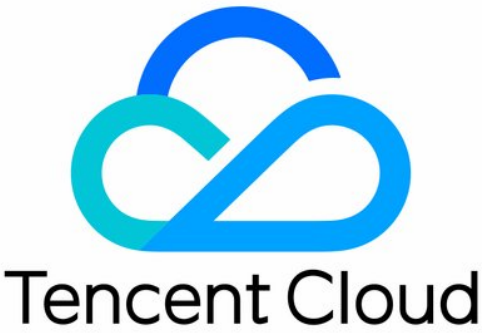
Cloud Log Service

Operation Guide

Product Documentation



Contents

Operation Guide
Resource Management
Logset

Last updated：2024-01-20 16:42:10

Overview

CLS manages the log data stored on it in the form of log topics and logsets. A log topic is the basic unit for log data collection, storage, search, and analysis, and a logset is used to categorize log topics for easier management.
This document describes how to manage logsets in the console. We recommend you read Log Topic and Logset first to learn more about concepts and use cases.

Prerequisites

You have logged in to the CLS console.

Directions

Creating logset

1. Click **Log Topic** on the left sidebar.
2. On the log topic management page, select the region, and click **Manage Logset**.
3. On the logset management page on the right, click **Create Logset**.
4. In the **Create Logset** pop-up window, enter the logset name and tag.
5. Click **OK**.

Viewing logset

1. Click **Log Topic** on the left sidebar.
2. On the log topic management page, select the region, and click **Manage Logset**.
3. Then you can view the logsets in the selected region on the right.

Editing logset

1. Click **Log Topic** on the left sidebar.
2. On the log topic management page, select the region, and click **Manage Logset**.

3. On the logset management page on the right, find the target logset ID/name and click **Edit**.
4. In the **Edit Logset** pop-up window, rename the logset.
5. Click **OK**.

Deleting logset

1. Click **Log Topic** on the left sidebar.
2. On the log topic management page, select the region, and click **Manage Logset**.
3. On the logset management page on the right, find the target logset ID/name and click **Delete**.

Note:

A logset can be deleted only when **no log topics are under it**. Otherwise, it cannot be deleted.

4. In the pop-up window, click **OK**.

Log Topic

Last updated：2024-01-20 16:42:10

Overview

CLS manages the log data stored on it in the form of log topics and logsets. A log topic is the basic unit for log data collection, storage, search, and analysis, and a logset is used to categorize log topics for easier management.
This document describes how to manage log topics in the console. We recommend you read Log Topic and Logset first to learn more about concepts and use cases.

Prerequisites

You have logged in to the CLS console.

Directions

Creating a log topic

1. Click **Log Topic** on the left sidebar.
2. On the log topic management page, select a region and click **Create Log Topic**.
3. In the pop-up window, enter the following information:
Log Topic Name: For example, enter `nginx` .
Storage Class: STANDARD by default. For more information, see Storage Class Overview.
Log Retention Period: 30 days by default. You can specify a log storage period, after which logs are automatically cleared. If LogListener is used, and a time field in the log content instead of the collection time is used as the logging time, the time specified by the time field will be used to determine whether a log expires.
Logset Operation:
Select an existing logset: select a target logset from the **Logset** drop-down list.
Create logset:
Logset Name: for example, enter `cls_test` .
Log Topic Tag: set a tag for the current log topic to be created so that resources can be managed by category from different dimensions.
Advanced settings:
Partitions: Enter a positive integer. It defaults to `1` . For more information on topic partitions, see Topic Partition.
Partition Auto-Split: Enabled by default.

Maximum Partitions: Enter an integer up to 50.

Logset Tag: this field is available only when **Logset Operation** is **Create logset**. You can set a tag for the logset to be created.

4. Click **OK**.

After the log topic is created, you can click its ID/name on the **Log Topic** page to view its details.

Note:

You're advised to select the same region as CVM or any other Tencent Cloud service from which you collect logs.

After the log topic is created, its region and logset cannot be modified.

Logs can be retained for 1 to 3600 days or permanently.

Editing log topic

1. Click **Log Topic** on the left sidebar.
2. On the log topic management page, find the target log topic ID/name and click **Edit**.
3. In the pop-up window, modify the basic information of the log topic.
4. Click **OK**.

Deleting a log topic

1. Click **Log Topic** on the left sidebar.

2 On the log topic management page, find the target log topic ID/name, and click **Delete**.

Note:

You can switch to different regions in the top-left corner of the page to delete log topics accordingly. To delete all log topics, we recommend you use an account with permissions of all resources; otherwise, some log topics may not be deleted as you don't have permissions to view them.

2. In the pop-up window, click **OK**.

Note:

Once a log topic is deleted, you cannot recover its topic configuration and log data. Proceed with caution.

Metric topic

Last updated：2024-09-20 17:48:27

Overview

CLS manages its stored metric data by using Metric Topics and Logsets. A Metric Topic is the basic unit for collecting, storing, and querying metric data, while a Logset categorizes metric topics to help users manage them more easily. A Logset can contain both metric topics and log topics simultaneously. This document describes how to manage metric topics.

Note:

The metric topics service ended its public beta on July 1, 2024, and is now officially a paid service. For more details, see the Billing Overview.

Prerequisites

You have logged in to the CLS Console.

Directions

Creating a Metric Topic

1. In the left sidebar, click **Metric Topic**.
2. On the Metric Topic management page, select the region for the Metric Topic, and click **Create Metric topic**.
3. In the pop-up window, fill in the following information:

Metric topic name: for example, nginx_metric.

Metric topicID: a random ID is generated by default; you can also define an ID. Only lowercase letters, numbers, and - are supported, and it cannot start or end with -. Length 3 to 40 characters.

Metric save policy:

Retention Time: It is 30 days by default. You can customize the data retention period. Data will be automatically deleted after expiration, or it can be set to be retained permanently.

Logset Operation:

Select an existing logset: select a target logset from the Logset dropdown list.

Create logset: Enter the name of the new logset, and set a tag for the current logset to be created so that resources can be managed by category from different dimensions.

Metric topic tag: set a tag for the newly created Metric Topic to facilitate resource classification management from different dimensions.
Advanced Settings:
Metric topic description: description information to understand the purpose of the Metric Topic in the future.
Topic partition:
Initial partitions: the number of partitions automatically created when the topic is created, supporting 1~10, default is 1. For more information on topic partitions, see Topic Partition.
Auto-split: whether to automatically add partitions based on the topic's write traffic size, enabled by default.
Maximum partitions: When auto split is enabled, the maximum number of partitions is 50 by default.
4. Click **OK** to create the metric topic.
After the metric topic is successfully added, you can go to the metric topic management page, click the created metric topic name/ID to view detailed information.

Note:

Once a metric topic is created, the associated region and logset cannot be modified.

Editing a Metric Topic

1. In the left sidebar, click **Metric Topic**.
2. In the metric topic management page, find the metric topic name/ID you want to edit and click **Edit**.
3. In the pop-up window, modify the basic information of the metric topic.
4. Click **OK** to update the metric topic information.

Deleting a Metric Topic

1. In the left sidebar, click **Metric Topic**.
2. In the metric topic management page, find the metric topic name/ID you want to delete and click **Delete**.
3. In the pop-up window, click **Next > Delete** to delete the current metric topic.

Note:

Once a metric topic is deleted, the associated configurations and data cannot be recovered. Operate with caution.

Topic Usage Monitoring

Last updated：2024-05-31 14:41:27

Traffic

Metric	Meaning	Unit	Dimension
Write traffic	Traffic incurred during log upload	MB	Log topic ID
Index traffic	Traffic incurred after the index feature is enabled	MB	Log topic ID
Private network read traffic	Private network read traffic incurred when a log is downloaded, consumed, or shipped via a private network	MB	Log topic ID
Public network read traffic	Public network read traffic incurred when a log is downloaded, consumed, or shipped via a public network	MB	Log topic ID
Read traffic	Total private and public network read traffic	MB	Log topic ID

Storage capacity

Metric	Meaning	Unit	Dimension
Log storage capacity	Storage capacity occupied by log data	MB	Log topic ID
Index storage capacity	Storage capacity occupied by index data	MB	Log topic ID
Storage Capacity	Total storage capacity occupied by log and index data	MB	Log topic ID

Service request quantity

Metric	Meaning	Unit	Dimension
Service requests	Number of requests that users use LogListener, API, or SDK to call CLS APIs, including read and write requests such as upload and download, creation and deletion, and search and analysis	COUNT	Log topic ID

Note:

The preceding monitoring metrics are reported to Cloud Monitor. For the corresponding detailed metrics and parameters, please see the CLS monitoring metrics.

Machine Group Management

Last updated：2024-01-20 16:42:10

Overview

A machine group is a server object configured to collect logs by LogListener of CLS. You can configure machine groups through the CLS console, and categorize them by use cases.

Directions

Creating a machine group by machine group IP

1. Log in to the CLS console.
2. Click **Machine Group** in the left sidebar.
3. Select the region of your CLS, such as Shanghai, and then click **Create Machine Group**.
4. Configure the following information in the pop-up:
Logset Name: enter `cls_test` for example.
Configuration Mode: select **Configure machine group IP**.
IP: enter IPs of machines.

Note:

Enter one IP per line. Do not enter IPs of Windows machines.
If you are using Tencent Cloud machines in the same region, you can enter private IPs directly, with one IP per line.
If you are using machine groups of different regions, enter public IPs.
LogListener Auto Upgrade: disclosed by default and can be configured.
LogListener Service Logs: enabled by default. This feature records logs of the running status and log collection among other aspects of LogListener. For details, see LogListener Service Logs.
5. Click **OK**.

Creating a machine group by machine ID

If your machine IPs change frequently, you need to modify machine group configuration once the IPs change.
To avoid the trouble, you can use CLS to configure machine groups dynamically by machine ID. You just need to enter the machine IDs when configuring LogListener, CLS will recognize and add these machines into the machine group automatically.

Note:

Configuring machine groups by machine ID is supported only on LogListener v2.3.0 and above. You need to upgrade lower versions manually to use this feature.

1. Log in to the CLS console.
2. Click **Machine Group** in the left sidebar.
3. Select the region of your CLS, such as Guangzhou, and then click **Create Machine Group**.
4. In the pop-up, configure the following items:

Logset Name: enter `cls_test` for example

Configuration Mode: select **Configure machine ID**

Machine ID: enter machine IDs.

Note:

Enter one machine ID per line, and do not enter Windows machine IDs.

LogListener Auto Upgrade: you're advised to enable this feature. For more information, see LogListener Upgrade Guide.

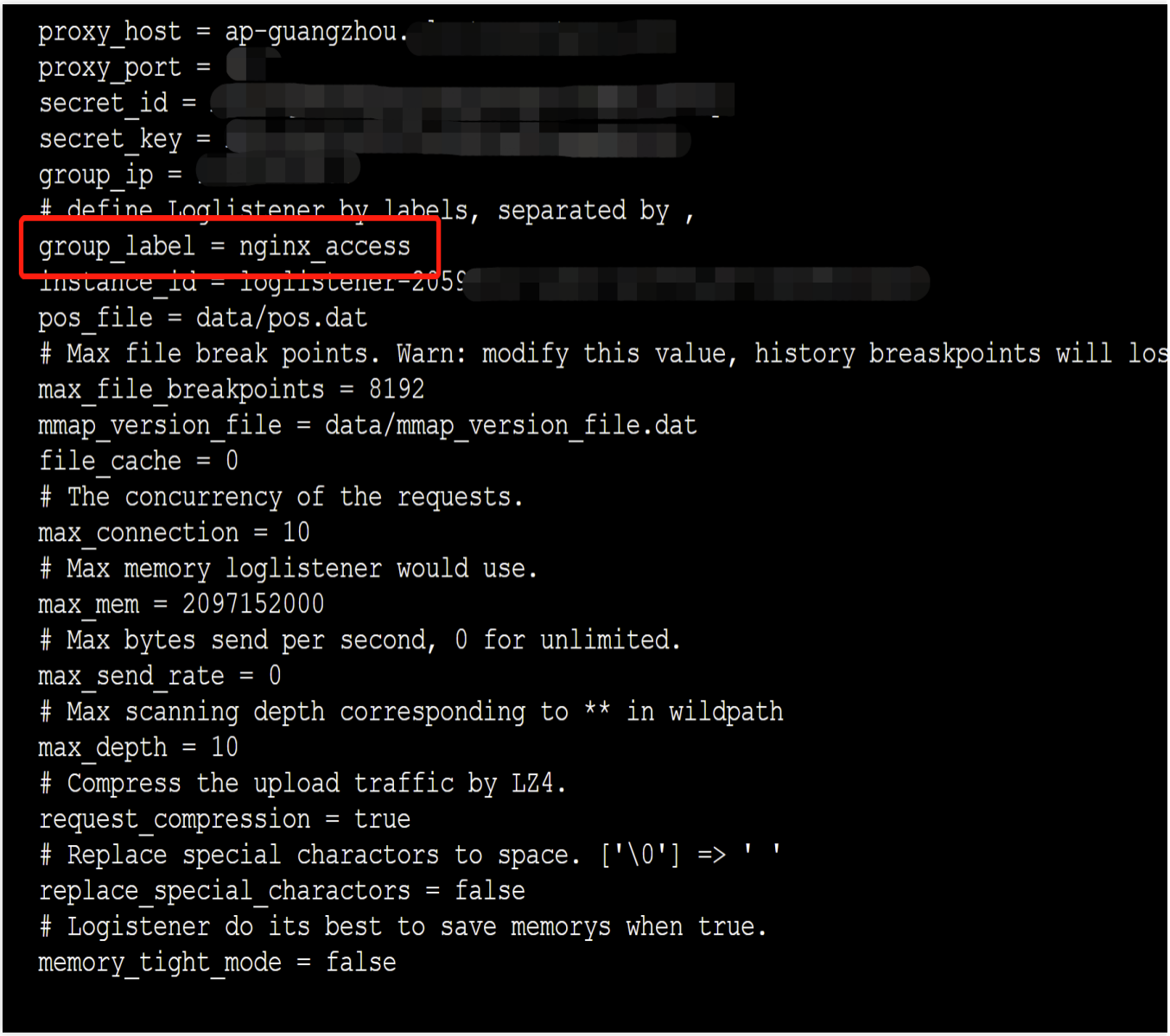
LogListener Service Logs: enabled by default. This feature records logs of the running status and log collection among other aspects of LogListener. For details, see LogListener Service Logs.

