', 'euclidean'}`. The metric used to measure the similarity between time series curves.	None

Query statement

```
* and m: NET and m: Tcp and (h: "nu4e01524.nu8" OR  h: "nu2i10267.nu8" OR  h : "nu4q10466.nu8") | sele
ct ts_similar_instance(stamp, metric_value, metric_name, 'nu4e01524.nu8' ) from ( select __time__ - __
time__ % 600 as stamp, sum(v) as metric_value, h as metric_name from log GROUP BY stamp, metric_name o
rder 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.

4.9.9. Frequent pattern statistics function

The frequent pattern statistics function combines representative attributes in a specified multi-attribute field sample.

pattern_stat

Syntax:

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

The following table lists the parameters of the function.

Parameter	Description	Value
array[col1, col2, col3]	A column of character values.	An array of values, for example, array[clientIP, sourceIP, path, logstore].
array['col1_name', 'col2_name', 'col3_name']	The field names of the character values.	An array of field names, for example, array['clientIP', 'sourceIP', 'path', 'logstore'].
array[col5, col6]	A column of numeric values.	An array of values, for example, array[Inflow, OutFlow].
array['col5_name', 'col6_name']	The field names of the numeric values.	An array of field names, for example, array['Inflow', 'OutFlow'].
support_score	The support ratio of samples for pattern mining.	The value is of the DOUBLE data type. Value range: (0,1].
sample_ratio	The sampling ratio. The default value is 0.1, which indicates that only 10% of the total samples are used.	The value is of the DOUBLE data type. Value range: (0,1].

Example:

- Query statement

```
* | select pattern_stat(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
```

- Display item

Display item	Description
count	The number of samples in the current pattern.
support_score	The score of the current pattern. The score indicates the degree to which the current pattern is supported.
pattern	The content of the pattern. The pattern is organized in the format that is defined by the query conditions.

4.9.10. Differential pattern statistics function

The differential pattern statistics function analyzes differential patterns of specified multi-field samples based on the specified condition. The function helps you identify the causes of the differences under the current condition at the earliest opportunity.

pattern_diff

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
array_char_value	A column of values of the character data type.	The value of this parameter is an array. Example: array[clientIP, sourceIP, path, logstore].
array_char_name	The column names of the values of the character data type.	The value of this parameter is an array. Example: array['clientIP', 'sourceIP', 'path', 'logstore'].
array_numeric_value	A column of numeric values.	The value of this parameter is an array. Example: array[Inflow, OutFlow].
array_numeric_name	The column names of the numeric values.	The value of this parameter is an array. Example: array['Inflow', 'OutFlow'].
condition	The condition that is used to filter data. The value True indicates positive samples and the value False indicates negative samples.	Example: Latency <= 300.
supportScore	The support ratio of positive and negative samples for pattern mining.	The value of this parameter is of the double type. Valid values: (0,1].
posSampleRatio	The sampling ratio of positive samples. Default value: 0.5. This value indicates that 50% of positive samples are collected.	The value of this parameter is of the double type. Valid values: (0,1].
negSampleRatio	The sampling ratio of negative samples. Default value: 0.5. This value indicates that 50% of negative samples are collected.	The value of this parameter is of the double type. Valid values: (0,1].

Example

- Query statement

```
* | 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
```

- Display item

Display item	Description
possupport	The support ratio of positive samples for the mined patterns.
posconfidence	The confidence level of the mined patterns in positive samples.
negsupport	The support ratio of negative samples for the mined patterns.
diffpattern	The content of the mined patterns.

4.9.11. Root cause analysis function

Log Service provides alerting and analysis capabilities that allow you to analyze and identify anomalies in specific subdimensions of a metric at the earliest opportunity. You can use the root cause analysis function to identify and analyze the subdimension attributes that cause the anomalies.

rca_kpi_search

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
varchar_array	The array of subdimension attributes.	Example: array[col1, col2, col3].
name_array	The array of subdimension attribute names.	Example: array['col1', 'col2', 'col3'].
real	The actual value of each subdimension attribute that is specified by the varchar_array parameter. The value of this parameter is of the double type.	Valid values: all real numbers.
forecast	The predicted value of each subdimension attribute that is specified by the varchar_array parameter. The value of this parameter is of the double type.	Valid values: all real numbers.
level	The number of subdimension attributes identified in the returned root cause set. The value 0 indicates that the function returns all root causes that are found. The value of this parameter is of the double type.	Valid values: [0, number of analyzed subdimensions]. The number of analysis dimensions is based on the length of the array that is specified by the varchar_array parameter.

Example

- Query statement

  Use a subquery to obtain the actual value and predicted value of each subdimension attribute, and then call the rca_kpi_search function to analyze the root causes of anomalies.

```
* 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)
```

- Query result

The following figure shows the structure of the query 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
rcSets	The root cause sets. Each value is an array.
rcItems	A root cause set.
kpi	An item in the root cause set. Each item is formatted in an array where each element is of the JSON type. The attr parameter indicates the name of a subdimension. The val parameter indicates the attribute name that corresponds to the subdimension.
nleaf	The number of leaf nodes that a kpi in the root cause set covers in the raw data. **Note** A leaf node is a log entry that contains the finest-grained attributes.
change	The ratio of the number of anomaly changes in the leaf nodes that are covered by a kpi to the total number of anomaly changes in the root cause set at the same point in time.
score	The abnormality score of the current kpi. Valid values: [0,1].

The following example shows the query result that is in the JSON format:

```
{
  "rcSets": [
  {
    "rcItems": [
    {
      "kpi": [
      {
        "attr": "country",
        "val": "*"
      },
      {
        "attr": "province",
        "val": "*"
      },
      {
        "attr": "provider",
        "val": "*"
      },
      {
        "attr": "domain",
        "val": "download.huya.com"
      },
      {
        "attr": "method",
        "val": "*"
      }
      ],
      "nleaf": 119,
      "change": 0.3180687806279939,
      "score": 0.14436007709620113
    }
    ]
  }
  ]
}
```


4.9.12. Correlation analysis functions

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

Functions

Function	Description
ts_association_analysis	Identifies the metrics that are correlated to a specified metric among multiple observed metrics in the system.
ts_similar	Identifies the metrics that are correlated to specified time series data among multiple observed metrics in the system.

ts_association_analysis

Syntax

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

The following table describes the parameters in the function.

Parameter	Description	Value
stamp	The UNIX timestamp that is of the long type.	None
params	The metrics that you want to analyze. The value of this parameter is an array. Each element in the array is of the double type.	Example: Latency, QPS, and NetFlow.
names	The names of the metrics that you want to analyze. The value of this parameter is an array. Each element in the array is of the varchar type.	Example: Latency, QPS, and NetFlow.
indexName	The name of the target metric. The value of this parameter is of the varchar type.	Example: Latency.
threshold	The threshold of correlation between the target metric and the metrics that you want to analyze.	Valid values: [0,1].

- Query statement

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

- Query result

```
| results               |
| --------------------- |
| ['latency', '1.0']    |
| ['outflow', '0.6265'] |
| ['status', '0.2270']  |
```

- Description of the query result
  - name: the name of the metric that meets the specified correlation condition of the target metric.
  - score: the value of correlation between the returned metric and the target metric. Valid values: [0, 1].

ts_similar

Syntax 1

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

The following table describes the parameters in the function.

Parameter	Description	Value
stamp	The UNIX timestamp that is of the long type.	None

Parameter	Description	Value
value	The value of the metric that you want to analyze. The value of this parameter is of the double type.	None
ts	The time sequence of the specified time series curve. The value of this parameter is an array. Each element in the array is of the double type.	None
ds	The sequence of numeric data of the specified time series curve. The value of this parameter is an array. Each element in the array is of the double type.	None
metricType	The type of correlation between the measured curves. The value of this parameter is of the varchar type. Valid values: SHAPE, RMSE, PEARSON, SPEARMAN, R2, and KENDALL.	Example: SHAPE.

Syntax 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 in the function.

Parameter	Description	Value
stamp	The UNIX timestamp that is of the long type.	None
value	The value of the metric that you want to analyze. The value of this parameter is of the double type.	None
startStamp	The start timestamp of the specified time series curve. The value of this parameter is of the long type.	None
endStamp	The end timestamp of the specified time series curve. The value of this parameter is of the long type.	None
step	The time interval between two adjacent data points in a time series. The value of this parameter is of the long type.	None
ds	The sequence of numeric data of the specified time series curve. The value of this parameter is an array. Each element in the array is of the double type.	None

Parameter	Description	Value
metricType	The type of correlation between the measured curves. The value of this parameter is of the varchar type. Valid values: SHAPE, RMSE, PEARSON, SPEARMAN, R2, and KENDALL.	Example: SHAPE.

- Query statement

```
* | 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;
```

- Query result

```
| vhost  | metric          | score                |
| ------ | --------------- | -------------------- |
| vhost1 | redolog         | -0.3519082537204182  |
| vhost1 | kv_qps          | -0.15922168009772697 |
| vhost1 | file_meta_write | NaN                  |
```

- Description of the query result

  score: the correlation between the analyzed metric and the specified time series curve. Valid values: [-1, 1].

4.9.13. Kernel density estimation function

Kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable.

The Kernel density estimation function uses the smooth peak function to fit the observed data points. In this way, the function simulates the real probability distribution curve.

- Syntax

```
select kernel_density_estimation(bigint stamp, double value, varchar kernelType)
```

- Parameters

Parameter	Description
stamp	The Unix timestamp of observed data. Unit: second.
value	The observed value.
kernelType	○ box: rectangle window. ○ epanechniov: Epanechnikov curve. ○ gausener: Gaussian curve.

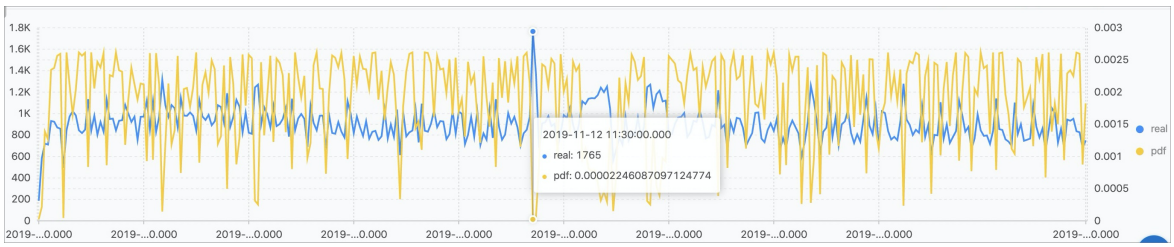
- Response

Display item	Description
unixtime	The Unix timestamp of observed data.
real	The observed value.
pdf	The probability of each observed data point.

- Example
  - Sample statement

```
* |
select
    date_trunc('second', cast(t1[1] as bigint)) as time, t1[2] as real, t1[3] as pdf from (
        select kernel_density_estimation(time, num, 'gaussian') as res from (
            select __time__ - __time__ % 10 as time, COUNT(*) * 1.0 as num from log group by time
order by time)
        ), unnest(res) as t(t1)  limit 1000
```

  - Response



4.10. Advanced analysis

4.10.1. Optimize queries

This topic describes how to optimize queries to improve query efficiency.

Increase the number of shards

More shards indicate more computing resources and faster computing speed. You can increase the number of shards to ensure that the average number of log entries that are scanned in each shard does not exceed 50 million. You can increase the number of shards by splitting shards. For more information, see Split a shard.

Reduce the query time range and data volume

- A larger time range means a slower query. If you query data within a year or a month, data is computed on a daily basis. To improve the computing speed, you can reduce the query time range.
- Larger data volumes slow down queries. Reduce the amount of data that you want to query 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. The underlying acceleration mechanism ensures that each query uses the previous query result to analyze data. This way, multiple queries can improve the accuracy of the query result.

Optimize SQL statements for queries

A time-consuming query statement has the following characteristics:

- Uses GROUP BY clauses to group string-formatted columns.
- Use GROUP BY clauses to group more than five columns.
- Includes operations that generate strings.

We recommend that you use the following methods to optimize SQL statements for queries:

- Avoid operations that generate strings if possible.

- If you use the date_format function to generate a formatted timestamp, the query is inefficient.

  ```
  * | select date_format(from_unixtime(__time__) , '%H_%i') as t, count(1) group by t
  ```

- If you use the substr() function, strings are generated. We recommend that you use the date_trunc or time_series function in a query statement.

- Avoid using GROUP BY clauses to group string-formatted columns if possible.

  If you use a GROUP BY clause to group strings, a large number of hash calculations are required. The number of the hash calculations account for more than 50% of the total number of calculations. The following example shows two query statements:

  ```
  * | select count(1) as pv , date_trunc('hour',__time__) as time group by time
  * | select count(1) as pv , from_unixtime(__time__-__time__%3600) as time group by __time__-__time_
  _%3600
  ```

  Query 1 is less efficient than query 2 because query 1 needs to hash strings.

  - Query 1 and query 2 calculate the total number of log entries per hour.
  - Query 1 converts the time to a string, for example, 2017-12-12 00:00:00, and then uses a GROUP BY clause to group the string.
  - Query 2 calculates the on-the-hour time value, uses a GROUP BY clause to group the result, and then converts the value to a string.

- List fields alphabetically based on the initial letter when you use a GROUP BY clause to group multiple columns.

  For example, you need to query 100 million users who are from 13 provinces.

  ```
  Fast: * | select province,uid,count(1)groupby province,uid
  Slow: * | select province,uid,count(1) group by uid,province
  ```

- Use estimating functions.

  Estimating functions provide better performance than accurate calculation. In estimation, accuracy is compromised to an acceptable level to achieve fast calculation.

  ```
  Fast: * |select approx_distinct(ip)
  Slow: * | select count(distinct(ip))
  ```

- Specify only the required columns in an SQL statement if possible.

  When you use an SQL statement to query data, specify only the required columns to speed up the calculation.

  ```
  Fast: * | select a,b c
  Slow: * | select *
  ```

- Specify columns that do not need to be grouped in an aggregate function if possible.

  For example, a user ID is associated with a username. Therefore, you can use a GROUP BY clause to group data by userid.

  ```
  Fast: * | select userid, arbitrary(username), count(1)group by userid
  Slow: * | select userid, username, count(1) group by userid,username
  ```

- Avoid using the IN operator if possible.

  Use the OR operator in SQL statements instead of the IN operator if possible.

  ```
  Fast: key : a or key :b or key:c | select count(1)
  Slow: * | select count(1) where key in ('a','b')
  ```

4.10.2. Use cases

This topic describes some use cases of data analysis in Log Service.

Cases

- Trigger an alert if the error rate exceeds 40% over the last 5 minutes
- Calculate the amount of transferred data and configure alerts
- Calculate the average latency of traffic data in different sizes
- Obtain the percentages of different results
- Calculate the number of log entries that meet the query condition

Trigger an alert if the error rate exceeds 40% over the last 5 minutes

Calculate the percentage of 500 Internal Server Error every minute. If the error rate exceeds 40% over the last 5 minutes, an alert is triggered.

```
status :500 |
select
  __topic__,
  max_by(error_count, window_time) / 1.0 / sum(error_count) as error_ratio,
  sum(error_count) as total_error
FROM  (
    select
      __topic__,
      count(*) as error_count,
      __time__-__time__ % 300 as window_time
    FROM      log
    group by
      __topic__,
      window_time
  )
group by
  __topic__
having
  max_by(error_count, window_time) / 1.0 / sum(error_count) > 0.4
  and sum(error_count) > 500
order by
  total_error desc
limit
  100
```

- You can use the following clause to calculate the error rate: `max_by(error_count,window_time)/1.0/sum(error_count) as error_ratio` .
- You can use the following clause to calculate the total number of 500 Internal Server Error: `sum(error_count) as total_error` .
- You can use the following clause to query the number of errors every 5 minutes: `select __topic__, count(*) as error_count , __time__ - __time__ % 300 as window_time from log group by __topic__, window_time` .

Calculate the amount of transferred data and configure alerts

Calculate the amount of transferred data every minute. If the amount of transferred data sharply decreases, an alert is triggered. Transferred data counted in the last minute does not cover a full minute. The `(max(time) - min(time))` clause is used 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 traffic data in different sizes

Distribute traffic data to multiple buckets based on the data size and calculate the average latency of the data in each bucket.

```
* |
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
```

Obtain the percentages of different results

Obtain the number and percentage of each count result for different departments. The following query statement includes subqueries and window functions. The `sum(c) over()` clause is used 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
    group by
      department
  )
```

Calculate the number of log entries that meet the query condition

Use the count_if clause to calculate the number of URLs that meet specified conditions and obtain the number of URLs that meet each condition by minute.

```
* |
select
  count_if(uri like '%login') as login_num,
  count_if(uri like '%register') as register_num,
  date_format(date_trunc('minute', __time__), '%m-%d %H:%i') as time
group by
  time
order by
  time
limit
  100
```

- You can use the following clause to calculate the number of URLs that end with login: `count_if(uri like '%login')`.
- You can use the following clause to calculate the number of URLs that end with register: `count_if(uri like '%register')`.

4.10.3. Examples of time field conversion

In most cases, you need to process the time fields in log data when you query and analyze the log data. For example, you need to convert a timestamp to a specified time format. This topic provides some examples on how to convert the values of time fields.

A log may contain multiple fields that record points in time for different events. Examples:

- `__time__`: records the time when you call the API or use an SDK to write log data. You can use this field when you ship, query, and analyze log data.
- Original time field in log data: records the time when the log data is generated. This field exists in raw logs.

Time fields in different formats are difficult to read. To simplify the read process, you can convert the time format when you query and analyze log data. Examples:

1. Convert the value of __time__ to a timestamp
2. Display the value of __time__ in a specified format
3. Convert the time in a log to a specified format

Convert the value of __time__ to a timestamp

You can use the from_unixtime function to convert the value of the `__time__` field to a timestamp.

```
* | select from_unixtime(__time__)
```

Display the value of __time__ in a specified format

To display the value of the `__time__` field in the format of `YYYY-MM-DD HH:MM:SS`, you can use the date_format function.

```
* | select date_format(__time__, '%Y-%m-%d %H:%i:%S')
```

Convert the time in a log to a specified format

To convert the value of the time field in a log to a specified format, such as `YYYY-MM-DD HH:MM:SS`, and then perform the GROUP BY operation on the `YYYY-MM-DD` part, you can use the date_format function.

- Sample log

```
__topic__:
body_byte_sent:  307
hostname:  www.host1.com
http_user_agent:  Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_3 like Mac OS X) AppleWebKit/603.3.8 (KHT
ML, like Gecko) Mobile/14G60 QQ/7.1.8.452 V1_IPH_SQ_7.1.8_1_APP_A Pixel/750 Core/UIWebView NetType/
WIFI QBWebViewType/1
method:  GET
referer:  www.host0.com
remote_addr:  36.63.1.23
request_length:  111
request_time:  2.705
status:  200
upstream_response_time:  0.225582883754
url:  /?k0=v9&
time:2017-05-17 09:45:00
```

- Sample SQL statement

```
* | select date_format (date_parse(time,'%Y-%m-%d %H:%i:%S'), '%Y-%m-%d') as day, count(1) as uv gr
oup by day order by day asc
```

4.11. Associate Log Service with external data sources

4.11.1. Overview

Log Service provides the external storage feature. You can use the feature to associate Log Service with MySQL databases, Alibaba Cloud Object Storage Service (OSS) buckets, or hosted CSV files. This topic describes the scenarios, benefits, and supported external stores of the external storage feature.

Scenarios

When you analyze data, you may need to obtain different types of data from separate storage resources. For example, you need to obtain data on user operations and user behavior from Log Service, and obtain data on user properties, registration, funds, and props from a database. In this example, you must classify and analyze data and then write the analysis results to the report system of the database.

To do this, you can migrate data to a centralized storage system and then analyze the data. However, the migration process is time-consuming and labor-intensive. During migration, data must be cleansed and formatted, and network resources are consumed. To address these issues, Log Service provides API operations for external storage. You can call the API operations to achieve the following goals:

- Define mappings between data in external stores and data in Log Service. Data migration is not required.
- Use a unified query engine. You can use JOIN statements to perform JOIN queries on data in Log Service and data in external stores.
- Store query results in external stores.

Benefits

- Cost-effective
  - The external storage feature eliminates the need for data migration, which helps you reduce overall costs. Data in different storage systems is stored in different formats. The API operations that you can call to manage data also vary based on the storage systems. This results in complicated data conversion during data migration. If you use the external storage feature of Log Service, you do not need to migrate data.
  - The external storage feature eliminates the need for data maintenance, which helps you reduce overall costs. If you migrate data, you must update and maintain the data at the earliest opportunity.

- Convenient
  - You can use SQL statements to analyze data and obtain the analysis results within seconds.
  - You can add charts to a dashboard and view the charts when you open the dashboard.

Supported external stores

The external storage feature of Log Service allows you to associate Log Service with MySQL databases, OSS buckets, or hosted CSV files. The following table describes the supported external stores.

Supported external store	Read from the external store	Write to the external store	Method that is used to create an external store
MySQL	Supported	Supported	API, SDK, and CLI
OSS	Supported	Supported	SQL create table
Hosted CSV files	Supported	Not supported	SDK

4.11.2. Associate Log Service with a MySQL database

This topic describes how to create an external store to associate Log Service with a MySQL database.

Prerequisites

- Data is collected and stored in Log Service. For more information, see Data collection.
- Data is stored in a MySQL database.

Context

The external storage feature of Log Service allows you to associate Log Service with databases that are deployed on ApsaraDB RDS for MySQL instances, self-managed MySQL databases that are deployed on Elastic Compute Service (ECS) instances, and self-managed MySQL databases that are created in other scenarios. The external storage feature also allows you to write query and analysis results to the MySQL databases for processing. In the following descriptions, the ApsaraDB RDS for MySQL instances are referred to as RDS instances.

Procedure

1. Configure an allowlist for your MySQL database.
   - If you use a MySQL database that is deployed on an RDS instance, add the following CIDR blocks to the allowlist: 100.104.0.0/16, 11.194.0.0/16, and 11.201.0.0/16. For more information, see *Configure an allowlist* in *ApsaraDB RDS User Guide*.
   - If you use a self-managed MySQL database deployed on an ECS instance that resides in a virtual private cloud (VPC) and the ECS instance is added to a security group, configure security group rules to allow access from the following CIDR blocks: 100.104.0.0/16, 11.194.0.0/16, and 11.201.0.0/16. For more information, see *Add security group rules* in *ECS User Guide*.
   - If you use a self-managed MySQL database that is created in other scenarios, add the following CIDR blocks to the allowlist: 100.104.0.0/16, 11.194.0.0/16, and 11.201.0.0/16.
2. Create an external store.
   i. Install the Log Service CLI. For more information, see Aliyun Log Service CLI.
   ii. Create a configuration file named */root/config.json*.
   iii. Add the following script to the */root/config.json* file. Change the parameter values based on your business scenario.

```
{
"externalStoreName":"storename",
"storeType":"rds-vpc",
"parameter":
   {
   "region":"cn-qingdao-env******",
   "vpc-id":"vpc-m5eq4irc1pucp*******",
   "instance-id":"i-m5eeo2whsn*******",
   "host":"localhost",
   "port":"3306",
   "username":"root",
   "password":"****",
   "db":"scmc",
   "table":"join_meta"
   }
}
```

Parameter	Description
externalStoreName	The name of the external store. The name must be in lowercase.
storeType	The type of the data source. Set the value to rds-vpc.
region	The region. ■ If you use a MySQL database that is deployed on an RDS instance, set the region parameter to the region where the RDS instance resides. ■ If you use a self-managed MySQL database deployed on an ECS instance that resides in a VPC, set the region parameter to the region where the ECS instance resides. ■ If you use a self-managed MySQL database that is created in other scenarios, set the region parameter to an empty string. Format: **"region": ""**.
vpc-id	The ID of the VPC. ■ If you use a MySQL database deployed on an RDS instance that resides in a VPC, set the vpc-id parameter to the ID of the VPC where the RDS instance resides. ■ If you use a self-managed MySQL database deployed on an ECS instance that resides in a VPC, set the vpc-id parameter to the ID of the VPC where the ECS instance resides. ■ If you use a MySQL database deployed on an RDS instance that resides in the classic network or if you use a self-managed MySQL database that is created in other scenarios, set the vpc-id parameter to an empty string. Format: **"vpc-id": ""**.
instance-id	The ID of the instance. ■ If you use a MySQL database that is deployed on an RDS instance, set the instance-id parameter to the value of the VpcCloudInstanceId parameter that is specified for the RDS instance. ■ If you use a self-managed MySQL database deployed on an ECS instance that resides in a VPC, set the instance-id parameter to the ID of the ECS instance. ■ If you use a self-managed MySQL database that is created in other scenarios, set the instance-id parameter to an empty string. Format: **"instance-id": ""**.

Parameter	Description
host	The address of your MySQL database. ■ If you use a MySQL database deployed on an RDS instance that resides in a VPC, set the host parameter to an internal endpoint of the RDS instance. ■ If you use a self-managed MySQL database deployed on an ECS instance that resides in a VPC, set the host parameter to the private IP address of the ECS instance. ■ If you use a self-managed MySQL database that is created in other scenarios, set the host parameter to a host address of the database. Make sure that the host address is accessible.
port	The port number. ■ If you use a MySQL database that is deployed on an RDS instance, set the port parameter to the port of the RDS instance. ■ If you use a self-managed MySQL database deployed on an ECS instance that resides on a VPC, set the port parameter to the MySQL service port of the ECS instance. ■ If you use a self-managed MySQL database that is created in other scenarios, set the port parameter to the MySQL service port.
username	The username of the account that you use to log on to your MySQL database.
password	The password of the account that you use to log on to your MySQL database.
db	The name of your MySQL database.
table	The name of the table that you want to use in your MySQL database.

iv. Create an external store.

Replace the value of the project_name parameter with the name of an actual project.

```
aliyunlog log create_external_store --project_name="log-rds-demo" --config="file:///root/config.json"
```

Related operations

- Update the MySQL external store.

```
aliyunlog log update_external_store --project_name="log-rds-demo" --config="file:///root/config.json"
```

- Delete the MySQL external store.

```
aliyunlog log delete_external_store --project_name="log-rds-demo" --store_name=abc
```

What's next

JOIN operations between Logstores and Relational Database Service (RDS) tables

4.11.3. Associate Log Service with an OSS bucket

This topic describes how to create an external store to associate Log Service with an Object Storage Service (OSS) bucket.

Prerequisites

- Logs are collected. For more information, see Data collection.

- The indexing feature is enabled and indexes are configured. For more information, see Configure indexes.
- An OSS bucket is created. For more information, see *Create buckets* in *OSS User Guide*.
- CSV files are uploaded to the OSS bucket. For more information, see *Upload objects* in *OSS User Guide*.

Benefits

The external storage feature that is used to associate Log Service with OSS buckets provides the following benefits:

- Reduced O&M workload: You can perform lightweight association analysis without the need to store all data in one storage system.
- High efficiency: You can use SQL statements to analyze data and view the analysis results within seconds. You can also create charts based on analysis results that are commonly queried. Then, you can click the charts to view the analysis results.

Procedure

1. Log on to the Log Service console
2. In the Projects section, click the project that you want to manage.
3. Choose **Log Storage > Logstores**. On the Logstores tab, click the Logstore that you want to manage.
4. On the page that appears, enter a query statement in the search box and click **Search & Analyze**.

   Execute the following SQL statement to create a virtual external table named user_meta1 and map the table to the OSS object user.csv. If the value of the **result** parameter in the output is **true**, the SQL statement is successfully executed, and an external store is created.

   ```
   * | create table user_meta1 ( userid bigint, nick varchar, gender varchar, province varchar, gende
   r varchar,age bigint) with ( endpoint='oss0036-cn-qingdao-env17-d01-a.intra.env17e.shuguang.com',a
   ccessid='**************************',accesskey ='****************************',bucket='testoss****
   *****',objects=ARRAY['user.csv'],type='oss')
   ```

   Define the name and table schema of the external store in the SQL statement, and define the information that is required to access OSS objects in the WITH clause. The following table describes the parameters.

Parameter	Description
External store name	The name of the external store. The name is the same as the name of the virtual external table. Example: user_meta1.
Table schema	The properties of the virtual external table, including the column names and data types. Example: (userid bigint, nick varchar, gender varchar, province varchar, gender varchar,age bigint).
endpoint	The internal endpoint of OSS. For more information, see Manage a project.
accessid	The AccessKey ID of your account. For more information, see Obtain an AccessKey pair.
accesskey	The AccessKey secret of your account. For more information, see Obtain an AccessKey pair.
bucket	The OSS bucket in which the CSV object is stored.
objects	The path of the CSV object. **Note** The value of the objects parameter is an array. The array can contain multiple elements. Each element represents an OSS object.

Parameter	Description
type	The type of the external store. Set the value to oss.

5. Check whether the external store is created.

   Execute the following statement. If the table content that you defined is returned, the external store is created.

   ```
   * | select * from user_meta1
   ```

6. Perform a JOIN query on Log Service and OSS.

   Execute the following statement to perform a JOIN query. A Logstore is associated with OSS objects based on the ID field in the Logstore and the userid field in the OSS objects. test_accesslog indicates the name of the Logstore. l indicates the alias of the Logstore. user_meta1 indicates the name of the external store that you define. You can configure the parameters based on your business scenario.

   ```
   * | select * from test_accesslog l join user_meta1 u on l.userid = u.userid
   ```

4.11.4. Associate Log Service with a hosted CSV file

Log Service allows you to upload a CSV file from your computer to Log Service by using an SDK. This way, the CSV file is hosted on Log Service and can be associated with a Logstore of Log Service. This topic describes how to perform a JOIN query on a CSV file that is hosted on Log Service and data in a Logstore of Log Service.

Prerequisites

- Logs are collected. For more information, see Data collection.
- Indexes are configured. For more information, see Configure indexes.
- A CSV file is created.
- Log Service SDK for Python is installed. For more information, see *Log Service SDK for Python* in *Log Service Developer Guide*.

  Log Service SDK for Python V0.7.3 and later are supported. You can use the **pip install aliyun-log-python-sdk -U** command to upgrade the SDK.

Limits

- Only one CSV file can be associated at a time.
- After you delete an external store, you cannot create an external store that has the same name as the deleted external store.
- You can associate Log Service with a CSV file that contains no more than 50 MB of data. The CSV file is uploaded to Log Service after it is compressed by using the SDK. The size of the file after compression must be less than 9.9 MB.

Sample data

The Logstore stores the logon operations of a user, and the CSV file records the basic information about the user, such as the gender and age. After you associate the Logstore with the CSV file, you can analyze the metrics for user properties.

- Logstore

  ```
  userid:100001
  action:login
  __time__:1637737306
  ```

- CSV file

userid	nick	gender	province	age
100001	User_A	male	Liaoning	24
100002	User_B	male	Beijing	23
100003	User_C	female	Zhejiang	22
100004	User_D	female	Jiangxi	21
100005	User_E	male	Guangxi	20

Procedure

1. Use Log Service SDK for Python to create an external store.

   For more information about Log Service SDK for Python, see *Log Service SDK for Python* in *Log Service Developer Guide*.

```
from aliyun.log import *
endpoint='data.cn-qingdao-env17-d01.sls-pub.inter.env17e.shuguang.com'
accessKeyId='test-project'
accessKey='TAI****YDw'
project='lr****VM'
ext_logstore='user_meta'
csv_file='./user.csv'
client = LogClient(endpoint, accessKeyId, accessKey)
res = client.create_external_store(project,
    ExternalStoreCsvConfig(ext_logstore, csv_file,
        [
            {"name" : "userid", "type" : "bigint"},
            {"name" : "nick", "type" : "varchar"},
            {"name" : "gender", "type" : "varchar"},
            {"name" : "province", "type" : "varchar"},
            {"name" : "age", "type" : "bigint"}
        ]))
res.log_print()
```

Parameter	Description
endpoint	The Log Service endpoint. For more information, see Obtain an endpoint in Log Service Developer Guide.
accessKeyId	The AccessKey ID that is used to access Log Service. For more information, see Obtain an AccessKey pair. **Warning** We recommend that you use the AccessKey pair of a RAM user to call API operations. This prevents the AccessKey pair of your Apsara Stack tenant account from being leaked.
accessKey	The AccessKey secret that is used to access Log Service. For more information, see Obtain an AccessKey pair.
project	The project to which the Logstore belongs.

Parameter	Description
ext_logstore	The name of the external store. The name is the same as the name of the virtual external table. The name must meet the following requirements: ○ The name can contain only lowercase letters, digits, hyphens (-), and underscores (_). ○ The name must start and end with a lowercase letter or a digit. ○ The name must be 3 to 63 characters in length.
csv_file	The path and name of the CSV file.
Table schema	The properties of the virtual external table, including the column names and data types. The following section shows an example of a table schema. You can replace the table schema based on your business scenario. `[ {"name" : "userid", "type" : "bigint"}, {"name" : "nick", "type" : "varchar"}, {"name" : "gender", "type" : "varchar"}, {"name" : "province", "type" : "varchar"}, {"name" : "age", "type" : "bigint"} ]`

2. Log on to the Log Service console
3. In the Projects section, click the project that you want to manage.
4. Choose **Log Storage > Logstores**. On the Logstores tab, click the Logstore where logs are stored.
5. Execute the following statement to check whether the external store is created.

   In the following statement, `user_meta` specifies the name of the external store. Replace the value with the name that you specified when you created the external store.

   ```
   * | SELECT * FROM user_meta
   ```

   If the content of the CSV file is returned, the external store is created.

userid	nick	gender	province	age
100001	User_A	male	Liaoning	24
100002	User_B	male	Beijing	23
100003	User_C	female	Zhejiang	22
100004	User_D	female	Jiangxi	21
100005	User_E	male	Guangxi	20

6. Execute the following statement to perform a JOIN query on the Logstore and the CSV file.

   The Logstore is associated with the CSV file based on the value of the userid field in the Logstore and the value of the userid field in the CSV file. website_log is the name of the Logstore. user_meta is the name of the external store that you created. You can configure the parameters based on your business scenario.

   ```
   * | SELECT * FROM website_log JOIN user_meta ON website_log.userid = user_meta.userid
   ```

action	userid	_time_	userid	nick	gender	province	age
login	100003	1637738249	100003	User_C	female	Zhejiang	22
login	100004	1637738249	100004	User_D	female	Jiangxi	21
login	100005	1637738249	100005	User_E	male	Guangxi	20
login	100002	1637738249	100002	User_B	male	Beijing	23
login	100004	1637738249	100004	User_D	female	Jiangxi	21
login	100003	1637738249	100003	User_C	female	Zhejiang	22
login	100001	1637738249	100001	User_A	male	Liaoning	24

4.12. Visual analysis

4.12.1. Charts

4.12.1.1. Chart overview

Log Service allows you to render query and analysis results into visualized charts.

Prerequisites

Indexes are configured and the analysis feature is enabled. To enable the analysis feature, turn on **Enable Analytics** for the fields in the **Search & Analysis** panel. For more information, see Enable the index feature and configure indexes for a Logstore.

> Note
>
> - Before you configure charts, we recommend that you are familiar with the log analysis feature. For more information, see Log analysis overview.
> - You must specify an analytic statement in a query statement. Log Service cannot display charts based on query results.

Usage notes

When you execute multiple query statements in sequence, the **Value Column**, **X Axis**, or **Y Axis** configurations are not automatically modified based on the current query statement. The X-axis and Y-axis configurations may remain the same as the configurations in the previous query statement. In this case, the query and analysis result of the current query statement cannot be automatically displayed on a chart. If the following errors occur, you must modify the **Properties** settings based on the current query statement:

- The dimensions that you selected are not in the query results. Check and modify the Properties settings.
- X Axis or Y Axis is unavailable. Check and modify the Properties settings.

Chart configurations

To go to the Graph tab, perform the following steps:

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project that you want to manage.
3. Click the Logstore that you want to manage.
4. Click **Graph** to go to the Graph tab.

On the **Graph** tab, multiple charts are provided to display query and analysis results. You can select a type of chart from the chart bar based on your business requirements.

- On the **Graph** tab, query and analysis results are displayed in the **Chart Preview** and **Data Preview** sections. The **Chart Preview** section shows the query and analysis results that are displayed in the specified type of chart. The **Data Preview** section shows the data of the related chart in a table.
- On the **Graph** tab, you can configure the following settings:

- On the **Data Source** tab, you can specify placeholder variables. For example, you can configure the drill-down event of Chart A to redirect to the dashboard on which Chart B is located. After you configure the drill-down event of Chart A, the placeholder variable is replaced by the variable that you click to trigger the drill-down event and execute the query statement of Chart B. To trigger the drill-down event, you must click the placeholder variable that you configured for Chart B. For more information, see Drill-down analysis.

  This feature is suitable for scenarios in which you want to configure drill-down events to redirect to specified dashboards.

- On the **Properties** tab, you can configure the chart properties that you want to display. You can configure the X-axis, left Y-axis and right Y-axis, margins, and other parameters. Different types of charts have different properties. For more information, see the related topic of each chart.

  This feature is suitable for all query and analysis scenarios.

- On the **Interactive Behavior** tab, you can configure drill-down events for a chart. Then, you can click the variable value in the chart to trigger the specified drill-down event. For more information, see Drill-down analysis.

  This feature is suitable for scenarios in which you want to trigger drill-down events for charts.

Supported chart types

- Table
- Line chart
- Column chart
- Bar chart
- Pie chart
- Area chart
- Individual value plot
- Progress bar
- Map
- Sankey diagram
- Word cloud
- Treemap chart

4.12.1.2. Display query results in a table

Tables are used to sort and display data for quick reference and analysis. All query results that match specified query statements can be rendered into visualized charts. By default, query results are displayed in a table.

Components

- Table header
- Row
- Column

Where:

- The number of columns can be specified by using a `SELECT` statement.
- The number of rows is calculated based on the number of log entries in a specified time range. The default clause is `LIMIT 100`.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.

3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. View the query result.

   By default, the query result is displayed in a table on the **Graph** tab.
6. On the **Properties** tab, configure the properties of the table.

   If you want to modify the rows and columns of the table or the entries to return on each page, you can set the parameters on the Properties tab.

Parameter	Description
Items per Page	The number of entries to return on each page.
Zebra Striping	Specifies whether to display the query result in a zebra-striped table.
Transpose Rows and Columns	Specifies whether to transpose rows and columns.
Hide Reserved Fields	Specifies whether to hide reserved fields, such as `__time__` and `__source__`.
Disable Sorting	Specifies whether to disable the sorting feature.
Disable Search	Specifies whether to disable the search feature.
Highlight Settings	If you turn on Highlight Settings, you can create rules to highlight matched rows or columns.
Sparkline	If you turn on **Sparkline**, you can add an area chart, a line chart, or a column chart for columns in the table.

Example

To query and analyze the distribution of page views (PVs) for different users based on status, execute the following query statement:

```
* |select Status,AlertDisplayName as name,COUNT(*) as count group by Status,name
```

4.12.1.3. Display query results on a line chart

This topic describes how to configure a line chart to display query results.

Background information

Line charts are used to analyze the changes of field values based on an ordered data type. In most cases, the analysis is based on a specified time range. You can use a line chart to analyze the following change characteristics of field values over a specified period of time:

- Increment or decrement
- Increment or decrement rate
- Increment or decrement pattern, for example, periodicity
- Peak value and bottom value

You can use line charts to analyze the changes of field values over a specified period of time. You can also use line charts to analyze the changes of multiple field values in multiple lines over the same period. This way, you can analyze the relationship between different fields. For example, the values of a field are proportional or inversely proportional to the values of another field.

Each line chart consists of the following elements:

- X-axis
- Left Y-axis
- Right Y-axis (optional)
- Data point
- Line of trend changes
- Legend

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the line chart.

Parameter	Description
X Axis	The sequential data. In most cases, a time series is selected.
Left Y Axis	The numeric data. You can select one or more fields for the left Y-axis.
Right Y Axis	The numeric data. You can select one or more fields for the right Y-axis. The layer of the right Y-axis is higher than the layer of the left Y-axis.
Column Marker	The column on the left or right Y-axis. The column is displayed as a histogram.
Legend	The position of the legend in the chart. Valid values: Top, Bottom, Left, and Right.
Format Left Y-axis	The format in which the data on the left and right Y-axis is displayed.
Format Right Y-axis	
Line Type	The type of line that is displayed in the line chart. Valid values: **Straight Line** and **Curve**.
Anomaly Point Column	The column where anomaly points are located. You can set the **Anomaly Point Lower Limit** and **Anomaly Point Upper Limit** parameters for a column. ○ **Anomaly Point Lower Limit**: Values that are less than the lower limit are highlighted in red. ○ **Anomaly Point Upper Limit**: Values that exceed the upper limit are highlighted in red.
Upper Limit Column	The area that is formed based on the values.
Lower Limit Column	
Time Series	A series of data points that are listed in chronological order.
Time Format	The time format of the time series fields.

Parameter	Description
Margin	The distance between the axis and the borders of the chart. The parameters include Top Margin, Bottom Margin, Left Margin, and Right Margin.

Note Each line in a line chart must contain more than two data points. Otherwise, the data trend cannot be generated. We recommend that you select no more than five lines for a line chart.

Example of a simple line chart

To query the page views (PVs) of the IP address `203.0.113.10` in the previous 24 hours, execute the following query statement:

```
remote_addr: 203.0.113.10 | 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 Left Y Axis. Set the Legend parameter and adjust the margins based on your business requirements.

Example of a dual Y-axis line chart

To query the PVs and number of unique visitors (UVs) in the previous 24 hours, execute the following query statement:

```
* | select date_format(date_trunc('hour', __time__), '%m-%d %H:%i') as time, count(1) as PV, approx_di
stinct(remote_addr) 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.

4.12.1.4. Display query results on a column chart

This topic describes how to configure a column chart to display query results.

Background information

A column chart uses vertical or horizontal bars to show the values of different categories. You can use a column chart to count the number of values in each category.

Each column chart consists of the following elements:

- X-axis (horizontal)
- Y-axis (vertical)
- Rectangular bar
- Legend

By default, column charts in Log Service use vertical bars. Each rectangular bar has a fixed width and a variable height that indicates a value. If you select multiple columns of data for the Y-axis, a grouped column chart is used to display the data.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.

5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the column chart.

> **Note** If a query statement returns no more than 20 log entries, you can use a column chart to display the query results. You can use a `LIMIT` clause to limit the number of rectangular bars. If the chart contains a large number of rectangular bars, the analysis results may not be displayed as expected. We recommend that you select no more than five fields for the Y-axis.

Parameter	Description
X Axis	The categorical data.
Y Axis	The numeric data. You can select one or more fields for the left Y-axis.
Legend	The position of the legend in the chart. Valid values: Top, Bottom, Left, and Right.
Format	The format in which the data on the Y-axis is displayed.
Margin	The distance between the axis and the borders of the chart. The parameters include Top Margin, Bottom Margin, Left Margin, and Right Margin.

Example of a simple column chart

To query the number of visits for each `http_referer` in the specified time range, execute the following query statement:

```
* | select http_referer, count(1) as count group by http_referer
```

Select `http_referer` for X Axis and `count` for Y Axis.

Example of a grouped column chart

To query the number of visits and the average bytes for each `http_referer` in the specified time range, execute the following query 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.

4.12.1.5. Display query results on a bar chart

This topic describes how to configure a bar chart to display query results.

Background information

A bar chart is a horizontal column chart that is used to analyze the top N values of fields. You can configure a bar chart in a similar manner in which you configure a column chart.

Each bar chart consists of the following elements:

- X-axis (vertical)
- Y-axis (horizontal)
- Rectangular bar
- Legend

Each rectangular bar has a fixed height and a variable width. The variable width indicates a value. If multiple columns of data are mapped to the Y-axis, you can use a grouped bar chart to display the data.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the bar chart.

> **Note**
>
> - If a query statement returns no more than 20 log entries , you can use a bar chart to display the query results. You can also use a LIMIT clause to limit the number of rectangular bars. If the chart contains a large number of rectangular bars, analysis results may not be displayed as expected. You can also use an `ORDER BY` clause to analyze Top N values of fields. We recommend that you select no more than five fields for the Y-axis.
> - You can use a grouped bar chart to display query results. However, the values represented by each rectangular bar in a group must be positively or negatively associated with each other.

Parameter	Description
X Axis	The categorical data.
Y Axis	The numeric data. You can select one or more fields for the Y-axis.
Legend	The position of the legend in the chart. Valid values: Top, Bottom, Left, and Right.
Format X-axis	The format in which the data on the X-axis is displayed.
Margin	The distance between the axis and the borders of the chart. The parameters include Top Margin, Bottom Margin, Left Margin, and Right Margin.

Example

To analyze the top 10 request URIs (`request_uri`) that are most frequently visited, execute the following query statement:

```
* | select request_uri, count(1) as count group by request_uri order by count desc limit 10
```

4.12.1.6. Display query results on a pie chart

This topic describes how to configure a pie chart to display query results.

Background information

A pie chart is used to show the percentages of different categories of data. The arc length of each segment in a pie chart is proportionate to the quantity that is represented by each category. A pie chart is divided into multiple segments based on the percentages of categories. Each segment shows the percentage of a category. The sum of all percentages is equal to 100%.

Each pie chart consists of the following elements:

- Segment
- Percentage in the text format
- Legend

Types

Log Service provides the following types of standard pie charts: pie chart, donut chart, and polar area chart.

- Donut chart

  A donut chart is a variant of a pie chart that has a hollow center. Compared with a pie chart, a donut chart provides the following advantages:

  - Displays more information, such as the total number of occurrences of all field values.
  - Allows you to compare data between two donut charts based on ring lengths. Data across different pie charts is difficult to compare.

- Polar area chart

  A polar area chart is a column chart in the polar coordinate system. Each category of data is represented by a segment with the same angle, and the radius of each segment varies based on the value. Compared with a pie chart, a polar area chart provides the following advantages:

  - If a query statement returns no more than 10 log entries, you can use a pie chart to display the query results. If a query statement returns 10 to 30 log entries, you can use a polar area chart to display the query results.
  - Enlarges the differences among the values of categories because the area of the segment correlates with the square of the radius. Therefore, the polar area chart is suitable for the comparison of similar values.
  - A circle can be used to display a periodic pattern. Therefore, you can use a polar area chart to analyze value changes in different periods, such as weeks and months.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the pie chart.

> **Note**
>
> - If a query statement returns no more than 10 log entries, you can use a pie chart or donut chart to display the query results. You can use a `LIMIT` clause to limit the number of segments. If a chart contains a large number of segments with different colors, the analysis results may not be displayed as expected.
> - If the number of log entries exceeds 10, we recommend that you use a polar area chart or column chart.

Parameter	Description
Chart Types	The type of the chart. Valid values: Pie Chart, Donut Chart, and Polar Area Chart.
Legend Filter	The categorical data.
Value Column	The values that correspond to different categories of data.
Legend	If you turn on **Show Legend**, you can set this parameter to adjust the position of the legend in the chart.
Format	The format in which data is displayed.
Tick Text Format	Valid values: **Percentage** and **Category: Percentage**.
Margin	The distance between the axis and the borders of the chart. The parameters include Top Margin, Bottom Margin, Left Margin, and Right Margin.