5. Click **OK**.
6. Log in to a machine added above, run the following command, open the `/etc/loglistener.conf` file under the installation directory of LogListener.

Take the `/user/local` installation directory as an example:

```
vi /usr/local/loglistener-2.3.0/etc/loglistener.conf
```

7. Press **i** to enter the edit mode.
8. Find the `group_label` parameter, and enter your custom machine IDs and separate them with a comma (,).



9. Press **Esc** to exit the edit mode.

10. Enter **:wq**, and press **Enter** to save the settings.

11. Run the following command, restart LogListener, and then the machine group will be created.

```
/etc/init.d/loglistenerd restart
```

Viewing Machine Status

A machine group uses the heartbeat mechanism to maintain its connection with the CLS system. A machine group with LogListener installed regularly sends heartbeats to CLS.

1. Log in to the CLS console.

2. Click **Machine Group** in the left sidebar.

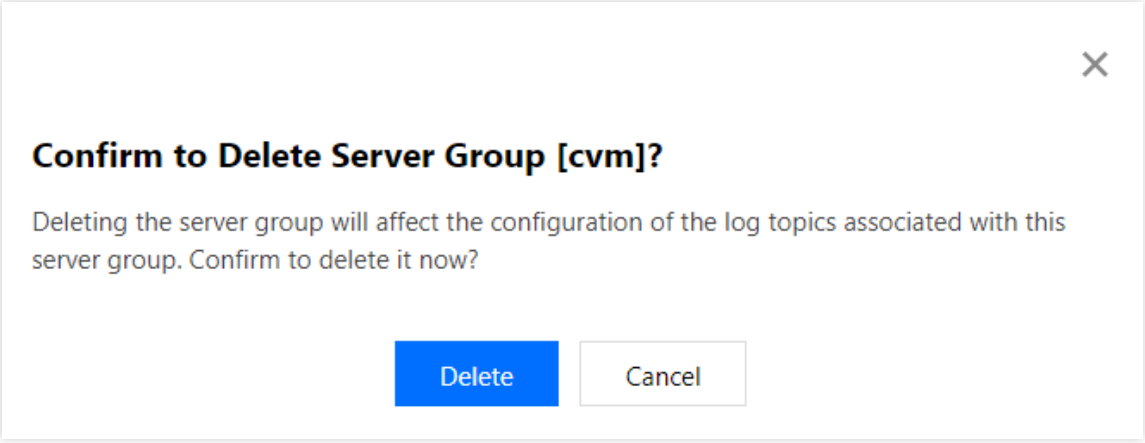
3. Find the target machine group, click **View**.

4. View the machine group status in the pop-up.

You can view the machine group status to see if it works normally. If so, your server can communicate with CLS normally.

Deleting a machine group

1. Log in to the CLS console.
2. Click **Machine Group** in the left sidebar.
3. Find the target machine group, click **Delete**.
4. In the pop-up, click **OK**.



Note:

Once the machine group is deleted, logs will no longer be collected under its associated log topics.

Permission Management
Sub-Account Authorization

Last updated：2024-01-20 16:56:41

Overview

CAM is a web-based Tencent Cloud service that helps you securely manage and control access to your Tencent Cloud resources. Using CAM, you can create, manage, and terminate users (user groups), and control who can access and use your Tencent Cloud resources through identity and policy management. For more information on CAM policies and how to use them, see Concepts.
A root account can grant a sub-account or collaborator access to specified CLS resources.

Preset access policies

CLS offers two preset access policies to meet your basic access management demand.
QcloudCLSFullAccess: access to all CLS resources and actions, including creating log topics, modifying index configuration, deleting log topics, searching for logs, uploading logs, etc.
QcloudCLSReadOnlyAccess: only read access to CLS data; no CRUD access
For how to use the policies, see Authorization Management.

Custom access policies

You can use a custom access policy to grant access at a finer granularity, for example, to allow a specific user to view the data of a specific log topic.
A custom access policy consists of two parts:
Action: The action a user is allowed to perform, such as searching for logs, modifying index configuration, uploading logs, and creating alarm policies.
Resource: The resources a user is allowed to operate on, such as a specific log topic, dashboard, and data processing task.
For more information on authorizable resource types and APIs of CLS, see Authorizable Resource Types. For more information on configuration methods, see Creating Custom Policy.
Configuring custom access policies can be a demanding process. The examples we offer in Access Policy Templates should meet most access management needs. You can also modify the examples based on your requirements.
Detailed directions are as follows:

1. Log in to the console with the root account (or an account with CAM access). On the Policies page, click **Create Custom Policy**.
2. In the pop-up window, click **Create by Policy Syntax**.
3. On the **Select Policy Template** page, select **Blank Template** and click **Next**.
4. On the **Edit Policy** page, enter a policy name and policy content. For the latter, you can copy the content from Access Policy Templates. For example, to grant the sub-account permission to use LogListener, copy the policy as shown below:
5. Click **Complete** to save the policy. Then, you can associate it with a user/user group to grant the user/user group the corresponding operation permissions as instructed in Authorization Management.

Authorizable Resource Types

Last updated：2024-12-03 14:57:20

Overview

CLS provides various types of resources. Some of its APIs allow you to configure user permissions based on resources. See here for examples.
The following table lists the types of resources that can be authorized in Cloud Access Management (CAM). Note that authorization by tag indicates whether a tag can be used to specify the range of resources on which users have operation permissions.

Resource Type	Resource Description Method in Access Policies	Authorization by Tag
Logset	`qcs::cls:$region:$account:logset/*` `qcs::cls:$region:$account:logset/$logsetId`	Supported
Log topic	`qcs::cls:$region:$account:topic/*` `qcs::cls:$region:$account:topic/$topicId`	Supported
Machine group	`qcs::cvm:$region:$account:machinegroup/*` `qcs::cvm:$region:$account:machinegroup/$machinegroupId`	Supported
Collection configuration	`qcs::cls:$region:$account:config/*` `qcs::cls:$region:$account:config/$configId`	Not supported
Dashboard	`qcs::cls:$region:$account:dashboard/*` `qcs::cls:$region:$account:dashboard/$dashboardId`	Supported
Alarm policy	`qcs::cls:$region:$account:alarm/*` `qcs::cls:$region:$account:alarm/$alarmId`	Not supported
Notification channel group	`qcs::cls:$region:$account:alarmNotice/*` `qcs::cls:$region:$account:alarmNotice/$alarmNoticeId`	Not supported
Data processing task	`qcs::cls:$region:uin/$account:datatransform/*` `qcs::cls:$region:uin/$account:datatransform/$TaskId`	Not supported
Shipping task (COS)	`qcs::cls:$region:$account:shipper/*` `qcs::cls:$region:$account:shipper/$shipperId`	Not supported

Other resource types (disused; used by APIs of earlier versions only)	Single chart in the dashboard:
	`qcs::cls:$region:$account:chart/*`
	`qcs::cls:$region:$account:chart/$chartId`
		Not supported

You need to change the variable parameters such as `$region` and `$account` to your actual parameter information.

For all the APIs supported by CLS and their resource description methods, see here. APIs adopting the authorization granularity of resource support user permission configuration by using the methods of the corresponding resource types described above. For APIs adopting the authorization granularity of API, the corresponding resource range in a CAM permission policy must be `*` .

Practice

Different types of resources in CLS are associated with each other. For example, log sets contain log topics, and log topics must apply collection configuration to machine groups. Directly configuring user permissions in CAM permission policies according to resource IDs results in difficult management and is likely to cause the error where users do not have permissions on some APIs. Therefore, you are advised to configure CAM permission policies as follows:

For resource types and corresponding APIs that support authorization by tag, bind related resources with tags and use tags to specify the ranges of resources on which users have operation permissions. For example, you can bind log topics, logsets, and related dashboards with tags so that you can assign management permissions on log topics with specified tags or assign management permissions on log topics and dashboards with specified tags. In either way, you can enable users to have the operation permissions on the APIs of the three types of resources.

For resource types and corresponding APIs that do not support authorization by tag, to simplify management, you can directly set the resource range in the CAM permission policy to `*` , indicating all resources. To avoid misoperations by ordinary users, you can configure read-only permissions for ordinary users and management permissions for admins. For example, you can assign admins the management permissions on all data processing tasks and assign ordinary users the read-only permissions on all data processing tasks.

Note:

For more use cases, see Examples of Custom Access Policies.

Access Policy Templates

Last updated：2024-12-17 20:53:57

For custom permission policies, the following permission policy templates can be used based on the scenario.

Module	Application Scenario
Overall Operation (Best Practices)	Classify topics, machine groups, and dashboards by using tags, and configure permissions by tag: Management permission for resources with specified tags Read-only permission for resources with specified tags
Data collection	Server data collection by using LogListener Self-built Kubernetes data collection by using LogListener Data upload by using APIs/SDKs Data upload by using Kafka Data upload through cloud product metric subscription Subscription to MySQL binlog data Subscription to Kafka data Fluent Bit log upload Logstash log upload Managing collection configurations and machine groups
Topic management and search/analysis	Viewing/Managing Topics and Performing Search/Analysis Management permission: operations on all topics Management permission: operations on specified topics Management permission: operations on topics with specified tags Read-only permission: operations on all topics Read-only permission: operations on specified topics Read-only permission: operations on topics with specified tags Using APIs to Perform Search and Analysis Read-only permission: search and analysis on all topics Read-only permission: search and analysis on specified topics Read-only permission: search and analysis on topics with specified tags
Dashboard	Management permission: operations on all dashboards Management permission: operations on dashboards with specified tags Management permission: operations on specified dashboards Read-only permission: operations on all dashboards Read-only permission: operations on dashboards with specified tags Read-only permission: operations on specified dashboards
Monitoring alarm	Management permission: operations on all alarm policies Management permission: operations on alarm policies with specified tags Management permission: operations on specified alarm policies

	Read-only permission: operations on all alarm policies Read-only permission: operations on alarm policies with specified tags Read-only permission: operations on specified alarm policies
Data Processing	Data Processing Management permission: operations on all data processing tasks Read-only permission: operations on all data processing tasks Performing Scheduled SQL Analysis Management permission: scheduled SQL analysis on all log topics Management permission: scheduled SQL analysis on log topics with specified tags
Data shipping and consumption	Shipping to CKafka Management permission: shipping all log topics to CKafka Management permission: shipping log topics with specified tags to CKafka Read-only permission: shipping all log topics to CKafka Read-only permission: shipping log topics with specified tags to CKafka Shipping to COS Management permission: shipping all log topics to COS Management permission: shipping log topics with specified tags to COS Read-only permission: shipping all log topics to COS Read-only permission: shipping log topics with specified tags to COS Shipping to SCF Management permission: shipping all log topics to SCF Management permission: shipping log topics with specified tags to SCF Read-only permission: shipping all log topics to SCF Read-only permission: shipping log topics with specified tags to SCF Kafka Protocol Consumption Management permission: consuming all log topics over Kafka protocol Management permission: consuming log topics with specific tags over Kafka protocol Management permission: consuming specific resources over Kafka protocol Minimum permission for consumption over Kafka protocol (not for console but for API calls) Shipping Metric Topics Management permission: shipping all metric topics Management permission: shipping metric topics with specific tags Custom Consumption Management permission: custom consumption of all metric topics
Independent DataSight console	Manage DataSight consoles: Management permission: operations on all independent DataSight consoles Management permission: operations on specific independent DataSight consoles

	Management permission: operations on independent DataSight consoles with specific tags Read-only permission: operations on all independent DataSight consoles Read-only permission: operations on specific independent DataSight consoles Read-only permission: operations on independent DataSight consoles with specific tags
Developer	Using CLS Through Grafana Displaying data of all topics through Grafana Displaying data of topics with specified tags through Grafana

Overall operation (best practices)

Users can classify topics, machine groups, and dashboards by using tags and configure permissions by tag. Tags are required for resources during resource creation. Users have management or read-only permissions only for resources with specified tags, which helps them manage various types of resources in CLS in batches.