Example of a pie chart

To analyze the percentages of the `requestURI` field values, execute the following query statement:

```
* | select requestURI as uri , count(1) as c group by uri limit 10
```

Example of a donut chart

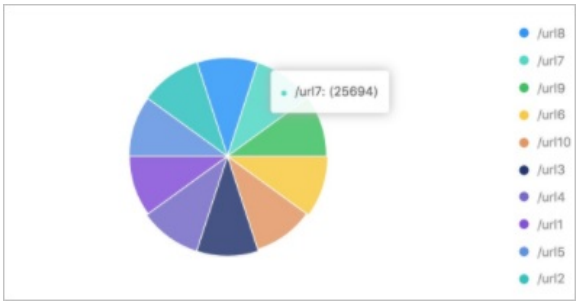
To analyze the percentages of the `requestURI` field values, execute the following query statement:

```
* | select requestURI as uri , count(1) as c group by uri limit 10
```

Example of a polar area chart

To analyze the percentages of the `requestURI` field values, execute the following query statement:

```
* | select requestURI as uri , count(1) as c group by uri limit 10
```



4.12.1.7. Display query results on an area chart

This topic describes how to configure an area chart to display query results.

Background information

An area chart is built based on a line chart. The colored section between a line and the axis is an area. The color is used to highlight the trend. An area chart is similar to a line chart and shows the changes of numeric values over a specified period of time. An area chart is used to highlight the overall data trend. Both line charts and area charts display the trend and relationship between numeric values instead of specific values.

Each area chart consists of the following elements:

- X-axis (horizontal)
- Y-axis (vertical)
- Area segment

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the icon.
6. On the **Properties** tab, configure the properties of the area chart.

> **Note** In an area chart, a single area segment must contain more than two data points. If a single area segment contains two or fewer data points, the data trend cannot be analyzed. We recommend that you select less than five area segments in an area chart.

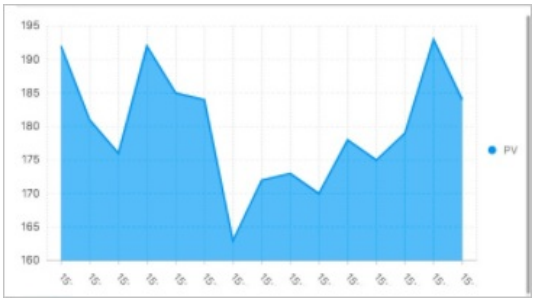
Parameter	Description
X Axis	The sequential data. In most cases, a time series is selected.
Y Axis	The numeric data. You can select one or more fields for the Y-axis.
Legend	The position of the legend in the chart. Valid values: Top, Bottom, Left, and Right.
Format	The format in which data is displayed.
Margin	The distance between the axis and the borders of the chart. The parameters include Top Margin, Bottom Margin, Left Margin, and Right Margin.

Example of a simple area chart

To query the page views (PVs) of the IP address `203.0.113.10` in the previous 24 hours, execute the following query statement:

```
remote_addr: 203.0.113.10 | select date_format(date_trunc('hour', __time__), '%m-%d %H:%i') as time, c
ount(1) as PV group by time order by time limit 1000
```

Select `time` for X Axis and `PV` for Y Axis.

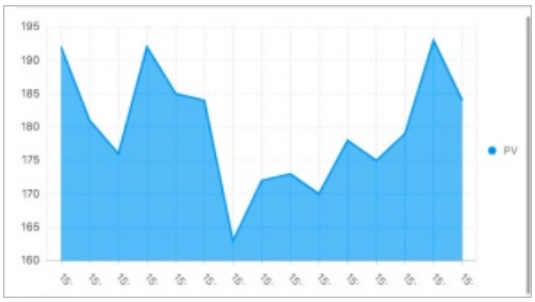


Example of a cascade chart

To query the number of PVs and the number of unique visitors (UVs) for each hour within one day, execute the following query statement:

```
* | select date_format(date_trunc('hour', __time__), '%m-%d %H:%i') as time, count(1) as PV, approx_di
stinct(remote_addr) as UV group by time order by time limit 1000
```

Select `time` for X Axis. Select `PV` and `UV` for Y Axis.



4.12.1.8. Display query results on a single value chart

This topic describes how to configure a single value chart to display query results.

Background information

A single value chart highlights a single value. Log Service supports the following types of single value charts:

- Rectangle Frame: shows a general value.
- Dial: shows the difference between the current value and a specified threshold value.
- Compare Numb Chart: shows the SQL query results of interval-valued comparison and periodicity-valued comparison functions. For more information, see Interval-valued comparison and periodicity-valued comparison functions.

By default, Rectangle Frame is selected. Rectangle Frame is the most basic type of single value chart to display data at a specified point. In most cases, this chart type is used to show the key information at a specified point in time. To display a proportional metric, you can select Dial.

Each single value chart consists of the following elements:

- Numeric value
- Unit (optional)
- Description (optional)
- Chart type

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the 123 icon.
6. On the **Properties** tab, configure the properties of the single value chart.

> **Note** Log Service normalizes data in charts that contain numeric values. For example, `230000` is processed as `230K`. You can include mathematical calculation functions in query statements to customize numeric formats. For more information, see Mathematical calculation functions.

- The following table describes the parameters of a rectangle frame.

Parameter	Description
Chart Types	The type of the single value chart. If you select **Rectangle Frame**, query results are displayed in a rectangle frame.
Value Column	The value that is displayed in the chart. By default, the data in the first row of the specified column is displayed.
Unit	The unit of the data.
Unit Font Size	The font size of the unit. You can drag the slider to adjust the font size. Valid values: 10 to 100. Unit: pixels.
Description	The description of the value.
Description Font Size	The font size of the description. You can drag the slider to adjust the font size. Valid values: 10 to 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 to 100. Unit: pixels.
Font Color	The color of the value, unit, and description in the chart. You can use the default color or select a color.
Background Color	The color of the background. You can use the default color or select a color.

- The following table describes the parameters of a dial.

Parameter	Description
Chart Types	The type of the single value chart. If you select **Dial**, query results are displayed on a dial.

Parameter	Description
Actual Value	The value that is displayed in the chart. By default, the data in the first row of the specified column is displayed.
Unit	The unit of the value on the dial.
Font Size	The font size of the value and unit. Valid values: 10 to 100. Unit: pixels.
Description	The description of the value.
Description Font Size	The font size of the description. You can drag the slider to adjust the font size. Valid values: 10 to 100. Unit: pixels.
Dial Maximum	The maximum value of the scale on the dial. Default value: 100.
Use Query Results	If you turn on Use Query Results, Dial Maximum is replaced by Maximum Value Column. Then, you can select the maximum value from query results for this parameter.
Format	The format in which data is displayed.
Colored Regions	The number of segments that divide the dial. Each segment is displayed in a different color. Valid values: 2, 3, 4, and 5. Default value: 3.
Region Max Value	The maximum value of the scale in each colored segment of the dial. By default, the maximum value in the last segment is the maximum value on the dial. You do not need to specify this value. **Note** By default, a dial is evenly divided into three colored segments. If you change the value of **Colored Regions**, Region Max Value is not automatically adjusted. You can manually set the maximum value for each colored segment based on your business requirements.
Font Color	The color of the value on the dial.
Region	The colored segments that divide the dial. By default, a dial is evenly divided into three segments. The segments are displayed in blue, yellow, and red. If you set **Colored Regions** to a value greater than 3, the added segments are displayed in blue by default. You can change the color of each segment.

- The following table describes the parameters of a compare numb chart.

Parameter	Description
Chart Types	The type of the single value chart. If you select Compare Numb Chart, query results are displayed on a compare numb chart.
Show Value	The value that is displayed in the center of the compare numb chart. In most cases, this value is set to the statistical result that is calculated by the related comparison function in the specified time range.

Parameter	Description
Compare Value	The value that is compared with the threshold. In most cases, this value is set to the result of the comparison between the statistical results that are calculated by the related comparison function in the specified time range and in the previously specified time range.
Font Size	The font size of the show value. Valid values: 10 to 100. Unit: pixels.
Unit	The unit of the show value.
Unit Font Size	The font size of the unit for the show value. Valid values: 10 to 100. Unit: pixels.
Compare Unit	The unit of the compare value.
Compare Font Size	The font size of the compare value and unit. Valid values: 10 to 100. Unit: pixels.
Description	The description of the show value and its growth trend.
Description Font Size	The font size of the description. Valid values: 10 to 100. Unit: pixels.
Trend Comparison Threshold	The value that is used to measure the variation trend of the compare value. For example, the compare value is -1. ■ If you set **Trend Comparison Threshold** to 0, a down arrow that indicates a value decrease is displayed on the page. ■ If you set **Trend Comparison Threshold** to -1, the value remains unchanged. The system does not display the trend on the page. ■ If you set **Trend Comparison Threshold** to -2, an up arrow that indicates a value increase is displayed on the page.
Format	The format in which data is displayed.
Font Color	The color of the show value and its description.
Growth Font Color	The font color of the compare value that is greater than the threshold.
Growth Background Color	The background color that is displayed when the compare value is greater than the threshold.
Decrease Font Color	The font color that is displayed when the compare value is less than the threshold.
Decrease Background Color	The background color that is displayed when the compare value is less than the threshold.
Equal Background Color	The background color that is displayed when the compare value is equal to the threshold.

Examples

To view the number of page views (PVs), execute the following query statements. The analysis results are displayed in charts.

- Rectangle frame

  To view the number of PVs in the previous 15 minutes, execute the following query statement:

```
* | select count(1) as pv
```

- Dial

  To view the number of PVs in the previous 15 minutes, execute the following query statement:

  ```
  * | select count(1) as pv
  ```

- Compare numb chart

  To view and compare the PVs on the current day and in the previous day, execute the following query statement:

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

4.12.1.9. Display query results on a progress bar

This topic describes how to configure a progress bar to display query results.

Background information

A progress bar shows the percentage of the actual value of a field to the maximum value of the field. You can configure the properties of a progress bar to change the style and configure display rules for the progress bar.

Each progress bar consists of the following elements:

- Actual value
- Unit (optional)
- Total value

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the progress bar.

Parameter	Description
Actual Value	The actual value in the chart. By default, data in the first row of the specified column is displayed.
Unit	The unit of the value in the progress bar.
Total Value	The maximum value indicated by the progress bar. Default value: 100.
Maximum Value Column	The maximum value in the specified column. If you turn on **Use Query Results**, **Total Value** is replaced by **Maximum Value Column**. Then, you can select the maximum value from the query results for this parameter.

Parameter	Description
Use Query Results	If you turn on Use Query Results, Total Value is replaced by Maximum Value Column. Then, you can select the maximum value from the query results for this parameter.
Edge Shape	The edge shape of the progress bar.
Vertical Display	Specifies whether to display the progress bar in vertical display mode.
Font Size	The font size of the value in 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 value in the progress bar.
Default Color	The default color of the progress bar.
Color Display Mode	The display mode of the progress bar.
Start Color	The start color of the progress bar. This parameter is available if you select **Gradient** for **Color Display Mode**.
End Color	The end color of the progress bar. This parameter is available if you select **Gradient** for **Color Display Mode**.
Display Color	The display color of the progress bar. This parameter is available if you select **Display by Rule** for **Color Display Mode**. **Note** The value of **Actual Value** is compared with the value 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. If the actual value does not match the condition, the progress bar is displayed in the default color.
Operator	The condition that is used to determine whether to display the progress bar in the color specified by Display Color. This parameter is available if you select **Display by Rule** for **Color Display Mode**.
Threshold	The threshold based on which the color of the progress bar is determined. This parameter is available if you select **Display by Rule** for **Color Display Mode**.

Example

To calculate the ratio of the page views (PVs) of the current hour to the PVs of the same period of time on the previous day, execute the following query statement:

```
* | SELECT diff[1] AS today, diff[2] AS yesterday, diff[3] AS ratio FROM (SELECT compare(PV,86400) AS diff FROM (SELECT count(*) AS PV FROM log))
```

4.12.1.10. Display query results on a map

This topic describes how to configure a map to display query results.

Background information

You can color and mark a map to display geographic data. Log Service provides the map of China. The display modes of an AMap include the anchor point and heat map. You can use specific functions in query statements to display analysis results as maps.

Each map consists of the following elements:

- Map canvas
- Colored area

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analytics**.
5. On the **Graph** tab, use the map of China, click the  icon..
6. On the **Properties** tab, configure the properties of the map.

   In this example, the map of China is selected.

Parameter	Description
Provinces	The location information that is recorded in logs. The information is displayed in one of the following dimensions based on the map type:
Value Column	The amount of data at the location.
Show Legend	If you turn on Show Legend, the legend information is displayed.

Example of a map of China

To display query results on a map of China, you can execute the following query statement in which the ip_to_province function is used:

```
* | select ip_to_province(remote_addr) as address, count(1) as count group by address order by count desc limit 10
```

Select *address* for Provinces and *count* for Value Column.

4.12.1.11. Display query results in a Sankey diagram

This topic describes how to configure a Sankey diagram to display query results.

Background information

A Sankey diagram is a type of flow chart. A Sankey diagram shows the flow from one set of values to another set of values. You can use Sankey diagrams to collect statistics about network traffic flows. A Sankey diagram contains the values of the `source`, `target`, and `value` fields. The `source` and `target` fields describe the source and target nodes, and the `value` field describes the flows from the `source` node to the `target` node.

Each Sankey diagram consists of the following elements:

- Node
- Edge

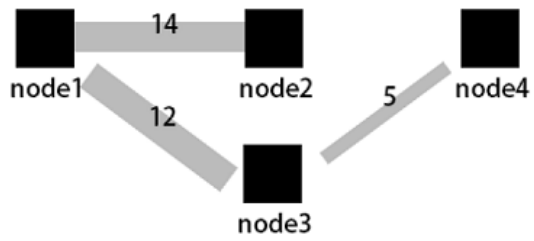
A Sankey diagram has the following features:

- The start flow is equal to the end flow. The sum of the widths of all main edges is equal to the sum of the widths of all branch edges. This allows you to manage and maintain a balanced flow of all traffic.
- The edge width in a row represents the volume of traffic in a specific status.
- The width of an edge between two nodes represents the flow volume in a status.

The following table describes the data that can be displayed in a Sankey diagram.

source	target	value
node1	node2	14
node1	node3	12
node3	node4	5

The following figure shows the data relationships in a Sankey diagram.



Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the icon.
6. On the **Properties** tab, configure the properties of the Sankey diagram.

Parameter	Description
Start Column	The start node.
End Column	The end node.

Parameter	Description
Value Column	The volume of traffic between the start node and end node.
Margin	The distance between an axis and the borders of the chart. The parameters include **Top Margin**, **Bottom Margin**, **Right Margin**, and **Left Margin**.

Example

If a log contains the `source`, `target`, and `value` fields, you can use a nested subquery to obtain the sum of all `steamValue` values.

```
* | select sourceValue, targetValue, sum(streamValue) as streamValue from (select sourceValue, targetV
alue,
 streamValue, __time__ from log group by  sourceValue, targetValue, streamValue, __time__ order by __t
ime__ desc) group by sourceValue,
 targetValue
```

4.12.1.12. Display query results on a word cloud

This topic describes how to configure a word cloud to display query results.

Background information

A word cloud shows text data. A word cloud is a cloud-like and colored image composed of words. You can use a word cloud to display a large amount of text data. The font size or color of a word indicates the significance of the word. This allows you to identify whether a word is significant in an efficient manner.

The words in a word cloud are sorted.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the word cloud.

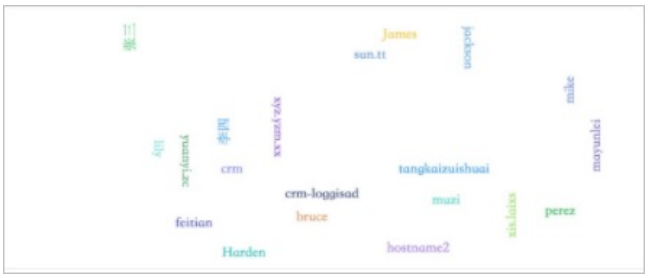
Parameter	Description
Word Column	The words that you want to display.
Value Column	The numeric value that corresponds to a word.
Font Size	The font size of a word. ○ The minimum font size ranges from 10 pixels to 24 pixels. ○ The maximum font size ranges from 50 pixels to 80 pixels.

Example

To query the distribution of hostnames in NGINX logs, execute the following query statement:

```
* | select hostname, count(1) as count group by hostname order by count desc limit 1000
```

Select `hostname` for Word Column and `count` for Value Column.



4.12.1.13. Display query results on a treemap chart

This topic describes how to configure a treemap chart to display query results.

Background information

A treemap chart consists of multiple rectangles that represent the data volumes. A larger rectangle area represents a larger proportion of the categorical data.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project in which you want to configure a chart.
3. Click the name of the Logstore in which you want to configure a chart.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, click the  icon.
6. On the **Properties** tab, configure the properties of the treemap chart.

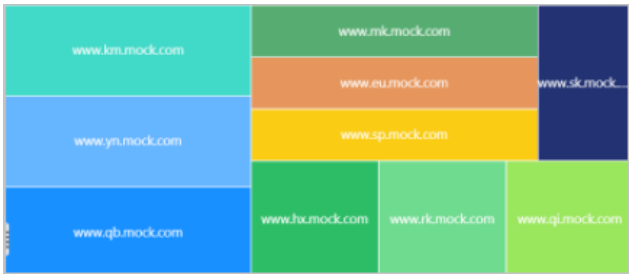
Parameter	Description
Legend Filter	The field that includes categorical data.
Value Column	The numeric value of a field. A greater field value represents a larger rectangle.

Example

To query the distribution of hostnames in NGINX logs, execute the following query statement:

```
* | select host, count(1) as count group by host order by count desc limit 1000
```

Select `host` for **Legend Filter** and select `count` for **Value Column**.



4.12.2. Dashboard

4.12.2.1. Overview

A dashboard provided by Log Service is a platform where you can analyze data in real time. You can add multiple charts to a dashboard for data analysis. Each chart is a visualized search and analytic statement.

A dashboard allows you to view the charts of multiple search and analytic statements at one time. When you open or refresh the dashboard, the statements of the charts run automatically.

After you add a chart to a dashboard, you can configure Drill-down analysis for the chart. Then you can click the chart on the dashboard to further analyze data and obtain more fine-grained analysis results.

Limits

- You can create a maximum of 50 dashboards for a project.
- Each dashboard can contain a maximum of 50 analysis charts.

Features

A dashboard has two modes: display mode and edit mode.

- Configure the display mode of a dashboard

  In the display mode, you can configure multiple display settings on the dashboard page.

  - Dashboard: You can specify the time range, the automatic refresh interval, full screen, and the display mode of the title for the dashboard, configure alerts for all charts on the dashboard, and filter chart data based on the Configure and use a filter on a dashboard of a Logstore.
  - Chart: You can view the analysis details of a specified chart, specify the time range and configure alerts for the chart, download logs and the chart, and check whether drill-down analysis is configured for the chart.

- Edit mode

  In the edit mode, you can change the configurations of the dashboard and charts.

  - Dashboard: You can use a dashboard as a canvas and add Markdown chart, custom charts, text, icons, and other chart elements to the dashboard. You can also add lines between chart elements that are self-adaptive to the positions of the charts. You can also add Configure and use a filter on a dashboard of a Logstore, which can be used to filter chart data in the display mode. In addition, you can configure display gridlines to help arrange chart elements such as icons in an orderly manner.
  - Chart: You can also edit a chart on the dashboard. You can modify the statement, properties, and interactive behavior such as drill-down analysis of the chart.

4.12.2.2. Create and delete a dashboard

This topic describes how to create a dashboard. After you create a dashboard, you can follow, view, and delete the dashboard.

Prerequisites

The indexing feature is enabled and indexes are configured. For more information, see Enable the indexing feature and configure indexes for a Logstore.

Create a dashboard

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project in which you want to create a dashboard.
3. In the left-side navigation pane, click the  icon.

4. Click the plus sign (+) to create a dashboard.
5. In the **Add to New Dashboard** dialog box, enter a name for the dashboard in the Dashboard Name field and click **OK**.

Related operations

After you create a dashboard, you can follow, view, and delete the dashboard.

- In the Dashboard list, find the dashboard that you created and choose > **Delete** to delete the dashboard.

  > **Notice** After you delete a dashboard, the dashboard cannot be restored. Proceed with caution.

- In the Dashboard list, find the dashboard that you created and choose > **Details** to view the dashboard.

4.12.2.3. Manage a dashboard in display mode

This topic describes how to configure a dashboard in display mode. By default, a dashboard shows all charts in display mode. You can perform multiple operations on a dashboard in display mode.

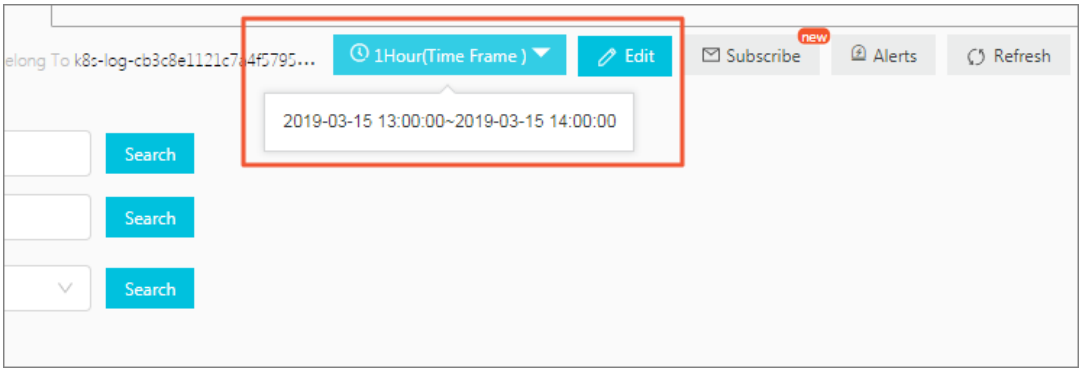
Specify a time range for a dashboard

By default, the time range that you specify for a dashboard applies to all charts on the dashboard. After you specify a time range for a dashboard, all charts on the dashboard display the query and analysis results of the time range. For information about how to specify a time range for a single chart, see Specify a time range for a chart.

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project in which you want to manage a dashboard.
3. In the left-side navigation pane, click the icon.
4. In the Dashboard list, click the dashboard that you want to manage.
5. Click **Please Select** to specify a time range.

   Log Service supports the following types of time ranges:

   - Relative: queries log data that is generated within a time range that ends with the current time, such as the previous 1, 5, or 15 minutes. The time range is accurate to seconds. For example, if the current time is 19:20:31 and you select 1Hour(Relative) as the time range, the charts on the dashboard display the log data that is generated from 18:20:31 to 19:20:31.
   - Time Frame: queries log data that is generated within a time range that ends with the current time, such as the previous 1 or 15 minutes. The time range is accurate to minutes, hours, or days. For example, if the current time is 19:20:31 and you select 1Hour(Time Frame) as the time range, the charts on the dashboard display the log data that is generated from 18:00:00 to 19:00:00.
   - Custom: queries log data that is generated within a custom time range.
6. Mover the pointer over the **Please Select** button to confirm the specified time range.



Enter the edit mode

In the Dashboard list, click the dashboard that you want to modify. On the page that appears, click **Edit** to enter the edit mode. In edit mode, you can perform multiple operations on the dashboard and the charts on the dashboard. For more information, see Edit mode.

Configure an alert rule

In the upper-right corner of the dashboard page, click **Alerts** and select **Create** to create an alert rule for the charts on the dashboard. For more information, see Configure an alert rule.

Configure a refresh method

You can manually refresh a dashboard or select an interval to automatically refresh the dashboard.

- In the upper-right corner of the dashboard page, choose **Refresh > Once**. The dashboard is immediately refreshed.
- In the upper-right corner of the dashboard page, choose **Refresh > Auto Refresh** and select an interval at which the dashboard is automatically refreshed.

  The interval can be 15 seconds, 60 seconds, 5 minutes, or 15 minutes.

  > **Note** If your browser is inactive, the dashboard may not refresh at the specified interval as expected.

Share a dashboard

In the upper-right corner of the dashboard page, click **Share** to copy the link of the dashboard to the clipboard. Then, you can send the link to authorized users. The shared dashboard page uses the settings of the dashboard at the point in time when you share the dashboard. The settings include the time range of charts and the display format of chart titles.

> **Note** Before you share a dashboard with other users, you must grant the read permissions to the users.

Display a dashboard in full screen

In the upper-right corner of the dashboard page, click **Full Screen**. Then, the dashboard enters the full-screen mode. This mode is suitable for scenarios such as presentations and reporting.

Select a display format for chart titles

In the upper-right corner of the dashboard page, click **Title Configuration** to select a display format for chart titles. Valid values:

- Single-line Title and Time Display
- Title Only
- Time Only

Reset the time range

In the upper-right corner of the dashboard page, click **Reset Time** to restore the saved time range of all charts on the dashboard. You can use this feature to restore time settings.

Configure charts

You can select a chart and perform the following operations on the chart.

> **Note** Different types of charts on a dashboard can display different information. You cannot view the analysis details of non-statistical charts such as custom charts and Markdown charts.

- View analysis details

  Find the chart whose details you want to view, click the ⋮ icon and select **View Analysis Details**. On the page that appears, you can view the query statement and the properties of the chart.

- Specify a time range for a chart

  Find the chart that you want to manage, click the ⋮ icon and select **Select Time Range** to specify a time range for the chart.

- Configure an alert rule for a chart

  Find the chart for which you want to configure an alert rule, click the ⋮ icon and select **Create Alert** to create an alert rule for the chart. For more information, see Configure an alert rule.

- Download log data

  Find the chart whose log data you want to download, click the ⋮ icon and select **Download Log**. The log data that is returned by the query statement of the chart within the current time range is downloaded in a comma-separated values (CSV) file.

- Check whether a drill-down event is configured for a chart

  Find the chart that you want to check, click the ⋮ icon and move the pointer over the  icon to check whether a drill-down event is configured for the chart. If the icon is red, a drill-down event is configured for the chart.

- Preview the query statement of a chart

  Find the chart whose query statement you want to preview, click the ⋮ icon, and then click the 👁 icon. In the Preview Query Statement dialog box, you can view the query statement of the chart.

4.12.2.4. Manage a dashboard in edit mode

You can manage a dashboard in edit mode. For example, you can add chart elements, adjust chart layouts, edit charts, and change the dashboard name.

Add chart elements

1. In the Projects section, click the project in which you want to modify a dashboard.
2. In the left-side navigation pane, click the  icon.
3. In the Dashboard list, click the dashboard that you want to manage.
4. In the upper-right corner of the dashboard page, click **Edit**.

   You can add the following chart elements on a dashboard in edit mode.

> **Notice** If you modify a dashboard in edit mode, you must save the modifications before the modifications can take effect. To save modifications, click **Save** in the upper-right corner of the dashboard page.

- Rectangles and diamonds

  Drag the rectangular icon or the diamond icon to a position. Then, double-click the icon and enter text. You can also modify the text properties and the border properties of rectangles and diamonds.

- Common icons

  Log Service allows you to display common icons on a dashboard page. You can drag an icon to a position.

- Text

  Drag the text icon to a position. Then, double-click the text box and enter text. You can also modify the properties of the text. The properties include the font size, font style, alignment, and font color.

- Markdown chart

  Drag the Markdown icon to a position. Then, double-click the text box and insert elements such as text, charts, and videos. For more information, see Markdown chart.

- Filter

  Click the filter icon to add a filter. For more information, see Add a filter.

  After you add a filter to a dashboard, you can use the filter to refine search results or replace placeholder variables in query statements.

- Custom SVG

  Click the SVG icon. In the Customize SVG dialog box, click the box or drag a Scalable Vector Graphics (SVG) file to the box to upload the file.

  > **Note** The size of an SVG file cannot exceed 10 KB.

- Custom image's HTTP link

  Click the Customize image's HTTP link icon in the menu bar. On the page that appears, enter the HTTP link of an image and click OK.

Adjust chart layouts

On a dashboard in edit mode, all charts and chart elements are displayed on a canvas. You can drag and scale each chart. The width of the canvas cannot exceed the width of your browser. The height of the canvas is unlimited and is measured in pixels.

On the canvas, you can perform the following operations:

- Adjust the position of a chart.
  - You can drag a chart to a position.
  - You can select a chart and set the **L** and **T** parameters to adjust the chart position.
- Adjust the width and height of a chart.
  - You can select a chart and drag the lower-right corner of the chart to resize the chart.
  - You can select a chart and set the **W** and **H** parameters to resize the chart.
- Add lines to connect charts.

  You can add a directional line between two charts. When you adjust the position or size of the charts, the line automatically moves to show the relative position between the two charts.
- Configure chart levels.

You can select a chart and click the Move Layer to Top icon or the Move Layer to Down icon in the menu bar to move the chart to the upper part or lower part of the dashboard.

Configure charts

You can modify, copy, and delete a chart on a dashboard in edit mode.

- Modify the query statement, properties, data source, and interactive behavior for a chart.
  i. In the upper-right corner of the dashboard page, click Edit to enter the edit mode. Find the chart that you want to modify and choose > **Edit**.
  ii. Modify the query statement, properties, data source, and interactive behavior for the chart.
  For information about how to configure interactive behavior for a chart, see Drill-down analysis.
  iii. Click **Preview** to check the configuration results.
  iv. Click **OK**.
  v. In the upper-right corner of the dashboard page, click **Save**.
- Create a copy of a chart. The copy uses the same configurations as the chart.
  i. In the upper-right corner of the dashboard page, click Edit to enter the edit mode. Find the chart that you want to copy and choose > **Copy**.
  ii. Drag the copy to a position. Then, specify the margins and size of the copy.
  iii. In the upper-right corner of the dashboard page, click **Save**.
- Delete a chart.
  i. In the upper-right corner of the dashboard page, click Edit to enter the edit mode. Find the chart that you want to delete and choose > **Delete**.
  ii. In the upper-right corner of the dashboard page, click **Save**.

4.12.2.5. Configure a drill-down event

Log Service allows you to configure a drill-down event for a chart to obtain more details in analysis results. This topic describes how to configure a drill-down event in the Log Service console.

Prerequisites

- The indexing feature is enabled and indexes are configured. For more information, see Configure indexes.
- A Logstore is created. This prerequisite must be met if you want to configure a drill-down event to open a Logstore. For more information, see Create a Logstore.
- A saved search is created. This prerequisite must be met if you want to configure a drill-down event to open a saved search. For more information, see Saved search.
  Placeholder variables are configured in the query statement of the saved search. This prerequisite must be met if you want to configure variables. For more information, see Configure a placeholder variable.
- A dashboard is created. This prerequisite must be met if you want to configure a drill-down event to open a dashboard. For more information, see Create a dashboard.
  Placeholder variables are configured in the related chart on the dashboard. This prerequisite must be met if you want to configure variables. For more information, see Configure a placeholder variable.
- If you want to configure a drill-down event to open a custom HTTP URL, you must create the HTTP URL.

Context

Drilling is required for data analysis. This feature allows you to analyze data in a fine-grained manner or coarse-grained manner. Drilling includes roll-up and drill-down. Drill-down allows you to obtain more details in analysis results. This way, you extract more value from data and make better decisions for your business.

Procedure

1. Log on to the Log Service console.
2. Click the name of the project that you want to manage.
3. Click the icon next to the name of the Logstore in which you want to query and analyze data, and then select **Search & Analysis**.
4. Enter a query statement in the search box, specify a time range, and then click **Search & Analyze**.
5. On the **Graph** tab, select a chart type. On the Properties tab, configure the parameters.

   For information about the parameters of a chart, see Chart configurations.
6. Click the **Interactive Behavior** tab. On this tab, configure a drill-down event for the chart.

   You can set the Event Action parameter to Disable, Open Logstore, Open Saved Search, Open Dashboard, Open Dashboard, or Custom HTTP Link.

   - **Disable**: disables the drill-down feature.
   - **Open Logstore**: configures the drill-down event to open a Logstore. The following table describes the parameters that you can configure if you set the Event Action parameter to Open Logstore.

Parameter	Description
Open in New Tab	If you turn on this switch, the Logstore that you specify is opened on a new tab when the drill-down event is triggered.
Select Logstore	The name of the Logstore to which you want to be redirected. When a drill-down event is triggered, you are redirected to the Search & Analysis page of the Logstore.
Time Range	The time range. The system queries the data that is generated within the time range. Valid values: ■ Default: queries data in the Logstore to which you are redirected based on the default time range. The default time range is 15 minutes (relative) and accurate to seconds. ■ Inherit table time: queries data in the Logstore to which you are redirected based on the time range specified for the chart when the drill-down event is triggered. ■ Relative: queries data in the Logstore to which you are redirected based on the time range that you specify. The time range is accurate to seconds. ■ Time Frame: queries data in the Logstore to which you are redirected based on the time range that you specify. The time range is accurate to minutes, hours, or days.
Inherit Filtering Conditions	If you turn on **Inherit Filtering Conditions**, the filter conditions that are added to the dashboard are synchronized to the Search & Analysis page of the Logstore to which you are redirected when the drill-down event is triggered. The filter conditions are added to the start of the query statement by using the `AND` operator.
Filter	On the **Filter** tab, you can enter a filter statement in the Filter Statement field. When the drill-down event is triggered, the filter statement is synchronized to the Search & Analysis page of the Logstore to which you are redirected. The filter statement is added to the start of the query statement by using the `AND` operator. The filter statement can contain fields that you specify in the **Optional Parameter Fields** field.

- **Open Saved Search**: configures the drill-down event to open a saved search. The following table describes the parameters that you can configure if you set the Event Action parameter to Open Saved Search.

Parameter	Description
Open in New Tab	If you turn on this switch, the saved search that you specify is opened on a new tab when the drill-down event is triggered.
Select Saved Search	The name of the saved search to which you want to be redirected. When a drill-down event is triggered, you are redirected to the page of the saved search.
Time Range	The time range for the saved search. Valid values: ■ Default: queries data by using the saved search based on the default time range. The default time range is 15 minutes (relative) and accurate to seconds. ■ Inherit table time: queries data by using the saved search based on the time range specified for the chart when the drill-down event is triggered. ■ Relative: queries data by using the saved search based on the time range that you specify. The time range is accurate to seconds. ■ Time Frame: queries data by using the saved search based on the time range that you specify. The time range is accurate to minutes, hours, or days.
Inherit Filtering Conditions	If you turn on **Inherit Filtering Conditions**, the filter conditions that are added to the dashboard are synchronized to the saved search that you want to execute when the drill-down event is triggered. The filter conditions are added to the start of the saved search by using the `AND` operator.
Inherit Variables	If you turn on **Inherit Variables** and the variable that you configure on the dashboard is the same as the variable in the saved search, the variable value on the dashboard replaces the variable in the saved search. **Note** If you want to inherit variables, you must configure a placeholder variable in the saved search.
Filter	On the **Filter** tab, you can enter a filter statement in the Filter Statement field. When the drill-down event is triggered, the filter statement is added to the start of the saved search by using the `AND` operator. The filter statement can contain fields that you specify in the **Optional Parameter Fields** field.

Parameter	Description
Variable	Log Service allows you to modify a saved search by using variables. If you configure a variable that is the same as the variable in the saved search, the variable value that you click to trigger the drill-down event replaces the variable in the saved search. You can add variables on the **Variable** tab. **Note** ■ If you want to configure a variable, you must configure a placeholder variable for the saved search to which you want to be redirected. ■ You can add up to five dynamic variables and up to five static variables. ■ Dynamic variables ■ **Variable**: the name of the variable. ■ **Variable Value Column**: the column in which the variable values are located. The values are used to dynamically replace the variable in the saved search. ■ Static variables ■ **Variable**: the name of the variable. ■ **Value**: the fixed value of the static variable. The value is used to replace the placeholder variable in the saved search.

- **Open Dashboard**: configures the drill-down event to open a dashboard. The following table describes the parameters that you can configure if you set the Event Action parameter to Open Dashboard.

Parameter	Description
Open in New Tab	If you turn on this switch, the dashboard that you specify is opened on a new tab when the drill-down event is triggered.
Select Dashboard	The name of the dashboard to which you want to be redirected. When a drill-down event is triggered, you are redirected to the page of the dashboard.
Time Range	The time range to query data for the dashboard. Valid values: ■ Default: queries data for the dashboard to which you are redirected based on the default time range. The default time range is 15 minutes (relative) and accurate to seconds. ■ Inherit table time: queries data for the dashboard to which you are redirected based on the time range specified for the chart when the drill-down event is triggered. ■ Relative: queries data for the dashboard to which you are redirected based on the time range that you specify. The time range is accurate to seconds. ■ Time Frame: queries data for the dashboard to which you are redirected based on the time range that you specify. The time range is accurate to minutes, hours, or days.
Inherit Filtering Conditions	If you turn on **Inherit Filtering Conditions**, the filter conditions that are added to the current dashboard are synchronized to the dashboard to which you are redirected when the drill-down event is triggered.

Parameter	Description
Inherit Variables	If you turn on **Inherit Variables**, the variables that you configure on the current dashboard are synchronized to the dashboard to which you are redirected.
Filter	On the **Filter** tab, you can enter a filter statement in the Filter Statement field. When the drill-down event is triggered, the filter statement is synchronized to the dashboard to which you are redirected. The filter statement can contain fields that you specify in the **Optional Parameter Fields** field.
Variable	The variables that you configure are synchronized to the dashboard to which you are redirected when the drill-down event is triggered. You can add variables on the **Variable** tab. **Note** ■ If you want to configure a variable for the chart, you must configure a placeholder variable for the chart on the dashboard to which you are redirected. ■ You can add up to five dynamic variables and up to five static variables. ■ Dynamic variables ■ **Variable**: the name of the variable. ■ **Variable Value Column**: the column in which the variable values are located. The values are dynamically synchronized to the dashboard to which you are redirected. ■ Static variables ■ **Variable**: the name of the variable. ■ **Value**: the fixed value of the static variable. The value is synchronized to the dashboard to which you are redirected.

- **Custom HTTP Link**: configures the drill-down event to open a custom HTTP URL.

  The path in the custom HTTP URL is the path of the file that you want to access. You can add an optional parameter to the path. If you click a variable value on a chart to trigger the drill-down event, the parameter is replaced by the value, and you are redirected to the custom HTTP URL.

Parameter	Description
Enter Link	The URL to which you want to be redirected.
Use System Variables	If you turn on **Use System Variables**, you can insert the following variables that are provided by Log Service to the HTTP URL: ${sls_project}, ${sls_dashboard_title}, ${sls_chart_name}, ${sls_chart_title}, ${sls_region}, ${sls_start_time}, ${sls_end_time}, ${sls_realUid}, and ${sls_aliUid}.
Transcoding	If you turn on **Transcoding**, the custom HTTP URL is encoded.
Optional Parameter Fields	If you add an optional parameter to the path, the parameter is replaced by the value that you click to trigger the drill-down event.

7. Click **Add to New Dashboard**.
8. In the dialog box that appears, specify a dashboard name and a chart name, and then click **OK**.

Example

This section provides an example on how to store NGINX access logs in a Logstore named accesslog and how to create two dashboards named RequestMethod and destination_drilldown for drill-down analysis. Before you perform drill-down analysis, add a table of request methods to the RequestMethod dashboard, and configure a drill-down event for the table to open the destination_drilldown dashboard. Then, add a line chart to the destination_drilldown dashboard. The line chart displays the trend of page views (PVs) over a specified period of time. After you configure the settings, you can click a request method on the RequestMethod dashboard. Then, you are redirected to the destination_drilldown dashboard on which can view the trend of PVs over a specified period of time.

1. Create a dsahboard named destination_drilldown.

   Before you configure a drill-down event for the table of request methods, create a dashboard to which you want to be redirected and add a line chart to the dashboard. The line chart displays the trend of PVs over a specified period of time. You need to configure the following settings. For more information, see Create a dashboard.

   - Specify a query statement.

     The query statement queries logs by request type. You can view the trend of PVs over a specified period of time.

     ```
     request_method: * | SELECT date_format(date_trunc('minute', __time__), '%H:%i:%s') AS time, COU
     NT(1) AS PV GROUP BY time ORDER BY time
     ```

   - Configure a placeholder variable.

     Specify the asterisk (*) to generate a placeholder variable and set the variable name to method.

2. Configure a drill-down event for the table of request methods and add the table to the RequestMethod dashboard.

   You need to configure the following settings. For more information, see Procedure.

   - Specify a query statement.

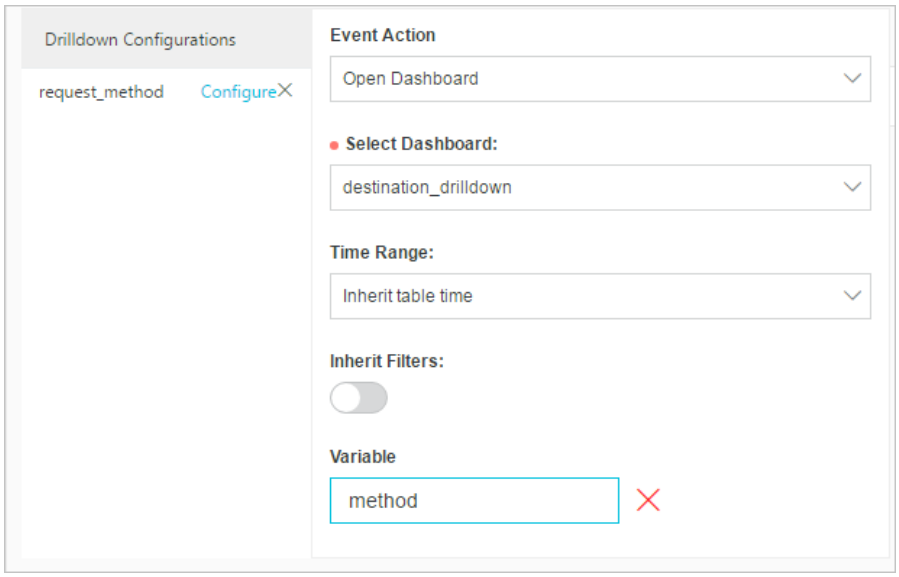
     The query statement queries the logs that are generated for each request method among the NGINX access logs.

```
*|SELECT request_method, COUNT(1) AS c GROUP BY request_method ORDER BY c DESC LIMIT 10
```

- Select a chart type.

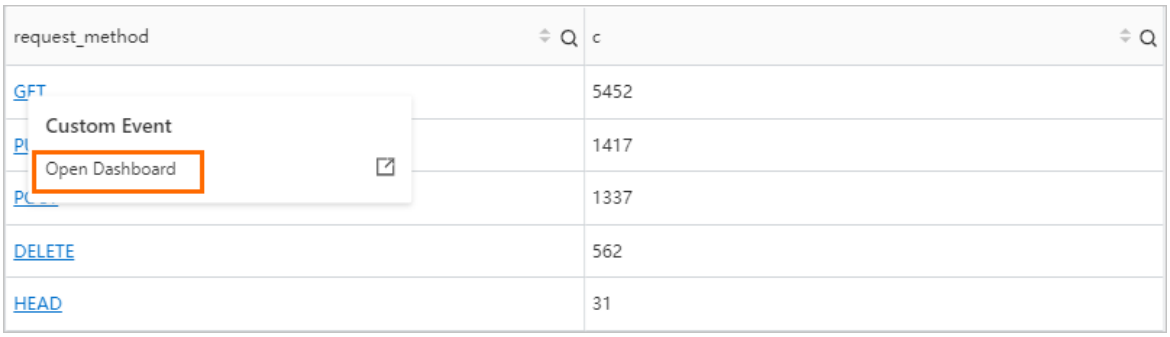
  In this example, a table is selected.

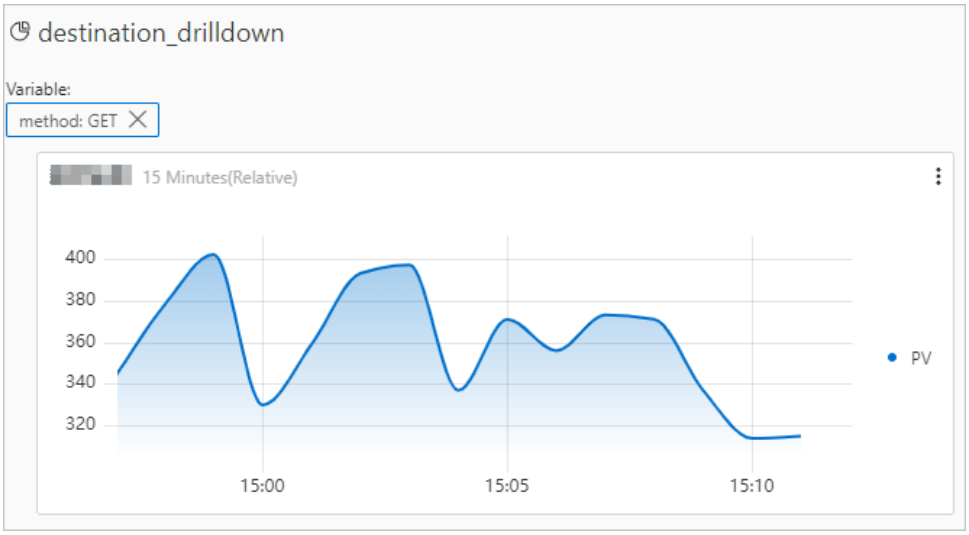
- Configure a drill-down event for the table.
  - Configure a drill-down event for the request_method column in the table.
  - Set the **Select Dashboard** parameter to **destination_drilldown**.
  - Set the **Variable** parameter to **method**.



3. View drill-down results.

   On the RequestMethod dashboard, click **GET**. You are redirected to the destination_drilldown dashboard. The asterisk (*) in the query statement is replaced by the value **GET**. The trend of PVs for GET requests over a specified period of time is displayed in a line chart.





4.12.2.6. Add a filter

You can add a filter to a dashboard. Then, you can use the filter to refine query results or replace placeholder variables with specific values. This topic describes how to add a filter to a dashboard.

Prerequisites

- Log data is collected. For more information, see Data collection overview.
- Indexes are configured. For more information, see Enable the indexing feature and configure indexes for a Logstore.
- Charts are added to a dashboard. For more information, see Add a chart to a dashboard.

> **Notice** If you set the filter type to Replace Variable, you must configure placeholder variables for the charts on the dashboard.

Context

A filter is used to modify query statements or replace placeholder variables for all charts on a dashboard. Each chart displays the query and analysis result of a query statement, which is in the [search query]|[sql query] format. After you add a filter to a dashboard, the filter condition or variables that you specify for the filter apply to the query statement that corresponds to each chart on the dashboard. The following types of filters are supported:

- Filter: uses key-value pairs as a filter condition.