Management Permission for Resources with Specified Tags

Note:

Delete comments to use this policy.

```
{
        "statement": [{
                        "action": [ //Required read-only permission for related pro
                                "monitor:GetMonitorData",
                "monitor:DescribeBaseMetrics",
                                "cam:ListGroups",
                                "cam:GetGroup",
                                "cam:DescribeSubAccountContacts",
                                "cam:ListAttachedRolePolicies",
                "cam:GetRole",
                "vpc:DescribeSubnetEx",//Required for creating DataSight consoles a
                "vpc:DescribeVpcEx",//Required for creating DataSight consoles acce
                                "tag:TagResources",
                                "tag:DescribeResourceTagsByResourceIds",
                                "tag:GetTags",
                                "tag:GetTagKeys",
                                "tag:GetTagValues",
                "kms:GetServiceStatus"
                        ],
                        "effect": "allow",
                        "resource": "*"
                },
```

```
                {
                        "action": [ //Specify that tags such as testCAM:test1 are r
                "cls:CreateDashboard",
                "cls:CreateLogset",
                "cls:CreateTopic",
                "cls:CreateAlarm",
                "cls:CreateAlarmNotice",
                "cls:CreateMachineGroup",
                "cls:CreateConsole"
                        ],
                        "condition": {
                                "for_any_value:string_equal": {
                                        "qcs:request_tag": [
                                                "testCAM&test1"
                                        ]
                                }
                        },
                        "effect": "allow",
                        "resource": "*"
                },
                {
                        "action": [ //Grant permission on all related APIs if tags
                                "cls:*"
                        ],
                        "condition": {
                                "for_any_value:string_equal": {
                                        "qcs:resource_tag": [
                                                "testCAM&test1"
                                        ]
                                }
                        },
                        "effect": "allow",
                        "resource": "*"
                },
                {
                        "action": [ //Some APIs do not support permission control b
                                "cls:CheckAlarmChannel",
                                "cls:CheckAlarmRule",
                                "cls:CheckDomainRepeat",
                                "cls:CheckFunction",
                                "cls:CheckRechargeKafkaServer",
                                "cls:DescribeClsPrePayDetails",
                                "cls:DescribeClsPrePayInfos",
                                "cls:DescribeConfigMachineGroups",
                                "cls:DescribeConfigs",
                                "cls:DescribeAgentConfigs",
                                "cls:DescribeTopicExtendConfig",
```

```
                        "cls:DescribeDataTransformFailLogInfo",
                        "cls:DescribeDataTransformInfo",
                        "cls:DescribeDataTransformPreviewDataInfo",
                        "cls:DescribeDataTransformPreviewInfo",
                        "cls:DescribeDataTransformProcessInfo",
                        "cls:DescribeDemonstrations",
                        "cls:DescribeExceptionResources",
                        "cls:DescribeExternalDataSourcePreview",
                        "cls:DescribeFunctions",
                        "cls:DescribeResources",
                        "cls:DescribeShipperPreview",
                        "cls:DescribeScheduledSqlProcessInfo",
                        "cls:DescribeConfigurationTemplates",
                        "cls:DescribeFolders",
                        "cls:GetClsService",
                        "cls:GetConfigurationTemplateApplyLog",
                        "cls:PreviewKafkaRecharge",
                        "cls:agentHeartBeat",
                        "cls:CreateDemonstrations",
                        "cls:DeleteDemonstrations",
                "cls:DescribeNoticeContents",
                "cls:DescribeWebCallbacks"
                        ],
                        "effect": "allow",
                        "resource": "*"
                },
                {
                        "action": [ //Some APIs do not support permission control b
                                "cls:RealtimeProducer", //Upload data by using Kafk
                                "cls:CreateConfigurationTemplate", //Configuration
                                "cls:ModifyConfigurationTemplate",
                                "cls:DeleteConfigurationTemplate",
                                "cls:CreateFolder", //Folder API
                                "cls:ModifyFolder",
                                "cls:DeleteFolder",
                                "cls:ModifyResourceAndFolderRelation",
                                "cls:CreateDataTransform",//Data processing API
                                "cls:ModifyDataTransform",
                                "cls:DeleteDataTransform",
                                "cls:RetryShipperTask",//COS shipping API
                                "cls:ModifyDashboardSubscribeAck",//Dashboard subsc
                                "cls:DeleteDashboardSubscribe",
                                "cls:ModifyConfigExtra",//Collection configuration
                                "cls:DeleteConfigExtra",
                                "cls:RemoveMachine",//Machine group API
                                "cls:UpgradeAgentNormal",
                "cls:CreateNoticeContent",//API related to alarm notification templ
```

```
                "cls:DeleteNoticeContent",
                "cls:ModifyNoticeContent",
                "cls:CreateWebCallback",//API related to alarm integration configur
                "cls:ModifyWebCallback",
                "cls:DeleteWebCallback"
                        ],
                        "effect": "allow",
                        "resource": "*"
                }
        ],
        "version": "2.0"
}
```

Read-Only Permission for Resources with Specified Tags

Note:

Delete comments to use this policy.

```
{
        "statement": [{
                        "action": [ //Required read-only permission for related pro
                                "monitor:GetMonitorData",
                "monitor:DescribeBaseMetrics",
                                "cam:ListGroups",
                                "cam:GetGroup",
                                "cam:DescribeSubAccountContacts",
                                "cam:ListAttachedRolePolicies",
                                "tag:DescribeResourceTagsByResourceIds",
                                "tag:GetTags",
                                "tag:GetTagKeys",
                                "tag:GetTagValues"
                        ],
                        "effect": "allow",
                        "resource": "*"
                },
                {
                        "action": [ //Grant read-only permission on related APIs if
                                "cls:DescribeConsumer",
                                "cls:DescribeConsumerPreview",
                                "cls:DescribeCosRecharges",
                                "cls:DescribeDashboardSubscribes",
                                "cls:DescribeDashboards",
                                "cls:DescribeExports",
                                "cls:DescribeIndex",
                                "cls:DescribeIndexs",
                                "cls:DescribeKafkaConsume",
```

```
                        "cls:DescribeKafkaConsumer",
                        "cls:DescribeKafkaRecharges",
                        "cls:DescribeLatestJsonLog",
                        "cls:DescribeLatestUserLog",
                        "cls:DescribeLogContext",
                        "cls:DescribeLogFastAnalysis",
                        "cls:DescribeLogHistogram",
                        "cls:DescribeMachineGroupConfigs",
                        "cls:DescribeMachines",
                        "cls:DescribePartitions",
                        "cls:DescribeScheduledSqlInfo",
                        "cls:DescribeScheduledSqlProcessInfo",
                        "cls:DescribeShipperPreview",
                        "cls:DescribeTopics",
                        "cls:EstimateRebuildIndexTask",
                        "cls:GetAlarm",
                        "cls:GetAlarmLog",
                        "cls:GetMetricLabelValues",
                        "cls:GetMetricSeries",
                        "cls:MetricsLabelValues",
                        "cls:MetricsLabels",
                        "cls:MetricsQuery",
                        "cls:MetricsQueryExemplars",
                        "cls:MetricsQueryRange",
                        "cls:MetricsSeries",
                        "cls:QueryMetric",
                        "cls:QueryRangeMetric",
                        "cls:SearchCosRechargeInfo",
                        "cls:SearchDashboardSubscribe",
                        "cls:SearchLog",
                        "cls:DescribeAlarmNotices",
                        "cls:DescribeAlarms",
                        "cls:DescribeAlertRecordHistory",
                        "cls:DescribeExternalDataSources",
                        "cls:DescribeLogsets",
                        "cls:DescribeMachineGroups",
                "cls:DescribeConsoles",
                        "cls:DescribeShipperTasks",
                        "cls:DescribeShippers",
                        "cls:DescribeRebuildIndexTasks"
                ],
                "condition": {
                        "for_any_value:string_equal": {
                                "qcs:resource_tag": [
                                        "testCAM&test1"
                                ]
                        }
```

```
                        },
                        "effect": "allow",
                        "resource": "*"
                },
                {
                        "action": [ //Some APIs do not support permission control b
                                "cls:CheckAlarmChannel",
                                "cls:CheckAlarmRule",
                                "cls:CheckDomainRepeat",
                                "cls:CheckFunction",
                                "cls:CheckRechargeKafkaServer",
                                "cls:DescribeClsPrePayDetails",
                                "cls:DescribeClsPrePayInfos",
                                "cls:DescribeConfigMachineGroups",
                                "cls:DescribeConfigs",
                                "cls:DescribeAgentConfigs",
                                "cls:DescribeTopicExtendConfig",
                                "cls:DescribeDataTransformFailLogInfo",
                                "cls:DescribeDataTransformInfo",
                                "cls:DescribeDataTransformPreviewDataInfo",
                                "cls:DescribeDataTransformPreviewInfo",
                                "cls:DescribeDataTransformProcessInfo",
                                "cls:DescribeDemonstrations",
                                "cls:DescribeExceptionResources",
                                "cls:DescribeExternalDataSourcePreview",
                                "cls:DescribeFunctions",
                                "cls:DescribeResources",
                                "cls:DescribeShipperPreview",
                                "cls:DescribeScheduledSqlProcessInfo",
                                "cls:DescribeConfigurationTemplates",
                                "cls:DescribeFolders",
                                "cls:GetClsService",
                                "cls:GetConfigurationTemplateApplyLog",
                                "cls:PreviewKafkaRecharge",
                                "cls:CreateDemonstrations",
                                "cls:DeleteDemonstrations",
                                "cls:CreateExport",
                                "cls:DeleteExport"
                "cls:DescribeNoticeContents",
                "cls:DescribeWebCallbacks"
                        ],
                        "effect": "allow",
                        "resource": "*"
                }
        ],
        "version": "2.0"
}
```

Data Collection

Server Data Collection by Using LogListener

Users can use LogListener on Agent to collect and upload log data. (The sample code below demonstrates the minimum permission for data upload by using LogListener installed on Agent.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
            "cls:getConfig",
            "cls:agentHeartBeat"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Note:

If the LogListener version is earlier than 2.6.5, add cls:listLogset to the code.

Self-built Kubernetes Data Collection by Using LogListener

Users can use Logagent to collect and upload log data from self-built Kubernetes clusters. (The sample code below demonstrates the minimum permission for data upload from a self-built Kubernetes cluster.)

```
{
  "version": "2.0",
  "statement": [
    {
      "action": [
        "cls:pushLog",
        "cls:agentHeartBeat",
        "cls:getConfig",
        "cls:CreateConfig",
        "cls:DeleteConfig",
        "cls:ModifyConfig",
        "cls:DescribeConfigs",
        "cls:DescribeMachineGroupConfigs",
        "cls:DeleteConfigFromMachineGroup",
        "cls:ApplyConfigToMachineGroup",
        "cls:DescribeConfigMachineGroups",
```

```
            "cls:ModifyTopic",
            "cls:DeleteTopic",
            "cls:CreateTopic",
            "cls:DescribeTopics",
            "cls:CreateLogset",
            "cls:DeleteLogset",
            "cls:DescribeLogsets",
            "cls:CreateIndex",
            "cls:ModifyIndex",
            "cls:CreateMachineGroup",
            "cls:DeleteMachineGroup",
            "cls:DescribeMachineGroups",
            "cls:ModifyMachineGroup",
            "cls:CreateConfigExtra",
            "cls:DeleteConfigExtra",
            "cls:ModifyConfigExtra"
        ],
        "resource": "*",
        "effect": "allow"
    }
  ]
}
```

Data Upload by Using APIs/SDKs

Users can use APIs/SDKs to upload data to CLS. (The sample code below demonstrates the minimum permission for data upload by using APIs/SDKs.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
            "cls:UploadLog",
            "cls:MetricsRemoteWrite"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Data Upload by Using Kafka

Users can upload log data to CLS over Kafka protocol. (The sample code below demonstrates the minimum permission for data upload over Kafka protocol.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:RealtimeProducer"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Data Upload Through Cloud Product Metric Subscription

Users can upload metric data to CLS through cloud product metric subscription. (The sample code below demonstrates the minimum permission for data upload through cloud product metric subscription.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:CreateMetricSubscribe",
            "cls:DescribeMetricCorrectDimension",
            "cls:DescribeMetricSubscribePreview",
            "monitor:DescribeBaseMetrics",
            "monitor:DescribeProductList"
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Subscription to MySQL Binlog Data

Users can upload MySQL binlog data to CLS through subscription. (The sample code below demonstrates the minimum permission for MySQL binlog data upload through subscription.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:CreateBinlogSubscribe",
            "cls:DescribeBinlogSubscribes",
            "cls:ModifyBinlogSubscribe",
            "cls:DescribeBinlogSubscribeConnectivity",
            "cls:DescribeBinlogSubscribePreview",
```

```
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Subscription to Kafka Data

Users can upload Kafka cluster data to CLS through subscription. (The sample code below demonstrates the minimum permission for Kafka cluster data upload through subscription.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:PreviewKafkaRecharge",
            "cls:CreateKafkaRecharge",
            "cls:ModifyKafkaRecharge",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

FluentBit Log Uploading

Users can upload Fluent Bit data to CLS by using Fluent Bit plugins in Go. (The sample code below demonstrates the minimum permission for data upload by using Fluent Bit plugins in Go.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Logstash Log Upload

Users can upload Logstash data to CLS by using Logstash plugins. (The sample code below demonstrates the minimum permission for data upload by using Logstash plugins.)

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:pushLog",
        ],
        "resource": "*",
        "effect": "allow"
    }]
}
```

Managing Collection Configurations and Machine Groups

Related operations include creation, modification, and deletion of collection configurations and machine groups.
Config-related APIs correspond to resources related to collection configurations.
MachineGroup-related APIs correspond to resources related to machine groups.
The three ConfigExtra-related APIs are used to manage the cluster configuration for uploading self-built Kubernetes cluster data. They can be ignored if no self-built Kubernetes cluster data is uploaded.

```
{
    "version": "2.0",
    "statement": [{
        "action": [
            "cls:DescribeLogsets",
            "cls:DescribeTopics",
            "cls:CreateConfig",
            "cls:CreateConfig",
            "cls:DeleteConfig",
            "cls:DescribeConfigs",
            "cls:ModifyConfig",
            "cls:CreateConfigExtra",
            "cls:DeleteConfigExtra",
            "cls:ModifyConfigExtra",
            "cls:CreateMachineGroup",
            "cls:DeleteMachineGroup",
            "cls:DescribeMachineGroups",
            "cls:DeleteConfigFromMachineGroup",
            "cls:ApplyConfigToMachineGroup",
            "cls:ModifyMachineGroup"
        ],
        "resource": "*",
        "effect": "allow"
    }
    ]
}
```

Topic Management and Search/Analysis

View/manage topics and perform search/analysis:

Management Permission: Operations on All Topics

Users can search for and manage all topics. Related operations include topic creation, topic deletion, and index configuration modification but exclude collection configuration, log shipping, and log processing.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateLogset",
                "cls:CreateTopic",
                "cls:CreateExport",
                "cls:CreateIndex",
                "cls:DeleteLogset",
                "cls:DeleteTopic",
                "cls:DeleteExport",
                "cls:DeleteIndex",
                "cls:ModifyLogset",
                "cls:ModifyTopic",
                "cls:ModifyIndex",
                "cls:MergePartition",
                "cls:SplitPartition",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:CreateRebuildIndexTask",
                "cls:EstimateRebuildIndexTask",
                "cls:CancelRebuildIndexTask",
                                "cls:MetricsLabelValues",
```

```
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Management Permission: Operations on Specified Topics

Users can search for and manage specific topics. Related operations include topic creation, topic deletion, and index configuration modification but exclude collection configuration, log shipping, and log processing.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateLogset",
                "cls:CreateTopic",
                "cls:CreateExport",
                "cls:CreateIndex",
                "cls:DeleteLogset",
                "cls:DeleteTopic",
                "cls:DeleteExport",
                "cls:DeleteIndex",
                "cls:ModifyLogset",
                "cls:ModifyTopic",
                "cls:ModifyIndex",
                "cls:MergePartition",
                "cls:SplitPartition",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
```

```
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:CreateRebuildIndexTask",
                "cls:EstimateRebuildIndexTask",
                "cls:CancelRebuildIndexTask",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007*827:logset/1c012db7-2cfd-4418-
**-7342c7a42516",
                "qcs::cls:ap-guangzhou:100007*827:topic/380fe1f1-0c7b-4b0d-**-
d514959db1bb"
            ]
        }
    ]
}
```

Management Permission: Operations on Topics with Specified Tags

Users can search for and manage topics with specific tags. Related operations include topic creation, topic deletion, and index configuration modification but exclude collection configuration, log shipping, and log processing. Tags are required for both topics and their logsets.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateLogset",
                "cls:CreateTopic",
                "cls:CreateExport",
                "cls:CreateIndex",
```

```
                "cls:DeleteLogset",
                "cls:DeleteTopic",
                "cls:DeleteExport",
                "cls:DeleteIndex",
                "cls:ModifyLogset",
                "cls:ModifyTopic",
                "cls:ModifyIndex",
                "cls:MergePartition",
                "cls:SplitPartition",
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                "cls:CreateRebuildIndexTask",
                "cls:EstimateRebuildIndexTask",
                "cls:CancelRebuildIndexTask",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            }
        }
    ]
```

```
}
```

Read-Only Permission: Operations on All Topics

Users can search for all topics.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Read-Only Permission: Operations on Specified Topics

Users can search for specified topics.

```
{
```

```
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007*827:logset/1c012db7-2cfd-4418-
**-7342c7a42516",
                "qcs::cls:ap-guangzhou:100007*827:topic/380fe1f1-0c7b-4b0d-**-
d514959db1bb"
            ]
        }
    ]
}
```

Read-Only Permission: Operations on Topics with Specified Tags

Users can search for topics with specified tags.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
```

```
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeExports",
                "cls:DescribeIndex",
                "cls:DescribeIndexs",
                "cls:DescribePartitions",
                "cls:SearchLog",
                "cls:DescribeLogHistogram",
                "cls:DescribeLogContext",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeLatestJsonLog",
                "cls:DescribeRebuildIndexTasks",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            }
        }
    ]
}
```

Use APIs to perform search and analysis:

Read-Only Permission: Search and Analysis on All Topics

Users can perform search and analysis on all topics by using APIs.

```
{
    "version": "2.0",
    "statement": [
```

```
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries",
                        "cls:MetricsRemoteRead"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Read-Only Permission: Search and Analysis on Specified Topics

Users can perform search and analysis on specified topics by using APIs.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries",
                        "cls:MetricsRemoteRead"
            ],
            "resource": [
```

```
                "qcs::cls:ap-guangzhou:100007*827:logset/1c012db7-2cfd-4418-
**-7342c7a42516",
                "qcs::cls:ap-guangzhou:100007*827:topic/380fe1f1-0c7b-4b0d-**-
d514959db1bb"
            ]
        }
    ]
}
```

Read-Only Permission: Search and Analysis on Topics with Specified Tags

Users can perform search and analysis on topics with specified tags by using APIs.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                                "cls:MetricsLabelValues",
                                "cls:MetricsLabels",
                                "cls:MetricsQuery",
                                "cls:MetricsQueryRange",
                                "cls:MetricsSeries",
                                "cls:MetricsQueryExemplars",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries",
                        "cls:MetricsRemoteRead"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "testCAM&test1"
                    ]
                }
            }
        }
    ]
}
```

Dashboard

Management Permission: Operations on All Dashboards

Users can manage **all** dashboards. Related operations include creation, deletion, editing, viewing, and subscription. Dashboards can use data of **all** topics.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:CreateChart",
                "cls:CreateDashboard",
                "cls:DeleteChart",
                "cls:DeleteDashboard",
                "cls:ModifyChart",
                "cls:ModifyDashboard",
                "cls:DescribeDashboards",
                "cls:CreateFolder",
                "cls:DeleteFolder",
                "cls:DescribeFolders",
                "cls:ModifyFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:SearchDashboardSubscribe",
                "cls:CreateDashboardSubscribe",
                "cls:ModifyDashboardSubscribe",
                "cls:DescribeDashboardSubscribes",
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyDashboardSubscribeAck"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                                "cls:GetMetricLabelValues",
```

```
                "cls:QueryMetric",
                "cls:QueryRangeMetric",
                "cls:GetMetricSeries"
            ],
            "resource": "*"
        }
    ]
}
```

Management Permission: Operations on Dashboards with Specified Tags

Users can manage dashboards **with specified tags**. Related operations include creation, deletion, editing, viewing, and subscription. Dashboards can use data of topics **with specified tags**.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:CreateChart",
                "cls:CreateDashboard",
                "cls:DeleteChart",
                "cls:DeleteDashboard",
                "cls:ModifyChart",
                "cls:ModifyDashboard",
                "cls:DescribeDashboards",
                "cls:CreateFolder",
                "cls:DeleteFolder",
                "cls:DescribeFolders",
                "cls:ModifyFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:SearchDashboardSubscribe",
                "cls:CreateDashboardSubscribe",
                "cls:ModifyDashboardSubscribe",
                "cls:DescribeDashboardSubscribes",
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyDashboardSubscribeAck"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
```

```
                        ]
                    }
                }
            },
            {
                "effect": "allow",
                "action": [
                    "cls:SearchLog",
                    "cls:DescribeTopics",
                    "cls:DescribeLogFastAnalysis",
                    "cls:DescribeIndex",
                    "cls:DescribeLogsets",
                                    "cls:GetMetricLabelValues",
                                    "cls:QueryMetric",
                                    "cls:QueryRangeMetric",
                                    "cls:GetMetricSeries"
                ],
                "resource": "*",
                "condition": {
                    "for_any_value:string_equal": {
                        "qcs:resource_tag": [
                            "key&value"
                        ]
                    }
                }
            }
        ]
    }
```

Management Permission: Operations on Specified Dashboards

Users can manage **specified dashboards**. Related operations include creation, deletion, editing, viewing, and subscription. Dashboards can use data of **specified topics**.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:CreateChart",
                "cls:CreateDashboard",
                "cls:DeleteChart",
                "cls:DeleteDashboard",
```

```
                "cls:ModifyChart",
                "cls:ModifyDashboard",
                "cls:DescribeDashboards",
                "cls:CreateFolder",
                "cls:DeleteFolder",
                "cls:DescribeFolders",
                "cls:ModifyFolder",
                "cls:ModifyResourceAndFolderRelation",
                "cls:SearchDashboardSubscribe",
                "cls:CreateDashboardSubscribe",
                "cls:ModifyDashboardSubscribe",
                "cls:DescribeDashboardSubscribes",
                "cls:DeleteDashboardSubscribe",
                "cls:ModifyDashboardSubscribeAck"
            ],
            "resource": [
                "qcs::cls::uin/100000*001:dashboard/dashboard-0769a3ba-2514-
409d-**-f65b20b23736"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls::uin/100000*001:topic/174ca473-50d0-4fdf-
**-2ef681a1e02a"
            ]
        }
    ]
}
```

Read-Only Permission: Operations on All Dashboards

Users can view **all** dashboards, and the dashboards can use data of **all** topics.

```
{
    "version": "2.0",
```

```
    "statement": [
         {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:DescribeDashboards",
                "cls:DescribeFolders",
                "cls:SearchDashboardSubscribe",
                "cls:DescribeDashboardSubscribes"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Operations on Dashboards with Specified Tags

Users can view dashboards with **specified tags**, and the dashboards can use data of topics with **specified tags**.

```
{
    "version": "2.0",
    "statement": [
         {
            "effect": "allow",
            "action": [
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:DescribeDashboards",
                "cls:DescribeFolders",
```

```
                "cls:SearchDashboardSubscribe",
                "cls:DescribeDashboardSubscribes"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

Read-Only Permission: Operations on Specified Dashboards

Users can view **specified dashboards**, and the dashboards can use data of **specified topics**.

```
{
    "version": "2.0",
    "statement": [
         {
            "effect": "allow",
            "action": [
```

```
                "cls:GetChart",
                "cls:GetDashboard",
                "cls:ListChart",
                "cls:DescribeDashboards",
                "cls:DescribeFolders",
                "cls:SearchDashboardSubscribe",
                "cls:DescribeDashboardSubscribes"
            ],
            "resource": [
                "qcs::cls::uin/100000*001:dashboard/dashboard-0769a3ba-2514-
409d-**-f65b20b23736"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:DescribeTopics",
                "cls:DescribeLogFastAnalysis",
                "cls:DescribeIndex",
                "cls:DescribeLogsets",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "qcs::cls::uin/100000*001:topic/174ca473-50d0-4fdf-
**-2ef681a1e02a"
            ]
        }
    ]
}
```

Monitoring and Alarm

Management Permission: Operations on All Alarm Policies

Users can manage all alarm policies. Related operations include creating alarm policies, creating notification channel groups, and viewing alarm policies.

```
{
    "version": "2.0",
    "statement": [
        {
```

```
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:CreateAlarm",
                "cls:ModifyAlarm",
                "cls:DeleteAlarm",
                "cls:DescribeAlarmNotices",
                "cls:CreateAlarmNotice",
                "cls:ModifyAlarmNotice",
                "cls:DeleteAlarmNotice",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory",
                "cls:CheckAlarmRule",
                "cls:CheckAlarmChannel"
            ],
            "resource": "*"
        }
    ]
}
```

Management Permission: Operations on Alarm Policies with Specified Tags

Users can manage alarm policies with specified tags. Related operations include modifying alarm policies, modifying notification channel groups, and viewing alarm policies.

```
{
    "version": "2.0",
    "statement": [
        {
```

```
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup",
                "cls:CheckAlarmRule",
                "cls:CheckAlarmChannel",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:ModifyAlarm",
                "cls:DeleteAlarm",
                "cls:DescribeAlarmNotices",
                "cls:ModifyAlarmNotice",
                "cls:DeleteAlarmNotice",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

Management Permission: Operations on Specified Alarm Policies

Users can manage specified alarm policies. Related operations include modifying alarm policies, modifying notification channel groups, and viewing alarm policies.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup",
                "cls:CheckAlarmRule",
                "cls:CheckAlarmChannel",
                                "cls:GetMetricLabelValues",
                                "cls:QueryMetric",
                                "cls:QueryRangeMetric",
                                "cls:GetMetricSeries"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:ModifyAlarm",
                "cls:DeleteAlarm",
                "cls:DescribeAlarmNotices",
                "cls:ModifyAlarmNotice",
                "cls:DeleteAlarmNotice",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:alarm/alarm-xxx-9bbe-4625-ac29-
                "qcs::cls:ap-guangzhou:100007***827:alarmNotice/notice-xxx-ec2c-410
            ]
        }
    ]
}
```

Read-Only Permission: Operations on All Alarm Policies

Users can view all alarm policies.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:DescribeAlarmNotices",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Operations on Alarm Policies with Specified Tags

Users can view alarm policies with specified tags.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
```

```
                "cam:GetGroup"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:DescribeAlarmNotices",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

Read-Only Permission: Operations on Specified Alarm Policies

Users can view specified alarm policies.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cam:ListGroups",
                "cam:DescribeSubAccountContacts",
                "cam:GetGroup"
            ],
            "resource": [
                "*"
            ]
```

```
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeAlarms",
                "cls:DescribeAlarmNotices",
                "cls:GetAlarmLog",
                "cls:DescribeAlertRecordHistory"
            ],
            "resource": [
                "qcs::cls:ap-guangzhou:100007***827:alarm/alarm-xxx-9bbe-4625-ac29-
                "qcs::cls:ap-guangzhou:100007***827:alarmNotice/notice-xxx-ec2c-410
            ]
        }
    ]
}
```

Data Processing

Data Processing

Management Permission: Operations on All Data Processing Tasks

Users can manage data processing tasks of all log topics.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeDataTransformPreviewDataInfo",
                "cls:DescribeTopics",
                "cls:DescribeIndex",
                "cls:CreateDataTransform"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeFunctions",
```

```
                "cls:CheckFunction",
                "cls:DescribeDataTransformFailLogInfo",
                "cls:DescribeDataTransformInfo",
                "cls:DescribeDataTransformPreviewInfo",
                "cls:DescribeDataTransformProcessInfo",
                "cls:DeleteDataTransform",
                "cls:ModifyDataTransform"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Read-Only Permission: Operations on All Data Processing Tasks

Users can view data processing tasks of all log topics. DSL function authorization is not required.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "cls:DescribeDataTransformFailLogInfo",
                "cls:DescribeDataTransformInfo",
                "cls:DescribeDataTransformPreviewDataInfo",
                "cls:DescribeDataTransformPreviewInfo",
                "cls:DescribeDataTransformProcessInfo"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Perform scheduled SQL analysis:

Management permission: Scheduled SQL Analysis on All Log Topics

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:CreateScheduledSql",
                "cls:SearchLog",
                "cls:DescribeScheduledSqlInfo",
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:DeleteScheduledSql",
                "cls:ModifyScheduledSql",
                "cls:RetryScheduledSqlTask"
            ],
            "resource": [
                "*"
            ]
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Management Permission: Scheduled SQL Analysis on Log Topics with Specified Tags