  The filter condition is added to the start of a query statement by using the AND or NOT operator. For example, the **Key: Value AND [search query]|[sql query]** statement queries logs that contain **Key:Value** in the query result of the [search query]|[sql query] statement. For the Filter type, you can select or enter multiple key-value pairs. If you specify multiple key-value pairs, the logical OR operator is used between the pairs.

- Replace Variable: uses a variable and the value of the variable.

  If the variable that you specify for the filter is configured for existing charts on the dashboard, the variable in the query statement of each chart is automatically replaced by the variable value that you specify for the filter. This applies to all charts for which the same variable is configured.

Procedure

1. Log on to the Log Service console.
2. Click the name of the project that you want to manage.
3. In the left-side navigation pane, click the Dashboard icon.

4. In the Dashboard list, click the dashboard that you want to manage.
5. In the upper-right corner of the dashboard page, click **Edit**.
6. Click the  icon and configure the parameters. Then, click **OK**. The following table describes the parameters.

Parameter	Description
Filter Name	The name of the filter.
Display Settings	Valid values: ○ **Title**: specifies whether to add a title for the filter. You can turn on Title to add a title for the filter. ○ **Border**: specifies whether to add borders to the filter. You can turn on Border to add borders to the filter. ○ **Background**: specifies whether to add a white background to the filter. You can turn on Background to add a white background to the filter.
Type	The type of the filter. ○ **Filter**: uses key-value pairs to filter data. The key-value pairs are used as a filter condition and are added to the start of a query statement by using the AND or NOT operator. By default, the AND operator is used. ■ AND: **Key: Value AND [search query] | [sql query]** ■ NOT: **Key: Value NOT [search query] | [sql query]** You can specify multiple values for the key-value pairs in the **Static List Items** field. ○ **Replace Variable**: specifies a variable and the value of the variable. If the variable that you specify for the filter is configured for existing charts on the dashboard, the variable in the query statement of each chart is automatically replaced by the variable value that you specify for the filter. You can specify multiple values for the variables in the **Static List Items** field.
Key	○ If you select **Filter**, enter the key that you want to use to filter data in the **Key** field. ○ If you select **Replace Variable**, enter the variable that you want to use to filter data in the **Key** field. **Note** If you select **Replace Variable**, you must specify a placeholder variable when you add a chart to the dashboard. The placeholder variable must be the same as the variable that you specify in the Key field.
Alias	The alias of the key. This parameter is available only when you select **Filter**.
Global filter	The parameter is available only when you select **Filter**. ○ If you want to filter specific values in all fields, turn on **Global filter**. ○ If you want to filter specific values in specified keys, turn off **Global filter**.
Static List Items	The value of the **Key** field that is used to filter data. You can click the plus sign (**+**) to add more values for the specified key. If you turn on **Select by Default** for a value, the value is used to filter data each time you open a dashboard.

Parameter	Description
Add Dynamic List Item	If you turn on **Add Dynamic List Item**, dynamic values can be retrieved for **Key**. Dynamic list items are dynamic values that are retrieved by executing the specified query statement. The values vary based on the time ranges during which the query statement is executed. If you turn on **Add Dynamic List Item**, you must configure the following parameters: ○ **Select Logstore**: Select a Logstore from which data is queried. ○ **Inherit Filtering Conditions**: If you turn on **Inherit Filtering Conditions**, the filter condition on the dashboard is added before the query statement. ○ Query statement: Enter a query statement and specify a time range. ○ **Dynamic List Item Preview**: Preview query results.

4.12.2.7. Manage a Markdown chart

Log Service allows you to add a Markdown chart to a dashboard. In the Markdown chart, you can insert images, links, videos, and other elements to make your dashboard page more intuitive to use.

Context

You can add multiple analysis charts to a dashboard. This allows you to view multiple analysis results and monitor the status of multiple applications on a single dashboard. You can also add Markdown charts to a dashboard. A Markdown chart is edited by using the Markdown syntax.

You can create different Markdown charts based on your business requirements. Markdown charts can make a dashboard more intuitive to use. You can insert text such as background information, chart description, notes, and extension information into a Markdown chart. You can insert custom images and videos into a Markdown chart. You can also insert saved searches or the dashboard links of other projects to redirect to other query pages.

You can insert links into a Markdown chart to redirect to the other dashboard pages of the current project. You can also insert an image that corresponds to each link. In addition, you can use a Markdown chart to describe the parameters of analysis charts.

Add a Markdown chart

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project in which you want to manage a dashboard.
3. In the left-side navigation pane, click the icon.

4. In the Dashboard list, click the dashboard that you want to manage.
5. In the upper-right corner of the dashboard page, click **Edit**.
6. In edit mode, drag the  icon from the menu bar and drop the icon on a specified position to create a Markdown chart.
7. Double-click the Markdown chart.
8. In the **Markdown Edit** dialog box, set the parameters, and then click **OK**.

Parameter	Description
Chart Name	The name of the Markdown chart.
Show Border	Specifies whether to show the borders of the Markdown chart. You can turn on **Show Border** to show the borders of the Markdown chart.
Show Title	Specifies whether to show the title of the Markdown chart. You can turn on **Show Title** to show the title of the Markdown chart.
Show Background	Specifies whether to show the background of the Markdown chart. You can turn on **Show Background** to show the background of the Markdown chart.
Query Binding	Specifies whether to associate a query statement with a Markdown chart. You can turn on **Query Binding** and associate a query statement with a Markdown chart. Then, dynamic query results are displayed in the Markdown chart. i. Select a Logstore whose data you want to query. ii. Enter a query statement in the search box, specify a time range, and then click **Search**. For more information, see Overview. **Note** The query results may contain logs that are generated 1 minute earlier or later than the specified time range. The first returned log is displayed. iii. Click the plus sign next to a field to insert the corresponding query result into the **Markdown Content** column.
Markdown Content	Enter Markdown content in the **Markdown content** column on the left. The data preview is displayed in real time in the **Show Chart** column on the right. You can modify the Markdown content based on the data preview. For more information, see Common Markdown syntax.

9. Click **Save**.

Modify a Markdown chart

1. In the upper-right corner of the dashboard page, click **Edit**.
2. Modify the position and size of a Markdown chart

   Drag the Markdown icon to a position on the dashboard and drag the lower-right corner of the chart to adjust the size of the chart.
3. Modify the properties of a Markdown chart
   i. Double-click the Markdown chart that you want to modify.

ii. In the **Markdown Edit** dialog box, modify the parameters, and then click **OK**.

You can modify the chart name, display settings, query settings, and Markdown content. For more information, see Add a Markdown chart.

Delete a Markdown chart

1. In the upper-right corner of the dashboard page, click **Edit**.
2. Find the Markdown chart that you want to delete and choose > **Delete**.
3. In the upper-right corner of the dashboard page, click **Save**.

Common Markdown syntax

- Heading
  - Markdown syntax

```
# Level 1 heading
## Level 2 heading
### Level 3 heading
```

  - Result

- Link

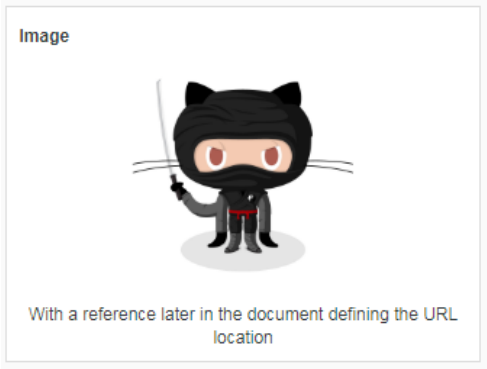
  Markdown syntax

```
### Contents
[Test](https://www.alibabacloud.com/)
```

- Image
  - Markdown syntax

```
<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"
```

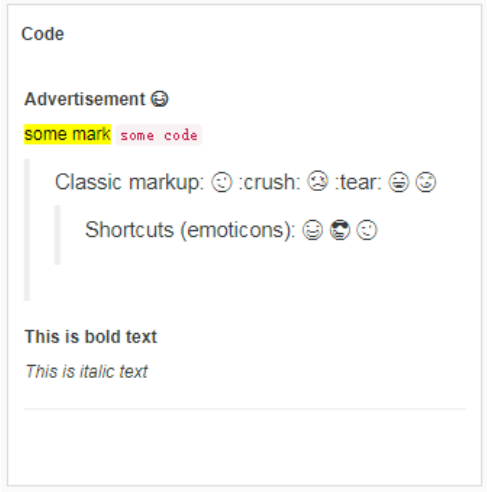
- Preview



- Special tag
  - Markdown syntax

```
---
__Advertisement :)__
==some mark==    `some code`
> Classic markup: :wink: :crush: :cry: :tear: :laughing: :yum:
>> Shortcuts (emoticons): :-)  8-) ;)
__This is bold text__
*This is italic text*
---
```

  - Result



5.Alerts

5.1. Overview

Log Service provides the alerting feature. You can configure alert rules to trigger alerts based on query and analysis results. After you create an alert rule, Log Service checks related query and analysis results on a regular basis. If a query and analysis result meets the trigger condition that you specify in the alert rule, Log Service sends an alert notification. This way, you can monitor the service status in real time.

Limits

The following table describes the limits of the alerting feature in Log Service.

Item	Description
Associated query statements	You can associate an alert rule with a maximum of three query statements.
Field value size	If the number of characters that are included in a field exceeds 1,024, Log Service extracts only the first 1,024 characters for data processing.
Trigger condition	Trigger conditions have the following limits: • Each trigger condition must be 1 to 128 characters in length. • If a query result includes more than 100 rows, Log Service only checks whether the first 100 rows meet the specified trigger condition. • Log Service checks whether a trigger condition is met for a maximum of 1,000 times for the specified query statements.
Query time range	The maximum time range that you can specify for each query is 24 hours.
Voice calls	If a voice call is not answered, Log Service sends an SMS notification. You are charged only once for the voice call regardless of whether the call is answered. You are not charged for SMS notifications.

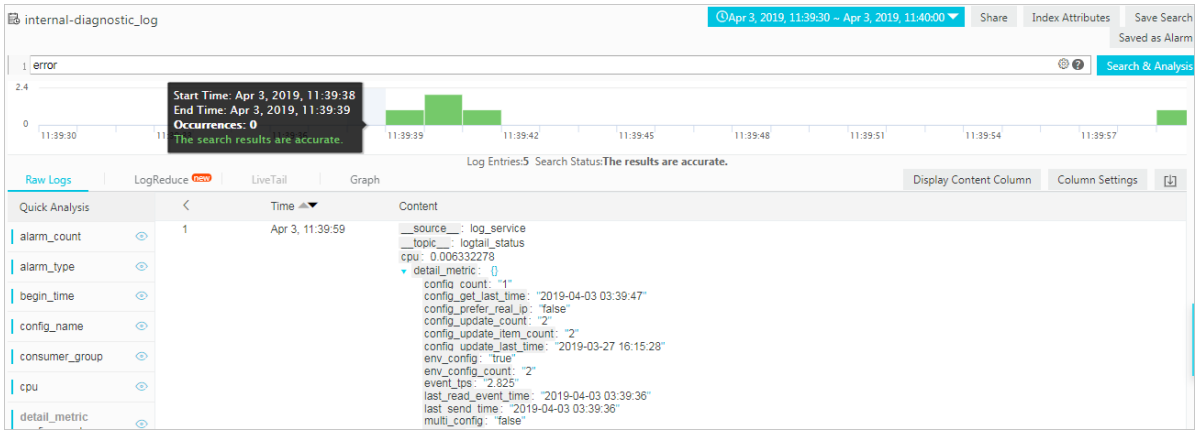
Query statements in alert rules

An alert rule is associated with one or more charts in a dashboard. Each chart displays the result of a query statement. You can associate an alert rule with one or more search statements or query statements.

- A search statement returns the log entries that meet the specified search condition.

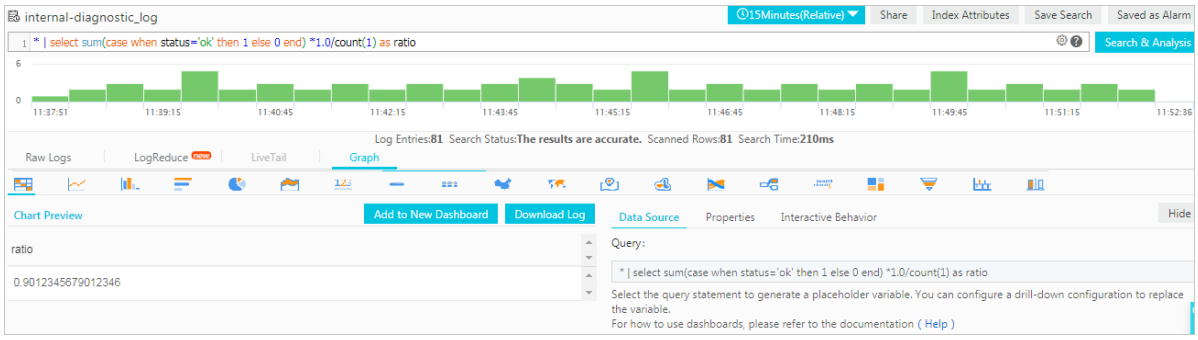
  For example, you can execute the error statement to search for the log entries that are generated in the previous 15 minutes and contain error. A total of 154 log entries are returned. Each log entry consists of key-value pairs. You can specify a trigger condition based on the value of a key.

  > **Note** If the number of returned log entries exceeds 100, Log Service checks only the first 100 log entries. If one of the log entries meets the specified condition, an alert is triggered.



- A query statement consists of a search statement and an analytic statement. The analytic statement analyzes the log entries that meet the search condition and returns a result.

  For example, the * | select sum(case when status='ok' then 1 else 0 end) *1.0/count(1) as ratio statement returns the percentage of the log entries in which the value of the status field is ok. If you set the trigger condition of an alert rule to ratio < 0.9, an alert is triggered if the percentage of the log entries whose status code is ok is less than 90%.



5.2. Configure an alarm

5.2.1. Configure an alert rule

You can create an alert rule on the Search & Analysis page of a Logstore or on a dashboard page in the Log Service console. After you create an alert rule, Log Service sends alert notifications when the trigger condition in the alert rule is met. This topic describes how to create an alert rule in the Log Service console.

Prerequisites

- Logs are collected and stored in a Logstore.
- The indexing feature is enabled and indexes are configured. For more information, see Enable the index feature and configure indexes for a Logstore.

Context

Log Service allows you to configure alert rules based on charts. You can create an alert rule for a query statement on the Search & Analysis page. After you create the alert rule, a chart that shows the query result of the query statement is automatically created on the specified dashboard. You can also create an alert rule for one or more existing charts on a dashboard.

- Create a chart and configure an alert rule for the chart

  After you create an alert rule for a query statement, a chart that shows the query result of the query statement is automatically created on the specified dashboard. When you create an alert rule for a query statement on the Search & Analysis page, you must specify a dashboard and a name for the chart.

- Create an alert rule for existing charts on a dashboard

  If you create an alert rule for multiple existing charts, you can specify one or more charts with which you want to associate the alert rule. You can specify a conditional expression for each chart, and combine the conditional expressions into a trigger condition of the alert rule.

The following section describes how to configure an alert rule for multiple existing charts on a dashboard.

> **Note** If you modify the query statement of a chart with which an alert rule is associated, you must update the query statement in the alert rule. For more information, see Modify an alert.

For information about common configurations for alert rules, see FAQ about alerts.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. In the left-side navigation pane, click the icon.
4. In the Dashboard list, click the dashboard that you want to manage.
5. In the upper-right corner of the page that appears, choose **Alerts > Create**.
6. In the **Alert Configuration** step, configure the parameters and click **Next**.

   The following table describes the parameters.

Alert Name	Description
Alert Name	The name of the alert rule. The name must be 1 to 64 characters in length.
Associated Chart	The chart with which you want to associate the alert rule. You can add up to three charts. You can configure an alert rule for up to three query statements at the same time. The number before the chart name is the serial number of the chart. The serial number of the chart is valid in the alert rule. You can use the serial number to specify a chart in the **Trigger Condition** parameter. You can click the icon next to the Query field to modify the query statement. The **Search Period** parameter specifies the time range of each query. You can select a relative time or a time frame. For example, the current time is 14:30:06. ◦ If you set the **Search Period** parameter to 15 Minutes(Relative), the time range of the query is 14:15:06-14:30:06. ◦ If you set the **Search Period** parameter to 15 Minutes(Time Frame), the time range of the query is 14:15:00-14:30:00.
Frequency	The frequency at which query results are checked.

Alert Name	Description
Trigger Condition	The trigger condition of an alert. If the specified trigger condition is met, an alert is triggered and alert notifications are sent based on the values of the **Frequency** and **Notification Interval** parameters. For example, you can set the trigger condition to `pv%100 > 0 && uv > 0`. **Note** You can use `$serial number` to specify a chart in a trigger condition. For example, `$0` specifies chart 0.
Advanced	
Notification Trigger Threshold	An alert is triggered only if the specified trigger condition is met during continuous check periods. If the number of continuous triggers reaches the specified threshold, alert notifications are sent at the specified notification interval. If the trigger condition is not met, no alert is triggered. Default value: 1. This value specifies that alert notifications are sent if the trigger condition is met. You can set the Notification Trigger Threshold parameter to an integer that is greater than 1. In this case, alert notifications are sent only if the number of continuous triggers reaches the threshold. For example, you set the **Notification Trigger Threshold** parameter to 100. In this case, if the trigger condition is met for 100 times during continuous check periods, the value of **Notification Trigger Threshold** is reached. If the interval between the current time and the last time when alert notifications are sent exceeds the specified value of the **Notification Interval** parameter, an alert notification is sent. After an alert notification is sent, Log Service resets the number of continuous triggers to zero. If a check fails due to network exceptions, the check is not counted.
Notification Interval	The interval at which Log Service sends alert notifications. If the trigger condition is met in a check, Log Service checks whether the number of continuous triggers reaches the specified value of the **Notification Trigger Threshold** parameter. Log Service also checks whether the interval between the current time and the last time when alert notifications are sent exceeds the specified value of the **Notification Interval** parameter. If you set the Notification Interval parameter to 5 minutes, only one alert notification is received once every 5 minutes. **Note** You can use the Notification Trigger Threshold and Notification Interval parameters to control the number of alert notifications that you receive.

7. In the **Notifications** step, configure alert notification methods and click **Submit**.

Log Service supports the following alert notification methods: WebHook-Custom and WebHook-DingTalk Bot. For more information, see Notification methods.

5.2.2. Authorize a RAM user to manage alert rules

You can use your Apsara Stack tenant account to authorize a RAM user to manage alert rules. This topic describes how to create a RAM user and authorize the RAM user to manage alert rules.

Procedure

1. Log on to the ASCM console ASCM console as an administrator.
2. Create a RAM role.
3. Create a permission policy.

   Replace the content in the Policy Document field with the following script. Replace *Project name* in the script with the name of your Log Service project.

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "log:CreateLogStore",
        "log:CreateIndex",
        "log:UpdateIndex"
      ],
      "Resource": "acs:log:*:*:project/Project name/logstore/internal-alert-history"
    },
    {
      "Effect": "Allow",
      "Action": [
        "log:CreateDashboard",
        "log:CreateChart",
        "log:UpdateDashboard"
      ],
      "Resource": "acs:log:*:*:project/Project name/dashboard/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "log:*"
      ],
      "Resource": "acs:log:*:*:project/Project name/job/*"
    }
  ]
}
```

4. Create a user.
5. Create a RAM user group.
6. Add a RAM user to a RAM user group
7. Grant permissions to a RAM role.

5.2.3. Configure alert notification methods

Log Service supports the following alert notification methods: Webhook-Custom and WebHook-DingTalk Bot. This topic describes how to configure alert notification methods.

Webhook-Custom

If you set the notification method to WebHook-Custom, Log Service sends alert notifications to a custom webhook URL.

> **Note** If Log Service does not receive a response within 5 seconds after a notification is sent, the request times out.

1. When you configure an alert rule, select **WebHook-Custom** from the **Notifications** drop-down list. For more information, see Configure an alert rule.
2. Configure the parameters. The following table describes the parameters.

Parameter	Description
Request URL	The custom webhook URL.
Request Method	The method that is used to send the notification. The following request methods are supported: GET, POST, DELETE, PUT, and OPTIONS. The default request header is Content-Type: application/json;charset=utf-8. To add request headers, click **Add Request Headers**.
Request Content	The content of the alert notification. Log Service provides default content. The content must be 1 to 500 characters in length. You can specify custom content. You can also use template variables in the content. For more information, see Template variables.

3. Click **Submit**.

WebHook-DingTalk Bot

If you set the notification method to Webhook-DingTalk Bot, Log Service sends alert notifications to the DingTalk group to which a specified webhook URL points by using a DingTalk chatbot. The chatbot can also remind the specified contacts of the alert notifications.

> **Note** Each DingTalk chatbot can send up to 20 alert notifications per minute.

1. Create a DingTalk chatbot.
   i. Open DingTalk and go to a DingTalk group.
   ii. In the upper-right corner of the chat window, click the **Group Settings** icon and choose **Group Assistant > Add Robot**.
   iii. In the **ChatBot** dialog box, click the **+** icon in the **Add Robot** section.
   iv. In the Robot details dialog box, select **Custom (Custom message services via Webhook)** and click **Add**.
   v. In the **Add Robot** dialog box, enter a chatbot name in the **Chatbot name** field and select security options in the **Security Settings** section based on your business requirements. Then, select **I have read and accepted DingTalk Custom Robot Service Terms of Service** and click **Finished**.

      > **Note** We recommend that you set the **Security Settings** parameter to **Custom Keywords**. You can specify up to 10 keywords. The chatbot sends only messages that contain at least one of the specified keywords. We recommend that you specify **Alert** as a keyword. 关于安全设置，更多信息，请参见钉钉开放平台。

   vi. Click **Copy** to copy the webhook URL.
2. Configure a notification method in the Log Service console.

i. When you configure an alert rule, select **WebHook-DingTalk Bot** from the **Notifications** drop-down list. For more information, see Configure an alert rule.

ii. Configure the parameters. The following table describes the parameters.

Parameter	Description
Request URL	The webhook URL of the DingTalk chatbot. Paste the webhook URL that you copied in Step 1.
Title	The alert topic. The title must be 1 to 100 characters in length. You can specify a custom title. You can also use template variables in the title. For more information, see Template variables.
Recipients	The group members whom you want to remind of the alert notification. Valid values: None, All, and Specified Members. If you select **Specified Members**, enter the mobile phone numbers of the group members in the **Tagged List** field. Separate multiple mobile phone numbers with commas (,).
Content	The content of the alert notification. Log Service provides default content. You can modify the content based on your business scenario. The content must be 1 to 500 characters in length. You can specify custom content. You can also use template variables in the content. For more information, see Template variables. **Note** If you want to remind a group member of the alert notification, use the `@<Mobile phone number of the group member>` syntax in the **Content** field.

iii. Click **Submit**.

Template variables

You can use template variables when you configure a notification method for an alert rule. When you configure the **Content** and **Subject** parameters, you can use the *${fieldName}* syntax to reference a template variable. When Log Service sends an alert notification, Log Service replaces the template variables that are referenced in the **Content** and **Subject** parameters with actual values. For example, Log Service replaces *${Project}* with the name of the project to which the alert rule belongs.

Notice You must reference valid variables. If a referenced variable does not exist or is invalid, Log Service processes the variable as an empty string. If the value of a referenced variable is of the object type, the value is converted and displayed as a JSON string.

The following table describes the supported template variables and the methods that are used to reference the variables.

Variable	Description	Example	Reference example
Aliuid	The ID of the Apsara Stack tenant account to which the project belongs.	1234567890	An alert is triggered for the Apsara Stack tenant account ${Aliuid}.
Project	The project to which an alert rule belongs.	my-project	An alert is triggered in the ${Project} project.
AlertID	The ID of an alert.	0fdd88063a611aa114938f9371daeeb6-1671a52eb23	The ID of the alert is ${AlertID}.

Variable	Description	Example	Reference example
AlertName	The ID of an alert rule. The ID is unique in a project.	alert-1542111415-153472	An alert is triggered based on the ${AlertName} alert rule.
AlertDisplayName	The display name of an alert rule.	My alert	An alert is triggered based on the ${AlertDisplayName} alert rule.
Condition	The conditional expression that triggers an alert. In an alert notification, a variable is replaced by an actual value that is enclosed in brackets [].	[5] > 1	The conditional expression that triggers an alert is ${Condition}.
RawCondition	The original conditional expression that triggers an alert.	count > 1	The original conditional expression that triggers an alert is ${RawCondition}.
Dashboard	The name of the dashboard that is associated with an alert rule.	mydashboard	The alert rule is associated with the ${Dashboard} dashboard.
DashboardUrl	The URL of the dashboard that is associated with an alert rule.	https://sls.console.aliyun.com/next/project/myproject/dashboard/mydashboard	The URL of the dashboard that is associated with the alert rule is ${DashboardUrl}.
FireTime	The time when an alert is triggered.	2018-01-02 15:04:05	The alert is triggered at ${FireTime}.
FullResultUrl	The URL that is used to query the details of an alert.	https://sls.console.aliyun.com/next/project/my-project/logsearch/internal-alert-history?endTime=1544083998&queryString=AlertID%3A9155ea1ec10167985519fccede4d5fc7-1678293caad&queryTimeType=99&startTime=1544083968	Click ${FullResultUrl} to view the alert details.

Variable	Description	Example	Reference example

Results	The parameters and results of a query. The value is of the array type. For more information, see Fields in alert log entries.
	Note The Results variable can contain the information of up to 100 alerts.
		<pre>[
		  {
		    "EndTime": 1542507580,
		    "FireResult": {
		      "__time__": "1542453580",
		      "count": "0"
		    },
		    "LogStore": "test-logstore",
		    "Query": "* | SELECT COUNT(*) as count",
		    "RawResultCount": 1,
		    "RawResults": [
		      {
		        "__time__": "1542453580",
		        "count": "0"
		      }
		    ],
		    "StartTime": 1542453580
		  }
		]</pre>
			The first query starts at ${Results[0].StartTime} and ends at ${Results[0].EndTime}. The alert is triggered ${Results[0].FireResult.count} times.
			Note In this example, 0 is the serial number of a chart. For more information, see How can I view the serial number of a chart?

5.3. Modify and view an alarm

5.3.1. Modify an alert rule

This topic describes how to modify an alert rule. After you create an alert rule, you can modify the chart that is associated with the alert rule, and then update the alert rule. If you associate a query statement with an alert rule, you can modify the query statement to modify the alert rule.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. In the left-side navigation pane, click the icon.
4. In the Alerts list, click the alert rule that you want to modify.
5. On the **Alert Overview** page, click **Modify Settings**.
6. In the **Modify Alert** panel, modify the chart that is associated with the alert rule.

In the **Associated Chart** section, select the chart that you want to associate with the alert rule.

7. Modify the query statement that is associated with the alert rule.
   i. In the **Associated Chart** section, find the query statement that you want to modify. Then, click the icon next to the Query field.

   > **Note** If the original statement consists of only a search statement, the new statement must also consist of only a search statement. If the original statement consists of a search statement and an analytic statement, the new statement must also consist of a search statement and an analytic statement. For example, after you associate the search statement **request_method: GET** with an alert rule, you can change the search statement to **error**. You cannot change the search statement to **error| select count(1) as c**.

   ii. In the Edit dialog box, enter a new query statement and click **Preview**.
   iii. If the expected query result is returned, click **OK**.
8. Modify the evaluation frequency, trigger condition, notification threshold, and notification interval. Then, click **Next**.

   For more information, see Configure alerts.
9. Modify the alert notification method and click **Submit**.

   For more information, see Configure alert notification methods.

5.3.2. View alert statistics

Log Service records historical alert statistics in logs and automatically creates a dashboard to display the execution results of all alert monitoring rules and the alert notifications that are sent.

Context

- View alert logs in a Logstore

  After you create an alert monitoring rule, Log Service automatically creates a Logstore named **internal-alert-history** in the project to which the alert monitoring rule belongs. A log is generated and written to the Logstore each time an alert monitoring rule is executed in the project, regardless of whether an alert is triggered. For more information about log fields, see Fields in alert log entries.

  > **Note** You are not charged for the internal-alert-history Logstore. The Logstore cannot be deleted.

- View alert statistics in a dashboard

  After you create an alert monitoring rule, Log Service automatically creates a dashboard named **Alert History Statistics** in the project to which the alert monitoring rule belongs. The dashboard displays information about all alerts that are triggered in the current project. The information includes Alerts, Execution Success Rate, Notification Rate upon Successful Execution, and Top 10 Alert Rule Executions.

  > **Note** The Alert History Statistics dashboard cannot be deleted or modified.

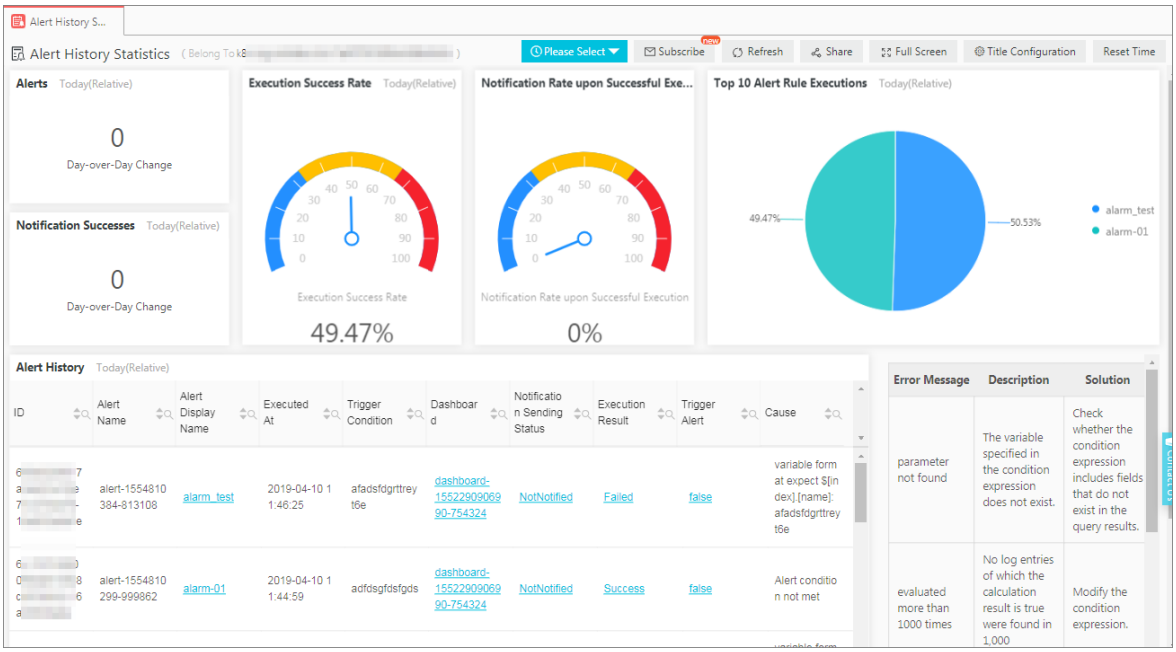
View alert logs in a Logstore

1. Log on to the Log Service console.
2. In the Projects section, click the required project.
3. On the **Log Storage > Logstores** tab, find the **internal-alert-history** Logstore and choose > **Search and Analysis**.
4. On the page that appears, query alert logs based on your business requirements.

View the Alert History Statistics dashboard

1. Log on to the Log Service console.
2. In the Projects section, click the required project.
3. In the left-side navigation pane, click the icon.
4. In the Dashboard list, click **Alert History Statistics**.

   The **Alert History Statistics** dashboard displays information about alerts, such as whether an alert is triggered, the reason why the alert is triggered, and the error information and description of the alert.



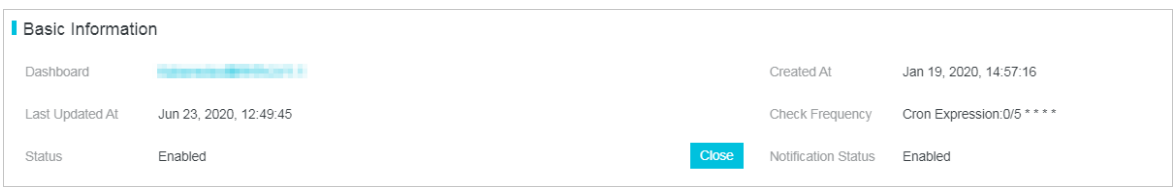
5.3.3. Manage alerts

After you configure an alert, you can manage the alert. For example, you can view information about the alert, modify the alert, and delete the alert.

View information about an alert

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. In the left-side navigation pane, click the **Alerts** icon.
4. In the list of alerts, click the alert whose information you want to view to go to the Alert Overview page.
5. On the Alert Overview page, view the information about the alert.

   On the **Alert Overview** page, you can view the dashboard to which the alert belongs, the point in time when the alert is configured, the most recent point in time when the alert is updated, the check frequency of the alert, the status of the alert, and the status of the alert notification feature for the alert.

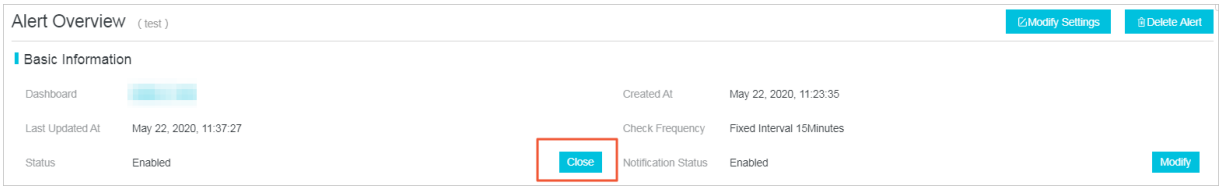


Disable or enable an alert

After you configure an alert, you can disable or enable the alert at any time.

> **Note** After an alert is disabled, Log Service no longer checks data for the alert and no longer sends alert notifications for the alert.

On the **Alert Overview** page, click **Disabled** or **Enabled** on the right side of **Status**.



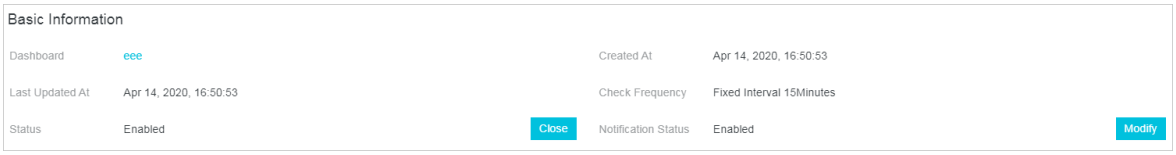
Suspend or resume the alert notification feature for an alert

If the monitoring status of an alert is **Enabled**, you can suspend the alert notification feature for the alert.

> **Note** If the alert notification feature is suspended for an alert, Log Service regularly checks data for the alert. However, Log Service does not send alert notifications even if the alert trigger condition is met.

1. On the **Alert Overview** page of an alert, click **Modify** on the right side of **Monitoring Status**.
2. Configure Disabled Duration and click **OK**.

   After the alert notification feature is suspended, you can view the value of **Monitoring Status** to determine the point in time when the alert notification feature is scheduled to resume. If you want to resume the alert notification feature before the scheduled point in time, you can click **Modify** on the right side of **Monitoring Status**. In the message that appears, click OK.



Delete an alert

> **Warning** After you delete an alert, the alert cannot be restored. Proceed with caution.

In the upper-right corner of the **Alert Overview** page, click **Delete Alert**.

5.4. Relevant syntax and fields for reference

5.4.1. Syntax of conditional expressions in alert rules

Log Service checks whether the specified trigger conditions are met based on the execution results of conditional expressions specified in alert rules. The result of a specified query statement is used as input, and the fields in the result of set operations are used as variables. If the condition that is specified in a conditional expression is met, an alert is triggered.

Limits

The conditional expressions that you can specify in an alert rule have the following limits:

- Negative numbers must be enclosed in parentheses (), for example, x+(-100)<100.
- Numeric values are converted to 64-bit floating-point numbers. If a comparison operator such as equal-to (==) is used, errors may occur.

- Variable names can contain only letters and digits, and must start with a letter.
- A conditional expression must be 1 to 128 characters in length.
- You can specify up to 1,000 conditions in a conditional expression. If the evaluation result of each condition in a conditional expression is false, the evaluation result of the conditional expression is false.
- An alert rule can be associated with a maximum of three charts or query statements.
- An alert is triggered only if the result of the specified conditional expression is true. For example, if a conditional expression is 100+100, the result is 200 and is not true, and no alert is triggered.
- Log Service reserves the words true and false. Log Service also reserves the special characters dollar sign ($) and period (.). You cannot use the reserved words and special characters as variables.

Syntax

The following table describes the types of syntax that is supported for the conditional expressions of an alert rule.

Syntax type	Description	Example
Arithmetic operators	The addition (+), subtraction (-), multiplication (*), division (/), and modulus (%) operators are supported. +-*/%	• `x * 100 + y > 200` • `x % 10 > 5`
Comparison operators	The following eight comparison operators are supported: greater-than (>), greater-than-or-equal-to (>=), less-than (<), less-than-or-equal-to (<=), equal-to (==), not-equal-to (!=), regex match (=~), and regex not match (!~). **Note** • Backslashes (\) in regular expressions must be escaped. • The Regular Expression 2 (RE2) syntax is used.	• `x >= 0` • `x < 100` • `x <= 100` • `x == 100` • `x == "foo"` • Regex match: `x =~ "\\w +"`
Logical operators	The AND (&&) and OR (||) operators are supported.	• `x >= 0 && y <= 100` • `x > 0 || y >0`
Not operator	The not operator (!) is supported.	`!(a < 1 && a > 100)`
Numeric constants	Numeric constants are supported. Log Service converts numeric constants to 64-bit floating-point numbers.	`x > 100`
String constants	String constants are supported. The string constants are in the format of 'String', for example, 'string'.	`foo == 'string'`
Boolean constants	Boolean constants are supported. Valid values: true and false.	`(x > 100) == true`
Parentheses	Parentheses () can be used to override the standard precedence order and force Log Service to evaluate the enclosed part of a trigger condition before an unenclosed part.	`x * (y + 100) > 100`
contains function	The contains function can be used to check whether a string contains a substring. For example, if you invoke contains(foo,'hello') and true is returned, this indicates that the foo string contains the hello substring.	`contains(foo, 'hello')`

Evaluate the results of multiple query statements

- Syntax

  An alert rule can be associated with multiple query statements. If you want to use a variable in the trigger condition to reference a field from the result of a query statement, you must prefix the variable with the serial number of the query statement in the $N.fieldname format. The serial number of a query statement is the same as the serial number of the chart that shows the result of the query statement. Each alert rule can be associated with up to three query statements. Therefore, the value range of N is 0 to 2. For example, $0.foo references the value of the foo field from the result of the first query statement. If an alert rule is associated with only one query statement, you do not need to specify the prefix in the trigger condition.

- Evaluate a conditional expression

  If multiple query results are returned, the variables specified in the conditional expression specify the results that need to be used to evaluate the conditional expression. For example, three query statements are specified and three sets of query results are returned. The number of log entries in the first set is x, the number of log entries in the second set is y, and the number of log entries in the third set is z. If the conditional expression that you specify is $0.foo > 100 && $1.bar < 100, only the first two sets are used to evaluate the conditional expression. Up to x × y times of evaluation ,or 1,000 if x × y is greater than 1,000, is performed. If the conditional expression is met within the maximum number of times of evaluation, true is returned. Otherwise, false is returned.

Operation methods

Note

- Log Service converts all numeric values to 64-bit floating-point numbers.
- A string constant must be enclosed in single quotation marks ('') or double quotation marks (""), for example, 'String' or "String".
- Boolean values include true and false.

Operator	Operation method		
	Operation between variables	Operation between a non-string constant and a variable	Operation between a string constant and a variable
Arithmetic operators: addition (+), subtraction (-), multiplication (*), division (/), and modulus (%)	Before an arithmetic operator is applied, the left and right operands are converted to 64-bit floating-point numbers.	Before an arithmetic operator is applied, the left and right operands are converted to 64-bit floating-point numbers.	Not supported.
Comparison operators: greater-than (>), greater-than-or-equal-to (>=), less-than (<), less-than-or-equal-to (<=), equal-to (==), and not-equal-to (!=)	Log Service uses the following comparison rules that are sorted in the precedence order: 1. The left and right operands are converted to 64-bit floating-point numbers, and then compared based on the numerical order. If the conversion fails, the operation of the next priority is performed. 2. The left and right operands are converted to strings, and then compared based on the lexicographic order.	The left and right operands are converted to 64-bit floating-point numbers, and then compared based on the numerical order.	The left and right operands are converted to strings, and then compared based on the lexicographic order.

Operator	Operation method		
	Operation between variables	Operation between a non-string constant and a variable	Operation between a string constant and a variable
Matching operators: regex match (=~) and regex not match (!~)	Before a matching operator is applied, the left and right operands are converted to strings.	Not supported.	Before a matching operator is applied, the left and right operands are converted to strings.
Logical operators: AND (&&) and OR (||)	A logical operator cannot be applied to log fields. The left and right operands must be sub-expressions and the result of the operation must be a Boolean value.	A logical operator cannot be applied to log fields. The left and right operands must be sub-expressions and the result of the operation must be a Boolean value.	A logical operator cannot be applied to log fields. The left and right operands must be sub-expressions and the result of the operation must be a Boolean value.
Not operator (!)	The not operator cannot be applied to log fields. The specified operand must be a sub-expression and the result of the operation must be a Boolean value.	The not operator cannot be applied to log fields. The specified operand must be a sub-expression and the result of the operation must be a Boolean value.	The not operator cannot be applied to log fields. The specified operand must be a sub-expression and the result of the operation must be a Boolean value.
contains function	Before the contains function is run, the left and right operands are converted to strings.	Not supported.	Before the contains function is run, the left and right operands are converted to strings.

Operator	Operation method		
	Operation between variables	Operation between a non-string constant and a variable	Operation between a string constant and a variable
Parentheses ()	Parentheses () are used to override the standard precedence order and force Log Service to evaluate the enclosed part of a trigger condition before an unenclosed part.	Parentheses () are used to override the standard precedence order and force Log Service to evaluate the enclosed part of a trigger condition before an unenclosed part.	Parentheses () are used to override the standard precedence order and force Log Service to evaluate the enclosed part of a trigger condition before an unenclosed part.

5.4.2. Fields in alert logs

After you configure an alert rule, Log Service automatically creates a Logstore to store log entries that are generated when the alert rule is executed and alert notifications are sent. This topic describes the fields in alert logs.

Fields in the logs that are generated when an alert rule is executed

Log field	Description	Example
AlertDisplayName	The display name of an alert rule.	Test alert rule
AlertID	The ID of an alert. The ID is unique.	0fdd88063a611aa114938f9371daeeb6-1671a52eb23
AlertName	The name of the alert rule. The name is unique within a project.	alert-1542111415-153472
Condition	The conditional expression of an alert rule.	$0.count > 1
Dashboard	The dashboard that is associated with the alert rule.	my-dashboard
FireCount	The cumulative number of triggers since the last alert notification was sent.	1
Fired	Indicates whether an alert was triggered. Valid values: true and false.	true
LastNotifiedAt	The time when the last alert notification was sent. The value is a UNIX timestamp.	1542164541

Log field	Description	Example
NotifyStatus	The notification status of an alert. Valid values: • Success: Alert notifications were sent. • Failed: Alert notifications failed to be sent. • NotNotified: No alert notification was sent. • PartialSuccess: Some of the alert notifications were sent.	Success
Reason	The reason why alert notifications failed to be sent or no alert notification was sent.	result type is not bool

Results	The parameters and results of each log query. The value is of the array type. For more information, see Results field.	<pre>[
		  {
		  "EndTime": 1542334900,
		  "FireResult": null,
		  "LogStore": "test-logstore",
		  "Query": "* | select count(1) as count",
		  "RawResultCount": 1,
		  "RawResults": [
		  {
		  "__time__": "1542334840",
		  "count": "0"
		  }
		  ],
		  "StartTime": 1542334840
		  }
		]</pre>

Status	The execution result of an alert. Valid values: Success and Failed.	Success

Results field

Log field	Description	Example
Query	The query statement.	* | select count(1) as count
LogStore	The Logstore in which data is queried.	my-logstore
StartTime	The beginning of the time range for a query.	2019-01-02 15:04:05
StartTimeTs	The beginning of the time range for a query. The value is a UNIX timestamp.	1542334840
EndTime	The end of the time range for a query.	2019-01-02 15:19:05
EndTimeTs	The end of the time range for a query. The value is a UNIX timestamp. The actual query time range is `[StartTime, EndTime)`.	1542334900

Log field	Description	Example

RawResults	The query result that is formatted in an array. Each element in the array is a log entry. An array can contain a maximum of 100 elements.	<pre>[
		  {
		  "__time__": "1542334840",
		  "count": "0"
		  }
		]</pre>

RawResultsAsKv	The query result that is formatted in key-value pairs. **Note** This field can be used as a template variable. No data is stored to a Logstore for this field.	[foo:0]
RawResultCount	The number of raw log entries that are returned.	1

FireResult	The log entry that records the triggers of an alert. If no alert is triggered, the value is NULL.	<pre>{
		  "__time__": "1542334840",
		  "count": "0"
		}</pre>

FireResultAsKv	The log entry that records the triggers of an alert. The log entry is formatted in key-value pairs. **Note** This field can be used as a template variable. No data is stored to a Logstore for this field.	[foo:0]

5.5. FAQ

5.5.1. A DingTalk alert notification fails to be sent and the error message "send webhook failed: unable to send request: Post" is reported. Why?

Problem description

A DingTalk alert notification fails to be sent and the error message `send webhook failed: unable to send request: Post ****` is reported.

Possible cause

The DingTalk certificate is not loaded on the physical machine on which Log Service is deployed.

Solution

Download the DingTalk certificate, import the certificate to the physical machine on which Log Service is deployed, and then restart the SlsEtlFramework# server role.

Procedure

1. Download the DingTalk certificate.
   i. Use a browser to visit *https://oapi.dingtalk.com*.
   The following information is returned:
   ```
   {"errcode":404,"errmsg":"The requested URI does not exist."}
   ```
   ii. Open the browser developer tool, click Security, and then view the certificate.
   iii. On the details tab of the **View certificate** dialog box, click **Copy to File**. Follow the wizard to save the DER-encoded binary certificate file to your computer and name the file *oapi.dingtalk.com-cert.pem.cer*.
2. Upload the DingTalk certificate to the physical machine on which Log Service is deployed and add the certificate to the trusted certificate list.
   i. Log on to the physical machine and upload the certificate to the physical machine.
   ii. Run the following command to convert the format of the certificate file:
   ```
   openssl x509 -inform der -in oapi.dingtalk.com-cert.pem.cer -out oapi.dingtalk.com-cert.pem
   ```
   In this command, *oapi.dingtalk.com-cert.pem.cer* specifies the certificate file that you download and *oapi.dingtalk.com-cert.pem* specifies the converted certificate file. You can change the name of the converted certificate file based on the actual situation.
   iii. Run the following command to obtain the path to the converted certificate file and the name of the certificate file that you need to back up:
   ```
   curl -v https://oapi.dingtalk.com/
   ```
   Information similar to the following example is returned.
   
   iv. Run the following command to back up the certificate file:
   ```
   cp -aL /etc/pki/tls/certs/ca-bundle.crt ./ca-bundle.crt.bak
   ```
   v. Run the following command to add the converted certificate to the trusted certificate list:
   ```
   cat oapi.dingtalk.com-cert.pem >> /etc/pki/tls/certs/ca-bundle.crt
   ```
3. Restart the SlsEtlFramework# server role.
   i. Log on to the Apsara Uni-manager Operations Console.
   ii. In the top navigation bar of the Apsara Uni-manager Operations Console, click **Products**. Then, choose **Base/Platforms > Apsara Infrastructure Management Framework**.
   iii. In the left-side navigation pane, click **Project & Cluster O&M**. On the page that appears, enter *sls* in the search box. In the search result, click the SlsEtlFramework# server role.
   iv. On the **Instances** tab, find the instance that you want to use and click Actions in the Actions column.
   v. Click **Restart**.

6.Real-time consumption

6.1. Overview

Log Service provides the real-time log consumption feature that allows you to read and write full data in the first-in, first-out (FIFO) order. This feature is similar to the features provided by Kafka. This topic describes the types of real-time consumption.

The following table describes the methods that you can use to process log data after the log data is sent to Log Service.

Method	Scenario	Timeliness	Retention period
Real-time log consumption	Stream computing and real-time computing	Real time	You can specify a retention period based on your business requirements.
Log query	Online query of recent hot data	Near real time with a latency of no more than 3 seconds in all cases and a latency of no more than 1 second in 99.9% cases	You can specify a retention period based on your business requirements.
Log shipping	Storage of full log data for offline analysis	A latency of 5 to 30 minutes	The retention period is based on the storage system.

Real-time log consumption

Log Service allows you to pull log data and consume the data in real time. The following procedure describes how log data is consumed from a shard:

1. Obtain a cursor based on the start time and end time of data consumption.
2. Read log data based on the cursor and step parameters and return the position of the next cursor.
3. Move the cursor to continuously consume log data.

- Use an SDK to consume log data

  Log Service provides SDKs in multiple programming languages, such as Java, Python, and Go. You can use an SDK to consume log data.

- Use consumer groups to consume log data

  Log Service provides an advanced method that allows you to consume log data by using consumer groups. A consumer group is a lightweight computing framework that allows multiple consumers to consume data from a Logstore at the same time. Shards are automatically allocated to consumers in a consumer group. Data is consumed in sequence based on the time when data is written to the Logstore. After a breakpoint, consumers can continue to consume data by using checkpoints. You can use consumer group SDKs in multiple programming languages, such as Go, Python, and Java, to consume log data.

- Use real-time computing systems to consume log data
  - Use Spark Streaming to consume log data.
  - Use Storm to consume log data
  - Use open source Flink to consume log data

- Use open source services to consume log data

  Use Flume to consume log data

6.2. Consume log data

This topic describes how to use an SDK to consume log data and preview log data in the Log Service console.

Use an SDK to consume log data

Log Service provides SDKs in various programming languages. You can use an SDK to consume log data. For more information, see the SDK Reference topic in *Log Service Developer Guide*. The following example shows how to use Log Service SDK for Java to consume log data from a shard:

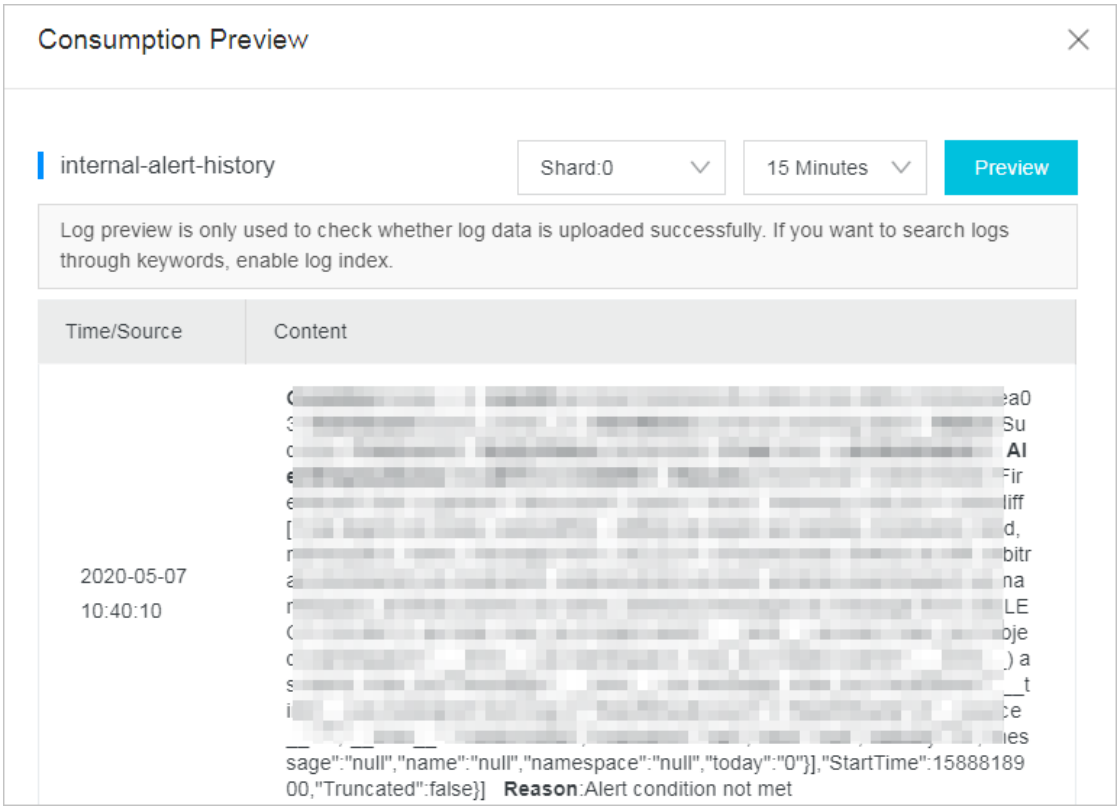
```
Client client = new Client(host, accessId, accessKey);
    String cursor = client.GetCursor(project, logStore, shardId, CursorMode.END).GetCursor();
    System.out.println("cursor = " +cursor);
    try {
      while (true) {
        PullLogsRequest request = new PullLogsRequest(project, logStore, shardId, 1000, cursor);
        PullLogsResponse response = client.pullLogs(request);
        System.out.println(response.getCount());
        System.out.println("cursor = " + cursor + " next_cursor = " + response.getNextCursor());
        if (cursor.equals(response.getNextCursor())) {
            break;
                }
        cursor = response.getNextCursor();
        Thread.sleep(200);
      }
    }
    catch(LogException e) {
      System.out.println(e.GetRequestId() + e.GetErrorMessage());
    }
```

Preview log data in the Log Service console

Consumption preview is a type of log data consumption. The consumption preview feature allows you to preview specific log data that is stored in a Logstore in the Log Service console.

1. Log on to the Log Service console.
2. In the Projects section, click the project from which you want to consume log data.
3. In the Logstores list, find the Logstore from which you want to consume log data, click the icon next to the Logstore, and then select **Consumption Preview**.
4. In the **Consumption Preview** panel, select a shard and a time range, and then click **Preview**.

   The Consumption Preview panel displays the log data of the first 10 packets in the specified time range.



6.3. Consumption by consumer groups

6.3.1. Use consumer groups to consume log data

If you use consumer groups to consume log data, you do not need to focus on factors such as Log Service implementation, load balancing among consumers, and failovers that may occur. This way, you can focus on the business logic during log data consumption.

Terms

Term	Description
consumer group	A consumer group consists of multiple consumers. Each consumer in a consumer group consumes different data in a Logstore. You can create a maximum of 30 consumer groups for a Logstore.
consumer	The consumers in a consumer group consume data. The name of each consumer in a consumer group must be unique.

A Logstore has multiple shards. A consumer library allocates shards to the consumers in a consumer group based on the following rules:

- You can allocate a shard to only one consumer.
- Each consumer can consume data from multiple shards.

After you add a consumer to a consumer group, shards that are allocated to the consumer group are reallocated to each consumer for load balancing. The shards are reallocated based on the preceding rules.

A consumer library stores checkpoints. This way, consumers can resume data consumption from a checkpoint and do not consume data after a program fault is resolved.

Procedure

You can use Java, Python, or Go to create consumers and consume data. The following procedure uses Java as an example:

1. Add Maven dependencies.