```
{
    "version": "2.0",
    "statement": [
        {
```

```
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:SearchLog",
                "cls:DescribeScheduledSqlProcessInfo",
                "cls:CreateScheduledSql",
                "cls:DeleteScheduledSql",
                "cls:ModifyScheduledSql",
                "cls:RetryScheduledSqlTask"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:DescribeScheduledSqlInfo"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Data Shipping and Consumption

Ship to CKafka:

Management Permission: Shipping All Log Topics to CKafka

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:CreateConsumer",
                "cls:ModifyConsumer",
                "cls:DeleteConsumer",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"

            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

Management Permission: Shipping Log Topics with Specified Tags to CKafka

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
```

```
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:CreateConsumer",
                "cls:ModifyConsumer",
                "cls:DeleteConsumer",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "age&13",
                        "name&vinson"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Shipping All Log Topics to CKafka

Users can perform read-only operations for shipping all log topics to CKafka.

```
{
    "version": "2.0",
    "statement": [
```

```
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Shipping Log Topics with Specified Tags to CKafka

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeConsumer",
                "cls:DescribeConsumerPreview"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
```

```
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "ckafka:DescribeInstances",
                "ckafka:DescribeTopic",
                "ckafka:DescribeInstanceAttributes",
                "ckafka:CreateToken",
                "ckafka:AuthorizeToken"
            ],
            "resource": "*"
        }
    ]
}
```

Ship to COS:

Management Permission: Shipping All Log Topics to COS

Users can perform all operations for shipping all log topics to COS.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeIndex",
                "cls:CreateShipper"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
```

```
                "cls:ModifyShipper",
                "cls:DescribeShippers",
                "cls:DeleteShipper",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cos:GetService",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList"
            ],
            "resource": "*"
        }
    ]
}
```

Management Permission: Shipping Log Topics with Specified Tags to COS

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets",
                "cls:DescribeIndex",
                "cls:CreateShipper"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:ModifyShipper",
```

```
                "cls:DescribeShippers",
                "cls:DeleteShipper",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cos:GetService",
                "cam:ListAttachedRolePolicies",
                "cam:AttachRolePolicy",
                "cam:CreateRole",
                "cam:DescribeRoleList"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Shipping All Log Topics to COS

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets" ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:DescribeShippers",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Shipping Log Topics with Specified Tags to COS

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cls:DescribeShippers",
                "cls:DescribeShipperTasks",
                "cls:RetryShipperTask",
                "cls:DescribeShipperPreview",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": "*"
        }
    ]
}
```

Ship to SCF:

Management Permission: Shipping All Log Topics to SCF

Users can perform all operations for shipping all log topics to SCF.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
```

```
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:CreateDeliverFunction",
                "cls:DeleteDeliverFunction",
                "cls:ModifyDeliverFunction",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

Management Permission: Shipping Log Topics with Specified Tags to SCF

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
```

```
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:CreateDeliverFunction",
                "cls:DeleteDeliverFunction",
                "cls:ModifyDeliverFunction",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Shipping All Log Topics to SCF

Users can perform read-only operations for shipping all log topics to SCF.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*"
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
```

```
            "resource": "*"
        }
    ]
}
```

Read-Only Permission: Shipping Log Topics with Specified Tags to SCF

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeTopics",
                "cls:DescribeLogsets"
            ],
            "resource": "*",
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies",
                "cls:GetDeliverFunction",
                "scf:ListFunctions",
                "scf:ListAliases",
                "scf:ListVersionByFunction"
            ],
            "resource": "*"
        }
    ]
}
```

Kafka Protocol Consumption

Management Permission: Consuming All Log Topics over Kafka Protocol

Users can consume all log topics over Kafka protocol.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeKafkaConsumer",
                "cls:CloseKafkaConsumer",
                "cls:ModifyKafkaConsumer",
                "cls:OpenKafkaConsumer"
            ],
            "resource": [
                "*"]
     },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Management Permission: Consuming Log Topics with Specific Tags over Kafka Protocol

Users can consume log topics with specific tags over Kafka protocol.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeKafkaConsumer",
                "cls:CloseKafkaConsumer",
```

```
                "cls:ModifyKafkaConsumer",
                "cls:OpenKafkaConsumer"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        },
        {
            "effect": "allow",
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Management Permission: Consuming Specific Resources over Kafka Protocol

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeLogsets",
                "cls:DescribeTopics",
                "cls:DescribeKafkaConsumer",
                "cls:CloseKafkaConsumer",
                "cls:ModifyKafkaConsumer",
                "cls:OpenKafkaConsumer"
            ],
            "effect": "allow",
            "resource": [
                "qcs::cls:ap-chengdu:100001127XXX:logset/axxxxxx-772e-4971-
ad9a-ddcfcfff691b",
```

```
                "qcs::cls:ap-chengdu:100001127XXX:topic/590xxxxxxx-36c4-447b-
a84f-172ee7340b22"
            ]
        },
        {
            "action": [
                "tag:DescribeResourceTagsByResourceIds",
                "tag:DescribeTagKeys",
                "tag:DescribeTagValues",
                "cam:ListAttachedRolePolicies"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ],
    "version": "2.0"
}
```

Minimum Permission for Consumption over Kafka Protocol (Not for Console but for API Calls)

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "cls:OpenKafkaConsumer"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ]
}
```

Ship metric topics:

Management Permission: Shipping All Metric Topics

```
{
    "statement": [
        {
            "action": [
```

```
                "cls:DescribeRemoteWriteTask",
                "cls:DescribeTopics",
                "cls:CreateRemoteWriteTask",
                "cls:ModifyRemoteWriteTask",
                "cls:DescribeLogsets",
                "cls:DeleteRemoteWriteTask",
                "cls:CheckRemoteWriteTaskConnect"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ],
    "version": "2.0"
}
```

Management Permission: Shipping Metric Topics with Specific Tags

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:DescribeRemoteWriteTask",
                "cls:DescribeTopics",
                "cls:CreateRemoteWriteTask",
                "cls:ModifyRemoteWriteTask",
                "cls:DescribeLogsets",
                "cls:DeleteRemoteWriteTask",
                "cls:CheckRemoteWriteTaskConnect"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "string_equal": {
                    "qcs:resource_tag": "key:value"
                }
            }
        }
    ]
}
```

Custom Consumption

Management Permission: Custom Consumption of All Metric Topics

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsumerGroup",
                "cls:ModifyConsumerGroup",
                "cls:DescribeConsumerGroups",
                "cls:DeleteConsumerGroup",
                "cls:DescribeConsumerOffsets",
                "cls:CommitConsumerOffsets",
                "cls:SendConsumerHeartbeat",
                "cls:pullLog"
            ],
            "resource": [
                "*"]
     }
    ]
}
```

DataSight Permissions

Management Permission: Operations on All Independent DataSight Consoles

Users can create, modify, view, and delete DataSight consoles in the Tencent Cloud console.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsole",
                "cls:DeleteConsole",
                "cls:DescribeConsoles",
                "vpc:DescribeSubnetEx",
                "vpc:DescribeVpcEx",
                "cls:ModifyConsole"
            ],
            "resource": [
```

```
                "*"
            ]
        }
    ]
}
```

Management Permission: Operations on Specific Independent DataSight Consoles

Users can create, modify, view, and delete specific DataSight consoles in the Tencent Cloud console.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsole",
                "cls:DeleteConsole",
                "cls:DescribeConsoles",
                "vpc:DescribeSubnetEx",
                "vpc:DescribeVpcEx",
                "cls:ModifyConsole"
            ],
            "resource": [
                "qcs::cls::uin/100******123:datasight/clsconsole-1234abcd"
            ]
        }
    ]
}
```

Management Permission: Operations on Independent DataSight Consoles with Specific Tags

Users can create, modify, view, and delete DataSight consoles with specific tags in the Tencent Cloud console.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:CreateConsole",
                "cls:DeleteConsole",
                "cls:DescribeConsoles",
                "vpc:DescribeSubnetEx",
                "vpc:DescribeVpcEx",
                "cls:ModifyConsole"
            ],
```

```
            "resource": [
                "*"
            ],
             "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

Read-Only Permission: Operations on All Independent DataSight Consoles

Users can view relevant information on DataSight consoles in the Tencent Cloud console.

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeConsoles"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ]
        }
    ],
    "version": "2.0"
}
```

Read-Only Permission: Operations on Specific Independent DataSight Consoles

Users can view relevant information on specific DataSight consoles in the Tencent Cloud console.

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeConsoles"
            ],
            "effect": "allow",
            "resource": [
                "qcs::cls::uin/100******123:datasight/clsconsole-1234abcd"
            ]
```

```
        }
    ],
    "version": "2.0"
}
```

Read-Only Permission: Operations on Independent DataSight Consoles with Specific Tags

Users can view relevant information on DataSight consoles with specific tags in the Tencent Cloud console.

```
{
    "statement": [
        {
            "action": [
                "cls:DescribeConsoles"
            ],
            "effect": "allow",
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ],
    "version": "2.0"
}
```

Developer

Use CLS through Grafana:

Displaying Data of All Topics Through Grafana

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
```

```
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:MetricsLabelValues",
                "cls:MetricsQueryRange",
                "cls:MetricsLabels",
                "cls:MetricsQuery"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Displaying Data of Topics with Specified Tags Through Grafana

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:SearchLog",
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:MetricsLabelValues",
                "cls:MetricsQueryRange",
                "cls:MetricsLabels",
                "cls:MetricsQuery"
            ],
            "resource": [
                "*"
            ],
            "condition": {
                "for_any_value:string_equal": {
                    "qcs:resource_tag": [
                        "key&value"
                    ]
                }
            }
        }
    ]
}
```

Log Collection
Collection Overview

Last updated：2024-01-20 17:14:28

Overview

CLS provides log collection clients that allow you to collect your application logs and import them into CLS through APIs or SDKs. Currently, CLS requires logs to be uploaded in a structured manner as key-value pairs.

Log Structuring

The structuring of logs is to store your log data on the CLS platform in key-value format. Structured logs can be searched for, analyzed, and shipped based on specified keys. CLS allows you to report structured data directly. For more information, see the following:
For example, a local raw log is as follows:

```
10.20.20.10;[Tue Jan 22 14:49:45 CST 2019 +0800];GET /online/sample HTTP/1.1;127.0.
```

Specify that the log is parsed by separator and select semicolon (;) as the separator. Then the log can be parsed into multiple field groups and each group is organized in key-value pairs. A key name is defined for each key as follows:

```
IP: 10.20.20.10
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
request: GET /online/sample HTTP/1.1
host: 127.0.0.1
status: 200
length: 647
bytes: 35
referer: http://127.0.0.1/
```

LogListener provides various parsing modes, in which reported logs with full text in a single line and full text in multi lines are both structured. LogListener adds `__CONTENT__` as a key and uses the original text as the value by default. The key-value pair is as follows:

LogListener Parsing Mode	Key	Description
Full text in a single line/Full text in multi	`__CONTENT__`	`__CONTENT__` is the key name by default.

lines		
JSON format	Name in the JSON file	The name and value in the original text of the JSON log are used as a key-value pair.
Separator format	Custom	After dividing fields using separators, you need to define a key name for each field group.
Full regular expression	Custom	After extracting fields based on a regular expression, you need to define a key name for each field group.

Collection Methods

CLS provides multiple methods for data collection:

Collection Method	Description
API	You can call CLS APIs to upload structured logs to CLS. For more information, see Uploading Log via API.
SDK	You can use SDKs to upload structured logs to CLS. For more information, see Collection via SDK.
LogListener client	LogListener is a log collection client provided by CLS. You can quickly access CLS by simply configuring LogListener in the console. For more information, see LogListener Use Process.

A comparison of the collection methods is as follows:

Category	Collection via LogListener	Collection via API
Code modification	Provides a non-intrusive collection method for applications, without code modification.	Reports logs only after modifying application code.
Resumable upload	Supports resumable upload of logs.	Automatically implemented by code.
Retransmission upon failure	Provides an inherent retry mechanism.	Automatically implemented by code.
Local cache	Supports local cache, ensuring data integrity during peak hours.	Automatically implemented by code.
Resource occupation	Occupies resources such as memory and CPU resources.	Occupies no additional resources.

Accessing Log Sources

You can access different log sources in different ways. For more information, see the following tables:

Log source environment

System Environment	Recommended Access Method
Linux/Unix	LogListener collection / Upload over Kafka / Log upload via API
Windows	Beats collection / Upload over Kafka / Log upload via API
iOS/Android/Web	Log upload via SDK

Tencent Cloud service logs

For more information, see Tencent Cloud Service Log Access.

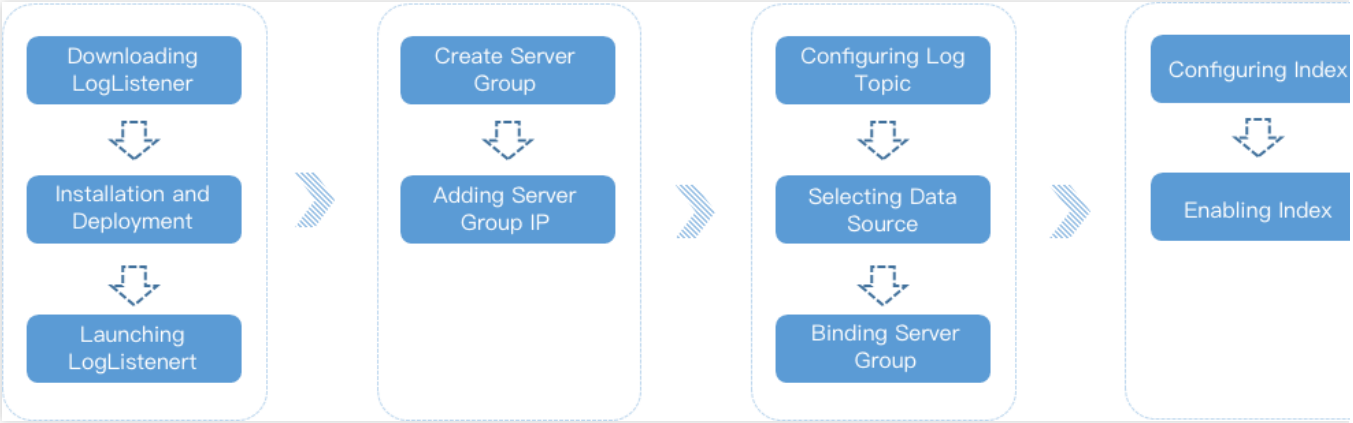
Collection by LogListener

LogListener Use Process

Last updated：2024-01-20 17:14:28

Overview

LogListener is a log collection client provided by CLS. You can install and deploy it to easily and quickly access CLS without modifying the run logic of applications. It is a non-intrusive collection method for application services.
The procedures for collecting logs with LogListener are shown in the following figure:



Procedure Description

1. Download the latest version of LogListener.
2. Install and deploy LogListener on the destination server. Upon successful installation, it automatically launches and maintains a heartbeat connection with the backend of CLS.
3. Go to the CLS Console, create a server group, and add the server IP.
4. Go to the **Collection Configuration** page of the log topic, enter the log path to determine the data source, and bind the server group. For a detailed operation sample, please see the Collection of Full Text in a Single Line document.
5. Go to the **Index Configuration** page of the log topic, configure the full text or key-value index, and enable the index. For a detailed operation sample, please see the Collection of Full Text in a Single Line document.
By now, LogListener monitors the log files that meet the rules according to the collection configuration of the log topics. Users may view the collected log data via log search.

LogListener Installation and Deployment
LogListener Installation Guide(Linux)

Last updated：2024-11-15 16:39:57

LogListener is a log collector provided by CLS. You can install and deploy it on a server to collect logs quickly.

Installation Environment

LogListener supports only Linux 64-bit operating systems and does not support Windows now. It is compatible with mainstream Linux operating system versions. If LogListener is incompatible with the Linux operating system version you use, submit a ticket for assistance.

OS	Compatible Versions
CentOS (64-bit)	CentOS_6.8_64-bit, CentOS_6.9_64-bit, CentOS_7.2_64-bit, CentOS_7.3_64-bit, CentOS_7.4_64-bit, CentOS_7.5_64-bit, CentOS_7.6_64-bit, CentOS_8.0_64-bit
Ubuntu (64-bit)	Ubuntu Server_14.04.1_LTS_64-bit, Ubuntu Server_16.04.1_LTS_64-bit, Ubuntu Server_18.04.1_LTS_64-bit
Debian (64-bit)	Debian_8.2_64-bit, Debian_9.0_64-bit
openSUSE (64-bit)	openSUSE_42.3_64-bit
TencentOS Server	TencentOS Server 3.1, TencentOS Server 2.4

Supported Features

Key features supported by different LogListener versions are as listed below. For more information, see LogListener Updates.

LogListener Version	Supported Feature	Feature Description	Documentation
v2.8.0	GBK encoding for collection; optimization of the	LogListener can collect GBK-encoded log text.	-

	escape character in JSON extraction mode		
v2.7.4	Host name collection (`hostname`)	LogListener collects and reports the machine host name as a default field and displays __HOSTNAME__ as a key, such as __HOSTNAME__:VM-108-centos.	-
v2.6.4	Combined parsing to customize complex log parsing rules	You can use LogListener's combined parsing mode to parse logs. This mode allows you to enter JSON code in the console to customize the log parsing pipeline logic.	Combined Parsing Format
v2.6.0	CVM batch deployment	You can select CVM instances in the console and batch distribute LogListener deployment tasks through an API to automatically complete LogListener installation and deployment (including `accesskey` , ID, and region configuration).	Deploying LogListener on CVMs in Batches
v2.5.4	LogListener service logs	LogListener service logs are used to record the operation, collection, and monitoring activities of LogListener, and you can configure visual graphs to display such log data.	LogListener Service Logs
v2.5.2	Uploading parsing-failed logs	Parsing-failed logs can be uploaded, using `LogParseFailure` as the key name (`Key`) and the raw log content as the key value (`Value`).	-
v2.5.0	LogListener auto-upgrade	Users can set a time period for Agent auto-upgrade or select specific machine groups to upgrade manually.	LogListener Upgrade Guide
v2.4.5	Multi-line log extraction with regex	The extraction mode of **Multi-line - Full regular expression** is added to LogListener collection configuration rules for log collection.	Full Regular Expression (Multi-Line)

Installation and startup

1. Downloading and installing LogListener

Download links of the latest version LogListener: Download via public network, Download via private network.

Download the LogListener installation package and decompress it to the installation path (`/usr/local/` in this

example). Then, go to the LogListener directory `/usr/local/loglistener/tools` and run the installation command.

Note:

No version number extensions are added to the installation package of LogListener on v2.8.3 and later. The latest version will be installed with `loglistener-linux-x64` by default. To install a specific version, specify the version number, for example, replace `loglistener-linux-x64` with `loglistener-linux-x64-2.8.0` to install the 2.8.0 version.

Operation command for the public network:

```
wget http://mirrors.tencent.com/install/cls/loglistener-linux-x64.tar.gz && tar zxv
```

Operation command for the private network:

```
wget http://mirrors.tencentyun.com/install/cls/loglistener-linux-x64.tar.gz && tar
```

2. Initializing LogListener

In the case of the `/usr/local/` installation path, go to the `/usr/local/loglistener/tools` path and run the following command to initialize LogListener as the root user (by default, the private network is used to access the service):

```
./loglistener.sh init -secretid AKID******************************** -secretkey ***
```

Note:

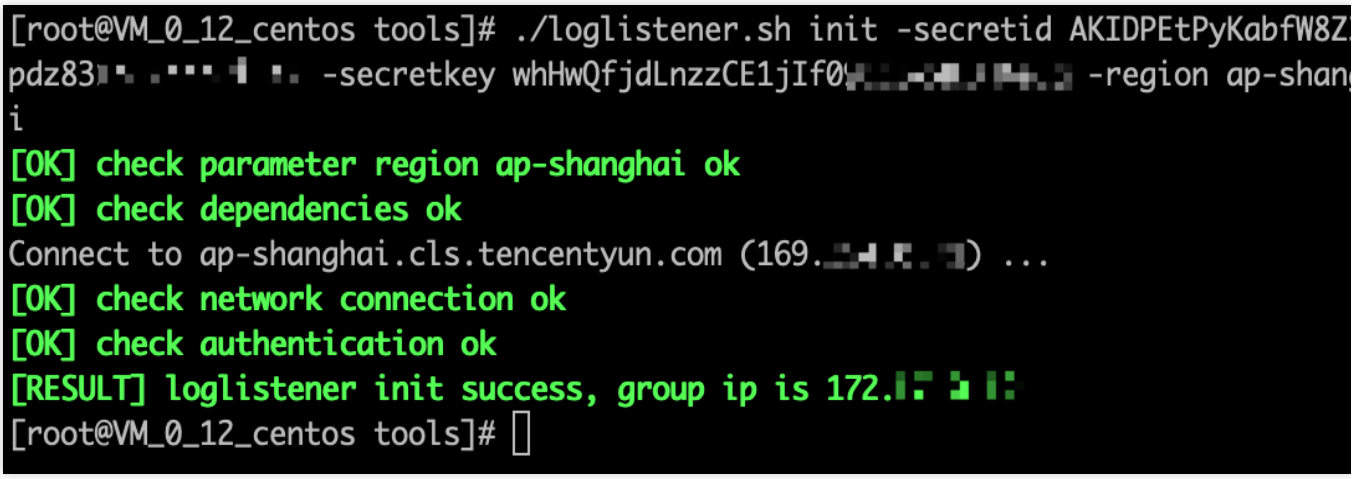
You need to replace **-secretid**, **-secretkey**, **-region**, and **-network** in the command with the actual values. For more information, please see Parameter description below.

Parameter description

Parameter	Description
secretid	Part of the Cloud API key. Used to identify who calls the API.
secretkey	Part of the Cloud API key. Used to encrypt signature strings and verify server-side signature strings.
region	Region where CLS resides. Enter a region abbreviation here, such as `ap-beijing` or `ap-guangzhou` .
network	Type of the network through which LogListener accesses the service by domain name. Valid values: `intra` (private network), `internet` (public network). Default value: `intra`
IP	Machine IP. If this parameter is left empty, LogListener will automatically get the local IP address.
label	Machine group label, which is required if you want to identify the machine group. Multiple labels

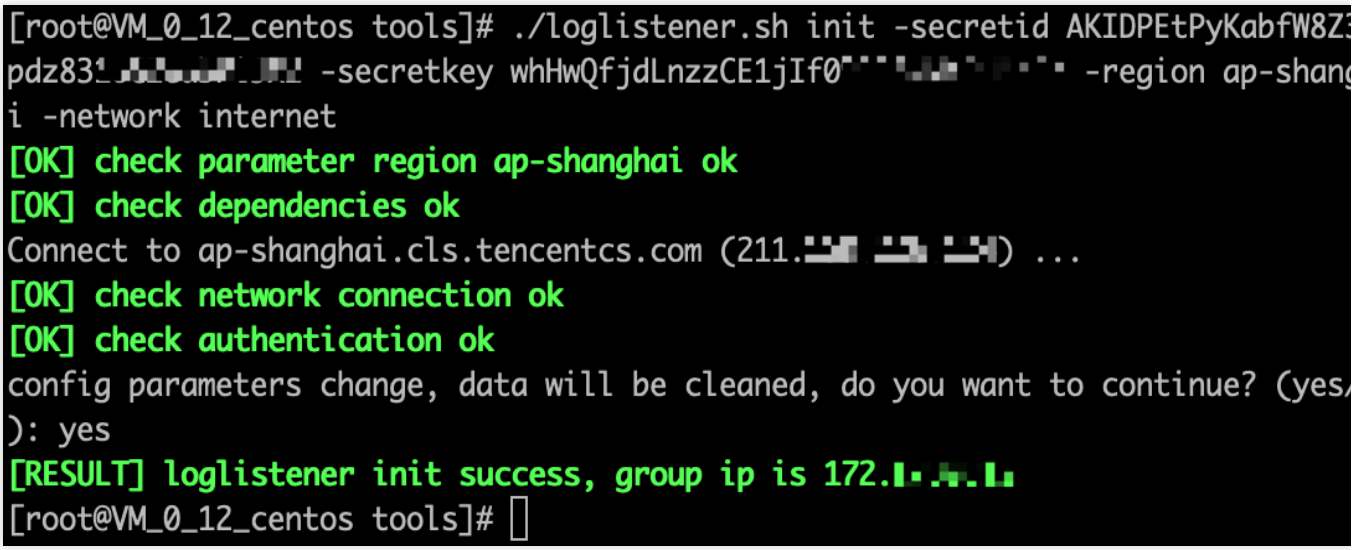
	should be separated by comma.

A private network domain name is used by default:



If you need to access the service by domain name through the public network, run the following command to set the network parameter `internet` explicitly:

```
./loglistener.sh init -secretid AKID******************************** -secretkey ***
```



Note:

We recommend that you use a collaborator key if the collaborator has been assigned the CLS read/write permission by the root account.

`region` indicates the region of the CLS you use, instead of the region where your business machine resides.

If your CVM instance and logset are in the same region, we recommend you access the service domain name over the private network; otherwise, use the public network.
For more information on log collection permissions, see Access Policy Templates.

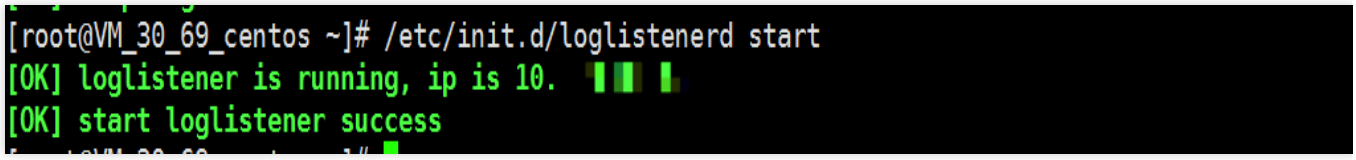
3. Starting LogListener

LogListener is on v2.8.3 or later and the operating system has systemd.

```
systemctl start loglistenerd
```

LogListener is earlier than v2.8.3, or LogListener is on v2.8.3 or later but the operating system does not have systemd.

```
/etc/init.d/loglistenerd start
```



Common LogListener Operations

Note:
The operation commands used in this document are applicable only to LogListener v2.2.4 and later versions. For operation commands applicable to earlier versions, see Earlier-Version LogListener Installation Guide.

1. Checking the LogListener version

```
/etc/init.d/loglistenerd -v
```

2. Viewing LogListener help documentation

```
/etc/init.d/loglistenerd -h
```

3. Managing LogListener process

LogListener is on v2.8.3 or later and the operating system has systemd.