```
<dependency>
  <groupId>com.google.protobuf</groupId>
  <artifactId>protobuf-java</artifactId>
  <version>2.5.0</version>
</dependency>
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>loghub-client-lib</artifactId>
  <version>0.6.33</version>
</dependency>
```

2. Create a file named Main.java.

```
import com.aliyun.openservices.loghub.client.ClientWorker;
import com.aliyun.openservices.loghub.client.config.LogHubConfig;
import com.aliyun.openservices.loghub.client.exceptions.LogHubClientWorkerException;
public class Main {
    // The endpoint of Log Service. Set the parameter based on your business requirements.
    private static String sEndpoint = "cn-hangzhou.log.aliyuncs.com";
    // The name of a Log Service project. Set the parameter based on your business requirements.
    private static String sProject = "ali-cn-hangzhou-sls-admin";
    // The name of a Logstore. Set the parameter based on your business requirements.
    private static String sLogstore = "sls_operation_log";
    // The name of a consumer group. Set the parameter based on your business requirements.
    private static String sConsumerGroup = "consumerGroupX";
    // The AccessKey pair that is used to access Log Service. Specify the AccessKey ID and AccessK
ey secret based on your business requirements.
    private static String sAccessKeyId = "";
    private static String sAccessKey = "";
    public static void main(String[] args) throws LogHubClientWorkerException, InterruptedExceptio
n {
        // consumer_1 is the name of the consumer. The name of each consumer in a consumer group m
ust be unique. If different consumers start multiple processes on multiple servers to consume the
data of a Logstore, you can use a server IP address to identify a consumer.
        The maxFetchLogGroupSize parameter specifies the maximum number of log groups that you can
obtain from the server at the same time. Valid values: (0,1000]. We recommend that you use the def
ault value.
        LogHubConfig config = new LogHubConfig(sConsumerGroup, "consumer_1", sEndpoint, sProject,
sLogstore, sAccessKeyId, sAccessKey, LogHubConfig.ConsumePosition.BEGIN_CURSOR);
        ClientWorker worker = new ClientWorker(new SampleLogHubProcessorFactory(), config);
        Thread thread = new Thread(worker);
        // After you execute the thread, the ClientWorker instance automatically runs and extends
the Runnable interface.
        thread.start();
        Thread.sleep(60 * 60 * 1000);
        // The shutdown function of the ClientWorker instance is called to exit the consumption in
stance. The associated thread is automatically stopped.
        worker.shutdown();
        // Multiple asynchronous tasks are generated when the ClientWorker instance is running. To
ensure that all running tasks exit after shutdown, we recommend that you set Thread.sleep to 30 se
conds.
        Thread.sleep(30 * 1000);
    }
}
```

3. Create a file named SampleLogHubProcessor.java.

```
import com.aliyun.openservices.log.common.FastLog;
import com.aliyun.openservices.log.common.FastLogContent;
import com.aliyun.openservices.log.common.FastLogGroup;
import com.aliyun.openservices.log.common.FastLogTag;
import com.aliyun.openservices.log.common.LogGroupData;
import com.aliyun.openservices.loghub.client.ILogHubCheckPointTracker;
import com.aliyun.openservices.loghub.client.exceptions.LogHubCheckPointException;
import com.aliyun.openservices.loghub.client.interfaces.ILogHubProcessor;
import com.aliyun.openservices.loghub.client.interfaces.ILogHubProcessorFactory;
import java.util.List;
public class SampleLogHubProcessor implements ILogHubProcessor {
    private int shardId;
    // The point in time when the last persistent checkpoint was saved.
    private long mLastCheckTime = 0;
    public void initialize(int shardId) {
        this.shardId = shardId;
    }
    // The main logic of data consumption. You must include the code to handle all exceptions that
may occur during log data consumption.
    public String process(List<LogGroupData> logGroups,
                          ILogHubCheckPointTracker checkPointTracker) {
        // Display the data that you obtained.
        for (LogGroupData logGroup : logGroups) {
            FastLogGroup flg = logGroup.GetFastLogGroup();
            System.out.println(String.format("\tcategory\t:\t%s\n\tsource\t:\t%s\n\ttopic\t:\t%s\n
\tmachineUUID\t:\t%s",
                    flg.getCategory(), flg.getSource(), flg.getTopic(), flg.getMachineUUID()));
            System.out.println("Tags");
            for (int tagIdx = 0; tagIdx < flg.getLogTagsCount(); ++tagIdx) {
                FastLogTag logtag = flg.getLogTags(tagIdx);
                System.out.println(String.format("\t%s\t:\t%s", logtag.getKey(), logtag.getValue()
));
            }
            for (int lIdx = 0; lIdx < flg.getLogsCount(); ++lIdx) {
                FastLog log = flg.getLogs(lIdx);
                System.out.println("--------\nLog: " + lIdx + ", time: " + log.getTime() + ", GetC
ontentCount: " + log.getContentsCount());
                for (int cIdx = 0; cIdx < log.getContentsCount(); ++cIdx) {
                    FastLogContent content = log.getContents(cIdx);
                    System.out.println(content.getKey() + "\t:\t" + content.getValue());
                }
            }
        }
        long curTime = System.currentTimeMillis();
        // Write a checkpoint to the server every 30 seconds. If the ClientWorker instance unexpec
tedly stops within 30 seconds, a newly started ClientWorker instance continues to consume data fro
m the last checkpoint. A small amount of data may be repeatedly consumed.
        if (curTime - mLastCheckTime > 30 * 1000) {
            try {
                // If you set the parameter to true, checkpoints are immediately synchronized to t
he server. If you set the parameter to false, checkpoints are locally cached. The default value of
a synchronization interval of checkpoints is 60 seconds.
                checkPointTracker.saveCheckPoint(true);
            } catch (LogHubCheckPointException e) {
                e.printStackTrace();
            }
            mLastCheckTime = curTime;
        }
        return null;
    }
```

```
    }
    // The ClientWorker instance calls this function when the instance exits. You can delete the c
heckpoints.
    public void shutdown(ILogHubCheckPointTracker checkPointTracker) {
        // Save checkpoints to the server.
        try {
            checkPointTracker.saveCheckPoint(true);
        } catch (LogHubCheckPointException e) {
            e.printStackTrace();
        }
    }
}
class SampleLogHubProcessorFactory implements ILogHubProcessorFactory {
    public ILogHubProcessor generatorProcessor() {
        // Generate a consumption instance.
        return new SampleLogHubProcessor();
    }
}
```

For more information, see Java, Python, and Go.

View the status of a consumer group

You can use the Log Service console, call the API, or use an SDK to view the progress of your data consumption. For more information, see View the status of a consumer group.

Related operations

- Handle exceptions.

  We recommend that you configure Log4j for the consumer program to return error messages in consumer groups. This way, you can handle exceptions at the earliest opportunity. The following code shows a configuration file of log4j.properties:

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

  After you configure Log4j, you can view the information of exceptions that occur when you run the consumer program. The following example shows an error message:

  ```
  [WARN ] 2018-03-14 12:01:52,747 method:com.aliyun.openservices.loghub.client.LogHubConsumer.sampleL
  ogError(LogHubConsumer.java:159)
  com.aliyun.openservices.log.exception.LogException: Invalid loggroup count, (0,1000]
  ```

- Use a consumer group to consume data that is generated from a certain point in time.

```
// consumerStartTimeInSeconds indicates that the data generated after the point in time is consumed
.
public LogHubConfig(String consumerGroupName,
                    String consumerName,
                    String loghubEndPoint,
                    String project, String logStore,
                    String accessId, String accessKey,
                    int consumerStartTimeInSeconds);
// The value of the position parameter is an enumeration variable. LogHubConfig.ConsumePosition.BEG
IN_CURSOR indicates that the consumption starts from the earliest data. LogHubConfig.ConsumePositio
n.END_CURSOR indicates that the consumption starts from the latest data.
public LogHubConfig(String consumerGroupName,
                    String consumerName,
                    String loghubEndPoint,
                    String project, String logStore,
                    String accessId, String accessKey,
                    ConsumePosition position);
```

> **Note**
> - You can use different constructors based on your business requirements.
> - If a checkpoint is stored on the server, data consumption starts from this checkpoint.

- Reset a checkpoint.

```
public static void updateCheckpoint() throws Exception {
        Client client = new Client(host, accessId, accessKey);
        long timestamp = Timestamp.valueOf("2017-11-15 00:00:00").getTime() / 1000;
        ListShardResponse response = client.ListShard(new ListShardRequest(project, logStore));
        for (Shard shard : response.GetShards()) {
            int shardId = shard.GetShardId();
            String cursor = client.GetCursor(project, logStore, shardId, timestamp).GetCursor();
            client.UpdateCheckPoint(project, logStore, consumerGroup, shardId, cursor);
        }
    }
```

Authorize a RAM user to access consumer groups

Before you use a RAM user to access consumer groups, you must grant the required permissions to the RAM user. For more information, see Grant permissions to a RAM role.

The following table describes the actions that a RAM user can perform.

Action	Description	Resource
log:GetCursorOrData (GetCursor and PullLogs)	Obtains a cursor based on the point in time when log data is generated.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*
log:CreateConsumerGroup	Creates a consumer group for a specified Logstore.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*/consumergroup/*
log:ListConsumerGroup	Queries all consumer groups in a specified Logstore.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*/consumergroup/*

Action	Description	Resource
log:UpdateCheckPoint	Updates the consumption checkpoint in a shard of a specified consumer group.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*/consumergroup/*${consumerGroupName}*
log:ConsumerGroupHeartBeat	Sends a heartbeat packet to Log Service for a specified consumer.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*/consumergroup/*${consumerGroupName}*
log:UpdateConsumerGroup	Modifies the attributes of a specified consumer group.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*/consumergroup/*${consumerGroupName}*
log:ConsumerGroupUpdateCheckPoint	Retrieves the consumption checkpoints in one or all shards of a specified consumer group.	acs:log:*${regionName}*:*${projectOwnerAliUid}*:project/*${projectName}*/logstore/*${logstoreName}*/consumergroup/*${consumerGroupName}*

For example, the project-test project resides in the China (Hangzhou) region. The ID of the Apsara Stack tenant account to which the project belongs is 174649****602745. The name of the Logstore from which you want to consume log data is logstore-test, and the consumer group name is consumergroup-test. To allow a RAM user to access the consumer group, you must grant the following permissions to the RAM user:

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "log:GetCursorOrData"
      ],
      "Resource": "acs:log:cn-hangzhou:174649****602745:project/project-test/logstore/logstore-test"
    },
    {
      "Effect": "Allow",
      "Action": [
        "log:CreateConsumerGroup",
        "log:ListConsumerGroup"
      ],
      "Resource": "acs:log:cn-hangzhou:174649****602745:project/project-test/logstore/logstore-test/consumergroup/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "log:ConsumerGroupUpdateCheckPoint",
        "log:ConsumerGroupHeartBeat",
        "log:UpdateConsumerGroup",
        "log:GetConsumerGroupCheckPoint"
      ],
      "Resource": "acs:log:cn-hangzhou:174649****602745:project/project-test/logstore/logstore-test/consumergroup/consumergroup-test"
    }
  ]
}
```

6.3.2. View the status of a consumer group

This topic describes how to view the status of a consumer group.

View the consumption progress in the Log Service console

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. In the Logstores list, find the Logstore that you want to manage and choose > **Data Consumption**.
4. Click the consumer group whose data consumption progress you want to view. The data consumption progress of each shard in the Logstore is displayed.

Call the API or use an SDK to view the data consumption progress

The following code shows how to call the API to view the data consumption progress. In this example, Log Service SDK for Java is used.

```
package test;
import java.util.ArrayList;
import com.aliyun.openservices.log.Client;
import com.aliyun.openservices.log.common.Consts.CursorMode;
import com.aliyun.openservices.log.common.ConsumerGroup;
import com.aliyun.openservices.log.common.ConsumerGroupShardCheckPoint;
import com.aliyun.openservices.log.exception.LogException;
public class ConsumerGroupTest {
    static String endpoint = "";
    static String project = "";
    static String logstore = "";
    static String accesskeyId = "";
    static String accesskey = "";
    public static void main(String[] args) throws LogException {
        Client client = new Client(endpoint, accesskeyId, accesskey);
        // Obtain all consumer groups that are created for the Logstore. If no consumer group exists,
an empty string is returned.
        List<ConsumerGroup> consumerGroups = client.ListConsumerGroup(project, logstore).GetConsumerGr
oups();
        for(ConsumerGroup c: consumerGroups){
            // Print the properties of consumer groups. The properties of a consumer group include the
name, heartbeat timeout, and whether the consumer group consumes data in order.
            System.out.println("Name: " + c.getConsumerGroupName());
            System.out.println("Heartbeat timeout: " + c.getTimeout());
            System.out.println("Consumption in order: " + c.isInOrder());
            for(ConsumerGroupShardCheckPoint cp: client.GetCheckPoint(project, logstore, c.getConsumer
GroupName()).GetCheckPoints()){
                System.out.println("shard: " + cp.getShard());
                // The consumption time. The time is a long integer and is accurate to milliseconds.
                System.out.println("The last time when data was consumed: " + cp.getUpdateTime());
                System.out.println("Consumer name: " + cp.getConsumer());
                String consumerPrg = "";
                if(cp.getCheckPoint().isEmpty())
                    consumerPrg = "Consumption not started";
                else{
                    // The UNIX timestamp. Unit: seconds. Format the output value of the timestamp.
                    try{
                        int prg = client.GetPrevCursorTime(project, logstore, cp.getShard(), cp.getChe
ckPoint()).GetCursorTime();
                        consumerPrg = "" + prg;
                    }
```

```
                    catch(LogException e){
                        if(e.GetErrorCode() == "InvalidCursor")
                            consumerPrg = "Invalid. The previous point in time when data was consumed
is out of the retention period of the data in the Logstore";
                        else{
                            //internal server error
                            throw e;
                        }
                    }
                }
                System.out.println("Consumption progress: " + consumerPrg);
                String endCursor = client.GetCursor(project, logstore, cp.getShard(), CursorMode.END).
GetCursor();
                int endPrg = 0;
                try{
                    endPrg = client.GetPrevCursorTime(project, logstore, cp.getShard(), endCursor).Get
CursorTime();
                }
                catch(LogException e){
                    //do nothing
                }
                // The UNIX timestamp. Unit: seconds. Format the output value of the timestamp.
                System.out.println("The point in time when the last data entry was received: " + endPr
g);
            }
        }
    }
}
```

6.4. Use Storm to consume log data

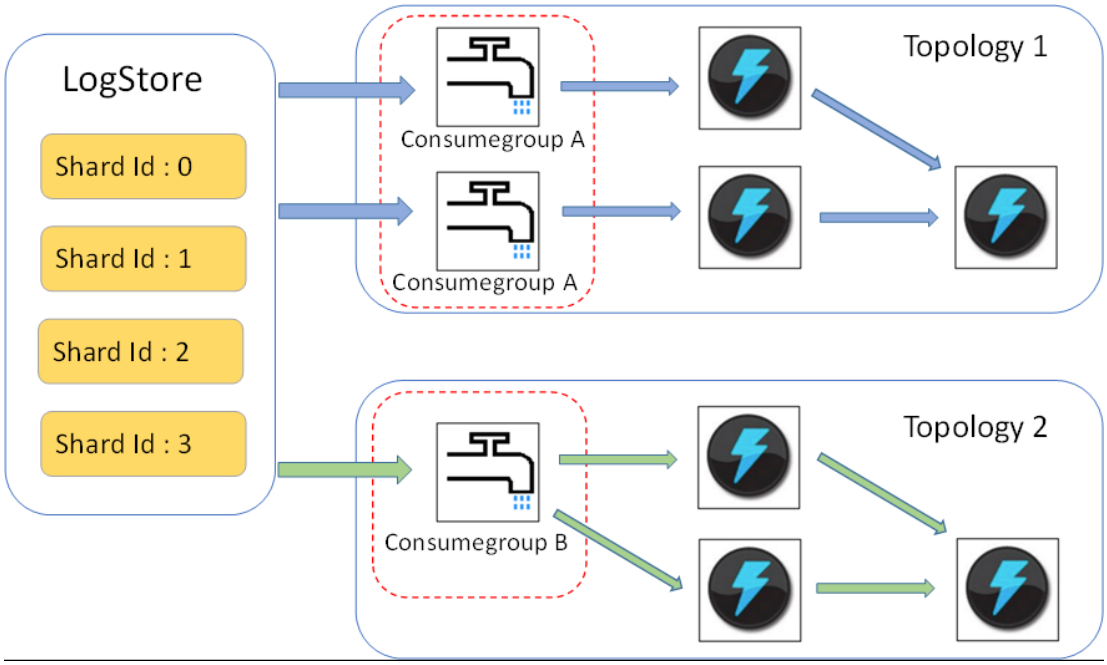
Log Service LogHub provides efficient and reliable log collection and consumption channels. You can use services such as Logtail or SDKs to collect log data in real time. After log data is collected and sent to Log Service, you can use stream computing systems such as Spark Streaming and Apache Storm to consume the log data.

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

Architecture and implementation

- In the following figure, LogHub Storm spouts are enclosed in red dashed-line boxes. Each Storm topology has a group of spouts that work together to read data from a Logstore. Spouts in different topologies are independent of each other.
- Each topology is identified by the unique name of a LogHub consumer group. Spouts in the same topology use a consumer group to perform load balancing and automatic failover. For more information, see Use consumer groups to consume log data.
- Spouts in a topology read data from a Logstore in real time, and then send the data to bolts in the topology. The spouts save consumption checkpoints to the LogHub server on a regular basis.

  Architecture and implementation



Limits

- You can create a maximum of 10 consumer groups to consume log data from a Logstore. If you no longer need a consumer group, you can call the DeleteConsumerGroup operation of the SDK for Java to delete the consumer group.
- We recommend that you configure the same number of spouts for a Logstore as the number of shards in the Logstore. This is because a single spout may not be able to process a large amount of data from multiple shards.
- If the data volume in a shard exceeds the processing capacity of a single spout, you can split the shard to reduce the data volume in each shard.
- LogHub Storm spouts and bolts must use the ack method to check whether log data is sent from spouts to bolts and whether the data is processed by the bolts.

Examples

- The following code provides an example to show how to construct a Storm topology:

```
     public static void main( String[] args )
    {
        String mode = "Local"; // Use the local test mode.
           String consumser_group_name = ""; // The name of the consumer group of a topology. The n
ame must be unique. The name cannot be an empty string. The name must be 3 to 63 characters in leng
th and can contain lowercase letters, digits, hyphens (-), and underscores (_). The name must start
and end with a lowercase letter or a digit.
        String project = ""; // The Log Service project.
        String logstore = ""; // The Log Service Logstore.
        String endpoint = ""; // The endpoint of Log Service.
        String access_id = ""; // The AccessKey ID.
        String access_key = "";
        // Configure a LogHub Storm spout.
        LogHubSpoutConfig config = new LogHubSpoutConfig(consumser_group_name,
                endpoint, project, logstore, access_id,
                access_key, LogHubCursorPosition.END_CURSOR);
        TopologyBuilder builder = new TopologyBuilder();
        // Create a LogHub Storm spout.
        LogHubSpout spout = new LogHubSpout(config);
        // In actual scenarios, we recommend that you create the same number of spouts for a Logsto
re as the number of shards in the Logstore.
```

```
        builder.setSpout("spout", spout, 1);
        builder.setBolt("exclaim", new SampleBolt()).shuffleGrouping("spout");
        Config conf = new Config();
        conf.setDebug(false);
        conf.setMaxSpoutPending(1);
        // If you use Kryo to serialize and deserialize data, configure the serialization method of
LogGroupData by using the LogGroupDataSerializSerializer class.
        Config.registerSerialization(conf, LogGroupData.class, LogGroupDataSerializSerializer.class
);
        if (mode.equals("Local")) {
            logger.info("Local mode...");
            LocalCluster cluster  = new LocalCluster();
            cluster.submitTopology("test-jstorm-spout", conf, builder.createTopology());
            try {
                Thread.sleep(6000 * 1000);    //waiting for several minutes
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            cluster.killTopology("test-jstorm-spout");
            cluster.shutdown();
        } else if (mode.equals("Remote")) {
            logger.info("Remote mode...");
            conf.setNumWorkers(2);
            try {
                StormSubmitter.submitTopology("stt-jstorm-spout-4", conf, builder.createTopology())
;
            } catch (AlreadyAliveException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InvalidTopologyException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } else {
            logger.error("invalid mode: " + mode);
        }
    }
}
```

- The following code provides an example to show how to consume log data, and then display the content of each log entry by using bolts:

```
public class SampleBolt extends BaseRichBolt {
    private static final long serialVersionUID = 4752656887774402264L;
    private static final Logger logger = Logger.getLogger(BaseBasicBolt.class);
    private OutputCollector mCollector;
    @Override
    public void prepare(@SuppressWarnings("rawtypes") Map stormConf, TopologyContext context,
            OutputCollector collector) {
        mCollector = collector;
    }
    @Override
    public void execute(Tuple tuple) {
        String shardId = (String) tuple
                .getValueByField(LogHubSpout.FIELD_SHARD_ID);
        @SuppressWarnings("unchecked")
        List<LogGroupData> logGroupDatas = (ArrayList<LogGroupData>) tuple.getValueByField(LogHubSp
out.FIELD_LOGGROUPS);
        for (LogGroupData groupData : logGroupDatas) {
            // Each log group consists of one or more log entries.
            LogGroup logGroup = groupData.GetLogGroup();
            for (Log log : logGroup.getLogsList()) {
                StringBuilder sb = new StringBuilder();
                // Each log entry contains a time field and other fields that are formatted in key-
value pairs.
                int log_time = log.getTime();
                sb.append("LogTime:").append(log_time);
                for (Content content : log.getContentsList()) {
                    sb.append("\t").append(content.getKey()).append(":")
                            .append(content.getValue());
                }
                logger.info(sb.toString());
            }
        }
        // LogHub spouts and bolts must use the ack method to check whether log data is sent from s
pouts to bolts and whether the data is processed by the bolts.
        mCollector.ack(tuple);
    }
    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        //do nothing
    }
}
```

Maven

The following code provides an example to show how to add Maven dependencies for Storm 1.0 or earlier, such as Storm 0.9.6:

```
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>loghub-storm-spout</artifactId>
  <version>0.6.6</version>
</dependency>
```

The following code provides an example to show how to add Maven dependencies for Storm 1.0 or later:

```
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>loghub-storm-1.0-spout</artifactId>
  <version>0.1.3</version>
</dependency>
```

6.5. Use Flume to consume log data

This topic describes how to use Flume to consume log data. You can use the aliyun-log-flume plug-in to connect Log Service to Flume and write log data to Log Service or consume log data from Log Service.

Context

The aliyun-log-flume plug-in connects Log Service to Flume. When Log Service is connected to Flume, you can use Flume to connect Log Service to other systems such as Hadoop distributed file system (HDFS) and Kafka. The aliyun-log-flume plug-in provides sinks and sources to connect Log Service to Flume.

- Sink: reads data from other data sources and writes the data to Log Service.
- Source: consumes log data from Log Service and writes the log data to other systems.

For more information, visit GitHub.

Procedure

1. Download and install Flume. For more information, see Flume.
2. Download the aliyun-log-flume plug-in and save the plug-in in the *cd/***/flume/lib* directory. To download the plug-in, click aliyun-log-flume-1.3.jar.
3. In the *cd/***/flume/conf* directory, create a configuration file named flumejob.conf.
   - For information about how to configure a sink, see Sink.
   - For information about how to configure a source, see Source.
4. Start Flume.

Sink

You can configure a sink for Flume to write data from other data sources to Log Service. Data can be parsed into the following two formats:

- SIMPLE: A Flume event is written to Log Service as a field.
- DELIMITED: A Flume event is parsed into fields based on the configured column names and written to Log Service.

The following table describes the parameters of a sink.

Parameter	Required	Description
type	Yes	Default value: `com.aliyun.Loghub.flume.sink.LoghubSink`.
endpoint	Yes	The endpoint of the region where the Log Service project resides.
project	Yes	The name of the project.
logstore	Yes	The name of the Logstore.
accessKeyId	Yes	The AccessKey ID that is used to access Log Service. For more information, see Obtain an AccessKey pair.

Parameter	Required	Description
accessKey	Yes	The AccessKey secret that is used to access Log Service. For more information, see Obtain an AccessKey pair.
batchSize	No	The number of data entries that are written to Log Service at a time. Default value: 1000.
maxBufferSize	No	The maximum number of data entries in the buffer. Default value: 1000.
serializer	No	The serialization format of the Flume event. Default value: SIMPLE. Valid values: • **DELIMITED**: delimiter mode. • **SIMPLE**: single-line mode. • Custom **serializer**: custom serialization mode. In this mode, you must specify the full names of columns.
columns	No	The names of colums. If you set the serializer parameter to **DELIMITED**, you must specify this parameter. Separate multiple columns with commas (,). The columns are sorted in the same order that the columns are sorted in the data entries.
separatorChar	No	The delimiter. If you set the serializer parameter to **DELIMITED**, you must specify a single character for this parameter. The default value is a comma (,).
quoteChar	No	The quote character. If you set the serializer parameter to **DELIMITED**, you must specify this parameter. The default value is double quotation marks (").
escapeChar	No	The escape character. If you set the serializer parameter to **DELIMITED**, you must specify this parameter. The default value is double quotation marks (").
useRecordTime	No	Specifies whether to use the value of the timestamp field in the data entries as the log time when data is written to Log Service. Default value: false. This value indicates that the current time is used as the log time.

For more information about how to configure a sink, visit GitHub.

Source

You can configure a source for Flume to ship data from Log Service to other data sources. Data can be parsed into the following two formats:

- DELIMITED: Log data is written to Flume in delimiter mode.
- JSON: Log data is written to Flume in the JSON format.

The following table describes the parameters of a source.

Parameter	Required	Description
type	Yes	Default value: `com.aliyun.loghub.flume.source.LoghubSource`.
endpoint	Yes	The endpoint of the region where the Log Service project resides.

Parameter	Required	Description
project	Yes	The name of the project.
logstore	Yes	The name of the Logstore.
accessKeyId	Yes	The AccessKey ID that is used to access Log Service. For more information, see Obtain an AccessKey pair.
accessKey	Yes	The AccessKey secret that is used to access Log Service. For more information, see Obtain an AccessKey pair.
heartbeatIntervalMs	No	The heartbeat interval between the client and Log Service. Default value: 30000. Unit: milliseconds.
fetchIntervalMs	No	The interval at which data is pulled from Log Service. Default value: 100. Unit: milliseconds.
fetchInOrder	No	Specifies whether to consume log data in the order that the log data is written to Log Service. Default value: false.
batchSize	No	The number of log entries that are read at a time. Default value: 100.
consumerGroup	No	The name of the consumer group that reads log data.
initialPosition	No	The start point from which data is read. Valid values: **begin**, **end**, and **timestamp**. Default value: **begin**. **Note** If a checkpoint exists on the server, the checkpoint is preferentially used.
timestamp	No	The UNIX timestamp. If you set the initialPosition parameter to **timestamp**, you must specify this parameter.
deserializer	Yes	The deserialization format of the Flume event. Default value: DELIMITED. Valid values: • **DELIMITED**: delimiter mode. • **JSON**: JSON format. • Custom **deserializer**: custom deserialization mode. In this mode, you must specify the full names of the columns.
columns	No	The names of colums. If you set the deserializer parameter to **DELIMITED**, you must specify this parameter. Separate multiple columns with commas (,). The columns are sorted in the same order that the columns are sorted in the log entries.
separatorChar	No	The delimiter. If you set the deserializer parameter to **DELIMITED**, you must specify a single character for this parameter. The default value is a comma (,).
quoteChar	No	The quote character. If you set the deserializer parameter to **DELIMITED**, you must specify this parameter. The default value is double quotation marks (").

Parameter	Required	Description
escapeChar	No	The escape character. If you set the deserializer parameter to **DELIMITED**, you must specify this parameter. The default value is double quotation marks (").
appendTimestamp	No	Specifies whether to append the timestamp as a field to the end of each log entry. If you set the deserializer parameter to **DELIMITED**, you must specify this parameter. Default value: false.
sourceAsField	No	Specifies whether to add the log source as a field named __source__. If you set the deserializer parameter to **JSON**, you must specify this parameter. Default value: false.
tagAsField	No	Specifies whether to add the log tag as a field named __tag__:{name of the tag}. If you set the deserializer parameter to **JSON**, you must specify this parameter. Default value: false.
timeAsField	No	Specifies whether to add the log time as a field named __time__. If you set the deserializer parameter to **JSON**, you must specify this parameter. Default value: false.
useRecordTime	No	Specifies whether to use the value of the timestamp field in the log entries as the log time when log data is read from Log Service. Default value: false. This value indicates that the current time is used as the log time.

For more information about how to configure a source, visit GitHub.

6.6. Use open source Flink to consume log data

Log Service provides the flink-log-connector agent to connect with Flink. This topic describes how to connect Log Service with Flink to consume log data.

Prerequisites

- An AccessKey pair is obtained. For more information, see Obtain an AccessKey pair.
- A project and a Logstore are created. For more information, see Create a project and Create a Logstore.
- A RAM user is authorized to access the Logstore from which you want to consume data. For more information, see Grant a RAM user the permissions to consume data from a specified Logstore.

Context

The flink-log-connector agent consists of the flink-log-consumer and flink-log-producer agents. The two agents have the following differences:

- The flink-log-consumer agent reads data from Log Service. This agent supports the exactly-once semantics and load balancing among shards.
- The flink-log-producer agent writes data to Log Service.

Before you can use the flink-log-producer agent to write data to Log Service, you must add Maven dependencies. The following example shows sample Maven dependencies:

```
<dependency>
   <groupId>com.aliyun.openservices</groupId>
   <artifactId>flink-log-connector</artifactId>
   <version>0.1.13</version>
</dependency>
<dependency>
   <groupId>com.google.protobuf</groupId>
   <artifactId>protobuf-java</artifactId>
   <version>2.5.0</version>
</dependency>
```

For more information, visit GitHub.

Flink Log Consumer

The flink-log-consumer agent can consume log data from a Logstore based on the exactly-once semantics. The flink-log-consumer agent detects the change in the number of shards in a Logstore.

Each Flink subtask consumes data from some shards in a Logstore. If the shards in a Logstore are split or merged, the shards from which the subtask consumes data also change.

If you use the flink-log-consumer agent to consume data from Log Service, you can call the following API operations:

- GetCursorOrData

  You can call this operation to pull log data from a shard. If you frequently call this operation, the amount of data that is transferred may exceed the capacity of shards. You can use the ConfigConstants.LOG_FETCH_DATA_INTERVAL_MILLIS parameter to specify the interval of API calls. You can use the ConfigConstants.LOG_MAX_NUMBER_PER_FETCH parameter to specify the number of log entries pulled by each call.

  Example:

  ```
  configProps.put(ConfigConstants.LOG_FETCH_DATA_INTERVAL_MILLIS, "100");
  configProps.put(ConfigConstants.LOG_MAX_NUMBER_PER_FETCH, "100");
  ```

- ListShards

  You can call this operation to view all shards in a Logstore and the status of each shard. If the shards are frequently split and merged, you can adjust the call interval to detect the changes in the number of shards at the earliest opportunity. Example:

  ```
  // Call the ListShards operation every 30,000 milliseconds.
  configProps.put(ConfigConstants.LOG_SHARDS_DISCOVERY_INTERVAL_MILLIS, "30000");
  ```

- CreateConsumerGroup

  You can call this operation to create a consumer group that is used to synchronize checkpoints.

- ConsumerGroupUpdateCheckPoint

  You can call this operation to synchronize snapshots of Flink to a consumer group.

1. Set startup parameters.

   The following example shows how to consume log data. The java.util.Properties class is used as a configuration tool. All constants must be configured in the ConfigConstants class.

```
Properties configProps = new Properties();
// Specify the endpoint of Log Service.
configProps.put(ConfigConstants.LOG_ENDPOINT, "cn-hangzhou.log.aliyuncs.com");
// Specify the AccessKey ID and AccessKey secret.
configProps.put(ConfigConstants.LOG_ACCESSKEYID, "");
configProps.put(ConfigConstants.LOG_ACCESSKEY, "");
// Specify the project.
configProps.put(ConfigConstants.LOG_PROJECT, "ali-cn-hangzhou-sls-admin");
// Specify the Logstore.
configProps.put(ConfigConstants.LOG_LOGSTORE, "sls_consumergroup_log");
// Specify the start position of log consumption.
configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.LOG_END_CURSOR);
// Specify the data deserialization method.
RawLogGroupListDeserializer deserializer = new RawLogGroupListDeserializer();
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
DataStream<RawLogGroupList> logTestStream = env.addSource(
        new FlinkLogConsumer<RawLogGroupList>(deserializer, configProps));
```

Note The number of Flink subtasks is independent of the number of shards in a Logstore. If the number of shards is greater than the number of subtasks, each subtask consumes one or more shards. If the number of shards is less than the number of subtasks, some subtasks remains idle until new shards are generated. The data of each shard is consumed by only one subtask.

2. Specify the start position of log consumption.

   If you use the flink-log-consumer agent to consume data from a Logstore, you can use the ConfigConstants.LOG_CONSUMER_BEGIN_POSITION parameter to specify the start position of log consumption. You can start to consume data from the earliest entry, the latest entry, or from a specific point in time. The flink-log-consumer agent also allows you to resume consumption from a specific consumer group. You can set the parameter to one of the following values:

   - Consts.LOG_BEGIN_CURSOR: starts to consume data from the earliest entry.
   - Consts.LOG_END_CURSOR: starts to consume data from the latest entry.
   - Consts.LOG_FROM_CHECKPOINT: starts to consume data from a checkpoint that is stored in a specified consumer group. You can use the ConfigConstants.LOG_CONSUMERGROUP parameter to specify the consumer group.
   - UnixTimestamp: a string of the integer data type. The timestamp follows the UNIX time format. It is the number of seconds that have elapsed since 00:00:00 Thursday, January 1, 1970. The value indicates that the data in a shard is consumed from this point in time.

   Example:

   ```
   configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.LOG_BEGIN_CURSOR);
   configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.LOG_END_CURSOR);
   configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, "1512439000");
   configProps.put(ConfigConstants.LOG_CONSUMER_BEGIN_POSITION, Consts.LOG_FROM_CHECKPOINT);
   ```

   Note If you configure to resume consumption from a state backend of Flink when you start a Flink job, the flink-log-connector agent uses checkpoints that are stored in the state backend.

3. (Optional)Configure consumption progress monitoring.

   The flink-log-connector agent allows you to monitor the consumption progress. You can use the monitoring feature to obtain the consumption position of each shard in real time. The consumption position is indicated by a timestamp. For more information, see View the status of a consumer group.

   Example:

```
configProps.put(ConfigConstants.LOG_CONSUMERGROUP, "your consumer group name");
```

> **Note** This setting is optional. If you configure consumption progress monitoring and no consumer group exists, the flink-log-connector agent creates a consumer group. If a consumer group is available, the agent synchronizes snapshots to the consumer group. You can view the consumption progress of the agent in the Log Service console.

4. Configure consumption resumption and the exactly-once semantics.

   If the checkpointing feature of Flink is enabled, the flink-log-consumer agent periodically saves the consumption progress of each shard. If a subtask fails, Flink restores the subtask and starts to consume data from the latest checkpoint.

   When you specify the checkpoint period, you can specify the maximum amount of data that can be re-consumed if a subtask fails. You can use the following code to specify the checkpoint period:

   ```
   final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
   // Configure the exactly-once semantics.
   env.getCheckpointConfig().setCheckpointingMode(CheckpointingMode.EXACTLY_ONCE);
   // Save checkpoints every 5 seconds.
   env.enableCheckpointing(5000);
   ```

   For more information, see Checkpoints.

Flink Log Producer

The flink-log-producer agent writes data to Log Service.

> **Note** The flink-log-producer agent supports only the Flink at-least-once semantics. If a subtask fails, duplicate data may be written to Log Service. However, no data is lost.

If you use the flink-log-producer agent to write data to Log Service, you can call the following API operations:

- PostLogStoreLogs
- ListShards

1. Initialize the flink-log-producer agent.
   i. Set the initialization parameters of the flink-log-producer agent.

   The flink-log-producer agent is initialized the same way as the flink-log-consumer agent. The following example shows how to configure the initialization parameters of the flink-log-producer agent. In most cases, you can use the default values of the parameters. Example:

   ```
   // The number of I/O threads that are used to send data. Default value: 8.
   ConfigConstants.LOG_SENDER_IO_THREAD_COUNT
   // The time that is required to send cached logs. Default value: 3000.
   ConfigConstants.LOG_PACKAGE_TIMEOUT_MILLIS
   // The number of logs in the cached packet. Default value: 4096.
   ConfigConstants.LOG_LOGS_COUNT_PER_PACKAGE
   // The size of the cached packet. Default value: 3. Unit: MB.
   ConfigConstants.LOG_LOGS_BYTES_PER_PACKAGE
   // The total size of memory that can be used by the job. Default value: 100. Unit: MB.
   ConfigConstants.LOG_MEM_POOL_BYTES
   ```

ii. Reload LogSerializationSchema and define the method that is used to serialize data into raw log groups.

A raw log group is a collection of log entries.

If you want to write data to a specific shard, you can use the LogPartitioner parameter to generate hash keys for log data. LogPartitioner is an optional parameter. If you do not specify this parameter, data is written to a random shard.

Example:

```
FlinkLogProducer<String> logProducer = new FlinkLogProducer<String>(new SimpleLogSerializer(),
configProps);
logProducer.setCustomPartitioner(new LogPartitioner<String>() {
      // Generate a 32-bit hash value.
      public String getHashKey(String element) {
          try {
              MessageDigest md = MessageDigest.getInstance("MD5");
              md.update(element.getBytes());
              String hash = new BigInteger(1, md.digest()).toString(16);
              while(hash.length() < 32) hash = "0" + hash;
              return hash;
          } catch (NoSuchAlgorithmException e) {
          }
          return  "00000000000000000000000000000000000000000000000000000000000000000";
      }
  });
```

2. Write simulated data to Log Service, as shown in the following example:

```
// Serialize data into the format of raw log groups.
class SimpleLogSerializer implements LogSerializationSchema<String> {
    public RawLogGroup serialize(String element) {
        RawLogGroup rlg = new RawLogGroup();
        RawLog rl = new RawLog();
        rl.setTime((int)(System.currentTimeMillis() / 1000));
        rl.addContent("message", element);
        rlg.addLog(rl);
        return rlg;
    }
}
public class ProducerSample {
    public static String sEndpoint = "cn-hangzhou.log.aliyuncs.com";
    public static String sAccessKeyId = "";
    public static String sAccessKey = "";
    public static String sProject = "ali-cn-hangzhou-sls-admin";
    public static String sLogstore = "test-flink-producer";
    private static final Logger LOG = LoggerFactory.getLogger(ConsumerSample.class);
    public static void main(String[] args) throws Exception {
        final ParameterTool params = ParameterTool.fromArgs(args);
        final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment(
);
        env.getConfig().setGlobalJobParameters(params);
        env.setParallelism(3);
        DataStream<String> simpleStringStream = env.addSource(new EventsGenerator());
        Properties configProps = new Properties();
        // Specify the endpoint of Log Service.
        configProps.put(ConfigConstants.LOG_ENDPOINT, sEndpoint);
        // Specify the AccessKey ID and AccessKey secret.
        configProps.put(ConfigConstants.LOG_ACCESSKEYID, sAccessKeyId);
        configProps.put(ConfigConstants.LOG_ACCESSKEY, sAccessKey);
        // Specify the project to which logs are written.
        configProps.put(ConfigConstants.LOG_PROJECT, sProject);
        // Specify the Logstore to which logs are written.
        configProps.put(ConfigConstants.LOG_LOGSTORE, sLogstore);
        FlinkLogProducer<String> logProducer = new FlinkLogProducer<String>(new SimpleLogSerialize
r(), configProps);
        simpleStringStream.addSink(logProducer);
        env.execute("flink log producer");
    }
    // Simulate log generation.
    public static class EventsGenerator implements SourceFunction<String> {
        private boolean running = true;
        @Override
        public void run(SourceContext<String> ctx) throws Exception {
            long seq = 0;
            while (running) {
                Thread.sleep(10);
                ctx.collect((seq++) + "-" + RandomStringUtils.randomAlphabetic(12));
            }
        }
        @Override
        public void cancel() {
            running = false;
        }
    }
}
```

6.7. Use Logstash to consume log data

Log Service provides Logstash that you can use to consume log data. You can configure the Logstash input plug-in to read log data from Log Service, and then write the data to other systems, such as Kafka and Hadoop Distributed File System (HDFS).

Features

- Distributed collaborative consumption: You can configure multiple servers to consume log data from a Logstore at the same time.
- High performance: If you use a Java consumer group, the consumption speed of a single-core CPU can reach 20 MB/s.
- High reliability: Log Service saves consumption checkpoints. This mechanism resumes log consumption from the last checkpoint after a consumption exception is resolved.
- Automatic load balancing: Shards are automatically allocated based on the number of consumers in a consumer group. If you add a consumer to a consumer group or remove a consumer from the consumer group, shards are automatically reallocated.

Procedure

1. Install Logstash.
   i. Download the Logstash installation package.
   ii. Decompress the package that you downloaded to the specified directory.
2. Install the Logstash input plug-in.
   i. Download the input plug-in logstash-input-sls.
   ii. Install the Logstash input plug-in.

   ```
   logstash-plugin install logstash-input-sls
   ```

   Note For information about the causes of installation failures and solutions, see Plug-in installation and configuration.
3. Start Logstash.

   ```
   logstash -f logstash.conf
   ```

   The following table describes the parameters of the Logstash input plug-in.

Parameter	Type	Required	Description
endpoint	String	Yes	The endpoint of the region where the Log Service project resides.
access_id	String	Yes	The AccessKey ID of an Apsara Stack tenant account or RAM user that is used to access the project. The Apsara Stack tenant account or RAM user must have the permissions to consume log data by using consumer groups. For more information, see Permission to consume data of a specified Logstore.
access_key	String	Yes	The AccessKey secret of an Apsara Stack tenant account or RAM user that is used to access the project. The Apsara Stack tenant account or RAM user must have the permissions to consume log data by using consumer groups. For more information, see Permission to consume data of a specified Logstore.

Parameter	Type	Required	Description
project	String	Yes	The name of the Log Service project.
logstore	String	Yes	The name of the Log Service Logstore.
consumer_group	String	Yes	The name of the consumer group.
consumer_name	String	Yes	The name of the consumer. The name of each consumer in a consumer group must be unique.
position	String	Yes	The position where data consumption starts. Valid values: ◦ **begin**: Data is consumed from the first log entry that is written to the Logstore. ◦ **end**: Data is consumed from the current point in time. ◦ **yyyy-MM-dd HH:mm:ss**: Data is consumed from the specified point in time.
checkpoint_second	Number	No	The interval at which checkpoints are recorded. We recommend that you set the interval to a value between 10 and 60. Minimum value: 10. Default value: 30. Unit: seconds.
include_meta	Boolean	No	Specifies whether input log data contains metadata, such as the log source, time, tags, and topic fields. Default value: true. ◦ true: The log data contains metadata. ◦ false: The log data does not contain metadata.
consumer_name_with_ip	Boolean	No	Specifies whether to include an IP address in a consumer name. Default value: true. You must set this parameter to true if you want to apply distributed collaborative consumption. ◦ true: The name of the consumer contains an IP address. ◦ false: The name of the consumer does not contain an IP address.

Example

The following script shows how to configure Logstash to consume log data from a Logstore, and then print the data in stdout logs:

```
input {
  logservice{
  endpoint => "your project endpoint"
  access_id => "your access id"
  access_key => "your access key"
  project => "your project name"
  logstore => "your logstore name"
  consumer_group => "consumer group name"
  consumer_name => "consumer name"
  position => "end"
  checkpoint_second => 30
  include_meta => true
  consumer_name_with_ip => true
  }
}
output {
  stdout {}
}
```

6.8. Use Spark Streaming to consume log data

After Log Service collects log data, you can use Spark Streaming to consume the data.

The Spark SDK provided by Alibaba Cloud allows you to consume log data from Log Service in Receiver or Direct mode. You must add the following Maven dependency:

```
<dependency>
  <groupId>com.aliyun.emr</groupId>
  <artifactId>emr-logservice_2.11</artifactId>
  <version>1.7.2</version>
</dependency>
```

Consume log data in Receiver mode

In Receiver mode, a consumer group consumes data from Log Service and temporarily stores the data in a Spark executor. After a Spark Streaming job is started, the consumer group reads and processes data from the Spark executor. Each log entry is returned as a JSON string. The consumer group periodically saves checkpoints to Log Service. You do not need to update checkpoints. For more information, see Use consumer groups to consume log data.

- Parameters

Parameter	Type	Description
project	String	The name of the Log Service project.
logstore	String	The name of the Log Service Logstore.
consumerGroup	String	The name of the consumer group.
endpoint	String	The endpoint of the region where the Log Service project resides.
accessKeyId	String	The AccessKey ID that is used to access Log Service.

Parameter	Type	Description
accessKeySecret	String	The AccessKey secret that is used to access Log Service.

- Example

> **Note** In Receiver mode, data loss may occur if the default configurations are used. To avoid data loss, you can enable the Write-Ahead Logs feature. This feature is available in Spark 1.2 or later. For more information, see Spark.

```
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.aliyun.logservice.LoghubUtils
import org.apache.spark.streaming.{Milliseconds, StreamingContext}
import org.apache.spark.SparkConf
object TestLoghub {
  def main(args: Array[String]): Unit = {
    if (args.length < 7) {
      System.err.println(
        """Usage: TestLoghub <project> <logstore> <loghub group name> <endpoint>
          |         <access key id> <access key secret> <batch interval seconds>
        """.stripMargin)
      System.exit(1)
    }
    val project = args(0)
    val logstore = args(1)
    val consumerGroup = args(2)
    val endpoint = args(3)
    val accessKeyId = args(4)
    val accessKeySecret = args(5)
    val batchInterval = Milliseconds(args(6).toInt * 1000)
    def functionToCreateContext(): StreamingContext = {
      val conf = new SparkConf().setAppName("Test Loghub")
      val ssc = new StreamingContext(conf, batchInterval)
      val loghubStream = LoghubUtils.createStream(
        ssc,
        project,
        logstore,
        consumerGroup,
        endpoint,
        accessKeyId,
        accessKeySecret,
        StorageLevel.MEMORY_AND_DISK)
      loghubStream.checkpoint(batchInterval * 2).foreachRDD(rdd =>
        rdd.map(bytes => new String(bytes)).top(10).foreach(println)
      )
      ssc.checkpoint("hdfs:///tmp/spark/streaming") // set checkpoint directory
      ssc
    }
    val ssc = StreamingContext.getOrCreate("hdfs:///tmp/spark/streaming", functionToCreateContext _
)
    ssc.start()
    ssc.awaitTermination()
  }
}
```

Consume log data in Direct mode

In Direct mode, you can consume log data from Log Service without the need of consumer groups. You can call API operations to request data from Log Service. Consuming log data in Direct mode has the following benefits:

- Simplified concurrency. The number of Spark partitions is the same as the number of shards in a Logstore. You can split shards to improve the concurrency of tasks.
- Increased efficiency. You no longer need to enable the Write-Ahead Logs feature to prevent data loss.
- Exactly-once semantics. Data is directly read from Log Service. Checkpoints are submitted after a task is successful.

  In some cases, data may be repeatedly consumed if a task ends due to an unexpected exit of Spark.

If you want to consume data in Direct mode, you must configure the ZooKeeper service to store intermediate data. In addition, you must specify a checkpoint directory in the ZooKeeper service. Intermediate data is stored in the checkpoint directory. To re-consume data after you restart a task, you must delete the checkpoint directory from ZooKeeper and change the name of the consumer group.

- Parameters

Parameter	Type	Description
project	String	The name of the Log Service project.
logstore	String	The name of the Log Service Logstore.
consumerGroup	String	The name of the consumer group. This name is used only to save consumption checkpoints.
endpoint	String	The endpoint of the region where the Log Service project resides.
accessKeyId	String	The AccessKey ID that is used to access Log Service.
accessKeySecret	String	The AccessKey secret that is used to access Log Service.
zkAddress	String	The connection URL of the ZooKeeper service.

- Throttling configuration

  Spark Streaming consumes data from each shard in a single batch. You must specify the number of log entries that are consumed in each batch.

  In the underlying storage model of Log Service, a log group serves as the basic storage unit. Each log group corresponds to a write request. For example, a write request may contain multiple log entries. These log entries are stored and consumed as a log group. When you use web tracking to write logs, each write request contains only one log entry. In this case, the log group that corresponds to the request contains only one log entry. You can specify parameters to limit the amount of log data in a single batch. The following table describes the two parameters.

Parameter	Description	Default
spark.loghub.batchGet.step	The maximum number of log groups that are returned for a single consumption request.	100

Parameter	Description	Default
spark.streaming.loghub.maxRatePerShard	The maximum number of log entries that are consumed from each shard in a single batch.	10000

You can set the spark.streaming.loghub.maxRatePerShard parameter to specify the maximum number of log entries that are consumed from each shard in each batch. The Spark SDK obtains the number of log groups from the spark.loghub.batchGet.step parameter before it consumes log data from Log Service, and accumulates the number of log entries in these log groups during the consumption. When the accumulated number reaches or exceeds the specified number in the spark.streaming.loghub.maxRatePerShard parameter, the Spark SDK stops consuming log data. The spark.streaming.loghub.maxRatePerShard parameter does not precisely control the number of consumed log entries in each batch. The number of consumed log entries in each batch varies based on the value of the spark.loghub.batchGet.step parameter and the number of log entries in each log group.

- Example

```
import com.aliyun.openservices.loghub.client.config.LogHubCursorPosition
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Milliseconds, StreamingContext}
import org.apache.spark.streaming.aliyun.logservice.{CanCommitOffsets, LoghubUtils}
object TestDirectLoghub {
  def main(args: Array[String]): Unit = {
    if (args.length < 7) {
      System.err.println(
        """Usage: TestDirectLoghub <project> <logstore> <loghub group name> <endpoint>
          |         <access key id> <access key secret> <batch interval seconds> <zookeeper host:po
rt=localhost:2181>
        """.stripMargin)
      System.exit(1)
    }
    val project = args(0)
    val logstore = args(1)
    val consumerGroup = args(2)
    val endpoint = args(3)
    val accessKeyId = args(4)
    val accessKeySecret = args(5)
    val batchInterval = Milliseconds(args(6).toInt * 1000)
    val zkAddress = if (args.length >= 8) args(7) else "localhost:2181"
    def functionToCreateContext(): StreamingContext = {
      val conf = new SparkConf().setAppName("Test Direct Loghub")
      val ssc = new StreamingContext(conf, batchInterval)
      val zkParas = Map("zookeeper.connect" -> zkAddress,
        "enable.auto.commit" -> "false")
      val loghubStream = LoghubUtils.createDirectStream(
        ssc,
        project,
        logStore,
        consumerGroup,
        accessKeyId,
        accessKeySecret,
        endpoint,
        zkParas,
        LogHubCursorPosition.END_CURSOR)
      loghubStream.checkpoint(batchInterval).foreachRDD(rdd => {
        println(s"count by key: ${rdd.map(s => {
          s.sorted
          (s.length, s)
        }).countByKey().size}")
        loghubStream.asInstanceOf[CanCommitOffsets].commitAsync()
      })
      ssc.checkpoint("hdfs:///tmp/spark/streaming") // set checkpoint directory
      ssc
    }
    val ssc = StreamingContext.getOrCreate("hdfs:///tmp/spark/streaming", functionToCreateContext _
)
    ssc.start()
    ssc.awaitTermination()
  }
}
```

For more information, visit GitHub.

6.9. Use Realtime Compute to consume log data

You can use Realtime Compute (Blink) to create a schema for data in Log Service and consume the data. This topic describes how to use Realtime Compute to create a schema for data in Log Service. This topic also describes the attribute fields and data type mapping that you can configure when you create a schema.

Create a schema for data in Log Service

Log Service stores streaming data. Realtime Compute can use the streaming data as input data. The following example is a sample log entry:

```
__source__:  203.0.113.10
__tag__:__receive_time__:  1562125591
__topic__:  test-topic
a:  1234
b:  0
c:  hello
```

The following example is a DDL statement that is used to create a schema for data in Log Service:

```
create table sls_stream(
  a int,
  b int,
  c varchar
) with (
  type ='sls',
  endPoint ='your endpoint',
  accessId ='your AccessKey ID',
  accessKey ='your AccessKey Secret',
  startTime = '2017-07-05 00:00:00',
  project ='ali-cloud-streamtest',
  logStore ='stream-test',
  consumerGroup ='consumerGroupTest1'
);
```

The following table describes the parameters in the WITH clause.

Parameter	Required	Description
endPoint	Yes	The endpoint of Log Service. For more information, see **Obtain an endpoint** in *Log Service Developer Guide*.
accessId	Yes	The AccessKey ID that is used to access Log Service.
accessKey	Yes	The AccessKey secret that is used to access Log Service.
project	Yes	The name of the Log Service project.
logStore	Yes	The name of the Log Service Logstore.
consumerGroup	No	The name of the consumer group.
startTime	No	The point in time when Realtime Compute starts to consume log data.

Parameter	Required	Description
heartBeatIntervalMills	No	The heartbeat interval of the client that consumes log data. Default value: 10. Unit: seconds.
maxRetryTimes	No	The maximum number of retries to read data. Default value: 5.
batchGetSize	No	The number of log groups that you want to read at a time. Default value: 10. If the version of Blink is 1.4.2 or later, the default value is 100 and the maximum value is 1000. **Note** If the size of a single log entry and the number of log groups in a batch are large, the Java system may frequently recycle the data that is stored in the memory.
columnErrorDebug	No	Specifies whether to enable debugging. If you set the value to true, debugging is enabled and log entries that fail to be parsed are displayed. Default value: false. This value indicates that debugging is not enabled.

Attribute fields

Realtime Compute can extract fields from log data. Realtime Compute can also extract three attribute fields and custom tag fields. The following table describes the three attribute fields.

Attribute field	Description
`__source__`	The source of the log entry.
`__topic__`	The topic of the log entry.
`__timestamp__`	The point in time when the log entry is generated.

To extract the three attribute fields, you must add HEADERs in the DDL statement. Example:

```
create table sls_stream(
  __timestamp__ bigint HEADER,
  __receive_time__ bigint HEADER
  a int,
  b int,
  c varchar
) with (
  type ='sls',
  endPoint ='your endpoint',
  accessId ='your AccessKey ID',
  accessKey ='your AccessKey Secret',
  startTime = '2017-07-05 00:00:00',
  project ='ali-cloud-streamtest',
  logStore ='stream-test',
  consumerGroup ='consumerGroupTest1'
);
```

Data type mapping

The string data type in Log Service is mapped to the varchar data type in Realtime Compute. We recommend that you declare the mapping in a DDL statement. If you specify another data type to convert data in Log Service, Realtime Compute attempts to automatically convert the data. For example, you can specify bigint as the data type to convert the string `1000` and specify timestamp as the data type to convert the string `2018-01-12 12:00:00`.

Usage notes

- Blink 2.2.0 or earlier versions do not support shard scaling. If you split or merge shards when a job is reading data from a Logstore, the job fails and cannot continue. In this case, you must restart the job.
- You cannot delete or recreate a Logstore whose log data is being consumed, regardless of the Blink version.
- In Blink version 1.6.0 and earlier, the read performance may be affected if you specify a consumer group to consume log data from a Logstore that contains a large number of shards.
- You cannot define the map data type in Realtime Compute when you create a schema for data in Log Service.
- Fields that do not exist are set to null.
- Fields can be converted in a random order. However, we recommend that you convert the fields in the same order as the fields in the schema.
- If no new data is written to a shard, the latency of a job increases. In this case, you must change the number of concurrent tasks in the job to the number of shards in which data is read and written.
- To extract fields from tags such as `__tag__:__hostname__` and `__tag__:__path__`, you can delete the __tag__: prefix and follow the method used to extract attribute fields.

  > **Note** You cannot extract this type of data during debugging. We recommend that you use the on-premises debugging method and the print method to display data in logs.

7.Data shipping

7.1. Ship logs to OSS

7.1.1. Overview

Log Service provides the data shipping feature. You can use this feature to ship logs to Object Storage Service (OSS) in real time by using the Log Service console. This topic describes the benefits and scenarios of the data shipping feature.

In the Log Service console, you can ship logs to other Apsara Stack services. Then, you can store or consume the log data by using other systems such as E-MapReduce. After you enable the log shipping feature, Log Service ships the collected logs to the specified cloud service at regular intervals.

Scenarios

The data shipping feature can be used to connect Log Service with data warehouses.

Benefits

The data shipping feature of Log Service has the following benefits:

- Ease of use

  You only need to complete a few settings in the Log Service console before you can ship logs from Logstores to other Apsara Stack services such as OSS.

- High efficiency

  Log Service stores logs that are collected from multiple servers. This improves efficiency when you ship log data to Apsara Stack services such as OSS.

- Effective management

  You can ship logs from different projects or Logstores to different OSS buckets. This way, you can efficiently manage the logs by log type or log source.

Log shipping destinations

For information about how to ship logs to OSS, see Ship log data from Log Service to OSS.

7.1.2. Ship log data from Log Service to OSS

You can use Log Service to collect log data and ship the log data to Object Storage Service (OSS) for storage and analysis. This topic describes how to ship log data from Log Service to OSS.

Prerequisites

- Log data is collected. For more information, see Log collection methods.
- OSS is activated. A bucket is created in the region where the Log Service project resides. For more information, see the **Create buckets** section in the *Service User Guide -Ojbect Storage Service(OSS)*.
- A Resource Access Management (RAM) role is created for the level-1 organization. For more information, see Obtain the ARN of a RAM role.

Context

Log Service can automatically ship log data from a Logstore to an OSS bucket.

- You can set a custom retention period for the log data in the OSS bucket. Permanent retention is supported.
- You can use data processing platforms such as E-MapReduce and Data Lake Analytics (DLA) or use custom programs to consume log data from the OSS bucket.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project from which you want to ship log data to OSS.
3. On the Logstores tab, click the > icon on the left of the specific Logstore and choose **Data Transformation > Export > Object Storage Service(OSS)**.
4. On the OSS Shipper page, click **Enable**.
5. In the **OSS LogShipper** pane, configure the shipping rules.

   The following table describes the required parameters.

Parameter	Description
OSS Shipper Name	The name of the shipping rule. The name can contain only lowercase letters, digits, hyphens (-), and underscores (_). The name must start and end with a lowercase letter or digit and must be 2 to 128 characters in length.
OSS Bucket	The name of the OSS bucket to which you want to ship log data. **Notice** You must specify the name of an existing OSS bucket. The specified OSS bucket must reside in the same region as the Log Service project.
OSS Prefix	The directory to which log data is shipped in the OSS bucket.
Partition format	The partition format of the bucket directory for the shipping task. The directory is automatically generated based on the time when the shipping task is created. The default format is %Y/%m/%d/%H/%M. The partition format cannot start with a forward slash (/). For information about partition format examples, see Partition format. For more information about parameters, see strptime API.
(Resource Access Management) RAM Role	The Alibaba Cloud Resource Name (ARN) of the RAM role. The RAM role is the identity that the OSS bucket owner creates for access control. Example: acs:ram::45643:role/aliyunlogdefaultrole. For information about how to obtain the ARN, see Obtain the ARN of a RAM role.
Shipping Size	The maximum size of raw log data that can be shipped to the OSS bucket in a shipping task. Valid values: 5 to 256. Unit: MB. If the size of shipped data exceeds the specified value, a new shipping task is automatically created.
Compress	Specifies whether to compress log data that is shipped to OSS. Valid values: ○ No Compress: The log data that is shipped to OSS is not compressed. ○ Compress (snappy): The snappy utility is used to compress the log data that is shipped to OSS. This way, the log data occupies less storage space of the OSS bucket.
Storage Format	The storage format of the log data that is shipped to OSS. Valid values: JSON, CSV, and Parquet. For more information, see Storage Formats.
Ship Tags	Specifies whether to ship log tags.

Parameter	Description
Shipping Time	The time period during which a shipping task runs. Valid values: 300 to 900. Default value: 300. Unit: seconds. If the specified time period expires, another shipping task is created.

6. Click **OK**.

> **Note**
> - After you configure a shipping rule, multiple shipping tasks can concurrently run. If the size of the data shipped from a shard reaches the specified threshold or the specified time period expires, another task is created.
> - After you create a shipping task, you can check whether the shipping rule satisfies your business requirements based on the task status and the data shipped to OSS.

View OSS data

After log data is shipped to OSS, you can access the log data in the OSS console, or by using the OSS API, an SDK, or another method. For more information, see the **Objects > Search for objects** section of the *Service User Guide -Ojbect Storage Service(OSS)*.

The following script shows a sample OSS directory:

```
oss://OSS-BUCKET/OSS-PREFIX/PARTITION-FORMAT_RANDOM-ID
```

OSS-BUCKET is the name of the OSS bucket. OSS-PREFIX is the prefix of the directory in the OSS bucket. PARTITION-FORMAT is the partition format of the directory for a shipping task. The partition format is calculated based on the time when the shipping task is created. For more information, see strptime API. RANDOM-ID is the unique identifier of the shipping task.

> **Note** The directory in the OSS bucket is created based on the time when the shipping task is created. For example, the shipping task is created at 00:00:00 on June 23, 2016 to ship data to OSS. The data is written to Log Service after 23:55:00 on June 22, 2016. The shipping interval is 5 minutes. To retrieve all logs shipped on June 22, 2016, you must check all objects in the *2016/06/22* directory. You must also check the *2016/06/23/00/* directory for the objects that are generated in the first 10 minutes of June 23, 2020.

Partition format

For each shipping task, log data is written to a directory of an OSS bucket. The directory is in the *oss://OSS-BUCKET/OSS-PREFIX/PARTITION-FORMAT_RANDOM-ID* format. A partition format is obtained by formatting the time when a shipping task is created. The following table describes the partition formats and directories that are obtained when a shipping task is created at 19:50:43 on January 20, 2017.

OSS Bucket	OSS Prefix	Partition format	OSS directory
test-bucket	test-table	%Y/%m/%d/%H/%M	*oss://test-bucket/test-table/2017/01/20/19/50_1484913043351525351_2850008*
test-bucket	log_ship_oss_example	year=%Y/mon=%m/day=%d/log_%H%M%s	*oss://test-bucket/log_ship_oss_example/year=2017/mon=01/day=20/log_195043_1484913043351525351_2850008.parquet*

OSS Bucket	OSS Prefix	Partition format	OSS directory
test-bucket	log_ship_oss_example	ds=%Y%m%d/%H	*oss://test-bucket/log_ship_oss_example/ds=20170120/19_1484913043351525351_2850008.snappy*
test-bucket	log_ship_oss_example	%Y%m%d/	*oss://test-bucket/log_ship_oss_example/20170120/_1484913043351525351_2850008* **Note** This format may prevent platforms such as Hive from parsing the log data in the OSS bucket. We recommend that you do not use this format.
test-bucket	log_ship_oss_example	%Y%m%d%H	*oss://test-bucket/log_ship_oss_example/2017012019_1484913043351525351_2850008*

You can use Hive, MaxCompute, or Data Lake Analytics (DLA) to analyze OSS data. In this case, if you want to use partition information, you can set PARTITION-FORMAT in the key=value format. For example, you can set the partition format to *oss://test-bucket/log_ship_oss_example/year=2017/mon=01/day=20/log_195043_1484913043351525351_2850008.parquet*. In this example, year, mon, and day are specified as three partition keys.

What to do next

After shipping tasks are created based on a shipping rule, you can modify the shipping rule. You can also disable the data shipping feature, view the statuses and error messages of the tasks, and retry failed tasks on the **OSS Shipper** page of a Logstore.

- Modify the shipping rule.

  Click **Settings** to modify the shipping rule. For information about the parameters, see Procedure.

- Disable the data shipping feature.

  Click **Disable**. The data in the Logstore is no longer shipped to OSS.

- View the statuses and error messages of the tasks.

  You can view the log shipping tasks of the last two days and their statuses.

  - Statuses of a shipping task

Status	Equivalent
Succeeded	The shipping task has succeeded.
Running	The shipping task is running. Check whether the task succeeds later.
Failed	The shipping task has failed. If the task cannot be restarted due to external causes, troubleshoot the failure based on the error message and retry the task.

- Error messages

  If a shipping task fails, an error message is returned for the task.

Error message	Error cause	Solution
UnAuthorized	The error message returned because the AliyunLogDefaultRole role does not have the required permissions.	To fix the error, check the following configurations: ■ Check whether the AliyunLogDefaultRole role is created by the OSS bucket owner. ■ Check whether the specified ID of the Alibaba Cloud account in the permission policy is valid. ■ Check whether the AliyunLogDefaultRole role is granted the write permissions on the OSS bucket. ■ Check whether the ARN of the AliyunLogDefaultRole role that you entered in the RAM Role field is valid.
ConfigNotExist	The error message returned because the task does not exist.	Check whether the data shipping feature is disabled. If the feature is disabled, enable the feature, configure a shipping rule, and then retry the shipping task.
InvalidOssBucket	The error message returned because the specified OSS bucket does not exist.	To fix the error, check the following configurations: ■ Check whether the OSS bucket resides in the same region as the Log Service project. ■ Check whether the specified bucket name is valid.
InternalServerError	The error message returned because an internal error has occurred in Log Service.	Retry the failed shipping task.

- Retry a shipping task

  By default, if a shipping task fails, Log Service retries the task based on the retry policy. You can also manually retry the task. By default, Log Service retries all tasks of the last two days. The minimum interval between two consecutive retries is 15 minutes. If a task fails for the first time, Log Service retries the task 15 minutes later. If the task fails for the second time, Log Service retries the task 30 minutes later. If the task fails for the third time, Log Service retries the task 60 minutes later. A similar method is used for subsequent attempts.

  To immediately retry a failed task, you can click **Retry All Failed Tasks** or **Retry** on the right of the task. You can also use the Log Service API or an SDK to retry a task.

7.1.3. Obtain the ARN of a RAM role

When you use a RAM user to ship data from Log Service to Object Storage Service (OSS), you must first create a Resource Access Management (RAM) role and specify the ARN of the RAM role. This topic describes how to create a RAM role and obtain the ARN of a RAM role.

Procedure

1. Log on to the Log Service console.
2. In the top navigation bar, click **Configurations**.

3. On the **Service-Linked Roles** page, click **Create RAM Role**.
4. In the **Organization Name** drop-down list, select the organization that you created. In the **Service Name** drop-down list, select **Log Service**, and click **OK**.
5. On the **RAM Service Role** page, enter `AliyunLogDefaultRole` in the **Role Name** search box and click **Search**.
6. Obtain the ARN of the RAM role.

   In the search results, the value in the **role identifier** column is the ARN of the RAM role.

7.1.4. Storage Formats

Different storage formats are supported when Log Service ships logs to OSS, including JSON, CSV, and Parquet. This topic describes the field details of the formats.

JSON format

You can set the storage format for the data that is shipped to OSS. The following table shows how to set the **storage format** to **JSON**. For more information, see Configure a data shipping rule.

Compression type	File extension	Example file address	Description

Uncompressed	N/A	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937	You can download the raw JSON object to the local host and open each object as a text file. The following example is the content of a sample file:
			`{"__time__":1453809242,"__topic__":"","__source__":"10.170.***.***","ip":"10.200.**.***","time":"26/Jan/2016:19:54:02 +0800","url":"POST /PutData?Category=YunOsAccountOpLog&AccessKeyId=<yourAccessKeyId>&Date=Fri%2C%2028%20Jun%202013%2006%3A53%3A30%20GMT&Topic=raw&Signature=<yourSignature>`
			
			`HTTP/1.1","status":"200","user-agent":"aliyun-sdk-java"}`

snappy	.snappy	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy	JSON objects are compressed by using Snappy. For more information, see Decompression tools for Snappy-compressed files.

CSV-format

You can set the storage format for the data that is shipped to OSS. The following table shows how to set the **storage format** to **CSV**. For more information, see Configure a data shipping rule.

The following table describes the parameters. For more information, see Common Format and MIME Type for Comma-Separated Values (CSV) Files and PostgreSQL 9.4.26 Documentation.

Parameter	Description
CSV Fields	The names of the log fields that you want to ship to OSS. You can view log fields on the **Raw Logs** tab of a Logstore and enter the names of the fields that you want to ship to OSS in the Key Name column. The log fields that you can ship to OSS include the fields in the log content and the reserved fields such as __time__, __topic__, and __source__. **Note** The keys that you enter in the **CSV Fields** section must be unique.
Delimiter	You can use commas (,), vertical bars (|), spaces, or tabs to delimit fields.
Escape Character	If a field contains a delimiter, you must use an escape character to enclose the field. This ensures that the field is not delimited.
Invalid Fields	If a key that you specify in the **CSV Fields** section does not exist, enter the value of the key in the Invalid Fields field.
Shipped Fields	If you turn on the **Shipped Fields** switch, field names are written in a CSV file.

The following table lists the directories in OSS buckets that store the data shipped from Log Service.

Compression type	File extension	Example	Description
No	.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.csv	You can download the raw JSON object to the local host and open the object as a text file.
snappy	.snappy.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.csv	Decompression tools for Snappy compressed files For more information, see Decompression tools for Snappy-compressed files.

Parquet-format

You can set the storage format for the data that is shipped to OSS. The following figure shows how to set the **storage format** to **Parquet**. For more information, see Configure a data shipping rule.

The following table describes the related parameters.

Parameter	Description

Parameter	Description
Key Name	The name of the log field that you want to ship to OSS. You can view log fields on the **Raw Logs** tab of a Logstore. You can also enter the names of the fields that you want to ship to OSS in the Key Name column. When the fields are shipped to OSS, they are stored in the Parquet format in the order that the field names are entered. The names of the fields are the column names in OSS. The log fields that you can ship to OSS include the fields in the log content and the reserved fields such as __time__, _topic__, and __source__. The value of a field in the Parquet format is null in the following two scenarios: • The field does not exist in logs. • The value of the field fails to be converted from the string type to a non-string type, for example, double or Int64. **Note** The keys that you enter in the **Parquet Keys** field must be unique.
Type	The Parquet storage format supports six data types: string, Boolean, Int32, Int64, float, and double. Log fields are converted from the string type to a data type that the Parquet storage format supports. If the data type of a log field fails to be converted, the value of the log field is null.

The following table lists the directories in OSS buckets that store data shipped from Log Service.

Compression type	File extension	Example	Description
Uncompressed	.parquet	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.parquet	You can download the OSS buckets to the local host and use the parquet-tools utility to open the objects. For more information about the parquet-tools utility, visit parquet-tools.
Snappy	.snappy.parquet	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.parquet	You can download the OSS buckets to the local host and use the parquet-tools utility to open the objects. For more information about the parquet-tools utility, visit parquet-tools.

7.1.5. Decompress Snappy compressed files

When you ship data from Log Service to Object Storage Service (OSS), you can use Snappy to compress OSS objects. After the data is shipped to OSS, you can decompress OSS objects by using the C++ library, Java library, Python library, and decompression tool for Linux.

Use the C++ library to decompress OSS objects

Download the C++ library from the snappy page and use the Snappy.Uncompress method to decompress Snappy compressed OSS objects.

Use the Java library to decompress OSS objects

Download the Java library from the xerial snappy-java page and use the Snappy.Uncompress or Snappy.SnappyInputStream method to decompress Snappy compressed OSS objects. The SnappyFramedInputStream method is not supported.

> **Note** If you use Java Library 1.1.2.1, some Snappy compressed OSS objects may fail to be decompressed. For more information, see Bad handling of the MAGIC HEADER. To fix this issue, you can use Java Library 1.1.2.6 or later.

```
<dependency>
<groupId>org.xerial.snappy</groupId>
<artifactId>snappy-java</artifactId>
<version>1.0.4.1</version>
<type>jar</type>
<scope>compile</scope>
</dependency>
```

- Snappy.Uncompress

```
String fileName = "C:\\Downloads\\36_1474212963188600684_4451886.snappy";
RandomAccessFile randomFile = new RandomAccessFile(fileName, "r");
int fileLength = (int) randomFile.length();
randomFile.seek(0);
byte[] bytes = new byte[fileLength];
int byteread = randomFile.read(bytes);
System.out.println(fileLength);
System.out.println(byteread);
byte[] uncompressed = Snappy.uncompress(bytes);
String result = new String(uncompressed, "UTF-8");
System.out.println(result);
```

- Snappy.SnappyInputStream

```
String fileName = "C:\\Downloads\\36_1474212963188600684_4451886.snappy";
SnappyInputStream sis = new SnappyInputStream(new FileInputStream(fileName));
byte[] buffer = new byte[4096];
int len = 0;
while ((len = sis.read(buffer)) != -1) {
    System.out.println(new String(buffer, 0, len));
}
```

Use the Python Library to decompress OSS objects

1. Download and install the Python library.
2. Run the decompression script.

   The following example is a sample decompression script:

```
import snappy
compressed = open('/tmp/temp.snappy').read()
snappy.uncompress(compressed)
```

> **Note** The following two commands cannot be used to decompress Snappy compressed OSS objects. These commands can be used only in Hadoop mode (hadoop_stream_decompress) or streaming mode (stream_decompress).

```
python -m snappy -c uncompressed_file compressed_file.snappy
python -m snappy -d compressed_file.snappy uncompressed_file
```

Use decompression tools for Linux to decompress OSS buckets

Log Service allows you to decompress Snappy compressed files by using the decompression tool for Linux. Click snappy_tool to download the tool. Replace 03_145345700654807 8722_44148.snappy and 03_1453457006548078722_44148 in the following code with the values specific to your environment and then run the following code:

```
./snappy_tool 03_1453457006548078722_44148.snappy 03_1453457006548078722_44148
compressed.size: 2217186
snappy::Uncompress return: 1
uncompressed.size: 25223660
```

8.Time series storage

8.1. Data import

8.1.1. Collect metric data from hosts

Log Service allows you to collect metric data from hosts by using Logtail. The metric data includes CPU, memory, load, disk, and network data. This topic describes how to create a Logtail configuration in the Log Service console to collect metric data from hosts.

Prerequisites

Logtail V0.16.40 or later is installed on a Linux server. For more information, see Install Logtail in Linux.

Limits

- Windows servers are not supported.
- Metric data related to GPUs and hardware status cannot be collected.

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, click **Host Monitoring Data**.
3. In the **Specify Logstore** step, select the project and the Metricstore that you want to use. Then, click **Next**.

   You can also click **Create Now** to create a project or a Metricstore. For more information, see Create a project and Create a Metricstore.
4. In the **Create Machine Group** step, create a machine group.
   - If a machine group is available, click **Use Existing Machine Groups**.
   - If no machine groups are available, perform the following steps to create a machine group. In this example, an Elastic Compute Service (ECS) instance is used.
     a. Select the ECS instance on which you want to install Logtail. For more information, see Install Logtail in Linux.

        If Logtail is installed on the ECS instance, click **Complete Installation**.

        > **Note** If you want to collect logs from self-managed clusters or servers that are provided by third-party cloud service providers, you must manually install Logtail. For more information, see Install Logtail in Linux.

     b. After Logtail is installed, click **Complete Installation**.
     c. Create a machine group.

        For more information about how to create a machine group, see Create a machine group based on a server IP address or Create a custom ID-based machine group.
5. Select the machine group from the **Source Server Groups** section and move the machine group to the **Applied Server Groups** section. Then, click **Next**.

   > **Notice** If you enable a machine group immediately after you create the machine group, the heartbeat status of the machine group may be **FAIL**. This issue occurs because the machine group is not connected to Log Service. To resolve this issue, you can click **Automatic Retry**. If the issue persists, troubleshoot the issue. For more information, see What can I do if Log Service does not receive heartbeats from a Logtail client?.
6. In the **Specify Data Source** step, configure the **Config Name** and **Plug-in Config** parameters.

inputs: specifies the collection configurations. This parameter is required. Configure the inputs parameter based on your data source.

> **Note** You can specify only one type of data source in the inputs parameter.

```
{
    "inputs": [
        {
            "detail": {
                "IntervalMs": 30000
            },
            "type": "metric_system_v2"
        }
    ]
}
```

Parameter	Type	Required	Description
type	string	Yes	The type of the data source. Set the value to metric_system_v2.
IntervalMs	int	Yes	The interval between two consecutive requests. Unit: milliseconds. The value must be greater than or equal to 5000. We recommend that you set the value to 30000.

7. Click **Next**.

Metrics

The following tables describe metrics that are related to CPUs, memory, loads, disks, and networks.

- CPU-related metrics

Metric	Description	Unit	Example
cpu_count	The number of CPU cores.	N/A	2.0
cpu_util	The CPU utilization. The CPU utilization is equal to one minus the sum of the idle, wait, and steal counters.	Percent (%)	7.68
cpu_guest_util	The guest counter of Linux. This counter indicates the percentage of the time that is spent by the CPU on processes with normal priority.	Percent (%)	0.0
cpu_guestnice_util	The guest_nice counter of Linux. This counter indicates the percentage of the time that is spent by the CPU on processes with nice priority.	Percent (%)	0.0

Metric	Description	Unit	Example
cpu_irq_util	The irq counter of Linux. This counter indicates the percentage of the time that is spent by the CPU to serve hardware interrupt requests.	Percent (%)	0.0
cpu_nice_util	The nice counter of Linux. This counter indicates the percentage of the time that is spent by the CPU on user-mode processes with nice priority.	Percent (%)	0.0
cpu_softirq_util	The softirq counter of Linux. This counter indicates the percentage of the time that is spent by the CPU to serve software interrupt requests.	Percent (%)	0.06
cpu_steal_util	The steal counter of Linux. This counter indicates the percentage of the time that is spent by the CPU to run other operating systems in a virtual environment.	Percent (%)	0.0
cpu_sys_util	The system counter of Linux. This counter indicates the percentage of the time that the CPU spends on kernel-mode processes.	Percent (%)	2.77
cpu_user_util	The user counter of Linux. This counter indicates the percentage of the time that is spent by the CPU on user-mode processes with normal priority.	Percent (%)	4.84
cpu_wait_util	The iowait counter of Linux. This counter indicates the percentage of the CPU idle time when outstanding disk I/O requests exist.	Percent (%)	0.11

- Memory-related metrics

Metric	Description	Unit	Example
mem_util	The memory usage.	Percent (%)	51.03
mem_cache	The amount of memory that is allocated but unused.	byte	3566386668.0
mem_free	The amount of the unused memory.	byte	177350084.0
mem_available	The amount of the available memory.	byte	3699885553.0

Metric	Description	Unit	Example
mem_used	The amount of the used memory.	byte	4041510463.0
mem_swap_util	The swap memory usage.	Percent (%)	0.0
mem_total	The size of the memory.	byte	7919128576.0

- Disk-related metrics

Metric	Description	Unit	Example
disk_rbps	The amount of data that is read from the disk per second.	byte/s	8376.81
disk_wbps	The amount of data that is written to the disk per second.	byte/s	247633.58
disk_riops	The number of read operations that are completed on the disk per second.	N/A	0.22
disk_wiops	The number of write operations that are completed on the disk per second.	N/A	43.39
disk_rlatency	The average read latency.	ms	2.83
disk_wlatency	The average write latency.	ms	2.15
disk_util	The I/O usage of the disk.	Percent (%)	0.27
disk_space_usage	The percentage of the used disk space.	Percent (%)	9.12
disk_inode_usage	The percentage of the used index node (inode) space.	Percent (%)	1.18
disk_space_used	The amount of the used disk space.	byte	11068512238.59
disk_space_total	The total amount of the disk space.	byte	126692061184.0
disk_inode_total	The total amount of the inode space.	byte	7864320.0
disk_inode_used	The amount of the used inode space.	byte	93054.78

- Network-related metrics

Metric	Description	Unit	Example
net_drop_util	The percentage of the number of discarded packets to the total number of packets.	Percent (%)	0.0

Metric	Description	Unit	Example
net_err_util	The percentage of the number of error packets to the total number of packets.	Percent (%)	0.0
net_in	The amount of data that is received per second.	byte/s	8440.91
net_in_pkt	The number of packets that are received per second.	N/A	40.83
net_out	The amount of data that is sent per second.	byte/s	12446.53
net_out_pkt	The number of packets that are sent per second.	N/A	39.95

- TCP-related metrics

Metric	Description	Unit	Example
protocol_tcp_established	The number of established connections.	N/A	205.0
protocol_tcp_insegs	The number of received packets.	N/A	4654.0
protocol_tcp_outsegs	The number of sent packets.	N/A	4870.0
protocol_tcp_retran_segs	The number of re-sent packets.	N/A	0.0
protocol_tcp_retran_util	The percentage of the number of re-sent packets to the number of sent packets.	Percent (%)	0.0

- System-related metrics

Metric	Description	Unit	Example
system_boot_time	The startup time of the system.	s	1578461935.0
system_load1	The average system load every minute.	N/A	0.58
system_load5	The average system load every 5 minutes.	N/A	0.68
system_load15	The average system load every 15 minutes.	N/A	0.60

What's next

- Query and analyze data

  After metric data is collected, you can query and analyze the data on the Query & Analysis page of the Metricstore. For more information, see Query and analyze time series data.

- Visualize query and analysis results in Log Service

You can create a time series chart and add the chart to a dashboard to visualize the query and analysis results. You can also configure alert rules for the chart.

8.1.2. Import metrics collected by Telegraf

8.1.2.1. Telegraf overview

Telegraf is an agent developed by InfluxData to collect metric data. Telegraf supports multiple input plug-ins and output plug-ins, such as MySQL, Redis, and Elasticsearch. This topic describes how Telegraf works, how to install Telegraf, and how to use Telegraf to collect metric data.

Implementation

You can use Telegraf to collect metric data from MySQL, Redis, and Elasticsearch. After you collect metric data, you can use the InfluxDB line protocol to write the collected data to Logtail. Then, Logtail uploads the metric data to a Metricstore of Log Service. Log Service allows you to configure plug-ins and create dashboards in the console for the collected data. The following figure shows how Telegraf works.

Collection methods

Telegraf provides the following collection methods:

- Local collection

  You can use Telegraf to collect metric data from a local server by using the local collection method. The server that is specified in your machine group is the server from which metric data is collected. When you create a Logtail configuration file, you can set the IP address of the server to 127.0.0.1. We recommend that you use this collection method.

- Remote collection

  You can use Telegraf that is installed on a server to collect metric data from other servers by using the remote collection method. When you create a Logtail configuration file, you can set the IP address to the actual IP address or the actual endpoint of the server. If you use the remote collection method, you can configure only one server in the machine group. If you configure more than one servers, duplicate data is generated. You can use the remote collection method in the following scenarios:

  - You want to collect metric data of a cloud service where you cannot install Logtail and Telegraf.
  - You do not want to install a metric collection agent on a running server.

8.1.2.2. Collect metric data from MySQL servers

You can use Telegraf to collect metric data from MySQL servers, and then use Logtail to send the metric data to a Metricstore in Log Service. This way, you can monitor metric data of MySQL servers in a visualized manner. This topic describes how to collect metric data from a MySQL server by using Log Service and visualize the data.

Prerequisites

- Logtail V0.16.48 or later is installed on a Linux server. For more information, see Install Logtail in Linux.
- Telegraf is installed on a server that is connected to the MySQL server over a private network.

Limit

Only MySQL 5.5 or later is supported.

Procedure

1. Log on to the Log Service console.
2. In the **Import Data** section, click **MySQL Monitoring Data**.
3. In the **Specify Logstore** step, select the project and the Metricstore that you want to use. Then, click **Next**.

You can also click **Create Now** to create a project or a Metricstore. For more information, see Create a project and Create a Metricstore.

4. In the **Create Machine Group** step, create a machine group.
   - If a machine group is available, click **Use Existing Machine Groups**.
   - If no machine groups are available, perform the following steps to create a machine group. In this example, an Elastic Compute Service (ECS) instance is used.
     a. Select the ECS instance on which you want to install Logtail. For more information, see Install Logtail in Linux.

        If Logtail is installed on the ECS instance, click **Complete Installation**.

        > **Note** If you want to collect logs from self-managed clusters or servers that are provided by third-party cloud service providers, you must manually install Logtail. For more information, see Install Logtail in Linux.

     b. After Logtail is installed, click **Complete Installation**.
     c. Create a machine group.

        For more information about how to create a machine group, see Create a machine group based on a server IP address or Create a custom ID-based machine group.
5. Select the machine group from the **Source Server Groups** section and move the machine group to the **Applied Server Groups** section. Then, click **Next**.

   > **Notice** If you enable a machine group immediately after you create the machine group, the heartbeat status of the machine group may be **FAIL**. This issue occurs because the machine group is not connected to Log Service. To resolve this issue, you can click **Automatic Retry**. If the issue persists, troubleshoot the issue. For more information, see What can I do if Log Service does not receive heartbeats from a Logtail client?.
6. In the **Specify Data Source** step, configure the parameters. The following table describes the parameters.