```
systemctl (start|restart|stop)  loglistenerd # Start, restart, stop
```

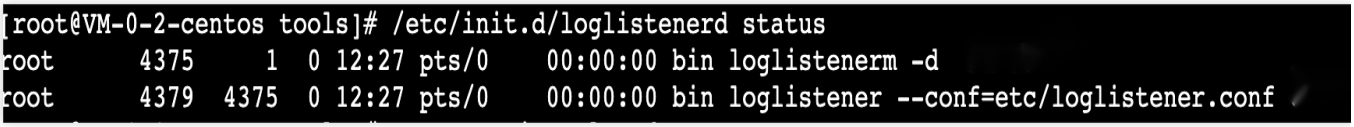
LogListener is earlier than v2.8.3, or LogListener is on v2.8.3 or later but the operating system does not have systemd.

```
/etc/init.d/loglistenerd (start|restart|stop) # Start, restart, stop
```

4. Checking LogListener process status

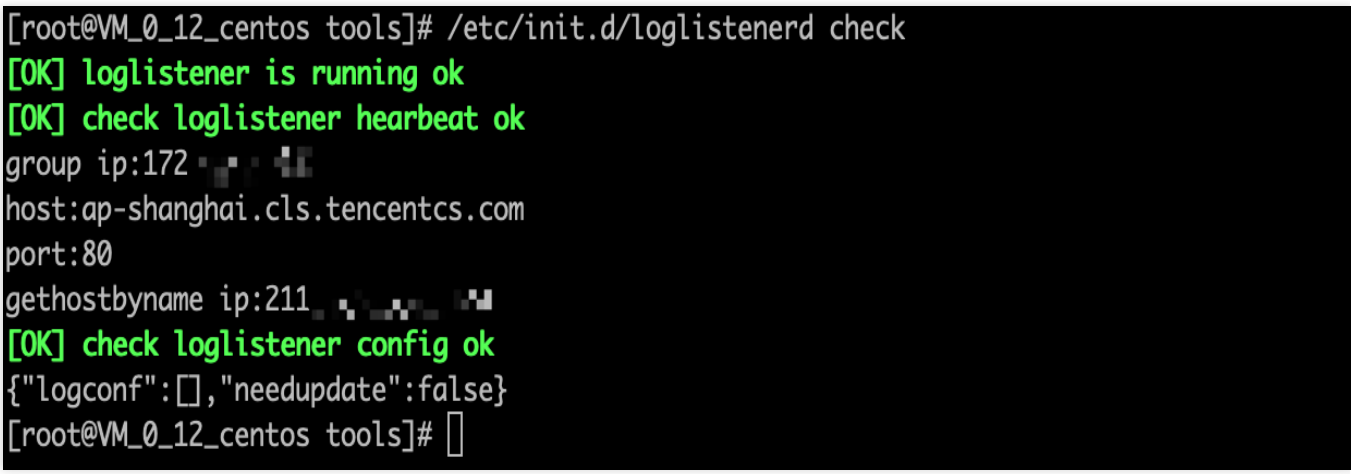
```
/etc/init.d/loglistenerd status
```

LogListener normally runs two processes:



5. Checking LogListener heartbeat and configuration

```
/etc/init.d/loglistenerd check
```



Uninstalling LogListener

In the case of the `/usr/local/` installation path, go to the `/usr/local/loglistener/tools` path and run the uninstallation command as the admin:

```
./loglistener.sh uninstall
```

Manually Updating LogListener

Reusing the breakpoint file (logs are not repeatedly collected)

1. Run the stop command to stop the existing LogListener.
2. Back up the breakpoint file directory (`loglistener/data`) on the earlier version; for example, back up the legacy breakpoint file to the `/tmp/loglistener-backup` directory.

```
cp -r loglistener-2.2.3/data /tmp/loglistener-backup/
```

3. Run the uninstallation command to uninstall the existing LogListener.
4. Download the latest version of LogListener. Then, install and initialize it with relevant commands.
5. Copy the breakpoint file directory backed up in step 2 to the new LogListener directory.

```
cp -r /tmp/loglistener-backup/data loglistener-<version>/
```

Change the value of `<version>` as required. The following is an example:

```
cp -r /tmp/loglistener-backup/data loglistener-2.8.2/
```

6. Run the start command to start the latest version of LogListener.

Not reusing the breakpoint file (logs may be repeatedly collected)

1. Run the stop command to stop the existing LogListener.
2. Run the uninstallation command to uninstall the earlier version of LogListener.
3. Download the latest version of LogListener. Then, install and initialize it with relevant commands.
4. Run the start command to start the latest version of LogListener.

Batch Deploying LogListener in CVM and Lighthouse

Last updated：2024-01-20 17:14:28

Overview

CLS allows you to use LogListener to collect CVM and Lighthouse logs. Before log collection, you need to install and deploy LogListener in CVM or Lighthouse. To quickly install and deploy LogListener in a number of CVM and Lighthouse instances, you can select CVM or Lighthouse instances in the console and batch distribute LogListener deployment tasks through an API to automatically complete LogListener installation and deployment (including `accesskey` , ID, and region configuration).

Prerequisites

You have installed TAT in CVM or Lighthouse.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Machine Group Management** to enter the management page.
3. Click **Deploy Instances** in the top-right corner of the page.
4. On the **Deploy Instances** page, select the server type (CVM or Lighthouse). You can batch deploy LogListener for CVM and Lighthouse instances at the same time.
5. Select the target CVM or Lighthouse instances, enter the `SecretId` information (`SecretId` and `SecretKey`), and set **Machine label** in **Advanced Settings** as needed.

Note:

To install LogListener, you need to provide `SecretId` and `SecretKey` for uploading logs. They can be obtained as instructed in Viewing Acquisition Method.

Make sure that the account of the entered key information has the permission to upload logs. For detailed directions on how to configure permissions, see CAM Access Management.

If your machine IP changes frequently, we recommend that you configure the machine label as instructed in Machine Group Management.

6. Click **Next**.

7. On the instance installation page, click **Next** when the installation **Status** changes to **Completed**, which indicates that the installation is completed.

Note:

If the installation fails, move the cursor over to the **Status** of the instance installation to view the failure causes.

8. On the machine group importing page, select an existing machine group or create a machine group as required, and click **Import**.

Note:

The version of the LogListener deployed via batch deployment is 2.6.0 or later.

Installing LogListener in Self-built Kubernetes Cluster

Last updated：2024-01-20 17:14:28

This document describes how to install LogListener on a self-built Kubernetes cluster to collect logs to CLS.

During the installation process, perform the following operations:

1. Dependencies

Standard Kubernetes cluster. MicroK8s, K3s, or other non-standard Kubernetes clusters are not supported.

Helm 3.1 or later is required.

2. Helm installation

For detailed directions, see Installing Helm.

3. LogListener installation

```
wget https://mirrors.tencent.com/install/cls/k8s/tencentcloud-cls-k8s-install.sh
```

```
bash +x tencentcloud-cls-k8s-install.sh
```

```
./tencentcloud-cls-k8s-install.sh --region ap-guangzhou --secretid xxx --secretkey xxx
```

4. Parameter description

Parameter	Description
secretid	Tencent Cloud account access ID
secretkey	Tencent Cloud account access key
region	CLS region
docker_root	The root directory of the cluster Docker. The default value is `/var/lib/docker`. If the actual directory is different from the default one, specify the root directory of Docker.
cluster_id	Cluster ID. If it is not specified, a default ID will be generated during installation (we recommend that you specify a cluster ID, as the generated default ID is less readable).
network	Private network or public network (default).
api_network	Private network or internet (default) for TencentCloud API.
api_region	TencentCloud API region. For more information, see Available Regions.

Sample:

Component deployment in Guangzhou

```
./tencentcloud-cls-k8s-install.sh --secretid xxx --secretkey xx --region ap-guangzhou  --network internet --api_region ap-guangzhou
```

5. Viewing

5.1 View the installed Helm package.

After the successful installation, view the `tencent-cloud-cls-log` Helm package.

```
helm list -n kube-system
```

5.2 View the components.

```
kubectl get pods -o wide -n kube-system | grep tke-log-agent
```

```
kubectl get pods -o wide -n kube-system | grep cls-provisioner
```

Run the above commands to check whether all components start properly. In normal cases, a `tke-log-agent` collection Pod and a `cls-provisioner` Pod will start on each host.

6. LogListener configuration

6.1 Run the following kubectl command to modify the environment variables of `tke-log-agent` .

```
kubectl edit ds tke-log-agent -n kube-system
```

6.2 Edit the environment variables to define LogListener collection configuration.

```
spec:
  containers:
  - args:
    - --cred-sources=norm
    - --loglistener-config=/usr/local/loglistener/etc/loglistener.conf
    command:
    - /log-agent
    env:
    - name: CRED_SOURCES
      value: norm
    - name: CLS_HOST
      value: ap-beijing.cls.tencentyun.com
    - name: TKE_CLUSTER_ID
      value: cls-4uoaa05v
    - name: MAX_MEM
      value: "2097152000"
    - name: CHECKPOINT_WINDOW_SIZE
      value: "512"
    - name: MAX_FILE_BREAKPOINTS
      value: "102400"
    - name: MAX_FILE
      value: "200000"
    - name: CLS_ENDPOINT
    - name: POD_NAME
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: metadata.name
    - name: POD_IP
      valueFrom:
```

6.3 Parameter description

Variable	Description
MAX_CONNECTION	Maximum number of connections, which is `10` by default.
CHECKPOINT_WINDOW_SIZE	The checkpoint window size of a file, which is `1024` by default.
MAX_FILE_BREAKPOINTS	Breakpoint file size, which is `N*2k`. `N` defaults to `8k`.
MAX_SENDRATE	Maximum sending rate (bytes/s), which is not limited by default.
MAX_FILE	Maximum number of monitored files, which is `15000` by default.
MAX_DIR	Maximum number of monitored directories, which is `5000` by default.
MAX_HTTPS_CONNECTION	Maximum number of HTTPS connections, which is `100` by default.
CONCURRENCY_TASKS	LogListener task pool, which is `256` by default (supported by v3.x or later).
PROCESS_TASKS_EVERY_LOOP	Number of tasks processed every loop, which is `4` by default.
CPU_USAGE_THRES	LogListener memory usage threshold, which is not limited by default.

7. Upgrade

7.1 For Kubernetes 1.13 or later:

```
wget http://mirrors.tencent.com/install/cls/k8s/upgrade/upgrade.sh
```

```
chmod +x upgrade.sh
```

```
./upgrade.sh
```

7.2 For Kubernetes versions earlier than 1.13:

```
wget http://mirrors.tencent.com/install/cls/k8s/upgrade/upgrade-1.13.sh
```

```
chmod +x upgrade-1.13.sh
```

```
./upgrade-1.13.sh
```

8. Uninstallation

Run the following command to uninstall the installed `tencent-cloud-cls-log` Helm package.

```
helm uninstall tencent-cloud-cls-log -n kube-system
```

Note:

To completely delete the `tencent-cloud-cls-log` package, regardless of whether you need to redeploy it after specifying the `-cluster_id` parameter incorrectly or you don't need to deploy it any more, you need to run the following command to delete the `secret` of `cls-k8s`, as it saves the `-cluster_id` parameter.

```
kubectl delete secret -n kube-system cls-k8s
```

LogListener Upgrade Guide

Last updated：2024-01-20 17:14:28

Overview

To experience better CLS log collection service, you can upgrade the LogListener automatically or manually via the console or semi-automatically via scripts. After the LogListener version is upgraded, you don't need to download the latest-version installer. You can set a time period for auto upgrade or select specific machines to upgrade manually by one click.

Note:

Auto upgrade is supported for LogListener v2.5.0 and later. For a better user experience, we recommend you install or upgrade to the latest version of LogListener.

The auto upgrade feature requires Python 2. If Python 3 is installed on the collection machine, the upgrade process will not be fully executed.

The script-based semi-auto upgrade feature requires Python 2.7. If this version is not installed on the collection machine, this upgrade mode cannot be used.

You can choose auto upgrade or manual upgrade for LogListener in the console.

If Agent upgrade fails or auto upgrade cannot be used due to version restrictions, you can use the semi-auto upgrade mode.

Directions

Auto upgrade

1. Log in to the CLS console.
2. On the left sidebar, click **Machine Group** to enter the management page.
3. Find the target machine group, move the cursor under

in the **Auto Upgrade** column, and click **Enable Now**.

4. In the pop-up, toggle the button on and specify the upgrade time period (the default period is from the current time to two hours later, such as 08:39-10:39).
5. Click **OK**. When the status of the target machine group changes to **Enabled** as displayed in the **Auto Upgrade** column, auto upgrade has been enabled successfully.

Note:

You can set any time period for the auto upgrade, and the system will check if upgrade is needed every day in the specified time period. If the upgrade conditions are met, the auto-upgrade will be performed; otherwise, the operation will be performed.
You can select multiple target machine groups and click **Auto Upgrade** to upgrade them in batches.

Manual upgrade

1. Log in to the CLS console.
2. On the left sidebar, click **Machine Group** to enter the management page.
3. Find the target machine group and click **Upgrade Now** in the **Operation** column.
4. In the pop-up window, select the target machine with status as **To be upgraded** and click **Manual Upgrade**.

The system will upgrade LogListener to the latest version by default. When you see **This is the latest version**, the manual upgrade is successful.
Note:
When the status is **Unable to upgrade**, you cannot upgrade the LogListener in the console, and you need to download the latest-version installer to upgrade. For details, see LogListener Installation Guide.
If **Abnormal heartbeat** appears, see Machine Group Exception for troubleshooting tips.

Semi-auto upgrade

1. Run the following command to check the Python version.

```
python -V
```

If the Python version is not 2.7, use auto or manual upgrade.
2. Run the following command to download the script.

```
wget http://mirrors.tencentyun.com/install/cls/agent-update.py
```

3. Run the following command to execute the script.