Parameter	Description
Configuration Name	Enter a name for the Logtail configuration.
Cluster Name	Enter the name of the MySQL cluster. After you configure this parameter, Log Service adds a *cluster=<Cluster name>* tag to the data that is collected. **Note** Make sure that the cluster name is unique. Otherwise, data conflicts may occur.

Parameter	Description
Server List	Click the + icon to add the MySQL server and configure the following parameters: ○ **Account**: the username of the account that is used to log on to the MySQL server. **Note** We recommend that you create a dedicated account to monitor the data of the MySQL server and grant the account only the permissions that are required to monitor data. ○ **Password**: the password of the account. ○ **Address**: the endpoint of the MySQL server. The endpoint can be the IP address, hostname, or domain name of the server. ○ **Port**: the port number of the MySQL server. The default value is 3306. You can add multiple MySQL servers based on your business requirements.
Custom Tags	Create a custom tag in **Custom Tags**. The created tag is added to the data that is collected based on the Logtail configuration. You can use custom tags to identify the data that is collected based on different Logtail configurations in a Metricstore. You can click the + icon to create a custom tag. You can create multiple custom tags. The custom tags are added to all data that is collected based on the Logtail configuration.

FAQ

How do I check whether Telegraf is collecting data as expected?

You can check the logs of the */etc/ilogtail/telegraf/telegraf.log* file on your server. You can use Log Service to collect this log file and search for the required information in Log Service.

What's next

- Query and analyze data

  After you configure the settings, Telegraf uses Logtail to upload collected metric data to the specified Metricstore in Log Service. You can query and analyze the data on the Query & Analysis page of the Metricstore. For more information, see Query and analyze time series data.

- Visualize query and analysis results

  You can create a time series chart and add the chart to a dashboard to view the query and analysis results. You can also configure alert rules.

8.1.2.3. Collect metric data from Java applications or Tomcat servers

You can use Telegraf to collect metric data from Java applications or Tomcat servers. Then, you can use Logtail to send the metric data to a Metricstore in Log Service. This way, you can monitor the metric data of Java applications and Tomcat servers in a visualized manner. This topic describes how to collect metric data from Java applications by using Log Service and visualize the data.

Prerequisites

- Logtail V0.16.48 or later is installed on a Linux server. For more information, see Install Logtail in Linux.
- Java 1.6 or later is installed on the server.

Step 1: Create a Logtail configuration

1. Log on to the Log Service console.
2. In the **Import Data** section, click **Java Application Monitoring Data**.

   If you want to collect metric data from Tomcat servers, click **Tomcat Monitoring Data**.
3. In the **Specify Logstore** step, select the project and the Metricstore that you want to use. Then, click **Next**.

   You can also click **Create Now** to create a project or a Metricstore. For more information, see Create a project and Create a Metricstore.
4. In the **Create Machine Group** step, create a machine group.
   - If a machine group is available, click **Use Existing Machine Groups**.
   - If no machine groups are available, perform the following steps to create a machine group. In this example, an Elastic Compute Service (ECS) instance is used.
     a. Select the ECS instance on which you want to install Logtail. For more information, see Install Logtail in Linux.

        If Logtail is installed on the ECS instance, click **Complete Installation**.

        > **Note** If you want to collect logs from self-managed clusters or servers that are provided by third-party cloud service providers, you must manually install Logtail. For more information, see Install Logtail in Linux.

     b. After Logtail is installed, click **Complete Installation**.

     c. Create a machine group.

        For more information about how to create a machine group, see Create a machine group based on a server IP address or Create a custom ID-based machine group.
5. Select the machine group from the **Source Server Groups** section and move the machine group to the **Applied Server Groups** section. Then, click **Next**.

   > **Notice** If you enable a machine group immediately after you create the machine group, the heartbeat status of the machine group may be **FAIL**. This issue occurs because the machine group is not connected to Log Service. To resolve this issue, you can click **Automatic Retry**. If the issue persists, troubleshoot the issue. For more information, see What can I do if Log Service does not receive heartbeats from a Logtail client?.
6. In the **Specify Data Source** step, configure the parameters. The following table describes the parameters.

Parameter	Description
Configuration Name	Enter a name for the Logtail configuration.
Application Name	Enter the name of your Java application. After you configure this parameter, Log Service adds a *cluster=<Application name>* tag to the data that is collected. **Note** Make sure that the name of the application is unique. Otherwise, data conflicts may occur.

Parameter	Description
Server List	Click the + icon to add a server on which your application resides. ○ **Address**: the address of the server. ○ **Port**: the port number of the server. You can specify a custom port number. The port number must be the same as the port number that you specify in Step 2: Configure JavaAgent. You can add multiple servers based on your business requirements.
Custom Tags	Create a custom tag in **Custom Tags**. The created tag is added to the data that is collected based on the Logtail configuration. You can use custom tags to identify the data that is collected based on different Logtail configurations in a Metricstore. You can click the + icon to create a custom tag. You can create multiple custom tags. The custom tags are added to all data that is collected based on the Logtail configuration.

Step 2: Configure JavaAgent

After the Logtail configuration is created, you must enable access to JMX data over HTTP. Log Service allows you to use Jolokia to access JMX data over HTTP. You can download and load Jolokia based on the official documentation of Jolokia. You can also use Jolokia JavaAgent that is provided in Log Service together with Logtail. Jolokia JavaAgent is stored in */etc/ilogtail/telegraf/javaagent/jolokia-jvm.jar*.

- If you want to collect metric data from Java applications, you must add the script `-javaagent:/etc/ilogtail/telegraf/javaagent/jolokia-jvm.jar=port=7777` to the Java startup parameters.
- If you want to collect metric data from Tomcat servers, you must configure the `JAVA_OPTS` environment variable. For example, specify `export JAVA_OPTS="-javaagent:/etc/ilogtail/telegraf/jolokia-jvm.jar=port=7777"`. In this example, 7777 indicates the port number of the application server. This port number must be the same as the port number that you specify in Step 1: Create a Logtail configuration.

> **Note** By default, Jolokia JavaAgent listens only on the IP address 127.0.0.1 and allows requests only from the local host. If Logtail and your Java application are installed on different servers, you can add the host= field to the added script. This way, Jolokia JavaAgent can listen on other IP addresses. If you add **host=0.0.0.0**, Jolokia JavaAgent listens on all IP addresses. Example:
>
> ```
> -javaagent:/tmp/jolokia-jvm.jar=port=7777,host=0.0.0.0
> ```

After you configure the settings, you must restart your Java application. If your Java application fails to restart, run the following command to connect Jolokia JavaAgent to a specified Java process. This way, the configuration immediately takes effect. Replace PID with the actual value.

> **Note** This command is used only for testing. You must complete the settings based on the preceding steps. Otherwise, the configuration becomes invalid after your application restarts.

```
java -jar /etc/ilogtail/telegraf/javaagent/jolokia-jvm.jar --port 7777 start PID
```

If the following output is returned, the connection is successful.

```
Jolokia is already attached to PID 752
http://127.0.0.1:7777/jolokia/
```

After the connection is established, you can access the following URL to verify the connection:

```
curl http://127.0.0.1:7777/jolokia/
# Sample response
{"request":{"type":"version"},"value":{"agent":"1.6.2","protocol":"7.2","config":{"listenForHttpServic
e":"true","maxCollectionSize":"0","authIgnoreCerts":"false","agentId":"30.43.124.186-752-5b091b5d-jvm"
,"debug":"false","agentType":"jvm","policyLocation":"classpath:\/jolokia-access.xml","agentContext":"\
/jolokia","serializeException":"false","mimeType":"text\/plain","maxDepth":"15","authMode":"basic","au
thMatch":"any","discoveryEnabled":"true","streaming":"true","canonicalNaming":"true","historyMaxEntrie
s":"10","allowErrorDetails":"true","allowDnsReverseLookup":"true","realm":"jolokia","includeStackTrace
":"true","maxObjects":"0","useRestrictorService":"false","debugMaxEntries":"100"},"info":{"product":"t
omcat","vendor":"Apache","version":"8.5.57"}},"timestamp":1602663330,"status":200}
```

FAQ

How do I check whether Telegraf is collecting data as expected?

You can check the logs of the */etc/ilogtail/telegraf/telegraf.log* file on your server. You can use Log Service to collect this log file and search for the required information in Log Service.

What's next

- Query and analyze data

  After you configure the settings, Telegraf uses Logtail to upload collected metric data to the specified Metricstore in Log Service. You can query and analyze the data on the Query & Analysis page of the Metricstore. For more information, see Query and analyze time series data.

- Visualize query and analysis results

  You can create a time series chart and add the chart to a dashboard to view the query and analysis results. You can also configure alert rules.

8.1.2.4. Collect metric data from NGINX servers

NGINX provides a built-in status page that allows you to monitor the status of NGINX metric data. You can use Telegraf to collect metric data from NGINX servers, and then use Logtail to send the metric data to a Metricstore in Log Service. This way, you can monitor metric data of NGINX servers in a visualized manner. This topic describes how to collect metric data from a NGINX server by using Log Service and visualize the data.

Prerequisites

Logtail V0.16.50 or later is installed on a Linux server. For more information, see Install Logtail in Linux.

Step 1: Configure the NGINX status module

1. Run the following command to check whether NGINX has the status module. For more information, see Module ngx_http_stub_status_module.

   ```
   nginx -V 2>&1 | grep -o with-http_stub_status_module
   with-http_stub_status_module
   ```

   If the message **with-http_stub_status_module** is returned, NGINX has the status module.

2. Configure the NGINX status module.

   Configure the status module in the NGINX configuration file. By default, this file is stored in the /etc/nginx/nginx.conf directory. Use the following sample code to configure the status module. For more information, see NGINX status.

```
location /private/nginx_status {
  stub_status on;
  access_log   off;
  allow 192.0.2.1;
  deny all;
}
```

- /private/nginx_status indicates the URI of the NGINX status module. Replace the value with the actual URI.
- allow 192.0.2.1 indicates that only the IP address 192.0.2.1 is allowed to access the NGINX status module. Replace the value with the actual IP address.

3. Run the following command to check whether the server on which Logtail is installed can access the NGINX status module:

```
$curl http://192.0.2.1/private/nginx_status
```

If the following message is returned, the NGINX status module is configured:

```
Active connections: 1
server accepts handled requests
2507455 2507455 2512972
Reading: 0 Writing: 1 Waiting: 0
```

Step 2: Import data

1. Log on to the Log Service console.
2. In the **Import Data** section, click **NGINX Monitoring**.
3. In the **Specify Logstore** step, select the project and the Metricstore that you want to use. Then, click **Next**.

   You can also click **Create Now** to create a project or a Metricstore. For more information, see Create a project and Create a Metricstore.
4. In the **Create Machine Group** step, create a machine group.
   - If a machine group is available, click **Use Existing Machine Groups**.
   - If no machine groups are available, perform the following steps to create a machine group. In this example, an Elastic Compute Service (ECS) instance is used.
     a. Select the ECS instance on which you want to install Logtail. For more information, see Install Logtail in Linux.

        If Logtail is installed on the ECS instance, click **Complete Installation**.

        > **Note** If you want to collect logs from self-managed clusters or servers that are provided by third-party cloud service providers, you must manually install Logtail. For more information, see Install Logtail in Linux.

     b. After Logtail is installed, click **Complete Installation**.

     c. Create a machine group.

        For more information about how to create a machine group, see Create a machine group based on a server IP address or Create a custom ID-based machine group.
5. Select the machine group from the **Source Server Groups** section and move the machine group to the **Applied Server Groups** section. Then, click **Next**.

> **Notice** If you enable a machine group immediately after you create the machine group, the heartbeat status of the machine group may be **FAIL**. This issue occurs because the machine group is not connected to Log Service. To resolve this issue, you can click **Automatic Retry**. If the issue persists, troubleshoot the issue. For more information, see What can I do if Log Service does not receive heartbeats from a Logtail client?.

6. In the **Specify Data Source** step, configure the parameters and click **Next**. The following table describes the parameters.

Parameter	Description
Configuration Name	Enter a name for the Logtail configuration.
Cluster Name	Enter the name of the cluster. After you configure this parameter, Log Service adds a *cluster=<Cluster name>* tag to the data that is collected. **Note** Make sure that the cluster name is unique. Otherwise, data conflicts may occur.
Server List	Click the **+** icon to add the NGINX server and configure the following parameters: ○ **Address**: the endpoint of the NGINX server. ○ **Port**: the port number of the NGINX server. ○ **Path**: the URI of the NGINX status module. Example: */private/nginx_status*. For information about how to configure the NGINX status module, see Step 1: Configure the NGINX status module. You can add multiple NGINX servers based on your business requirements.
Custom Tags	Create a custom tag in **Custom Tags**. The created tag is added to the data that is collected based on the Logtail configuration. You can use custom tags to identify the data that is collected based on different Logtail configurations in a Metricstore. You can click the **+** icon to create a custom tag. You can create multiple custom tags. The custom tags are added to all data that is collected based on the Logtail configuration.

FAQ

How do I check whether Telegraf is collecting data as expected?

You can check the logs of the */etc/ilogtail/telegraf/telegraf.log* file on your server. You can use Log Service to collect this log file and search for the required information in Log Service.

What's next

- Query and analyze data

  After you configure the settings, Telegraf uses Logtail to upload collected metric data to the specified Metricstore in Log Service. You can query and analyze the data on the Query & Analysis page of the Metricstore. For more information, see Query and analyze time series data.

- Visualize query and analysis results

  You can create a time series chart and add the chart to a dashboard to view the query and analysis results. You can also configure alert rules.

8.1.3. Collect metric data from Prometheus

8.1.3.1. Collect metric data from Prometheus by using the Remote Write Protocol

Prometheus is a cloud native application that you can use to collect and monitor metric data of various software and systems. This topic describes how to import metric data from Prometheus to Log Service. This topic also describes how to use Log Service to analyze and monitor the data.

Prerequisites

- A Metricstore is created. For more information, see Create a Metricstore.
- Prometheus is installed. For more information, see GETTING STARTED.
- Data collection rules are configured in Prometheus. For more information, see scrape_config.

Procedure

Log Service supports the Remote Write Protocol. You can use the remote write feature of Prometheus to import metric data to Log Service. Before you can use the remote write feature, you must perform the following steps to enable the feature in Prometheus:

1. Log on to the server on which Prometheus is installed.
2. Open the configuration file and configure the parameters based on your business scenario. The following table describes the parameters. For more information, see remote_write.

```
url: https://{project}.{sls-endpoint}/prometheus/{project}/{metricstore}/api/v1/write
basic_auth:
  username: access-key-id
  password: access-key-secret
queue_config:
  batch_send_deadline: 20s
  capacity: 20480
  max_backoff: 5s
  max_samples_per_send: 2048
  min_backoff: 100ms
  min_shards: 100
```

Parameter	Description
url	The URL of the Metricstore in Log Service. The URL must be in the following format: https://*{project}*.*{sls-endpoint}*/prometheus/*{project}*/*{metricstore}*/api/v1/write. Take note of the following variables: ○ *{sls-endpoint}*: the Log Service endpoint. For more information, see *Obtain an endpoint* in Log Service Developer Guide. ○ *{project}*: the project that you created. ○ *{metricstore}*: the Metricstore that you created. **Notice** ○ If you use an Alibaba Cloud internal network, we recommend that you use an internal Log Service endpoint. ○ To ensure secure transmission, you must use HTTPS.

Parameter	Description
basic_auth	The authentication information. If data is written to Log Service over the Remote Write Protocol, basic authentication is required. Take note of the following parameters: ◦ username: the AccessKey ID of your Apsara Stack tenant account. ◦ password: the AccessKey secret of your Apsara Stack tenant account. We recommend that you use the AccessKey pair of a RAM user that is granted only the write permissions on the Log Service project.

queue_config	queue_config specifies the cache and retry policies for data writes.
	To reduce invalid network requests, set min_backoff to a value that is greater than or equal to 100ms and set max_backoff to a value that is greater than or equal to 5s.
	If you want to collect a large amount of data from Prometheus, use the following settings for queue_config:
	`batch_send_deadline: 20s`
	`capacity: 20480`
	`max_backoff: 5s`
	`max_samples_per_send: 2048`
	`min_backoff: 100ms`
	`min_shards: 100`

3. Check whether data is imported to Log Service.

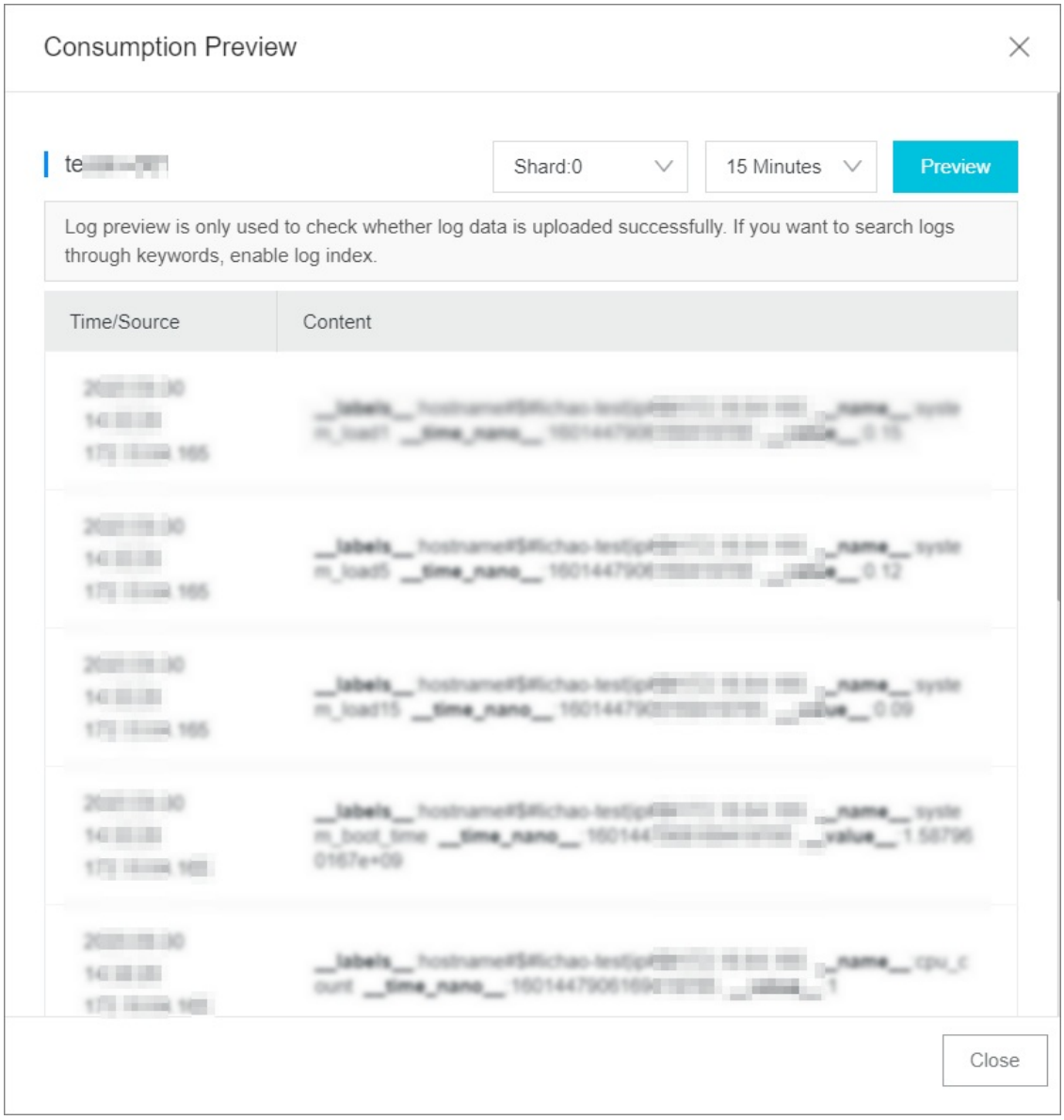
   After you configure Prometheus, you can use the preview feature in the Log Service console to check whether data is imported to Log Service.

   i. Log on to the Log Service console.

   ii. In the Projects section, click the project that you want to manage.

iii. Choose **Time Series Storage > Metricstore**. On the Metricstore tab, find the Metricstore that you want to manage and choose > **Consumption Preview**.

If data is displayed in the **Consumption Preview** panel, Prometheus is correctly configured.



What's next

After metric data is collected from Prometheus, you can perform the following operations on the data:

- Query and analyze Prometheus metric data in Log Service. For more information, see Query and analyze time series data.
- Visualize Prometheus metric data in Grafana. For more information, see Send time series data from Log Service to Grafana.

8.1.3.2. Collect metric data from Prometheus by using a Logtail plug-in

Log Service allows you to collect various types of Prometheus metrics by using a Logtail plug-in. The metrics include Prometheus-formatted metrics from Node Exporter and Kafka Exporter, and Prometheus metrics that are collected from applications. This topic describes how to create a Logtail configuration in the Log Service console to collect metric data from Prometheus.

Prerequisites

A Metricstore is created. For more information, see Manage a Metricstore.

Procedure

> **Notice** A Logtail plug-in supports only one Logtail configuration for Prometheus. If more than one configuration exists, Logtail uses a random configuration.

1. Log on to the Log Service console.
2. In the **Import Data** section, click **Prometheus Metric Scrape**.
3. In the **Specify Logstore** step, select the project and the Metricstore that you want to use. Then, click **Next**.
4. Create a machine group.
   - If a machine group is available, click **Use Existing Machine Groups**.
   - If no machine groups are available, perform the following steps to create a machine group. In this example, an Elastic Compute Service (ECS) instance is used.
     a. On the **ECS Instances** tab, select Manually Select Instances. Then, select the ECS instance that you want to use and click **Execute Now**.

        For more information, see *Obtain an endpoint* in Log Service Developer Guide.

        > **Note** If you want to use a server in a self-managed cluster or a server that is deployed on a third-party cloud, you must manually install Logtail V0.16.66 or later on the Linux server. For more information, see Install Logtail in Linux.

     b. After Logtail is installed, click **Complete Installation**.
     c. In the **Create Machine Group** step, configure the **Name** parameter and click **Next**.

        Log Service allows you to create IP address-based machine groups and custom ID-based machine groups. For more information, see Create a machine group based on a server IP address and Create a custom ID-based machine group.
5. Select the machine group from the **Source Server Groups** section and move the machine group to the **Applied Server Groups** section. Then, click **Next**.

   > **Notice** If you enable a machine group immediately after you create the machine group, the heartbeat status of the machine group may be **FAIL**. This issue occurs because the machine group is not connected to Log Service. To resolve this issue, you can click **Automatic Retry**. If the issue persists, see *What do I do if a Logtail machine group has no heartbeats?*
6. In the **Specify Data Source** step, configure the **Config Name** and **Plug-in Config** parameters. Then, click **Next**.

   Plug-in Config includes the inputs and processors parameters. Log Service provides a template for the inputs parameter. The template includes only the global and scrape_configs sections.
   - inputs: specifies the collection configurations. This parameter is required. Configure the inputs parameter based on your data source.

> **Notice**
>
> - You can configure fields only in the global and scrape_configs sections regardless of whether you collect Prometheus-formatted metrics or Prometheus metrics. For more information, see Prometheus configuration.
> - You can specify only one type of data source in the inputs parameter.

- processors: specifies the processing method. This parameter is optional.

  If you want to append custom fields, such as the IP address of the server on which Logtail is installed and the hostname of the server, to the collected metric data, you must turn on **Use Advanced Edit Mode** to add processors settings. In this case, the processor_appender plug-in must be used. Example:

```
{
  "processors":[
    {
      "type":"processor_appender",
      "detail": {
        "Key": "__labels__",
        "Value": "|host#$#{{__host__}}|ip#$#{{__ip__}}",
        "SortLabels": true
      }
    }
  ]
}
```

  For more information, see Add log fields.

What's next

- Query and analyze data

  After metric data is collected, you can query and analyze the data on the Query & Analysis page of the Metricstore. For more information, see Query and analyze time series data.

- Visualize query and analysis results in Log Service

  You can create a time series chart and add the chart to a dashboard to view the query and analysis results. You can also configure alert rules.

- Visualize data on Grafana

  Log Service allows you to send time series data to Grafana for visualization. For more information, see Send time series data from Log Service to Grafana.

8.2. Query and analysis

8.2.1. Overview of query and analysis of time series data

This topic describes the syntax and limits of query and analysis on time series data.

Log Service supports the following types of syntax for the query and analysis of time series data:

- SQL: You can use the SQL syntax to query and analyze time series data based on the encoding format of the data.
- Combination of SQL and PromQL: This combination allows you to query and analyze time series data in an efficient manner. The combination is implemented by using nested queries. PromQL is the query language that is provided by Prometheus. For more information, see *Prometheus documentation*.

SQL

Examples of SQL-based query statements:

- Query and analyze all data from a Metricstore.

```
*| SELECT * FROM "my_metric_store.prom" WHERE __name__ != ''
```

- Query the data in which the value of the __labels__, 'domain' field is *www.example.com* and obtain the sum of the values of the __value__ field.

```
*| SELECT sum(__value__) FROM "my_metric_store.prom" WHERE element_at(__labels__, 'domain')='www.example.com'
```

- Query the data in which the value of the __labels__, 'domain' field is *www.example.com*, obtain the sum of the __value__ field, and aggregate the data by hour.

```
*| SELECT sum(__value__),date_trunc('hour', __time_nano__/1000000) as t
FROM "my_metric_store.prom"
WHERE element_at(__labels__, 'domain')='www.example.com'
GROUP BY t
ORDER BY t DESC
```

Description:

- The SQL syntax for time series data is the same as the SQL syntax for log data. For more information, see *Log analysis overview*. When you query and analyze time series data by using the SQL syntax, the table name in a FROM clause must be {metrics_store_name}.prom. *{metrics_store_name}* specifies the name of the Metricstore that you created.

  > **Note** You must enclose the table name in double quotation marks ("").

- You can use the `element_at()` function to obtain the value of a key from the __labels__ field. Example: **element_at(__labels__, 'key')**.
- For more information about the table structure, see *Metric*.

Combination of SQL and PromQL

If you use the combination of SQL and PromQL, you can use security check functions and advanced features, such as the machine learning feature.

> **Notice**

- If you use the combination of SQL and PromQL, the table name in a FROM clause must be metrics.
- For information about the API endpoints and descriptions of PromQL functions, see *Prometheus documentation*.

The following table describes the PromQL functions that are supported by Log Service. Among the functions, the promql_query, promql_labels, promql_label_values, and promql_series functions can be invoked only on the Query & Analysis page of a Metricstore.

Function	Description	Example
promql_query(string)	Evaluates an instant query on the data at the point in time that is the closest to the end time of the query time range. This function is equivalent to the `/query API` of Prometheus. Parameter setting: `query=<string>`.	***| SELECT promql_query('up') FROM metrics**

Function	Description	Example
promql_query_range(string, string)	Evaluates a query on the data over a specified period of time. This function is equivalent to the `/query_range API` of Prometheus. Parameter settings: query=<string> and step=<duration>.	***| SELECT promql_query_range('up', '5m') FROM metrics**
promql_labels()	Returns all label keys.	***| SELECT promql_labels() FROM metrics**
promql_label_values(string)	Returns the values of a label.	***| SELECT promql_label_values('__name__') FROM metrics**
promql_series(string)	Returns the time series that is matched.	***| SELECT promql_series('up') FROM metrics**

A PromQL function is similar to a user-defined table generating function (UDTF) and returns a table.

- The following table describes the schema of a table that is returned by the `promql_query(string)` or `promql_query_range(string, string)` function.

Field	Data type	Description
metric	varchar	The metric name of the time series. If a GROUP BY clause is included in the query statement, this field may be empty.
labels	map<varchar, varchar>	The labels. The value is a map.
time	bigint	The time.
value	double	The value that represents a point in time.

- The following table describes the schema of a table that is returned by the `promql_labels()` or `promql_label_values(string)` function.

Field	Data type	Description
label	varchar	Label Key

- The following table describes the schema of a table that is returned by the `promql_series(string)` function.

Field	Type	Description
series	map<varchar, varchar>	The time series.

Limits

- A Metricstore supports only the query API endpoints of Prometheus, such as /query and /query_range. Other API endpoints, such as /admin, /alerts, and /rules, are not supported.
- If you use the combination of SQL and PromQL for query and analysis, a maximum of 11,000 points in time can be returned for a query.
- If you use the combination of SQL and PromQL for query and analysis, the metric name and label that you specify must comply with the naming conventions. For more information, see *Metric*.

8.2.2. Query and analyze time series data

This topic describes how to query and analyze time series data in a Metricstore and how to specify a legend format for a time series chart.

Prerequisites

Time series data is collected. For more information, see Collect time series data.

Procedure

> **Note** Only the combination of the SQL syntax and the PromQL syntax is supported on the query page of a Metricstore. If you want to use the standard SQL syntax, click the **Search & Analyze** button in the upper-right corner of the query page to go to the Search & Analysis page of the Metricstore.

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. Choose **Time Series Storage > Metricstore**. On the Metricstore tab, click the Metricstore that you want to manage.
4. In the upper-right corner of the page that appears, click **15 Minutes(Relative)** and specify a time range for data query and analysis.

   You can select a relative time or a time frame. You can also specify a custom time range.

   > **Note** The query and analysis results may contain time series data that is generated 1 minute earlier or 1 minute later than the specified time range.

5. On the **Query Statements** tab, query and analyze the time series data.

   You can use the following methods to query and analyze time series data:

   - Enter a query statement in the field next to the Metricstore name and click **Preview**.

     You can click the **Add Query Statement** button or the icon to add or copy a query statement. Then, enter the query statement and click **Preview**.

     The results of multiple query statements are displayed in the same time series chart.
   - Select a metric from the **Metrics** drop-down list. A query statement is automatically generated. Then, click **Preview**.

     You can modify the query statement that is generated.

Specify a legend format for a time series chart

After you execute a query statement on the **Query Statements** tab, you can specify a legend format for the time series chart.

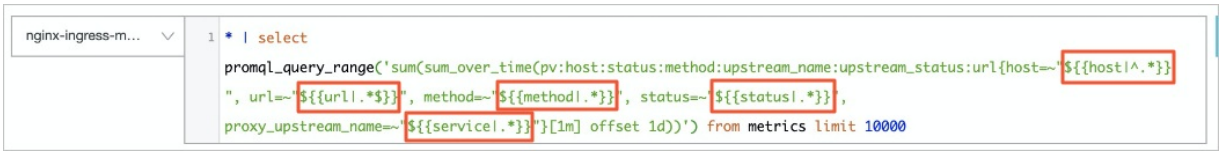
The default legend for each time series consists of a metric name and labels. You can change the legend to a label value by using a magic variable. The format is {{Label key}}. For example, the label of a time series chart is {ip="192.0.2.1"}. If you enter {{ip}} in the **Legend Format** field, the legend of the time series chart changes to 192.0.2.1.

Configure a placeholder variable

Log Service allows you to specify placeholder variables in query statements. For example, you can configure a drill-down event for Chart A to redirect to the dashboard on which Chart B is located. After you configure a drill-down event for Chart A, the variable that you click to trigger the drill-down event and execute the query statement of Chart B is replaced by the placeholder variable specified in Chart B. To trigger the drill-down event, you must click the variable that you configured for Chart A. The format of a placeholder variable is `${{Variable name|Default value}}`. For example, you can set `host=~"^.*"` to `host=~"${{host|^.*}}"`.

Example: In the following query statement, set the values of the host, url, method, status, and proxy_upstream_name fields to placeholder variables. The following figure shows the result.

```
* | select promql_query_range('sum(sum_over_time(pv:host:status:method:upstream_name:upstream_status:url{host=~"^.*", url=~".*$", method=~".*", status=~".*", proxy_upstream_name=~".*"}[1m]))') from metrics limit 10000
```



Related operations

Operation	Description
Copy a query statement	On the **Query Statements** tab, click the icon to copy the specified query statement.
View raw data	On the **Query Statements** tab, click the icon to view raw time series data.
Configure the properties of a time series chart	On the **Properties** tab, configure the properties of the time series chart. For more information, see Configure a time series chart.
Add query and analysis results to a dashboard	Click **Add to New Dashboard** to add the query and analysis results to a dashboard.
Create an alert rule based on query statements	Click **Save as Alert** to create an alert rule for the query and analysis results. For more information, see Configure an alert rule.
Refresh data	You can manually refresh the time series data in a Metricstore or enable the automatic refresh feature. • In the upper-right corner of the query page, choose **Refresh > Once** to refresh the time series data in the Metricstore. • In the upper-right corner of the query page, choose **Refresh > Auto Refresh** and select a refresh interval. The time series data in the Metricstore is automatically refreshed based on the selected interval. The interval can be 15 seconds, 60 seconds, 5 minutes, or 15 minutes.
Share query and analysis results	In the upper-right corner of the query page, click **Share** to copy the URL of the page. You can send the URL to other users who have the permissions to query the time series data in the Metricstore. The query and analysis results that are displayed when you copy the URL are also displayed when a user visits the URL.
Go to the Search & Analysis page	In the upper-right corner of the query page, click **Search & Analyze**.

8.3. Visualization

8.3.1. Configure a time series chart

This topic describes how to configure a time series chart.

Background information

Time series charts are designed for Metricstores. A time series chart displays the results of one or more queries on data that is collected from Prometheus and stored to a Metricstore.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. Choose **Time Series Storage > Metricstore**. On the Metricstore tab, click the Metricstore that you want to manage.
4. Click the **Properties** tab and configure the parameters of the time series chart. The following section describes the parameters.

Parameter	Description
Fill Missing Data	If you turn on **Fill Missing Data**, Log Service automatically generates substitutes for missing samples in a time series.
Y-axis Minimum Value	The minimum value allowed for the y-axis.
Y-axis Maximum Value	The maximum value allowed for the y-axis.
Format Left Y-axis	The format of the left y-axis.
X-axis Scale Density	The scale density of the x-axis. Valid values: 3 to 30.
Line Type	The type of the lines in the time series chart. Valid values: Straight Line and Curve.
Show Points	If you turn on **Show Points**, sample values are displayed in the time series chart.
Margin	The distance between an axis and a border of the time series chart.

5. On the **Query Statements** tab, query and analyze the time series data.

   You can click **Preview Raw Data** in the upper-right corner of the page to view the collected time series data. Example: `__labels__:hostname#$#hostname1|ip#$#192.0.2.0 __time_nano__:1644309671000000000 __value__:52.71 __name__:cpu_util`. Log Service generates a time series chart based on the collected time series data and the metrics that you select. For example, if you want to query the CPU utilization of different hosts, you can select the cpu_util metric. Then, Log Service displays a time series chart for the CPU utilization of different hosts.

8.3.2. Send time series data from Log Service to Grafana

Metricstores of Log Service are compatible with the query API of Prometheus. You can send data from Log Service to Grafana and visualize the data in Grafana. This topic describes how to connect Log Service as a Prometheus data source to Grafana.

Prerequisites

- Grafana is installed. For more information, see Install Grafana.
- Time series data is imported to Log Service. For more information, see Collect metric data from Prometheus.

Connect Log Service to Grafana

1. Log on to Grafana.
2. In the left-side navigation pane, choose **Configuration > Data Sources**.
3. On the **Data Sources** tab, click **Add data source**.
4. Move the pointer over the **Prometheus** card and click **Select**.
5. On the **Settings** tab, configure the parameters. The following table describes the parameters.

Parameter	Description
Name	Specify a name for the data source based on your business requirements. Example: Prometheus-01.
HTTP	◦ **URL**: Enter the URL of the Metricstore in the https://{project}.{sls-enpoint}/prometheus/{project}/{metricstore} format. Replace *{sls-enpoint}* with the Log Service endpoint in the region where the project resides. To obtain the list of Log Service endpoints in different regions, see *Obtain an endpoint* in Log Service Developer Guide. Replace *{project}* and *{metricstore}* with the actual project name and Metricstore name. Example: https://sls-prometheus-test.cn-hangzhou.log.aliyuncs.com/prometheus/sls-prometheus-test/prometheus. **Note** To ensure the security of transmissions, use `https`. ◦ **Whitelisted Cookies**: Add a whitelist. This parameter is optional.
Auth	Turn on **Basic auth**.
Basic Auth Details	◦ **User**: Enter the AccessKey ID of your Apsara Stack tenant account. ◦ **Password**: Enter the AccessKey secret of your Apsara Stack tenant account. We recommend that you use the AccessKey pair of a RAM user that is granted only the read-only permissions on the specified project.

6. Click **Save & Test**.

Import a Log Service dashboard template to Grafana

Perform the following steps to import a Log Service dashboard template to Grafana:

1. Copy the ID of a template.
   i. Go to the Dashboards page of Grafana.
   ii. Click the template that you want to import.
   iii. On the right side of the page, click **Copy ID to Clipboard**.
2. Log on to Grafana.
3. In the left-side navigation pane, choose **Create > Import**.
4. In the Grafana.com Dashboard field, paste the template ID that you copied in Step 1.

   Then, click a blank area to go to the page for data source configuration.
5. Configure the data source.

In this step, configure the parameters based on the data source that you added. For more information, see Connect Log Service to Grafana. The parameters vary based on the dashboard template. You can also configure the telegraf parameter and host parameter based on your business requirements.

6. Click **Import**.

Access Log Service by using the query API of Prometheus

Log Service is compatible with the query API of Prometheus. You can configure Log Service as a Prometheus data source in Grafana or use the Prometheus API to access Log Service. The following table describes the API operations that are supported.

Operation	Example
Instant queries	`GET /api/v1/query` `POST /api/v1/query`
Range queries	`GET /api/v1/query_range` `POST /api/v1/query_range`
Getting label names	`GET /api/v1/labels` `POST /api/v1/labels`
Querying label values	`GET /api/v1/label/<label_name>/values`
Finding series by label matchers	`GET /api/v1/series` `POST /api/v1/series`

9.RAM

9.1. Overview

Resource Access Management (RAM) is a resource access control service provided by Apsara Stack.

You can use RAM to manage users, including employees, systems, and applications. You can also use RAM to grant users permissions to access resources.

RAM provides the following features:

- RAM Role

  To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This role specifies the operations that the cloud service can perform on the resources.

  Only system administrators and level-1 organization administrators can create RAM roles.

- User group

  You can create multiple RAM users for an organization and grant the users different permissions on the same cloud resources in the organization.

  You can create RAM user groups to classify and authorize RAM users within your Apsara Stack tenant account. This simplifies the management of RAM users and their permissions.

  You can create RAM policies to grant permissions to different user groups.

9.2. Create a RAM role

To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This RAM role specifies the operations that the cloud service can perform on the resources.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the upper-right corner of the page, click **Create RAM Role**.
5. On the **Roles - Create RAM Role** page, configure the **Role Name** and **Description** parameters.
6. Click **Create**.

9.3. Create a user

You can create a user and assign the user different roles as an administrator to meet different requirements for system access control.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Users**.
4. On the Users page, click **Create a user**.
5. In the **Create a user** dialog box, configure the parameters. The following table describes the parameters.

Parameter	Description
User name	The username.
Display name	The display name of the user.
Role	The roles that you want to assign to the user.
Organization	The organization to which the user belongs.
Logon policy	The logon policy that restricts the logon time and IP address of the user. If you do not configure this parameter, the default policy is attached to the created user.
Phone	The mobile phone number of the user. If you want to send text messages about the resource requests and usage to the mobile phone number, make sure that the specified mobile phone number is valid.
Landline	Optional. The landline number of the user.
Email	The email address of the user. If you want to send emails about the resource requests and usage to the email address, make sure that the specified email address is valid.
DingTalk Key	Optional. The DingTalk key.
Notify User by Email	If you select **Notify User by Email**, emails about the resource requests and usage are sent to the specified email address.
Notify User by DingTalk	If you select **Notify User by DingTalk**, messages about the resource requests and usage are sent to the specified DingTalk user.

6. Click **OK**.

9.4. Create a user group

You can create a user group in a selected organization and grant permissions to the users within the group at the same time.

Prerequisites

An organization is created.

Context

Relationships between user groups and users:

- A user group can contain zero or more users.
- A user does not need to belong to a user group.
- A user can be added to multiple user groups.

Relationships between user groups and organizations:

- A user group belongs to only one organization.
- You can create multiple user groups within an organization.

Relationships between user groups and roles:

- Only one role can be assigned to each user group.
- A role can be assigned to multiple user groups.
- When a role is assigned to a user group, the permissions that the role has are automatically granted to the users within the user group.

Relationships between user groups and resource sets:

- Zero or more user groups can be added to a resource set.
- A user group can be added to multiple resource sets.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. In the upper-right corner of the page, click **Create a user group**.
5. In the dialog box that appears, configure the **User Group Name**, **Organization**, and **Role authorization** parameters.
6. Click **OK**.

9.5. Add a user to a user group

You can add a user to a user group.

Procedure

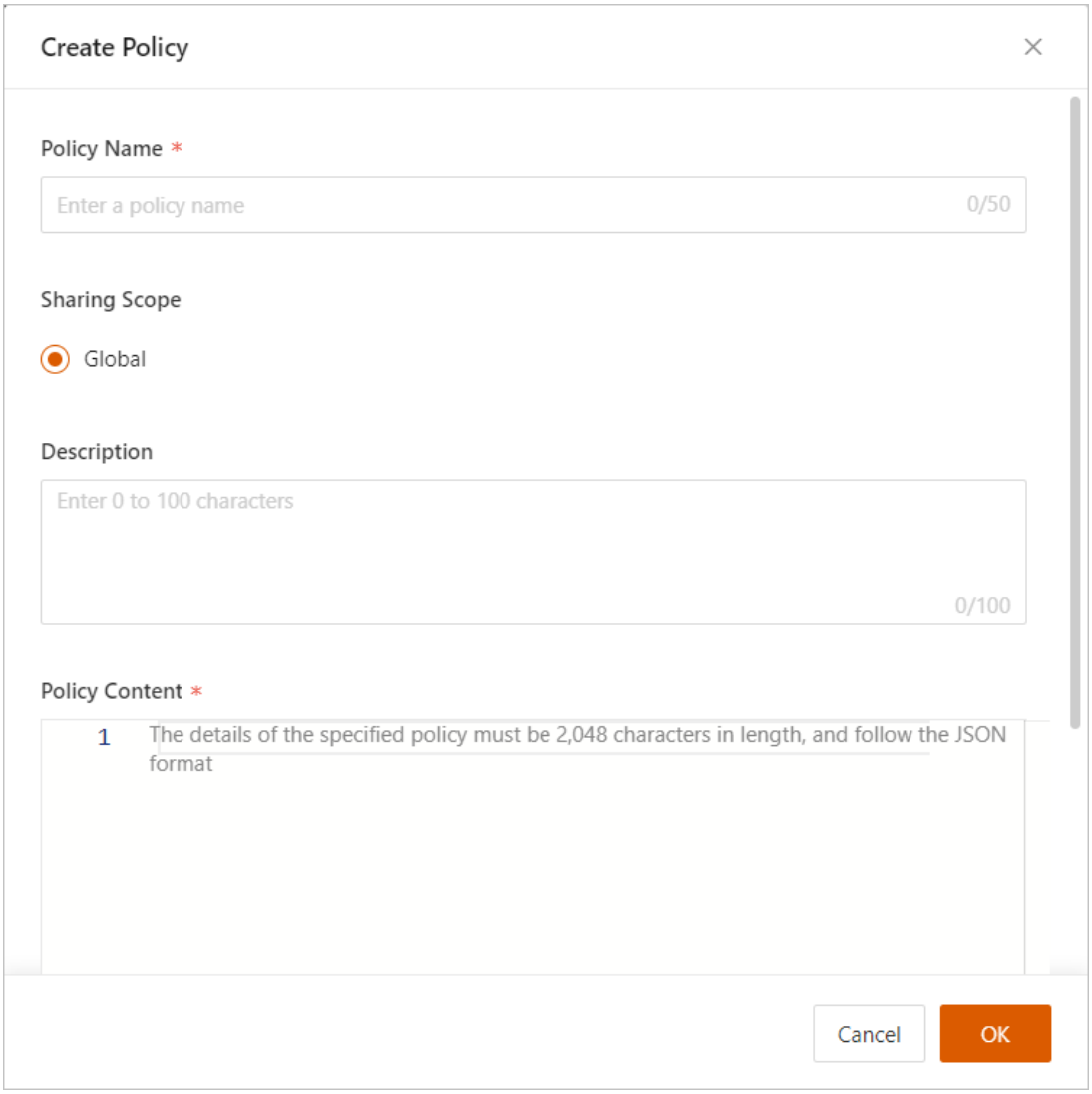
1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **User Groups**.
4. Find the user group to which you want to add a user, and click **Add User** in the **Actions** column.
5. In the dialog box that appears, select the user that you want to add from the left section, and click the right arrow to move the user to the right section.
6. Click **OK**.

9.6. Create a policy

To use a cloud service to access other cloud resources, you must create a policy and attach it to a user group.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, find the RAM role that you want to modify and choose **More** > **Modify** in the **Actions** column to go to the **Roles** page.
5. Click the **Permissions** tab.
6. Click **Add Permission Policy**.
7. In the Add Permission Policy dialog box, enter the information about the policy.



9.7. Grant a RAM user the permissions to manage a project

This topic describes how to grant a Resource Access Management (RAM) user the permissions to manage a specified project.

Prerequisites

- A RAM user is created. For more information, see Create a user.
- A resource set is created and the RAM user is added to the resource set. For more information, see **Enterprise Center > Resource Sets** in *Apsara Uni-manager Management Console User Guide*.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Data Permissions**.
4. Click the resource set that you want to manage, for example, ResourceSet(appstreaming).
5. In the **Product Type** section, click **Log Service**.

6. Find the instance that you want to manage and click **Authorize** in the Actions column.

   In this example, the instance is a Log Service project whose name is test-project.

7. Grant the RAM user the permissions to manage the project.

   If you turn on the Action switch of the **Update Project** permission, the RAM user can modify the project that is selected in Step .

9.8. Grant permissions to a RAM role

When you grant permissions to a RAM role, all users in the user groups that are assigned this role share the granted permissions.

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane of the **Enterprise** page, click **Roles**.
4. In the role name list, find the RAM role that you want to modify and choose **More** > **Modify** in the **Actions** column to go to the **Roles** page.
5. Click the **Permissions** tab.
6. Click **Select Existing Permission Policy**.
7. In the dialog box that appears, select a RAM policy and click **OK**.

   If no RAM policies are available, see Create a permission policy.

9.9. Use custom policies to grant permissions to a RAM user

This topic describes how to use custom policies to grant permissions to a RAM user. In the Resource Access Management (RAM) console, you can grant permissions to the RAM users that belong to your Apsara Stack tenant account.

Context

In terms of data security, we recommend that you follow the principle of least privilege (PoLP) when you grant permissions to RAM users. You must grant the read-only permissions on the project list to RAM users. Otherwise, the RAM users cannot view the projects in the project list.

Use the RAM console to grant permissions to a RAM user

- The read-only permissions on projects

  For example, you want to use your Apsara Stack tenant account to grant the following permissions to a RAM user:

  - The permissions to view the project list of the Apsara Stack tenant account
  - The read-only permissions on the projects that are specified by the Apsara Stack tenant account

  Use the following policy:

```
{
   "Version": "1",
   "Statement": [
     {
       "Action": ["log:ListProject"],
       "Resource": ["acs:log:*:*:project/*"],
       "Effect": "Alow"
      },
     {
       "Action": [
         "log:Get*",
         "log:List*"
       ],
       "Resource": "acs:log:*:*:project/Project name/*",
       "Effect": "Allow"
     }
   ]
 }
```

- The read-only permissions on a specified Logstore and the permissions to save a search and use the saved search

  For example, you want to use your Apsara Stack tenant account to grant the following permissions to a RAM user:

  - The permissions to view the project list of the Apsara Stack tenant account
  - The read-only permissions on a specified Logstore and the permissions to save a search and use the saved search

  Use the following policy.

  > **Note** If the content of the Resource element in a policy does not end with an asterisk (*), the RAM user can access only the specified resource of the current resource type. If the content of the Resource element ends with an asterisk (*), the RAM user can access all resources of the current resource type. Other resources are represented by an asterisk (*).

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:ListProject"
      ],
      "Resource": "acs:log:*:*:project/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*"
      ],
      "Resource": "acs:log:*:*:project/Project name/logstore/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/logstore/Logstore name>"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/dashboard",
        "acs:log:*:*:project/Project name/dashboard/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*",
        "log:Create*"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/savedsearch",
        "acs:log:*:*:project/Project name/savedsearch/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- The read-only permissions on a specified Logstore and the permissions to view all saved searches and dashboards in a project

  For example, you want to use your Apsara Stack tenant account to grant the following permissions to a RAM user:

  - The permissions to view the project list of the Apsara Stack tenant account

- The read-only permissions on a specified Logstore and the permissions to view all saved searches and dashboards in a project

  Use the following policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:ListProject"
      ],
      "Resource": "acs:log:*:*:project/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*"
      ],
      "Resource": "acs:log:*:*:project/Project name/logstore/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/logstore/Logstore name"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/dashboard",
        "acs:log:*:*:project/Project name/dashboard/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/savedsearch",
        "acs:log:*:*:project/Project name/savedsearch/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Use API operations to grant permissions to a RAM user

- The permissions to write data to a specified project

To grant a RAM user only the permissions to write data to a specified project, use the following policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:Post*"
      ],
      "Resource": "acs:log:*:*:project/Project name/*",
      "Effect": "Allow"
    }
  ]
}
```

- The permissions to consume data from a specified project

  To grant a RAM user only the permissions to consume data from a specified project, use the following policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:ListShards",
        "log:GetCursorOrData",
        "log:GetConsumerGroupCheckPoint",
        "log:UpdateConsumerGroup",
        "log:ConsumerGroupHeartBeat",
        "log:ConsumerGroupUpdateCheckPoint",
        "log:ListConsumerGroup",
        "log:CreateConsumerGroup"
      ],
      "Resource": "acs:log:*:*:project/Project name/*",
      "Effect": "Allow"
    }
  ]
}
```

- The permissions to consume data from a specified Logstore

  To grant a RAM user only the permissions to consume data from a specified Logstore, use the following policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:ListShards",
        "log:GetCursorOrData",
        "log:GetConsumerGroupCheckPoint",
        "log:UpdateConsumerGroup",
        "log:ConsumerGroupHeartBeat",
        "log:ConsumerGroupUpdateCheckPoint",
        "log:ListConsumerGroup",
        "log:CreateConsumerGroup"
      ],
      "Resource": [
        "acs:log:*:*:project/Project name/logstore/Logstore name",
        "acs:log:*:*:project/Project name/logstore/Logstore name/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

10.Monitor Log Service

10.1. Overview

The service log feature of Log Service helps you record log data about the operations that are performed on the resources of a project. This feature also provides dashboards that allow you to analyze data in multiple dimensions. You can use this feature to view the service status of Log Service in real time and improve O&M efficiency.

Default configurations

Default configuration item	Description
Logstore	When the service log feature is enabled for a project, the generated log data is classified and stored in one of the dedicated Logstores. By default, Log Service automatically creates the following dedicated Logstores: • **internal-operation_log**: stores operation logs. A log corresponds to an API request. By default, log data in the Logstore is retained for 30 days. The billing method for the Logstore is the same as regular Logstores. • **internal-diagnostic_log**: stores the consumption delay logs of consumer groups and Logtail heartbeat logs. The logs are classified by topic. By default, log data in the Logstore is retained for 30 days. The Logstore is free of charge. For more information about log types and fields, see Log types. **Note** The dedicated Logstores are used to store only the logs that are generated by Log Service. You cannot write other data to these Logstores. However, you can query, analyze, and consume the data in the dedicated Logstores and configure alert rules.
Region	• If you select **Automatic creation (recommended)**, Log Service automatically creates a project in the same region to store service logs. • You can also select a project from the Log Storage Location drop-down list to store service logs. The selected project must reside in the same region as the project for which the service log feature is enabled.
Shard	By default, the system creates two shards and enables the automatic sharding feature for each Logstore. For more information, see *Manage shards* in *Log Service User Guide*.
Log retention period	By default, log data is retained for 30 days. You can change the retention period. For more information, see *Manage a Logstore* in *Log Service User Guide*.
Index	By default, the indexing feature is enabled for all collected log data. If you no longer need to query and analyze data or configure alert rules, you can click **Index Attributes** in the upper-right corner of the **Search & Analysis** page to disable the indexing feature.
Dashboard	The following dashboards are automatically created: • Operations Statistics • Logtail Collection Statistics • Logtail Monitoring • Consumer Group Monitoring For more information, see Service log dashboards.

Scenarios

- **Check whether data is evenly written and consumed among shards**

  You can use predefined dashboards to view the data write and consumption trends of shards and check whether data is evenly written or consumed among shards.

  Multiple Logstores in a project may share the same shards. To view the data writes to multiple shards of a Logstore, you can specify the Logstore as a filter condition.

- **Monitor API request status**

  You can call API operations to write log data, consume log data, and create projects or Logstores. A log is generated in the **internal-operation_log** Logstore each time an API operation is called. If an API request fails, the value of the **Status** field in the generated log is an integer that is greater than 200. For example, the value of the field is 404. You can monitor API requests by viewing the number of logs in which the value of the **Status** field is greater than 200.

- **View Logtail status**

  By default, the following Logtail-related dashboards are created after you enable the service log feature: **Logtail Monitoring** and **Logtail Collection Statistics**. The Logtail Monitoring dashboard helps you monitor Logtail exceptions such as regular expression mismatches and log data parsing failures.

10.2. Manage service logs

This topic describes how to enable and disable the service log feature. This topic also describes how to modify service log configurations.

Prerequisites

- A project is created. For more information, see *Manage a project* in *Log Service User Guide*.
- The RAM user that you want to use to log on to the Log Service console is granted the required permissions. You must use an Apsara Stack tenant account to grant the required permissions to a RAM user. For more information, see *Grant permissions to a RAM role* in *Log Service User Guide*.

Enable the service log feature

1. Log on to the Log Service console.

   For more information, see *Log on to the Log Service console* in *Log Service User Guide*.
2. In the Projects section, click the name of the project that you want to manage.
3. In the **Operations Log** section of the **Overview** page, click **Enable Service Logs**.
4. In the **Enable Service Logs** panel, configure the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Service Logs	◦ **Detailed Logs**: records logs related to operations that are performed on the resources in your project, including create, modify, delete, read, and write operations. The logs are stored in the **internal-operation_log** Logstore of a specified project. ◦ **Important Logs**: records the consumption delay logs of consumer groups and Logtail heartbeat logs by Logstore. The logs are stored in the **internal-diagnostic_log** Logstore of a specified project.

Parameter	Description
Log Storage Location	◦ **Automatic creation (recommended)**: Log Service automatically creates a project named `log-service-{User ID}-{region}` in the same region to store service logs. We recommend that you store all service logs of the same region in this project. ◦ **Current Project**: Log Service stores service logs in the current project. ◦ Other projects in the drop-down list: Log Service stores service logs in another project that resides in the same region as the current project. You can specify only the project that resides in the same region as the project for which the service log feature is enabled.

Modify service log configurations

1. Log on to the Log Service console.

   For more information, see *Log on to the Log Service console* in *Log Service User Guide*.
2. In the Projects section, click the name of the project that you want to manage.
3. In the **Operations Log** section of the **Overview** page, click **Modify**.
4. In the **Service Logs** section of the Modify Service Logs Settings panel, select the log type that you want to record and clear the log type that you do not need to record.
5. Select a project in which you want to store service logs from the **Log Storage Location** drop-down list.

   > **Note**
   > - We recommend that you select **Automatic creation (recommended)** to store service logs in the project that is automatically created. You can store the service logs of different projects that reside in the same region in the same project.
   > - After you change the value of **Log Storage Location**, the service logs that are generated after the change are stored in the new project that you specify. The logs that are stored in the original project are not automatically deleted or migrated to the new project that you specify. If you no longer need the logs in the original project, you can manually delete the original project.
6. Click **OK**.

Disable the service log feature

1. Log on to the Log Service console.

   For more information, see *Log on to the Log Service console* in *Log Service User Guide*.
2. In the Projects section, click the name of the project that you want to manage.
3. In the **Operations Log** section of the **Overview** page, click **Modify**.
4. In the Modify Service Logs Settings panel, clear all log types that are selected in the **Service Logs** section.
5. Click **OK** to disable the service log feature.

   > **Note** After you disable the service log feature, Log Service does not delete the service logs that are stored in the specified project. You can manually delete the project to delete the service logs that you no longer need.

Grant permissions to a RAM user

Before you can use the service log feature as a RAM user, you must use your Apsara Stack tenant account to grant the required permissions to the RAM user. For more information, see *Grant permissions to a RAM role* in *Log Service User Guide*. The following sample code provides an example of a policy that contains the required permissions:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:CreateDashboard",
        "log:UpdateDashboard"
      ],
      "Resource": "acs:log:*:*:project/{The project in which logs are stored}/dashboard/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:GetProject",
        "log:CreateProject",
        "log:ListProject"
      ],
      "Resource": "acs:log:*:*:project/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:List*",
        "log:Create*",
        "log:Get*",
        "log:Update*"
      ],
      "Resource": "acs:log:*:*:project/{The project in which logs are stored}/logstore/*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:*"
      ],
      "Resource": "acs:log:*:*:project/{The project for which the service log feature is enabled}/logg
ing",
      "Effect": "Allow"
    }
  ]
}
```

10.3. Log types

Log Service provides the service log feature. You can use this feature to generate different types of logs. This topic describes the log types and the fields for each log type.

Log types

If you enable the service log feature, you must select the types of the logs that you want to generate. The following table describes the log types.

Log type	Overview	Logstore	Log source	Description

Log type	Overview	Logstore	Log source	Description
Detailed Logs	Records the operations that are performed on the resources in your project, including create, modify, delete, read, and write operations.	internal-operation_log	Operation logs	The logs of all API requests, including requests that are sent in the Log Service console and by using consumer groups and SDKs.
Important Logs	Records the consumption delay events of consumer groups and events that are related to the errors, heartbeats, and statistics of Logtail by Logstore.	internal-diagnostic_log	Consumption delay logs of consumer groups	The consumption delay logs of consumer groups. These logs are generated at 2-minute intervals. If you want to query the consumption delay logs of a consumer group, you must specify `__topic__: consumergroup_log` in the query statement.
			Logtail alert logs	The alert logs that record errors on Logtail. Alert logs are generated at 30-second intervals. If the same error occurs multiple times within 30 seconds, only one alert log is generated. The alert log contains the total number of times that the error occurs and one error message. If you want to query Logtail alert logs, you must specify `__topic__: logtail_alarm` in the query statement.
			Logtail collection logs	The collection logs that record statistics about Logtail configurations. These logs are generated at 10-minute intervals. If you want to query Logtail collection logs, you must specify `__topic__: logtail_profile` in the query statement.
			Logtail status logs	The status logs of Logtail. Logtail reports status at regular intervals. These logs are generated at 1-minute intervals. If you want to query Logtail status logs, you must specify `__topic__: logtail_status` in the query statement.

Notice To ensure the compatibility of a custom query statement, we recommend that you use `__topic__: XXX` to specify a log type in the query statement.

Operation logs

Operation logs are classified into the following categories based on the Method field: read operation logs, write operation logs, and resource operation logs. The following table describes the categories of operation logs.

Category	Request method

Category	Request method
Read operation log	Read operation logs are generated when you call the following API operations: • GetHistograms • GetLogs • PullLogs • GetCursor • GetCursorTime
Write operation log	Write operation logs are generated when you call the following API operations: • PutLogs • PutWebtracking
Resource operation log	Resource operation logs are generated when you call the following API operations: API operations such as CreateProject and DeleteProject

The following table describes the common fields in operation logs.

Field	Description	Example
APIVersion	The version of the API.	0.6.0
AccessKeyId	The AccessKey ID of the account that is used to access Log Service.	LTAI4FkSqNGBsVTqVZYx****
CallerType	The type of the API caller.	Subuser
InvokerUid	The ID of the account that is used to call the API operation.	175921811532****
Latency	The latency of the request. Unit: microseconds.	123279
LogStore	The name of the Logstore.	logstore-1
Method	The API operation for which the log is recorded.	GetLogs
NetOutFlow	The volume of read traffic. Unit: bytes.	120
NetworkOut	The volume of read traffic that is received over the Internet. Unit: bytes.	10
Project	The name of the project.	project-1
RequestId	The ID of the request.	8AEADC8B0AF2FA2592C9****
SourceIP	The IP address of the client that sends the request.	1.2.3.4
Status	The HTTP status code in the response to the request.	200

Field	Description	Example
UserAgent	The agent that is used by the client to call the API operation.	sls-java-sdk-v-0.6.1

The following table describes the fields that are specific to read operation logs.

Field	Description	Example
BeginTime	The start time of the request. The value is a UNIX timestamp.	1523868463
DataStatus	The response to the request. Valid values include Complete, OK, and Unknown.	OK
EndTime	The end time of the request. The value is a UNIX timestamp.	1523869363
Offset	The read offset that you specify when you call the GetLogs operation.	20
Query	The original query statement.	UserAgent: [consumer-group-java]*
RequestLines	The number of rows that are requested by the caller.	100
ResponseLines	The number of returned rows.	100
Reverse	Indicates whether logs are returned in descending order by timestamp. • 1: Logs are returned in descending order by timestamp. • 0: Logs are returned in ascending order by timestamp. This is the default value.	0
TermUnit	The number of delimited keywords that are included in the search statement.	0
Topic	The topic of the data that is read.	topic-1

The following table describes the fields that are specific to write operation logs.

Field	Description	Example
InFlow	The size of the raw data that you want to write. Unit: bytes.	200
InputLines	The number of lines that you want to write.	10
NetInflow	The size of the compressed data that you want to write. Unit: bytes.	100
Shard	The ID of the shard to which data is written.	1

Field	Description	Example
Topic	The topic of the data that is written.	topic-1

Consumption delay logs of consumer groups

The following table describes the fields in consumption delay logs.

Field	Description	Example
consumer_group	The name of the consumer group.	consumer-group-1
fallbehind	The interval between the current consumption checkpoint and the point in time at which the last write operation log is recorded. Unit: seconds.	12345
logstore	The name of the Logstore.	logstore-1
project	The name of the project.	project-1
shard	The ID of the shard whose data is consumed.	1

Logtail alert logs

The following table describes the fields in Logtail alert logs.

Field	Description	Example
alarm_count	The number of times that alerts are generated in the specified time window.	10
alarm_message	The sample raw log that triggers the alert.	M_INFO_COL,all_status_monitor,T22380,0,2018-04-17 10:48:25.0,AY66K,AM5,2018-04-17 10:48:25.0,2018-04-17 10:48:30.561,i-23xebl5ni.1569395.715455,901,00789b
alarm_type	The type of the alert.	REGISTER_INOTIFY_FAIL_ALARM
logstore	The name of the Logstore.	logstore-1
os	The operating system. Example: Linux or Windows.	Linux
project	The name of the project.	project-1
source_ip	The IP address of the server on which Logtail runs.	1.2.3.4
version	The version of Logtail.	0.14.2

Logtail collection logs

Logtail collection logs are classified into the following categories based on the `file_name` field:

- Statistics about a Logtail configuration for a log file.
- Statistics about a Logtail configuration for a Logstore. In the configuration, the `file_name` field is set to `log_store_statistics`.

The following table describes the fields in Logtail collection logs.

Field	Description	Example
logstore	The name of the Logstore.	logstore-1
config_name	The name of the Logtail configuration. The name is globally unique and must be in the following format: `##Logtail configuration version##Project name$Configuration name`.	##1.0##project-1$logstore-1
error_line	The raw log that causes an error.	M_INFO_COL,all_status_monitor,T22380,0,2018-04-17 10:48:25.0,AY66K,AM5,2018-04-17 10:48:25.0,2018-04-17 10:48:30.561,i-23xebl5ni.1569395.715455,901,00789b
file_dev	The device ID of the log file. **Note** If the `file_name` field is set to `logstore_statistics`, this field is invalid.	123
file_inode	The inode of the log file. **Note** If the `file_name` field is set to `logstore_statistics`, this field is invalid.	124
file_name	The full path of the log file or the value of `logstore_statistics`.	*/abc/file_1*
file_size	The size of the log file. Unit: bytes.	12345
history_data_failures	The number of times that data fails to be processed.	0
last_read_time	The last read time in the specified time window. The value is a UNIX timestamp.	1525346677
project	The name of the project.	project-1
logtail_version	The version of Logtail.	0.14.2
os	The operating system.	Windows

Field	Description	Example
parse_failures	The number of lines that fail to be parsed in the specified time window.	12
read_avg_delay	The average of the difference between the actual file size and the offset value that is generated each time log data is read in the specified time window.	65
read_count	The number of reads in the specified time window.	10
read_offset	The last read offset of the log file. Unit: bytes.	12345
regex_match_failures	The number of times that regular expressions fail to be matched.	1
send_failures	The number of times that logs fail to be sent in the specified time window.	12
source_ip	The IP address of the server on which Logtail runs.	1.2.3.4
succeed_lines	The number of log lines that are processed.	123
time_format_failures	The number of times that log times fail to be matched.	122
total_bytes	The total size of data that is read. Unit: bytes.	12345

The following table describes the fields that are specific to Logstore statistics collected when the `file_name` field is set to `logstore_statistics`.

Field	Description	Example
send_block_flag	Indicates whether the send queue is blocked when the specified time window ends.	false
send_discard_error	The number of packets that are discarded due to data errors or insufficient permissions in the specified time window.	0
send_network_error	The number of packets that fail to be sent due to network errors in the specified time window.	12
send_queue_size	The number of unsent packets in the current send queue when the specified time window ends.	3

Field	Description	Example
send_quota_error	The number of packets that fail to be sent because the Logtail quota is exceeded in the specified time window.	0
send_success_count	The number of packets that are sent in the specified time window.	12345
sender_valid_flag	Indicates whether the send flag of the current Logstore is valid when the specified time window ends. Valid values: • true: The flag is valid. • false: The flag is disabled due to network or quota errors.	true
max_send_success_time	The last time when data was sent in the specified time window. The value is a UNIX timestamp.	1525342763
max_unsend_time	The last time when packets in the send queue failed to be sent in the specified time window. The value is a UNIX timestamp. If the send queue is empty, the value is 0.	1525342764
min_unsend_time	The first time when packets in the send queue failed to be sent in the specified time window. The value is a UNIX timestamp. If the send queue is empty, the value is 0.	1525342764

Logtail status logs

The following table describes the fields in Logtail status logs.

Field	Description	Example
cpu	The CPU load of the Logtail process.	0.001333156
hostname	The hostname.	abc2.****
instance_id	The ID of the instance. This ID is randomly assigned.	05AFE618-0701-11E8-A95B-00163E025256_10.11.12.13_151745****
ip	The IP address.	1.0.1.0
load	The average system load.	0.01 0.04 0.05 2/376 5277
memory	The memory space that is occupied by the Logtail process. Unit: MB.	12
detail_metric	The metrics in the JSON format.	detail_metric
os	The operating system.	Linux

Field	Description	Example
os_cpu	The CPU utilization of the system.	0.004120005
os_detail	The details of the operating system.	2.6.32-220.23.8.tcp1.34.el6.x86_64
status	The status of Logtail. • ok • busy • many_log_files • process_block • send_block • send_error	busy
user	The username.	root
user_defined_id	The user-defined ID.	aliyun-log-id
uuid	The universally unique identifier (UUID) of the server.	64F28D10-D100-492C-8FDC-0C62907F****
version	The version of Logtail.	0.14.2
project	The project to which the Logtail configuration belongs.	my-project

The following table describes the fields that are included in the detail_metric field.

Field	Description	Example
config_count	The number of Logtail configurations.	1
config_get_last_time	The last time when the Logtail configuration was obtained.	2021-07-20 16:19:22
config_update_count	The number of Logtail configuration updates after Logtail was started.	1
config_update_item_count	The total number of configuration items that are updated after Logtail was started.	1
config_update_last_time	The time when the configuration was last updated after Logtail was started.	2021-07-20 16:18:42
env_config	Indicates whether environment variables are used to create the Logtail configuration.	false
event_tps	The transactions per second (TPS).	1
last_read_event_time	The last time when data was read.	2021-07-20 16:18:42
last_send_time	The last time when data was sent.	2021-07-20 16:18:42

Field	Description	Example
multi_config	Indicates whether multiple Logtail configurations are enabled to collect logs from the same file.	false
net_err_stat	The number of times that network sending errors occurred in the previous 1, 5, and 15 minutes.	0,0,0
open_fd	The number of log files that are open.	1
plugin_enabled	Indicates whether Logtail plug-ins are enabled.	false
poll_modify_size	The number of monitored log files that are modified.	1
polling_dir_cache	The number of scanned directories.	1
polling_file_cache	The number of scanned files.	1
process_bytes_ps	The size of log data that is processed per second. Unit: bytes.	1000
process_lines_ps	The number of logs that are processed per second.	1000
process_queue_full	The number of processing queues that reach the maximum processing capacity.	1
process_queue_total	The total number of processing queues.	10
process_tps	The number of data processing transactions per second.	0
reader_count	The number of log files that are being processed.	1
region	The region where Logtail resides.	cn-hangzhou,cn-shanghai
register_handler	The number of directories to be monitored.	1
send_bytes_ps	The size of raw log data that is sent per second. Unit: bytes.	11111
send_lines_ps	The number of logs that are sent per second.	1000
send_net_bytes_ps	The volume of network data that is sent per second. Unit: bytes.	1000
send_queue_full	The number of send queues that reach the maximum sending capacity.	1
send_queue_total	The total number of send queues.	12

Field	Description	Example
send_request_concurrency	The maximum number of packets that can be concurrently sent from send queues.	10
send_tps	The number of data sending transactions per second.	0.075
sender_invalid	The number of abnormal send queues.	0
start_time	The start time.	2021-07-20 16:19:22
used_sending_concurrency	The number of packets that are concurrently sent.	0

10.4. Service log dashboards

After you enable the service log feature, Log Service automatically creates the following dashboards based on the log types that you selected to display the related statistics: Operations Statistics, Logtail Collection Statistics, Logtail Monitoring, and Consumer Group Monitoring.

Dashboards

When you enable the service log feature for a project, you can select one or more of the following log types:

- If you turn on **Detailed Logs**, Log Service automatically creates the Operations Statistics dashboard. For more information, see Operations Statistics.
- If you turn on **Important Logs**, Log Service automatically creates the following dashboards: Logtail Collection Statistics, Logtail Monitoring, and Consumer Group Monitoring. For more information, see Logtail Collection Statistics, Logtail Monitoring, and Consumer Group Monitoring.

Operations Statistics

This dashboard displays the statistics about visits and operations, such as API requests and operations that are performed on projects.

Logtail Collection Statistics

This dashboard displays the statistics about the logs collected by Logtail.

Logtail Monitoring

This dashboard displays the statistics about Logtail errors and alerts to help you monitor the status of Logtail in real time.

Consumer Group Monitoring

This dashboard displays the statistics about consumer groups, including the volume of data consumed from shards, consumption delay, and the details of consumer groups.

11.FAQ

11.1. Log collection

11.1.1. How do I troubleshoot errors that occur when I use Logtail to collect logs?

If the preview page is blank or the No Data message appears on the query page after you create a Logtail configuration to collect logs, perform the steps that are described in the topic to troubleshoot the issue.

Procedure

1. Check whether Log Service receives heartbeats from the machine group.

   You can view the Logtail heartbeat status in the Log Service console. For more information, see View the status of a server group.

   If the heartbeat status is OK, perform the next step. If the heartbeat status is FAIL, identify the cause of the failure. For more information, see What can I do if no heartbeat packet is received from a Logtail client?.

2. Check whether the Logtail configuration is created.

   If the heartbeat status of Logtail is OK, check whether the Logtail configuration is created. Make sure that the path and name of monitored logs match those of the files stored on the server. The path can be a full path or a path that includes wildcards.

3. Make sure that the Logtail configuration is applied to the machine group.

   For more information, see Manage server group configurations.

4. Check collection errors.

   If the Logtail configuration is valid, check whether new logs are generated in real time. Logtail collects only incremental log data. Logtail does not read log files that are not updated. If a log file is updated but the updated data cannot be found in Log Service, you can use the following method to troubleshoot the issue:

   - View the logs of the Logtail client.

     Client logs include key INFO logs, all WARNING logs, and all ERROR logs. To view complete and real-time error information, view the client logs in the following paths:

     - Linux: */usr/local/ilogtail/ilogtail.LOG*.
     - Linux: */usr/local/ilogtail/logtail_plugin.LOG*. The file contains the logs such as HTTP logs, MySQL binary logs, and MySQL query results.
     - Windows x64 : *C:\Program Files (x86)\Alibaba\Logtail\logtail_*.log*
     - Windows x32 : *C:\Program Files\Alibaba\Logtail\logtail_*.log*

   - Check whether the amount of log data exceeds the limit.

     To collect large amounts of log data, you may need to modify the startup parameters of Logtail to increase the log collection throughput. For more information, see Set Logtail startup parameters.

11.1.2. What can I do if Log Service does not receive heartbeats from a Logtail client?

If Log Service does not receive heartbeats from a Logtail client, perform the steps that are described in the topic to troubleshoot the issue.

Context

After Logtail is installed on a server, the Logtail client sends heartbeats to Log Service. If the status page of the machine group shows that Log Service does not receive heartbeats from a Logtail client, the Logtail client is not installed or disconnected from the server.

Step 1: Check whether Logtail is installed

Use the following method to check whether Logtail is installed:

- On a Linux server, run the following command:

  ```
  sudo /etc/init.d/ilogtaild status
  ```

  If Logtail is installed, the following result appears:

  ```
  ilogtail is running
  ```

- On a Windows server, perform the following steps:
  i. On **Control Panel**, click **Administrative Tools**, and then click **Services**.
  ii. In the Services window, check the status of the LogtailDaemon and LogtailWorker services. If the services are in the Running state, Logtail is installed.

If Logtail is not installed, install Logtail. For more information, see Install Logtail in Linux or Install Logtail in Windows. Make sure that you install Logtail based on the region where your Log Service project resides. If Logtail is running, go to the next step.

Step 2: Check the Log Service endpoint in the Logtail installation command

When you install Logtail, you must specify a Log Service endpoint based on the region where your Log Service project resides. If the endpoint is incorrect or the Logtail installation command is invalid, Log Service cannot receive heartbeats from the Logtail client.

You can view the Log Service endpoint and the installation method in the Logtail configuration file named *ilogtail_config.json*. The file is stored in the following path:

- Linux: */usr/local/ilogtail/ilogtail_config.json*
- 64-bit Windows: *C:\Program Files (x86)\Alibaba\Logtail\ilogtail_config.json*
- 32-bit Windows: *C:\Program Files\Alibaba\Logtail\ilogtail_config.json*

In the *ilogtail_config.json* Logtail configuration file, check the endpoint that is specified for the `config_server_address` parameter. Then, check whether the Logtail client can use the endpoint to connect to Log Service. For example, if the endpoint that is recorded in the *ilogtail_config.json* Logtail configuration file is `logtail.cn-qingdao-env25-d01.sls-pub.inter.env25.shuguang.com`, you can run the following command to check the connection:

- Linux:

  ```
  curl logtail.cn-qingdao-env25-d01.sls-pub.inter.env25.shuguang.com
  ```

- Windows:

  ```
  telnet logtail.cn-qingdao-env25-d01.sls-pub.inter.env25.shuguang.com 80
  ```

If the Log Service endpoint in the Logtail installation command is incorrect, re-install Logtail. For more information, see Install Logtail in Linux or Install Logtail in Windows.

If the Log Service endpoint in the Logtail installation command is correct, go to the next step.

Step 3: Check the server IP addresses in the machine group

The server IP address that is obtained by a Logtail client must be configured in the machine group. Otherwise, Log Service cannot receive heartbeats or collect logs from the Logtail client. Logtail uses the following methods to obtain the IP address of a server:

- If the server is not bound with a hostname, Logtail obtains the IP address of the first network interface controller (NIC) card of the server.
- If the server is bound with a hostname, Logtail obtains the IP address that corresponds to the hostname. You can view the hostname and IP address in the */etc/hosts* file.

  > **Note** You can run the **hostname** command to view the hostname.

Perform the following steps to check whether the server IP address that is obtained by the Logtail client is configured in the machine group.

1. Check the server IP address that is obtained by Logtail.

   The value of the `ip` field in the *app_info.json* file is the server IP address that is obtained by Logtail. The file is stored in the following path:

   - Linux: */usr/local/ilogtail/app_info.json*
   - 64-bit Windows: *C:\Program Files (x86)\Alibaba\Logtail\app_info.json*
   - 32-bit Windows: *C:\Program Files\Alibaba\Logtail\app_info.json*

   > **Note**
   > - If the `ip` field in the `app_info.json` file is empty, Logtail cannot work. In this case, you must configure an IP address for the server and restart Logtail.
   > - The `app_info.json` file is used only to record information. If you modify the IP address in the file, the server IP address that is obtained by Logtail is not updated.

2. Check the server IP addresses in the machine group.

   Log on to the Log Service console. In the Projects section, click the project to which the machine group belongs. In the left-side navigation pane, click **Machine Groups**, and then click the name of the machine group. In the Machine Group Status section of the **Machine Group Settings** page, check the server IP addresses.

   If no server IP address in the machine group is the same as the IP address that is obtained by Logtail, perform the following step to modify the IP address configurations in the Log Service console:

   - If a server IP address in the machine group is incorrect, change the IP address to the IP address that is obtained by Logtail. Then, check the heartbeat status 1 minute after you save the change.
   - If you modify the IP address of the server where Logtail is installed, for example, the */etc/hosts* file, restart Logtail. After Logtail obtains the new server IP address, set a server IP address in the machine group to the value of the `ip` field in the `app_info.json` file.

You can use the following method to restart Logtail:

- On a Linux server, run the following commands:

  ```
  sudo /etc/init.d/ilogtaild stop
  sudo /etc/init.d/ilogtaild start
  ```

- On a Windows server, perform the following steps:

  On **Control Panel**, choose **Administrative Tools > Services**. In the list that appears, find LogtailWorker and restart LogtailWorker.

11.1.3. How do I query the status of local log collection?

You can use the status query feature of Logtail to query the health status of Logtail and the log collection status. You can also use this feature to troubleshoot log collection issues and customize status monitoring for log collection.

Usage notes

After you install a Logtail client that supports the status query feature, you can query the local log collection status by running commands on the client. For more information about how to install Logtail, see Install Logtail in Linux.

You can run the `/etc/init.d/ilogtaild -h` command on a client to check whether the client supports the status query feature. If the result includes the `logtail insight, version` keyword, the client supports the status query feature.

```
/etc/init.d/ilogtaild -h
Usage: ./ilogtaild { start | stop (graceful, flush data and save checkpoints) | force-stop | status |
-h for help}$
logtail insight, version : 0.1.0
commond list :
       status all [index]
             get logtail running status
       status active [--logstore | --logfile] index [project] [logstore]
             list all active logstore | logfile. if use --logfile, please add project and logstore. de
fault --logstore
       status logstore [--format=line | json] index project logstore
             get logstore status with line or json style. default --format=line
       status logfile [--format=line | json] index project logstore fileFullPath
             get log file status with line or json style. default --format=line
       status history beginIndex endIndex  project logstore [fileFullPath]
             query logstore | logfile history status.
index :   from 1 to 60. in all, it means last $(index) minutes; in active/logstore/logfile/history, it
means last $(index)*10 minutes
```

Logtail supports multiple query commands. The following table describes the query commands, command functionalities, time ranges, and time windows for query results.

Command	Functionality	Maximum time range that can be queried	Time window
all	Queries the status of Logtail.	Last 60 minutes	1 minute
active	Queries the active Logstores that are collecting logs and the active log files from which logs are being collected.	Last 600 minutes	10 minutes
logstore	Queries the collection status of a Logstore.	Last 600 minutes	10 minutes
logfile	Queries the collection status of a log file.	Last 600 minutes	10 minutes

Command	Functionality	Maximum time range that can be queried	Time window
history	Queries the collection status of a Logstore or log file within a specified period of time.	Last 600 minutes	10 minutes

> **Note**
> - The `index` parameter in the preceding commands specifies the index of the time window. Valid values: 1 to 60. The index of the latest time window is 1. The time window ends at the current system time. If you specify a 1-minute time window, the status in the previous interval of `(index, index-1]` minutes is returned. If you specify a 10-minute time window, the status in the previous interval of `(10*index, 10*(index-1)]` minutes is returned.
> - All commands in the preceding table are subcommands of the status command.

all command

- Syntax

```
/etc/init.d/ilogtaild status all [ index ]
```

> **Note** The all command is used to query the status of Logtail. The index parameter is optional. Default value: 1.

- Examples

```
/etc/init.d/ilogtaild status all 1
ok
/etc/init.d/ilogtaild status all 10
busy
```

- Response

Status	Description	Priority	Troubleshooting
ok	Logtail runs as expected.	N/A	No action is required.
busy	The collection speed is high, and Logtail runs as expected.	N/A	No action is required.
many_log_files	A large number of log files are being collected by Logtail.	Low	You can check whether Logtail is configured to collect log files that do not need to be collected.
process_block	Log parsing is blocked.	Low	You can check whether a large number of logs are generated in a short period of time. If you use the all command multiple times and the returned value is always process_block, you can modify the limit of CPU utilization or the limit of concurrent packet sending. For more information, see Set Logtail startup parameters.

Status	Description	Priority	Troubleshooting
send_block	The process of packet sending is blocked.	High	You can check whether a large number of logs are generated in a short period of time and whether the network is stable. If you use the all command multiple times and the returned value is always send_block, you can modify the limit of CPU utilization or the limit of concurrent packet sending. For more information, see Set Logtail startup parameters.

active command

- Syntax

> **Note** The active command is used to query log files. We recommend that you query active Logstores before you query the active log files in the Logstores.

```
/etc/init.d/ilogtaild status active [--logstore] index
```

You can use the `active [--logstore] index` command to query all active Logstores. The `--logstore` parameter is optional.

```
/etc/init.d/ilogtaild status active --logfile index project-name logstore-name
```

You can use the `active --logfile index project-name logstore-name` command to query all active log files in the Logstore of a project.

- Examples

```
/etc/init.d/ilogtaild status active 1
sls-zc-test : release-test
sls-zc-test : release-test-ant-rpc-3
sls-zc-test : release-test-same-regex-3
```

If you run the `active --logstore index` command, the names of the active Logstores are returned in the following format: `project-name : logstore-name`.

```
/etc/init.d/ilogtaild status active --logfile 1 sls-zc-test release-test
/disk2/test/normal/access.log
```

  - If you run the `active --logfile index project-name logstore-name` command, the full paths of active log files are returned.
  - The status of inactive Logstores or inactive log files in the query time window is not returned.

logstore command

- Syntax

```
/etc/init.d/ilogtaild status logstore [--format={line|json}] index project-name logstore-name
```

> **Note**
>
> - The logstore command is used to query the collection status of the specified project and Logstore in the `LINE` or `JSON` format.
> - The default value of the `--format=` parameter is `--format=line`. This value indicates that the status is returned in the LINE format.
> - If the Logstore specified in the preceding command does not exist or is not active in the query time window, an empty response in the `LINE` format or the `null` value in the `JSON` format is returned.

- Examples

```
/etc/init.d/ilogtaild status logstore 1 sls-zc-test release-test-same
time_begin_readable : 17-08-29 10:56:11
time_end_readable : 17-08-29 11:06:11
time_begin : 1503975371
time_end : 1503975971
project : sls-zc-test
logstore : release-test-same
status : ok
config : ##1.0##sls-zc-test$same
read_bytes : 65033430
parse_success_lines : 230615
parse_fail_lines : 0
last_read_time : 1503975970
read_count : 687
avg_delay_bytes : 0
max_unsend_time : 0
min_unsend_time : 0
max_send_success_time : 1503975968
send_queue_size : 0
send_network_error_count : 0
send_network_quota_count : 0
send_network_discard_count : 0
send_success_count : 302
send_block_flag : false
sender_valid_flag : true
/etc/init.d/ilogtaild status logstore --format=json 1 sls-zc-test release-test-same
{
   "avg_delay_bytes" : 0,
   "config" : "##1.0##sls-zc-test$same",
   "last_read_time" : 1503975970,
   "logstore" : "release-test-same",
   "max_send_success_time" : 1503975968,
   "max_unsend_time" : 0,
   "min_unsend_time" : 0,
   "parse_fail_lines" : 0,
   "parse_success_lines" : 230615,
   "project" : "sls-zc-test",
   "read_bytes" : 65033430,
   "read_count" : 687,
   "send_block_flag" : false,
   "send_network_discard_count" : 0,
   "send_network_error_count" : 0,
   "send_network_quota_count" : 0,
   "send_queue_size" : 0,
   "send_success_count" : 302,
   "sender_valid_flag" : true,
   "status" : "ok",
   "time_begin" : 1503975371,
   "time_begin_readable" : "17-08-29 10:56:11",
   "time_end" : 1503975971,
   "time_end_readable" : "17-08-29 11:06:11"
}
```

- Response

Parameter	Description	Unit

Parameter	Description	Unit
status	The status of the Logstore. For information about the different status of Logstore and the actions that are required to handle each status, see the following table.	N/A
time_begin_readable	The time when logs become readable.	N/A
time_end_readable	The time when logs become unreadable.	N/A
time_begin	The time when statistics collection starts.	UNIX timestamp in seconds
time_end	The time when statistics collection ends.	UNIX timestamp in seconds
project	The name of the project.	N/A
logstore	The name of the Logstore.	N/A
config	The name of the Logtail configuration. The name is globally unique. The format of the name is `##1.0##` + project + `$` + config.	N/A
read_bytes	The amount of log data that is read in the query time window.	Bytes
parse_success_lines	The number of log lines that are parsed in the query time window.	Lines
parse_fail_lines	The number of log lines that fail to be parsed in the query time window.	Lines
last_read_time	The time when logs are last read in the query time window.	UNIX timestamp in seconds
read_count	The number of times that the log file is read in the query time window.	N/A
avg_delay_bytes	The average of the difference between the actual file size and the offset value that is generated each time log data is read in the query time window.	Bytes
max_unsend_time	The maximum waiting period for an unsent packet in the sending queue. An unsent packet refers to a packet that is still in the sending queue at the end of the query time window. If no packets exist in the queue, the value is 0.	UNIX timestamp in seconds

Parameter	Description	Unit
min_unsend_time	The maximum waiting period for an unsent packet in the sending queue. An unsent packet refers to a packet that is still in the sending queue at the end of the query time window. If no packets exist in the queue, the value is 0.	UNIX timestamp in seconds
max_send_success_time	The maximum waiting period for an unsent packet in the sending queue.	UNIX timestamp in seconds
send_queue_size	The number of unsent packets in the sending queue at the end of the query time window.	N/A
send_network_error_count	The number of packets that cannot be sent due to network errors in the query time window.	N/A
send_network_quota_count	The number of packets that cannot be sent due to quota limit in the query time window.	N/A
send_network_discard_count	The number of packets that are discarded due to data errors or lack of permissions.	N/A
send_success_count	The number of packets that are sent in the query time window.	N/A
send_block_flag	Indicates whether the sending queue is blocked at the end of the query time window.	N/A
sender_valid_flag	Indicates whether the sender flag of the Logstore is valid. The value true indicates that the sender flag is valid. The value false indicates that the sender flag is invalid and disabled due to a network error or quota error.	N/A

Logstore status

Status	Description	Troubleshooting
ok	Logtail runs as expected.	No action is required.
process_block	Log parsing is blocked.	You can check whether a large number of logs are generated in a short period of time. If you use the all command multiple times and the returned value is always process_block, you can modify the limit of CPU utilization or the limit of concurrent packet sending. For more information, see Set Logtail startup parameters.
parse_fail	Logtail fails to parse logs.	You can check whether the format of logs is the same as the format that you specify in the Logtail configuration.

Status	Description	Troubleshooting
send_block	The process of packet sending is blocked.	You can check whether a large number of logs are generated in a short period time and whether the network is stable. If you use the all command multiple times and the returned value is always send_block, you can modify the limit of CPU utilization or the limit of concurrent packet sending. For more information, see Set Logtail startup parameters.

logfile command

- Syntax

```
/etc/init.d/ilogtaild status logfile [--format={line|json}] index project-name logstore-name fileFu
llPath
```

> **Note**
>
> - The logfile command is used to query the collection status of the specified log files in the `LINE` or `JSON` format.
> - The default value of the `--format=` parameter is `--format=line`. This value indicates that the status is returned in the LINE format.
> - If the log file specified in the command does not exist or is not active in the query time window, an empty response in the `LINE` format or the `null` value in the `JSON` format is returned.
> - The `--format` parameter is placed after the `logfile` parameter.
> - The value of the `filefullpath` parameter must be set to the full path of the log file.

- Examples

```
/etc/init.d/ilogtaild status logfile 1 sls-zc-test release-test-same /disk2/test/normal/access.log
time_begin_readable : 17-08-29 11:16:11
time_end_readable : 17-08-29 11:26:11
time_begin : 1503976571
time_end : 1503977171
project : sls-zc-test
logstore : release-test-same
status : ok
config : ##1.0##sls-zc-test$same
file_path : /disk2/test/normal/access.log
file_dev : 64800
file_inode : 22544456
file_size_bytes : 17154060
file_offset_bytes : 17154060
read_bytes : 65033430
parse_success_lines : 230615
parse_fail_lines : 0
last_read_time : 1503977170
read_count : 667
avg_delay_bytes : 0
/etc/init.d/ilogtaild status logfile --format=json 1 sls-zc-test release-test-same /disk2/test/norm
al/access.log
{
   "avg_delay_bytes" : 0,
   "config" : "##1.0##sls-zc-test$same",
   "file_dev" : 64800,
   "file_inode" : 22544456,
   "file_path" : "/disk2/test/normal/access.log",
   "file_size_bytes" : 17154060,
   "last_read_time" : 1503977170,
   "logstore" : "release-test-same",
   "parse_fail_lines" : 0,
   "parse_success_lines" : 230615,
   "project" : "sls-zc-test",
   "read_bytes" : 65033430,
   "read_count" : 667,
   "read_offset_bytes" : 17154060,
   "status" : "ok",
   "time_begin" : 1503976571,
   "time_begin_readable" : "17-08-29 11:16:11",
   "time_end" : 1503977171,
   "time_end_readable" : "17-08-29 11:26:11"
}
```

- Response

Parameter	Description	Unit
status	The collection status of the log file in the query time window. For more information, see the status parameter in the logstore command section.	N/A
time_begin_readable	The time when logs become readable.	N/A
time_end_readable	The time when logs become unreadable.	N/A

Parameter	Description	Unit
time_begin	The time when statistics collection starts.	UNIX timestamp in seconds
time_end	The time when statistics collection ends.	UNIX timestamp in seconds
project	The name of the project.	N/A
logstore	The name of the Logstore.	N/A
file_path	The path of the log file.	N/A
file_dev	The ID of the device from which the log file is collected.	N/A
file_inode	The inode of the log file.	N/A
file_size_bytes	The size of the last log file that is scanned in the query time window.	Bytes
read_offset_bytes	The parsing offset of the log file.	Bytes
config	The name of the Logtail configuration. The name is globally unique. The format of the name is `##1.0##` + project + `$` + config.	N/A
read_bytes	The amount of log data that is read in the query time window.	Bytes
parse_success_lines	The number of log lines that are parsed in the query time window.	Lines
parse_fail_lines	The number of log lines that fail to be parsed in the query time window.	Lines
last_read_time	The time when logs are last read in the query time window.	UNIX timestamp in seconds
read_count	The number of times that the log file is read in the query time window.	N/A
avg_delay_bytes	The average of the difference between the actual file size and the offset value that is generated each time log data is read in the query time window.	Bytes

history command

- Syntax

```
/etc/init.d/ilogtaild status history beginIndex endIndex project-name logstore-name [fileFullPath]
```

> **Note**
>
> - The history command is used to query the collection status of a Logstore or log file in the query time window.
> - The `beginIndex` and `endIndex` parameters specify the start and end indexes of the time windows that you want to query. You must ensure that beginIndex must be less than or equal to endIndex (`beginIndex <= endIndex`).
> - The `fileFullPath` parameter is optional. If you specify the path of a log file, the collection status of the log file is queried. Otherwise, the collection status of the Logstore is queried.

- Examples

  Query the collection status of a Logstore.

  - Command

    ```
    /etc/init.d/ilogtaild status history 1 3 sls-zc-test release-test-same /disk2/test/normal/access.
    log
    ```

  - Response

    ```
            begin_time          status      read  parse_success  parse_fail      last_read_time  read
    _count  avg_delay    device     inode  file_size  read_offset
     17-08-29 11:26:11              ok   62.12MB         231000           0   17-08-29 11:36:11
    671         0B     64800  22544459    18.22MB      18.22MB
     17-08-29 11:16:11              ok   62.02MB         230615           0   17-08-29 11:26:10
    667         0B     64800  22544456    16.36MB      16.36MB
     17-08-29 11:06:11              ok   62.12MB         231000           0   17-08-29 11:16:11
    687         0B     64800  22544452    14.46MB      14.46MB
    ```

  Query the collection status of a log file.

  - Command

    ```
    $/etc/init.d/ilogtaild status history 2 5 sls-zc-test release-test-same
    ```

  - Response

    ```
            begin_time          status      read  parse_success  parse_fail      last_read_time  read
    _count  avg_delay  send_queue  network_error  quota_error  discard_error  send_success  send_bloc
    k  send_valid          max_unsend          min_unsend    max_send_success
     17-08-29 11:16:11              ok   62.02MB         230615           0   17-08-29 11:26:10
    667         0B           0              0            0              0           300       false
    true   70-01-01 08:00:00   70-01-01 08:00:00   17-08-29 11:26:08
     17-08-29 11:06:11              ok   62.12MB         231000           0   17-08-29 11:16:11
    687         0B           0              0            0              0           303       false
    true   70-01-01 08:00:00   70-01-01 08:00:00   17-08-29 11:16:10
     17-08-29 10:56:11              ok   62.02MB         230615           0   17-08-29 11:06:10
    687         0B           0              0            0              0           302       false
    true   70-01-01 08:00:00   70-01-01 08:00:00   17-08-29 11:06:08
     17-08-29 10:46:11              ok   62.12MB         231000           0   17-08-29 10:56:11
    692         0B           0              0            0              0           302       false
    true   70-01-01 08:00:00   70-01-01 08:00:00   17-08-29 10:56:10
    ```

- Response
  - The collection status of the Logstore or log file in each query time window is listed in a line.
  - For more information about the response parameters, see the `logstore` command and `logfile` command sections.

Response status codes

- Success code

  If the parameters that you specify in a command is valid even if the queried Logstore or log file is not found, the code 0 is returned. Examples:

```
/etc/init.d/ilogtaild status logfile --format=json 1 error-project error-logstore /no/this/file
null
echo $?
0
/etc/init.d/ilogtaild status all
ok
echo $?
0
```

- Error codes

```
/etc/init.d/ilogtaild status nothiscmd
invalid param, use -h for help.
echo $?
10
/etc/init.d/ilogtaild status/all 99
invalid query interval
echo $?
1
```

  If a non-zero code is returned, an error occurs. The following table describes the possible non-zero codes.

Code	Description	Response	Troubleshooting
10	The command is invalid or the required parameters in the command are not specified.	`invalid param, use -h for help.`	You can run the `-h` command to obtain help information.
1	The value of the index parameter is not within the range of 1 to 60.	`invalid query interval`	You can run the `-h` command to obtain help information.
1	The collection status in the specified query time window cannot be queried.	`query fail, error: $(error)` . For more information, see errno.	The startup time of Logtail is earlier than the query time window. For more information, submit a ticket.
1	The start time of the query falls out of the query time window.	`no match time interval, please check logtail status`	You can check whether Logtail runs as expected. For more information, submit a ticket.
1	No data exists in the query time window that you specify.	`invalid profile, may be logtail restart`	You can check whether Logtail runs as expected. For more information, submit a ticket.

Scenarios

You can use the status query feature of Logtail to query the overall status of Logtail. You can also obtain specific metrics based on the collection status during log collection. You can customize a mechanism to monitor the log collection status based on the queried information.

Monitor the status of Logtail

You can monitor the status of Logtail by using the `all` command.

For example, you can run the command every minute to query the status of Logtail. If the `process_block`, `send_block`, or `send_error` value is returned for 5 consecutive minutes, an alert is triggered.

You can adjust the alert duration and monitoring scope based on the priorities of the collected log files.

Monitor the log collection status

You can monitor the log collection status of a Logstore by using the `logstore` command.

For example, you can run the `logstore` command every 10 minutes to query the status of the Logstore. If the value of the `avg_delay_bytes` parameter exceeds 1 MB (1024 × 1024 bytes) or the value of the `status` parameter is not `ok`, an alert is triggered.

You can adjust the alert threshold for the `avg_delay_bytes` metric based on the size of data that is generated during log collection.

Check whether Logtail has finished collecting log files

You can check whether Logtail has finished collecting log files by using the `logfile` command.

If Logtail no longer collects log files, you can run the `logfile` command every 10 minutes to query the status of the log file. If the value of the `read_offset_bytes` parameter is the same as the value of the `file_size_bytes` parameter, the log file is collected.

Troubleshoot log collection issues

If latency occurs on a server during log collection, you can use the `history` command to query the status history of log collection.

1. The value of the `send_block_flag` parameter is true. This indicates that log collection is blocked due to unstable network connections.
   - If the value of the `send_network_quota_count` parameter is greater than 0, split the shards in the Logstore. For more information, see Split a shard.
   - If the value of the `send_network_error_count` parameter is greater than 0, check the network connections.
   - If no network error occurs, adjust the limit of concurrent packet sending and the data transfer speed of Logtail. For more information, see Set Logtail startup parameters.
2. The parameters for packet sending are set to appropriate values. However, the value of the `avg_delay_bytes` parameter is large.
   - Use the value of the `read_bytes` parameter to calculate the average speed at which logs are parsed. You can determine whether a large amount of data is transferred during log collection based on the average speed.
   - Adjust the limits on resource usage for Logtail. For more information, see Set Logtail startup parameters.
3. The value of the `parse_fail_lines` parameter is greater than 0.

   Check whether the regular expression for log parsing can match all required log fields.

11.1.4. How do I debug a regular expression?

If you select the full regex mode when you configure Logtail to collect and parse text logs, you must specify a regular expression based on your sample log entries. This topic describes how to debug a regular expression.

Context

To debug the regular expression that you specified in the Log Service console, you can click **Validate** in the console and check the following results:

- If the regular expression is used to match the start part of the first line in a log entry, check whether the regular expression can match the expected number of log entries.
- If the fields are extracted by the regular expression, check whether the value of each field meets your requirements.

You can use online tools such as regex101.com and regextester.com to debug a regular expression. You can copy and paste the regular expression that is generated by Log Service to an online tool, and specify a sample log entry as the test string.

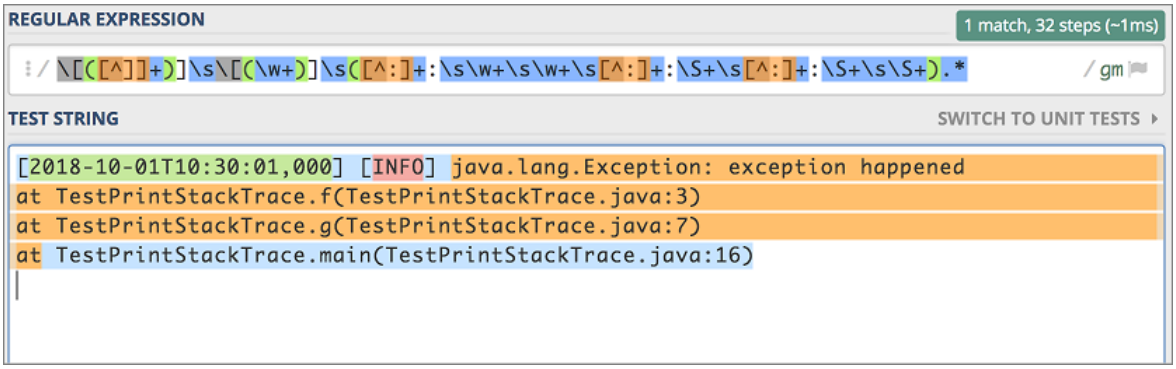
If you use the full regex mode, Log Service automatically generates a regular expression based on the sample log entry that you specify. However, the regular expression may fail to match the message field in multi-line log entries as expected. The following example shows how to use the regex101.com tool to debug a regular expression.

Procedure

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. In the Logstores list, find the Logstore that you want to manage and choose **Data Import > Logtail Configurations**.
4. Click the name of the Logtail configuration that you want to manage.
5. On the **Logtail Config** page, copy the regular expression that is automatically generated by Log Service based on sample log entries.
6. Visit the regex101.com website.
7. In the **REGULAR EXPRESSION** field, paste the regular expression.

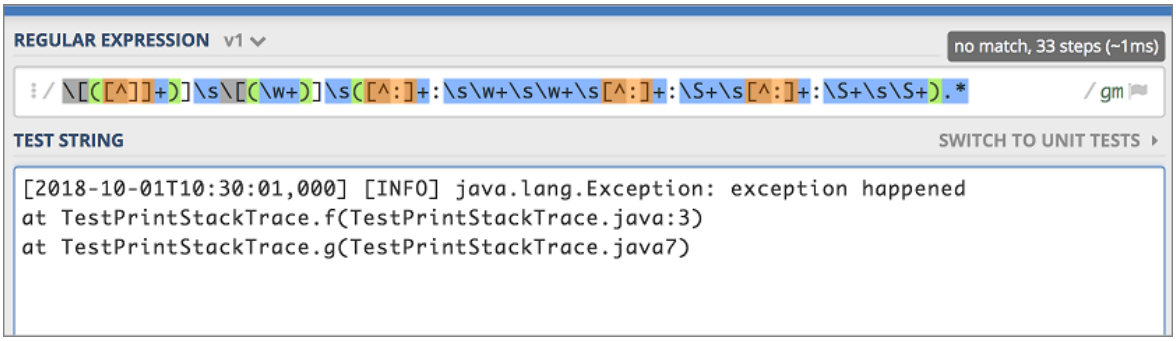
   On the right side of the page, you can view the explanation of the regular expression.
8. In the **TEST STRING** field, paste a sample log entry.

   In the following figure, the log content that is included in the message field is highlighted in orange, and the log content that is not included is highlighted in blue. The figure shows that the substring that follows the `at` word is not included in the `message` field. Therefore, the regular expression does not match fields in the sample log entry as expected and cannot be used to collect log data.

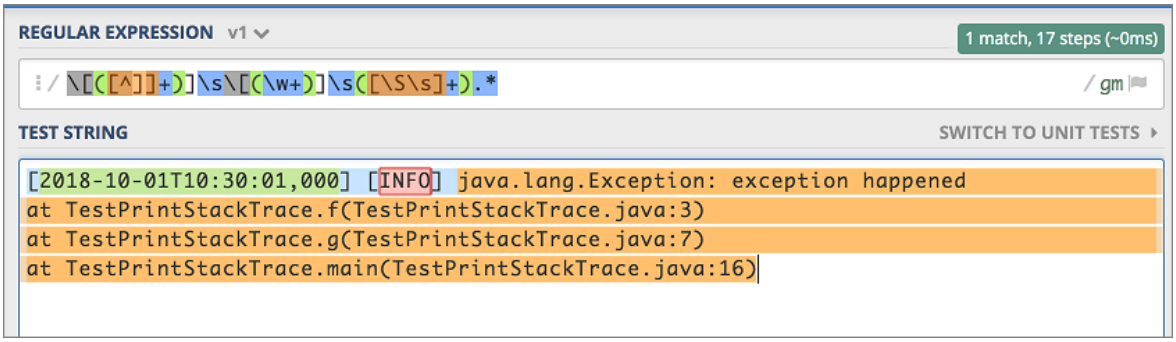
9. Check whether the regular expression can match fields in the sample log entry that contains two colons (::).

   The following figure shows that the regular expression fails to match fields in the sample log entry.

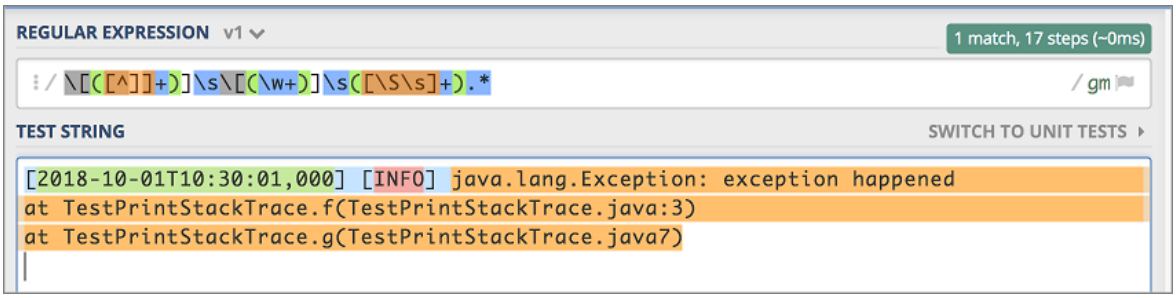


10. Replace the last subexpression in the regular expression with `[\S\s]+`, and check whether the regular expression can match fields in the sample log entries as expected.

    The following figure shows how the modified regular expression matches the substring that follows the `at` word.



    The following figure shows how the modified regular expression matches the sample log entry that contains two colons (::).



    You can perform the preceding steps to debug your regular expression. After you validate the regular expression, you can apply the expression to a Logtail configuration.

11.1.5. How do I optimize regular expressions?

A regular expression that accurately matches data with a high match rate can improve the performance of log collection.

When you optimize regular expressions, we recommend that you conform to the following rules:

- Use precise characters.

  We recommend that you do not use wildcard characters `.*` in a regular expression to match fields in log entries. Wildcard characters may cause mismatches that reduce the matching performance. For example, if you want to extract a field that consists of only letters, use `[A-Za-z]`.

- Use appropriate quantifiers.

  We recommend that you do not use `+,*`. For example, if you want to query data in a more efficient and accurate manner, use `\d` instead of `\d+` or `\d{1,3}` to match IP addresses.

- Debug regular expressions.

  If an error occurs when you use a regular expression, you can use online tools such as **regex101.com** to debug the regular expression. This way, you can troubleshoot the issue and optimize the regular expression in an efficient manner.

11.1.6. How do I use the full regex mode to collect log entries in multiple formats?

The full regex mode requires that log entries to be collected be in the same format. Therefore, if you want to collect log entries that are in multiple formats, you must use the schema-on-write or schema-on-read solution.

Taking Java logs as an example, the following section lists the types of error log entry and normal log entry.

- Multi-line WARNING log entries
- Simple text INFO log entries
- Key-value DEBUG log entries

```
[2018-10-01T10:30:31,000] [WARNING] java.lang.Exception: another exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
[2018-10-01T10:30:32,000] [INFO] info something
[2018-10-01T10:30:33,000] [DEBUG] key:value key2:value2
```

To collect log entries of these types, you can use the following solutions:

- Schema-on-write: To extract log fields, you must apply multiple Logtail configurations with different regular expressions to a log file.

  > **Note** However, Logtail cannot apply multiple Logtail configurations directly to the same log file. Therefore, you must set up multiple symbolic links for the directory in which the log file resides. Each Logtail configuration applies to a symbolic link to collect log entries in a specific format.

- Schema-on-read: you can use a common regular expression to collect log entries in different formats.

  For example, if you want to collect log entries in multiple formats, you can configure a regular expression that matches the time and log level fields as the first line, and specify the rest of the log entries as the log message. If you want to parse the message, create an index for the message, specify a regular expression to extract log messages, and then extract target fields.

  > **Note** We recommend that you use this solution only for scenarios in which tens of millions of log entries are collected, or fewer.

11.1.7. How do I specify time formats for logs?

If you configure Logtail to collect logs, you must specify a common time format for the time field of the logs.

- The timestamp of a log is accurate to seconds. Therefore, you can specify the time format only to seconds.
- You need to specify the time format only for the time in the time field.

The following examples show time formats that are commonly used:

```
Custom1 2017-12-11 15:05:07
%Y-%m-%d %H:%M:%S
Custom2 [2017-12-11 15:05:07.012]
[%Y-%m-%d %H:%M:%S]
RFC822     02 Jan 06 15:04 MST
%d %b %y %H:%M
RFC822Z    02 Jan 06 15:04 -0700
%d %b %y %H:%M
RFC850      Monday, 02-Jan-06 15:04:05 MST
%A, %d-%b-%y %H:%M:%S
RFC1123     Mon, 02 Jan 2006 15:04:05 MST
%A, %d-%b-%y %H:%M:%S
RFC3339     2006-01-02T15:04:05Z07:00
%Y-%m-%dT%H:%M:%S
RFC3339Nano 2006-01-02T15:04:05.999999999Z07:00
%Y-%m-%dT%H:%M:%S
```

11.1.8. How do I configure non-printable characters in a sample log?

This topic describes how to configure non-printable characters in a sample log.

Context

If you collect logs in delimiter mode, Log Service allows you to specify a non-printable character as the delimiter or quote. Non-printable characters are characters whose decimal ASCII codes are within the range of 1 to 31 and 127. If you use a non-printable character as a delimiter or quote, you must find the hexadecimal ASCII code of the character and enter the character in the following format: `0x`*Hexadecimal ASCII code of the non-printable character*. For example, a sample log is `123456780`. You can specify 0x01 as the delimiter and 0x02 as the quote, and then enter a non-printable character 0x01 between the digits 5 and 6.

Procedure

1. Log on to the Log Service console.
2. Right-click the blank space on the browser and select **Inspect** from the shortcut menu.
3. On the page that appears, click the **Console** tab.
4. Enter `"\x01"` in the code editor and press Enter.
5. Copy the returned result.

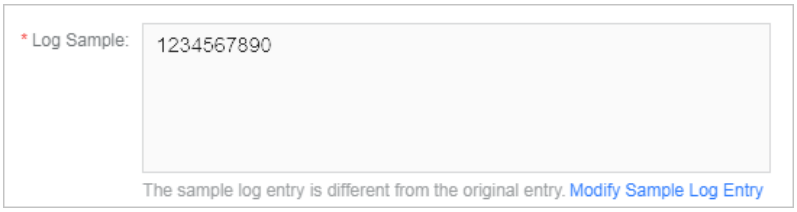
   A non-printable character is enclosed in double quotation marks (").

6. Paste the returned result between the digits 5 and 6.

   In the **Logtail Config** step, paste the result in the **Log Sample** field. For more information, see Collect DSV formatted logs.

7. Delete the double quotation marks (") between the digits 5 and 6.

   A non-printable character is configured in a sample log.



11.1.9. How do I troubleshoot errors that occur when I collect logs from containers?

If an error occurs when you use Logtail to collect logs from Docker containers, self-managed Kubernetes, or Container Service for Kubernetes (ACK), you can perform the steps that are described in this topic to troubleshoot the issue.

Troubleshoot an error if Log Service does not receive heartbeats from a Logtail client

To check whether Logtail is installed, perform the following steps:

1. View the heartbeat status of servers in a machine group.
   i. Log on to the Log Service console.
   ii. In the Projects section, click the project that you want to manage.
   iii. In the left-side navigation pane, click the **Machine Groups** icon.
   iv. In the Machine Groups list, click the name of the machine group that you want to view.

   In the **Machine Group Status** section, count the number of servers whose heartbeat status is **OK**.
2. Count the number of worker nodes in the related cluster.
   i. Log on to a master node in the Kubernetes cluster. For more information, see **Use kubectl to connect to a Kubernetes cluster** in *User Guide for Container Service for Kubernetes*.
   ii. Run the following command to view the number of worker nodes in the cluster:

   ```
   kubectl get node | grep -v master
   ```

   The following output is expected:

   ```
   NAME                                STATUS    ROLES     AGE       VERSION
   cn-hangzhou.i-bp17enxc2us3624wexh2  Ready     <none>    238d      v1.10.4
   cn-hangzhou.i-bp1ad2b02jtqd1shi2ut  Ready     <none>    220d      v1.10.4
   ```
3. Check whether the number of servers whose heartbeat status is **OK** in the machine group is equal to the number of worker nodes in the cluster. Troubleshoot the error based on the check result.
   - The heartbeat status of all servers in the machine group is **Failed**.

- If you use Logtail to collect standard Docker logs, check whether the values of the *${your_region_name}*, *${your_aliyun_user_id}*, and *${your_machine_group_user_defined_id}* parameters are valid. For more information, see the Parameters section in Collect standard Docker logs.
- If you use Logtail to collect ACK logs, submit a ticket.
- If you use Logtail to collect logs from self-managed Kubernetes, check whether the values of the *{your-project-suffix}*, *{aliuid}*, *{access-key-id}*, and *{access-key-secret}* parameters are valid. For more information, see the Parameters section in Collect Kubernetes logs.

  If the value of a parameter is invalid, run the `helm del --purge alibaba-log-controller` command to delete the installation package and re-install Logtail.

- The number of servers whose heartbeat status is**OK** is less than the number of worker nodes in the cluster.
  a. Check whether a DaemonSet is manually deployed by using a YAML file.

     Run the `kubectl get po -n kube-system -l k8s-app=logtail` command to perform the check. If the command returns pod information, a DaemonSet is manually deployed by using a YAML file.

  b. Download the latest version of the Logtail DaemonSet template.
  c. Set the *${your_region_name}*, *${your_aliyun_user_id}*, and *${your_machine_group_name}* parameters based on your business requirements.
  d. Run the `kubectl apply -f ./logtail-daemonset.yaml` command to update the DaemonSet YAML file.

  If the error persists, submit a ticket to contact Log Service technical support.

Troubleshoot an error if Log Service does not collect logs from containers

If no log is displayed in the **Consumption Preview** panel or on the **Search & Analysis** page of a Logstore, Log Service does not collect logs from the machine group of the Logstore. Check the status of the containers that correspond to the servers in the machine group. If the containers are working as expected, perform the following steps to troubleshoot the error:

1. Check the heartbeat status of the servers in the machine group. For more information, see View the heartbeat status of servers in a machine group.
2. Check whether the parameter settings in the related Logtail configuration are correct.

   Check whether the values of the IncludeLabel, ExcludeLabel, IncludeEnv, and ExcludeEnv parameters in the Logtail configuration meet your business requirements.

   > **Note** The `IncludeLabel` or ExcludeLabel parameter specifies whether to include the container images to which specified labels are attached. You can run the docker inspect command to retrieve a list of container image labels. These labels are not the labels that are defined by using Kubernetes. To check whether the parameter settings are valid in a Logtail configuration, delete the IncludeLabel, ExcludeLabel, IncludeEnv, and ExcludeEnv parameters of the Logtail configuration. If Log Service can collect logs from the containers after you delete the parameters, the parameter settings are invalid.

3. Check other items.

   Log Service does not collect logs from containers in the following scenarios:

   - Log files are not updated.
   - The log files of a container are not stored in the default storage or a storage attached to the container.

Other O&M operations

- Log on to a Logtail container
- View the operational logs of Logtail
- Ignore the stdout logs of a Logtail container
- View the status of Log Service components in a Kubernetes cluster

- View the version number, IP address, and startup time of Logtail

Log on to a Logtail container

Use one of the following methods based on your business requirements.

- Docker
  i. Log on to the host and run the following command to view and record the ID of the Logtail container:

  ```
  docker ps | grep logtail
  ```

  ii. Run the following command to log on to the Logtail container:

  ```
  docker exec -it [$ID] bash
  ```

  > **Note** [$ID] is the ID of the Logtail container.

- Kubernetes
  i. Run the following command to view and record the pod where the Logtail container resides:

  ```
  kubectl get po -n kube-system | grep logtail
  ```

  ii. Run the following command to log on to the pod:

  ```
  kubectl exec -it -n kube-system [$Pod_ID] bash
  ```

  > **Note** [$Pod_ID] is the ID of the pod.

View the operational logs of Logtail

The operational logs of Logtail are stored in the files named *ilogtail.LOG* and *logtail_plugin.LOG* in the */usr/local/ilogtail/* directory of a Logtail container.

1. Log on to a Logtail container. For more information, see Log on to a Logtail container.
2. Run the following command to go to the /usr/local/ilogtail/ directory:

   ```
   cd /usr/local/ilogtail
   ```

3. Run the following commands in sequence to view the ilogtail.LOG and logtail_plugin. LOG files:

   ```
   cat ilogtail.LOG
   cat logtail_plugin.LOG
   ```

Ignore the stdout logs of a Logtail container

The standard output of the container is irrelevant to this case. Ignore the following standard output:

```
start umount useless mount points, /shm$|/merged$|/mqueue$
umount: /logtail_host/var/lib/docker/overlay2/3fd0043af174cb0273c3c7869500fbe2bdb95d13b1e110172ef57fe840c82155/merged: must be superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/d5b10aa19399992755de1f85d25009528daa749c1bf8c16edff44beab6e69718/merged: must be superuser to unmount
umount: /logtail_host/var/lib/docker/overlay2/5c3125daddacedec29df72ad0c52fac800cd56c6e880dc4e8a640b1e16c22dbe/merged: must be superuser to unmount
......
xargs: umount: exited with status 255; aborting
umount done
start logtail
ilogtail is running
logtail status:
ilogtail is running
```

View the status of Log Service components in a Kubernetes cluster

1. Log on to a master node in the Kubernetes cluster. For more information, see **Use kubectl to connect to a Kubernetes cluster** in *User Guide for Container Service for Kubernetes*.
2. Run the following command to view the status of Log Service components in the Kubernetes cluster:

   ```
   helm status alibaba-log-controller
   ```

View the version number, IP address, and startup time of Logtail

1. Log on to a Logtail container. For more information, see Log on to a Logtail container.
2. Run the following command to view the version number, IP address, and start time of Logtail:

   ```
   kubectl exec logtail-ds-gb92k -n kube-system cat /usr/local/ilogtail/app_info.json
   ```

   The following output is expected:

   ```
   {
      "UUID" : "",
      "hostname" : "logtail-gb92k",
      "instance_id" : "0EBB2B0E-0A3B-11E8-B0CE-0A58AC140402_10.10.10.10_1517810940",
      "ip" : "203.0.113.10",
      "logtail_version" : "0.16.2",
      "os" : "Linux; 3.10.0-693.2.2.el7.x86_64; #1 SMP Tue Sep 12 22:26:13 UTC 2017; x86_64",
      "update_time" : "2018-02-05 06:09:01"
   }
   ```

11.1.10. How do I obtain the labels and environment variables of a container?

Log Service allows you to collect logs from containers. You can specify the containers by label or environment variable. Labels are retrieved by running the docker inspect command and environment variables are specified in the startup configuration of each container.

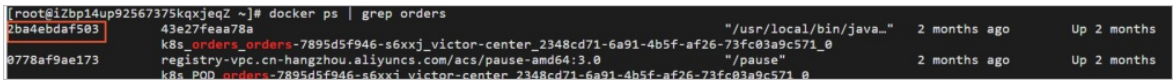
Obtain container labels

1. Log on to the host where the container whose labels you want to obtain resides. For example, the host is an Elastic Compute Service (ECS) instance.
2. Run the following command to obtain the ID of the container.

   The *orders* variable in the command is the name of a container group. Replace the value of the variable with an actual name.

```
docker ps | grep orders
```

2ba4ebdaf503 in the response indicates the ID of the container.



3. Run the following command to obtain the labels of the container.

   The *2ba4ebdaf503* variable in the command is the ID of a container. Replace the value of the variable with an actual ID.

   ```
   docker inspect 2ba4ebdaf503
   ```

   The Labels field in the response indicates the container labels.

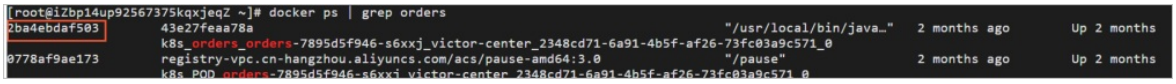
Obtain environment variables

1. Log on to the host where the container whose labels you want to obtain resides. For example, the host is an ECS instance.
2. Run the following command to obtain the ID of the container.

   The *orders* variable in the command is the name of a container group. Replace the value of the variable with an actual name.

   ```
   docker ps | grep orders
   ```

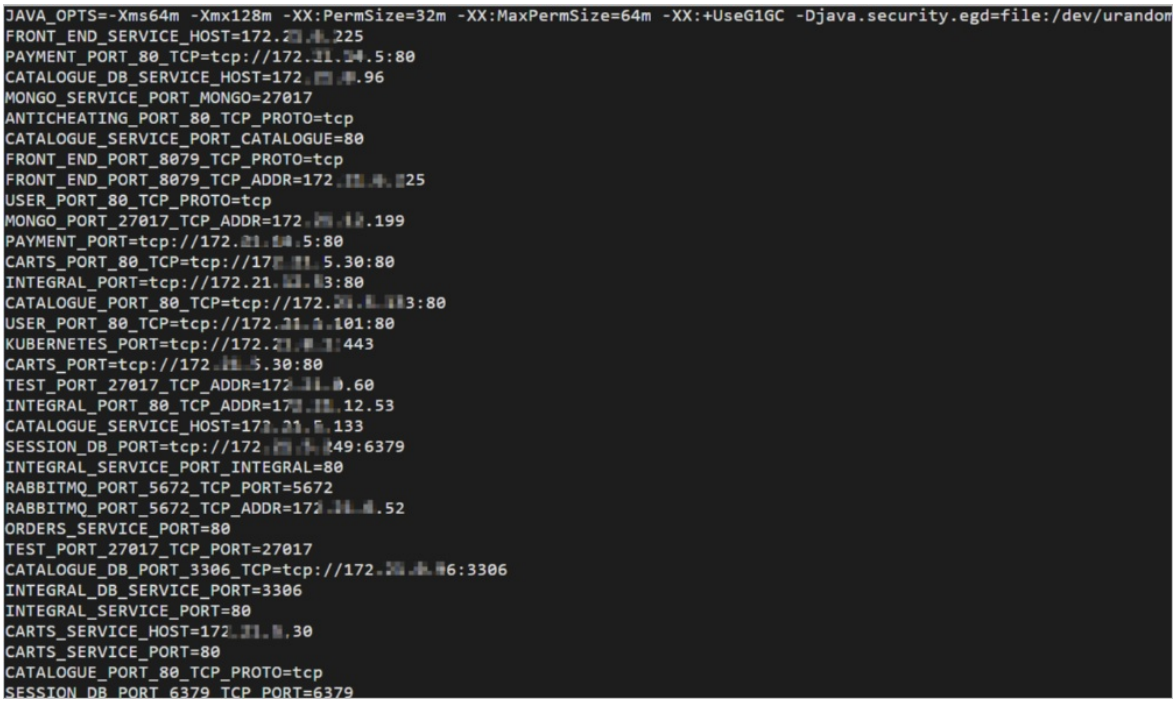
   2ba4ebdaf503 in the response indicates the ID of the container.



3. Run the following command to obtain the environment variables of the container.

   The *2ba4ebdaf503* variable in the command is the ID of a container. Replace the value of the variable with an actual ID.

   ```
   docker exec 2ba4ebdaf503 env
   ```



11.2. Log search and analysis

11.2.1. FAQ about log query

This topic provides answers to some frequently asked questions (FAQ) about log query in Log Service.

How do I identify the source server from which Logtail collects logs during a query?

If a machine group uses IP addresses as its identifiers when logs are collected by using Logtail, the servers in the machine group are distinguished by internal IP addresses. When you query logs, you can use the hostname and custom IP address to identify the source server from which logs are collected.

For example, you can use the following query statement to calculate the number of occurrences of each hostname.

> **Note** You must configure an index for the __tag__:__hostname__ field and enable the analysis feature.

```
* | select "__tag__:__hostname__" , count(1) as count group by "__tag__:__hostname__"
```

How do I query IP addresses in logs?

You can use the exact match method to query IP addresses in logs. You can search for log data by IP address. For example, you can specify whether to include or exclude an IP address. However, you cannot use the partial match method to query the IP addresses in logs. This is because decimal points contained in an IP address are not default delimiters in Log Service. You can also filter data by using other methods. For example, you can use an SDK to download data and then use a regular expression or the string.indexof() method to search for results.

For example, if you execute the following query statement, the logs that contain the 203.0.113 CIDR block are still returned.

```
not ip:203.0.113 not status:200 not 360jk not DNSPod-Monitor not status:302 not jiankongbao
        not 301 and status:403
```

How do I query log data by using a keyword that contains a space character?

If you use a keyword that contains a space character to query log data, log data that contains a part of the keyword on the left or right of the space character is returned. You can enclose the keyword that contains a space character in double quotation marks (""). The entire enclosed content is regarded as a keyword to query log data as expected.

For example, you want to query log data that contains the keyword `POS version` from the following log data:

```
post():351]: device_id: BTAddr : B6:xF:xx:65:xx:A1 IMEI : 35847xx22
xx81x9 WifiAddr : 4c:xx:0e:xx:4e:xx | user_id: bb07263xxd2axx43xx9exxea2
6e39e5f POS version:903
```

If you use `POS version` as the keyword, log data that contains `POS` and `version` is returned. This query result does not meet your requirements. If you use `"POS version"` as the keyword, log data that contains the keyword `POS version` is returned.

How do I use two conditions to query log data?

You can specify two conditions in a query statement to query log data.

For example, if you want to query logs in which the value of the status field is not 200 and the value of the request_method field is not GET in a Logstore, you can execute the `not status:200 not request_method:GET` statement to query logs as expected.

How do I query collected logs in Log Service?

You can use one of the following methods to query logs in Log Service:

- Use the Log Service console.
- Use an SDK.
- Use the Restful API.

11.2.2. What can I do if I cannot obtain the required results from a log query?

If you cannot find the required log data by using the query feature of Log Service, perform the steps that are described in this topic to troubleshoot the issue.

Log collection failure

If log data fails to be collected by Log Service, you cannot query the log data. Check whether log data is available on the consumption preview page of your Logstore.

If log data is available, log data is collected by Log Service.

If log data is unavailable, check whether the issue occurs due to the following causes:

- The log source does not generate log data.

  If no log data is generated by the log source, no log data can be sent to Log Service. Check your log source.

- Logtail has no heartbeat.

  On the **Machine Group Settings** page, check whether the heartbeat status of the related server is OK in the Machine Group Status section. For more information about how to troubleshoot the issue if Log Service does not receive heartbeats from a Logtail client, see What can I do if no heartbeat packet is received from a Logtail client?.

- Data is not written to the file that is monitored in real time.

If data is not written to the file that is monitored in real time, you can view error messages in the */usr/local/ilogtail/ilogtail.LOG* file. Common error messages:

- parse delimiter log fail: The error message returned because an error occurs when Log Service collects logs in delimiter mode.
- parse regex log fail: The error message returned because an error occurs when Log Service collects logs in full regex mode.

Delimiter setting errors

View the specified delimiters and check whether you can use a keyword to find a log after the log content is split by using the delimiters. In this example, the default delimiters `,;=()[]{}?@&<>/:'` are used. If a log contains abc"defg,hij, the log is split into the following two words: abc"defg and hij. If the log is split, you cannot use the keyword abc to find the log.

Fuzzy match is supported. For more information, see Query syntax.

> Note
>
> - The indexing feature of Log Service is optimized. If you configure both full-text indexes and field indexes, the configurations of the field indexes take precedence. This helps you reduce indexing costs. For example, you configure an index for a log field whose key is message and specify a space character as a delimiter. To use a space character as a delimiter, you must specify the space character in the middle of the delimiters that you specify for an index. You can find a log that contains "message: this is a test message" by using the keyword message:this in the key:value format. However, you cannot find the log by using the keyword this because you have configured a field index for the key and the full-text indexing feature does not take effect for the log field.
> - You can create indexes or modify existing indexes. However, new or modified indexes take effect only for new data.
>
>   You can click Index Attributes to check whether the specified delimiters meet your business requirements.

Other reasons

If log data is generated, modify the query time range and perform a query operation again. Log Service allows you to preview log data in real time. The maximum latency of the query feature is 1 minute. We recommend that you query log data at least 1 minute after logs are generated.

If the issue persists, submit a ticket.

11.2.3. What are the differences between log consumption and log query?

Log Service provides the log consumption and log query features that allow you to read log data from Log Service.

Log consumption

The log consumption feature allows you to read and write full data in the first-in, first-out (FIFO) order. This feature is similar to the features provided by Kafka.

- Each Logstore has one or more shards. Data is written to a random shard.
- You can read multiple logs at a time from a specified shard based on the order in which the logs were written to the shard.
- You can specify a start position (cursor) to pull logs from shards based on the time when Log Service receives the logs.

Log query

Log Service allows you to query and analyze a large amount of log data based on specific conditions.

- You can specify query conditions to find required log data.
- You can use multiple operators such as AND, NOT, and OR to specify query conditions and perform SQL analysis on query results.
- The log query feature is independent of shards.

Differences

Item	Log query	Log consumption
Search by keyword	Supported.	Not supported.
Data read (a small amount of data)	Fast.	Fast.
Data read (full data)	Slow. Log Service reads 100 logs in 100 milliseconds. We recommend that you do not use this method.	Fast. Log Service reads 1 MB of log data in 10 milliseconds. We recommend that you use this method.
Data read by topic	Yes.	No. Data is identified only by shard.
Data read by shard	No. Data in all shards of a Logstore is queried.	Yes. You need to specify a shard each time to read data.
Fee	Medium.	Low.
Scenario	Monitoring, issue troubleshooting, and analysis.	Full data processing scenarios, such as stream computing and batch processing.

11.2.4. How do I resolve common errors that occur when I query log data?

This topic describes the common error messages that are returned when you query log data in the Log Service console and provides related solutions.

Error messages

Error message	Cause	Solution
line 1:44: Column 'XXX' cannot be resolved;please add the column in the index attribute	The XXX key cannot be specified in the query statement because the key does not exist.	Click Index Attributes to configure an index for the field. For more information, see Enable the indexing feature and configure indexes for a Logstore.
ErrorType:QueryParseError.ErrorMessage:syntax error error position is from column:10 to column:11,error near < : >	The query statement contains unnecessary colons (:).	Delete the unnecessary colons (:) from the query statement, and then execute the query statement.

Error message	Cause	Solution
Column 'XXX' not in GROUP BY clause;please add the column in the index attribute	You use a GROUP BY clause and specify a non-GROUP BY field in a SELECT statement. For example, you do not specify the key1 field in the `select key1, avg(latency) group by key2` statement in the GROUP BY clause.	You must specify the same field that you specified in the SELECT statement in the GROUP BY clause. Example: `* | select key1,avg(latency) group by key1,key2`
sql query must follow search query,please read syntax doc	The syntax of the query statement is invalid because a search statement is not specified.	Invalid query statement: `select ip,count(*) group by ip`. Valid query statement: `*|select ip,count(*) group by ip`.
line 1:10: identifiers must not start with a digit; surround the identifier with double quotes	The column name or variable name that is referenced in an SQL statement cannot start with a digit.	Change the column name or variable name to a name that starts with a letter.
line 1:9: extraneous input '' expecting	One or more words are misspelled.	Correct the misspelled words.
key (XXX) is not config as key value config,if symbol : is in your log,please wrap : with quotation mark "	The XXX field cannot be referenced in the analytic statement because no field index is configured for the field.	Click Index Attributes to configure an index for the field. For more information, see Enable the indexing feature and configure indexes for a Logstore.
Query exceeded max memory size of 3GB	The size of the memory that is used by the query statement exceeds 3 GB. The issue occurs because a large number of values are returned in the query result after you use a `GROUP BY` clause to remove duplicates.	Optimize the `GROUP BY` clause. Reduce the number of keys that is specified in the `GROUP BY` clause.
ErrorType:ColumnNotExists.ErrorPosition,line:0,column:1.ErrorMessage:line 1:123: Column 'XXX' cannot be resolved; it seems XXX is wrapper by ";if XXX is a string ,not a key field, please use 'XXX'	XXX is not an indexed field. In an SQL statement, you must enclose an indexed field in double quotation marks ("") and you must enclose a string in single quotation marks ('').	If you want to reference the XXX field, make sure that you index the field and enable the analysis feature for the field. For more information, see Enable the indexing feature and configure indexes for a Logstore. **Note** If XXX is a string, you must replace the double quotation marks (") with single quotation marks (').
user can only run 15 query concurrently	More than 15 concurrent search statements are executed. A maximum of 15 concurrent search statements can be executed by a user in a project.	Reduce the number of search statements based on your business requirements.
unclosed string quote	The double quotation marks (") in the query statement are incomplete.	Check the query statement, specify double quotation marks (") in pairs, and then execute the query statement.

Error message	Cause	Solution
error after :.error detail:error after :.error detail:line 1:147: mismatched input 'in' expecting {<EOF>, 'GROUP', 'ORDER', 'HAVING', 'LIMIT', 'OR', 'AND', 'UNION', 'EXCEPT', 'INTERSECT'}	The syntax of the query statement is valid.	Modify the SQL statement as prompted and execute the SQL statement.
Duplicate keys (XXX) are notallowed	Indexes are not case-sensitive. For example, if the aBc index exists, an error message is returned when you create the abc index.	Check whether duplicate indexes exist.
only support * or ? in the middle or end of the query	You can specify only asterisks (*) and question marks (?) in the middle or end of a field value to perform a fuzzy match.	You can use the SQL LIKE operator in a query statement based on your business requirements. For example, you cannot use `Msg: *xxx` to search for logs that contains the Msg field and the field value ends with XXX. You can use the SQL LIKE operator to perform a fuzzy match, as shown in the following query statement: `Msg: * | SELECT Msg WHERE Msg LIKE '%xxx'`
logstore (xxx) is not found	The XXX Logstore does not exist or the analysis feature is not enabled.	Check whether the Logstore exists. If the Logstore exists, you must index at least one field and enable the analysis feature for the Logstore.
condition number 43 is more than 30	A maximum of 30 fields can be referenced by a user in a query statement.	Modify the query statement to reduce the number of the referenced fields, and then execute the query statement.

11.2.5. Why data queries are inaccurate?

This topic describes the causes for inaccurate data queries. It also includes solutions to these issues.

When you search and analyze log data, the message **The results are inaccurate** may prompt in the console. This indicates that the returned result is inaccurate because some log data in a Logstore was not queried.

Possible causes include:

The time range for queries is excessive.

- Cause

  The time range for a query is excessively wide, for example, three months or a year. In this case, Log Service cannot scan all log data generated within this time period for one query.

- Solution

  Narrow down the query time range and perform multiple queries.

Query statements are complex.

- Cause

The query statement is exceedingly complex or contains multiple frequently used words. In this case, Log Service cannot scan all related log data or read the query results at one time.

- Solution

  Narrow down the query scope and perform multiple queries.

The SQL computing needs to read an excessively large amount of data.

- Cause

  The SQL computing needs to read an excessively large amount of data. In this case, query results are likely to become inaccurate. A maximum of 1 GB of data can be read from each shard. For example, if the SQL computing needs to read strings from multiple columns, which exceed the threshold data volume, inaccurate query results will be returned.

- Solution

  Narrow down the query scope and perform multiple queries.

11.3. Alarm

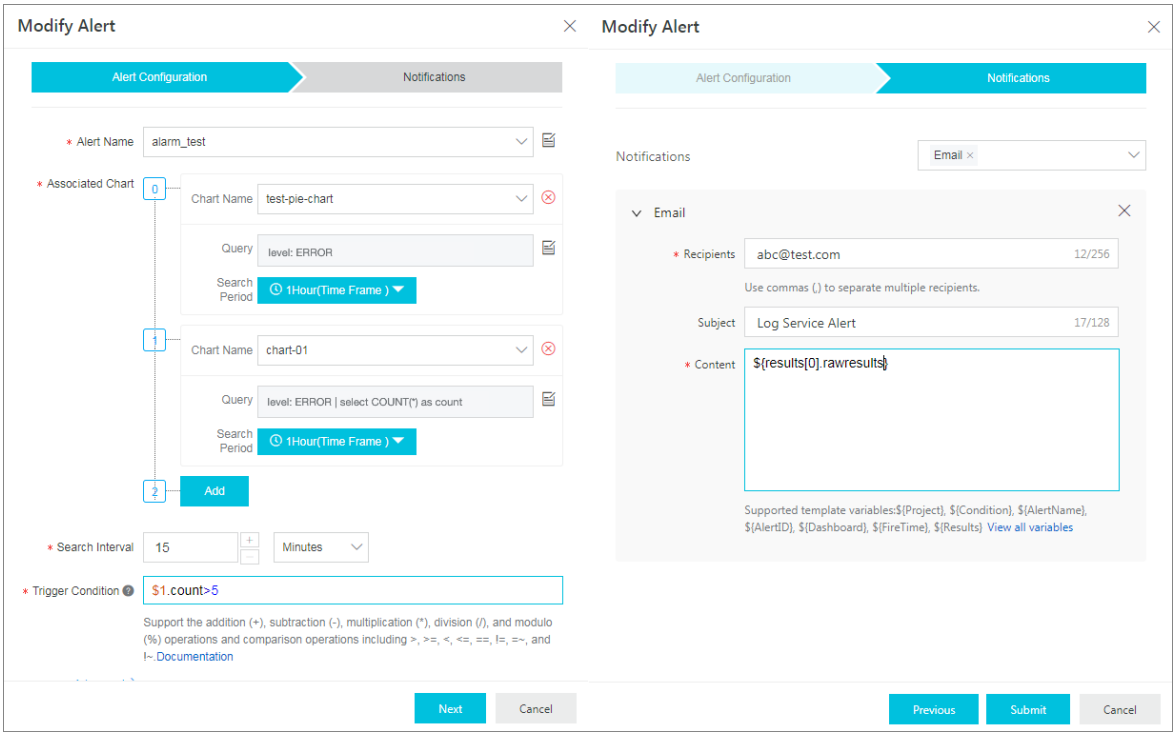
11.3.1. FAQ about alerts

This topic provides answers to some frequently asked questions (FAQ) about alerts in the Log Service console.

How do I add raw logs to an alert notification?

For example, if more than five error logs are detected in the previous 5 minutes, an alert is triggered and an alert notification is sent. To add the raw logs to the alert notification, perform the following steps:

1. Log on to the Log Service console.
2. In the Projects section, click the project that you want to manage.
3. In the left-side navigation pane, click the icon.
4. In the Dashboard list, click the dashboard for which you want to configure alert rules.
5. In the upper-right corner of the dashboard page, choose **Alerts > Create**.
6. In the **Create Alert** panel, set the parameters. The following figure shows the parameter settings.



The following example shows how to set the parameters:

- Query
  - Association Chart 0: **level:ERROR**
  - Association Chart 1: **level: ERROR | select COUNT(*) as count**
- Trigger Condition: **$1.count > 5**
- Content: **${results[0].rawresults}**

7. Click **Submit**.

What can I do if the DingTalk chatbot fails to send a notification?

For example, after you set the notification method to WebHook-DingTalk Bot, the following error message is returned when the DingTalk chatbot sends a notification:

```
{"errcode":310000,"errmsg":"sign not match"}
{"errcode":310000,"errmsg":"keywords not in content"}
```

This error message is returned because the security settings of the latest chatbot are invalid. You can reconfigure the security settings. For more information, see DingTalk chatbot webhooks. If the issue persists or other error messages are returned, submit a ticket.

11.4. What do I do if the Forbidden.SLS::ListProject error occurs when I log on to the Apsara Uni-manager Management Console?

Error description

If you are redirected to the Log Service console when you use a RAM user to log on to the Apsara Uni-manager Management Console, the Forbidden.SLS::ListProject error may occur.

Cause

The Log Service-related role that the RAM user assumes is not granted the permissions to access Log Service. Therefore, you cannot use the RAM user to access Log Service.

Solution

Procedure

1. Log on to the Apsara Uni-manager Management Console.
2. In the top navigation bar, click **Enterprise**.
3. In the left-side navigation pane, choose **Permissions > Role Permissions**. In the search box, enter SLS to view the role.
4. Click **Modify** in the Actions column.
5. On the details page, click the **Application Permissions** tab.
6. Select **View resources** in the Log Service section and click **Update**.
7. Use the RAM user to log on to the Apsara Uni-manager Management Console again.