```
/usr/bin/python2.7 agent-update.py
http://mirrors.tencentyun.com/install/cls/loglistener-linux-x64-x.tar.gz
```

Here, `x` in `loglistener-linux-x64-x.tar.gz` represents the version number of LogListener to upgrade to (such as 2.7.2). The latest version of LogListener is as displayed in LogListener Installation Guide. If the entered version does not exist, the download will fail. If the version entered is earlier than the current version installed on the machine, the upgrade will not take effect.

LogListener Service Logs

Last updated：2024-01-20 17:14:28

Overview

LogListener service logs are used to record the operation, collection, and monitoring activities of LogListener, and you can configure visualized graphs to display such log data.

Default configuration

Default Configuration Item	Description
Log Topic	When you enable LogListener service logs, the logset `cls_service_logging` will be created for you automatically, and all log data generated by associated machine groups will be categorized and stored in corresponding log topics. The following three log topics are created by default: `loglistener_status` : the heartbeat status logs for the corresponding LogListener. `loglistener_alarm` : logs corresponding to LogListener's monitoring by collection metric/error type . `loglistener_business` : logs corresponding to LogListener's collection operations, with each log corresponds to one request.
Region	After the LogListener log services are enabled, logsets and log topics will be created under the machine groups in the same region of LogListener.
Log Storage Duration	The default storage duration is 7 days, and the value cannot be modified.
Index	Full-text index and key-value index are enabled for all collected log data by default. You can modify the index configuration. For details, please see Configuring Index.
Dashboard	Dashboard `service_log_dashboard` will be created in the same region of LogListener by default.

Note:

LogListener service logs are only about collection and monitoring activities of LogListener, and do not support writing into other kinds of data.

Log data generated by this feature does not incur costs.

`cls_service_logging` is a unified logset for LogListener service logs.

Operations

Viewing LogListener status

When the LogListener service logs are enabled, you can view LogListener running status and collection statistics. You can view the number of active LogListeners, LogListener status distribution and other statistical metrics on the `service_log_dashboard` dashboard.

Log collection monitoring configuration

You can configure the collection and monitoring service logs by metric/error type. For example:

MEM, CPU, collection speed, collection latency, or other metrics.

The number of parsing errors of LogListener.

File-level monitoring

With the LogListener service logs are enabled, you can view the monitoring logs of files and directories. For example:

All collection statistics files of one IP

The amount of logs collected under a certain path on a certain IP, such as `app1` application logs located in `/var/log/app1/` . You can get the statistics of logs collected under this path.

The collection statistics of a topic.

Prerequisites

Only LogListener v2.5.4 and above support collection and monitoring service logs by machine/machine group. You're advised to upgrade LogListener to the latest version.

Directions

Enabling the service logs

1. Log in to the CLS console.
2. In the left sidebar, click **Machine Group** to go to the machine group management page.
3. On the machine group list page, select the target machine group and click

to enable the LogListener service logs.

Disabling LogListener service logs

1. Log in to the CLS console.
2. In the left sidebar, click **Machine Group** to view the machine group list.
3. Select the target machine group, and click

 to disable the LogListener service logs.

Note:

After this feature is disabled, the log data saved in the logset `cls_service_logging` will not be deleted automatically. You can manually delete the logset where the service logs are saved.

Dashboard of Service Logs

When the LogListener service logs are enabled, CLS will create a dashboard `service_log_dashboard` by the type of recorded logs to display LogListener's collection and monitoring statistics.

Collection statistics dashboard

You can go to the Dashboard page of the CLS console, click the ID of the target dashboard to view LogListener collection statistics, including its status, parsing failure rate, sending success rate, and other metrics.

Log Types

LogListener status logs

The parameters of the log topic `loglistener_status` are detailed as follows:

Parameter	Description
InstanceId	LogListener unique identifier
IP	Machine group IP
Label	An array of machine IDs
Version	Version number
MemoryUsed	Memory utilization of LogListener
MemMax	Memory utilization threshold on this machine set by the Agent
CpuUsage	LogListener CPU utilization
Status	LogListener running status
TotalSendLogSize	Size of logs sent
SendSuccessLogSize	Size of successfully sent logs

SendFailureLogSize	Size of sending-failed logs
SendTimeoutLogSize	Size of logs with sending timed out
TotalParseLogCount	Total number of logs parsed
ParseFailureLogCount	Number of parsing-failed logs
TotalSendLogCount	Number of logs sent
SendSuccessLogCount	Number of successful sent logs
SendFailureLogCount	Number of sending-failed logs
SendTimeoutLogCount	Number of logs with sending timed out
TotalSendReqs	Total number of requests sent
SendSuccessReqs	Number of successful sent requests
SendFailureReqs	The number of sending-failed requests
SendTimeoutReqs	Number of requests with sending timed out
TotalFinishRsps	Total number of RSP files received
TotalSuccessFromStart	Total number of successfully sent requests since LogListener was enabled
AvgReqSize	Average request packet size
SendAvgCost	Average sending time
AvailConnNum	Number of available connections
QueueSize	The size of queued requests

LogListener alarm logs

The parameters of the log topic `loglistener_alarm` are detailed as follows:

Monitoring Metric	Description
InstanceId	LogListener unique identifier
Label	An array of machine IDs
IP	Machine group IP
Version	LogListener version

AlarmType.count	Statistics of alarm types
AlarmType.example	Sample alarm type

AlarmType:

alarm type	type ID	Description
UnknownError	0	Initializing the alarm type.
UnknownError	1	Failed to parse.
CredInvalid	2	Failed to verify.
SendFailure	3	Failed to send.
RunException	4	Abnormal LogListener running.
MemLimited	5	Reached the memory utilization threshold.
FileProcException	6	Exceptions occurred in file processing.
FilePosGetError	7	Failed to get the file publishing info.
HostIpException	8	Exceptions occurred in the server IP thread.
StatException	9	Failed to get the process info.
UpdateException	10	Exceptions occurred in the CLS modification feature.
DoSendError	11	Failed to confirm sending.
FileAddError	12	Failed to create the file.
FileMetaError	13	Failed to create the metadata file.
FileOpenError	14	Failed to open the file.
FileOpenError	15	Failed to read the file.
FileStatError	16	Failed to get the file status.
getTimeError	17	Failed to get the time from the log content.
HandleEventError	18	Exceptions occurred in processing the file.
handleFileCreateError	19	Exceptions occurred in `handleFileCreateEvent()`.
LineParseError	20	Failed to parse the log directory.

Lz4CompressError	21	Failed to compress.
readEventException	22	Failed to read.
ReadFileBugOn	23	A bug exists.
ReadFileException	24	Exceptions occurred in the read file.
ReadFileInodeChange	25	File node changed.
ReadFileTruncate	26	The read file is truncated.
WildCardPathException	27	Exceptions occurred in `addWildcardPathInotify()` .

LogListener collection logs

The parameters of the log topic `loglistener_business` are detailed as follows:

Parameter	Description
InstanceId	LogListener unique identifier
Label	An array of machine IDs
IP	Machine group IP
Version	LogListener version
TopicId	The target topic of the collected file
FileName	File path name
FileName	Actual file path
FileInode	File node
FileSize	File size
LastReadTime	The most recent read time of the file
ParseFailLines	Number of parsing-failed logs within a time window
ParseFailSize	Size of parsing-failed logs within a time window
ParseSuccessLines	Number of logs successful parsed within a time window
ParseSuccessSize	Size of logs successful parsed within a time window
ReadOffset	Offset of file reading in bytes

TruncateSize	Size of truncated log files within a time window
ReadAvgDelay	Average time delay for reads within a time window
TimeFormatFailuresLines	Number of timestamp matching errors within a time window
SendSuccessSize	Size of logs successful sent within a time window
SendSuccessCount	Number of logs successful sent within a time window
SendFailureSize	Size of sending-failed logs within a time window
SendFailureCount	Number of sending-failed logs within a time window
SendTimeoutSize	Size of logs with sending timed out within a time window
SendTimeoutCount	Number of logs with sending timed out within a time window
DroppedLogSize	Size of dropped logs within a time window
DroppedLogCount	Number of dropped logs in a time window
ProcessBlock	Whether the current file has triggered collection blocking in a statistical period (collection blocking will be triggered if the sliding window of a file has not moved for 10 minutes)

Importing LogListener Collection Configuration

Last updated：2024-01-20 17:14:28

This document describes how to use LogListener to quickly import the collection configuration rules of other log topics.

Overview

LogListener collection configuration refers to the collection path, use limits, collection mode, and other collection rules configured in the LogListener collection server before log collection. The collection configuration rule import feature allows users to import the collection configuration of an existing log topic to quickly configure LogListener collection rules when they add or modify collection configuration. This eliminates repetitive and tedious operations for configuring multiple log topics and improves the efficiency of collection configuration.

Note:

By default, the collection of a log file can only be configured in only one LogListener.

To apply multiple collection configurations to one file, you need to add a soft link to the source file and add the soft link to another group of collection configuration.

Only LogListener 2.3.9 or above allows adding multiple collection paths.

Directions

1. Log in to the CLS console.
2. In the left sidebar, click **Log Topic** to go to the log topic management page.
3. Click the ID/name of an existing log topic to go to the log topic information page.
4. Click the **Collection Configuration** tab to go to the **Collection Configuration** tab page.
5. Click **Import Configuration Rule** in the upper-right corner.
6. In the configuration rule list displayed in the pop-up window, select the configuration rule of a log topic to import and click **OK** to import it to the collection configuration of the current log topic.

Note:

The configuration rule list displays all log topics that support the cross-region log topic configuration rule import feature in the current region by default.

Only the collection rules of log topics for which a collection path is configured can be imported to the collection configuration of the current log topic.

Collection Mode

LogListener can collect text logs in the following collection modes: Full Text in a Single Line, Full Text in Multi Lines, Full Regular Format (Single-Line), Full Regular Format (Multi-Line), JSON Format, and Separator Format.

LogListener Updates

Last updated：2024-01-20 17:14:28

This document describes CLS LogListener updates.

Note:

The __HOSTNAME__ collection feature is available starting from LogListener v2.7.4.

The combined parsing feature is available starting from LogListener v2.6.4.

Full/Incremental collection policies are available starting from LogListener v2.6.2.

CVM batch deployment is available starting from LogListener v2.6.0.

Multi-line - full regular expression collection mode is available starting from LogListener v2.4.5.

LogListener auto upgrade is available starting from LogListener v2.5.0.

Uploading parsing-failed logs is available starting from LogListener v2.5.2.

You are advised to install or upgrade to the latest version for a better user experience.

Version	Change Type	Description
v2.7.9	Experience optimization	Added LogListener file lock verification, so only one agent instance can be started by default. Fixed the empty row processing exception in `containerd stdout`. Fixed full disk and business exceptions caused by file handle leaks. Fixed the failure in parsing the second half of the log content when there were too many lines of logs.
v2.7.8	Experience optimization	Fixed the issue where logs didn't have tag metadata due to metadata file generation delay in container scenarios.
v2.7.7	Experience optimization	Fixed the issue where the collection program's network connection couldn't be reconnected after a DNS exception was fixed.
v2.7.6	Experience optimization	Optimized the line break processing during `hostname` extraction.
v2.7.5	Experience optimization	Fixed the processing exception in file rotation when the actual file and soft link in the same directory were collected at the same time with different collection configurations.
v2.7.4	New feature	Supported collecting `hostname` as the metadata. Added `meta_processor` for combined parsing and supported parsing custom metadata (path).
	Experience optimization	Fixed the missing collection problem in file deletion scenarios. Fixed the issue where a file was collected repeatedly as the file size calculated by the system was incorrect due to the lace of a line break at the end of the file.

v2.7.3	New feature	Supported log upload from multiple endpoints by a single agent instance.
v2.7.2	Experience optimization	Fixed the issue where the memory leaked as the corresponding configuration cache couldn't be cleared when a rotation file was removed.
v2.7.1	Experience optimization	Fixed the issue where a large number of empty service logs were printed.
v2.7.0	Experience optimization	Fixed the issue where collection was blocked due to possible exceptions when an empty string was uploaded.
v2.6.9	Experience optimization	Fixed the issue where excessive invalid logs were printed when multi-line log parsing failed.
v2.6.8	Experience optimization	Added a limit on the LogListener collection specification, so the protection mechanism will be enabled after the limit is exceeded. Fixed the Ubuntu startup failure. Optimized the blocklist feature to reduce the memory usage. Optimized the combined parsing mode and fixed processing exceptions when the root processor was a regular expression parsing plugin. Optimized the printing of certain logs.
v2.6.7	New feature	Supported the multi-tenancy collection capabilities under a single agent.
v2.6.6	Experience optimization	Fixed the issue where files with a small amount of written data might be missing or delayed during collection in soft link scenarios.
v2.6.5	New feature	Supported parsing the time zone information in the log time.
	Experience optimization	Fixed the empty pointer processing exception in advanced data processing. Fixed the exception when multiple files were rotated at the same time.
v2.6.4	New feature	Supported customizing log parsing rules through a plugin.
	Experience optimization	Optimized the log parsing format pipeline. Fixed the exception of parsing the millisecond timestamp (`%F`).
v2.6.3	Experience optimization	Fixed the issue where LogListener couldn't be started if the checkpoint file is corrupted. Fixed the issue where the blocklist didn't take effect for new files in special scenarios.
v2.6.2	New feature	Added support for incremental collection.
	Experience optimization	Optimized the issue where collection is ignored in the period from file scanning to processing. Optimized abnormal overriding during automatic upgrade.

v2.6.1	Experience optimization	Optimized the issue where backtracking collection may occur during log rotation in some scenarios. Adjusted the timeout duration for log upload on the collection end to avoid data duplication caused by timeout.
v2.6.0	New feature	Added support for CVM batch deployment. Added support for ciphertext storage of secret IDs/KEYs.
	Experience optimization	Optimized the LogListener installation and stop logic. Optimized the retry policy upon upload failures. Added a tool for detecting and rectifying dead locks caused by Glibc libraries of earlier versions. Optimized collection performance.
v2.5.9	Experience optimization	Optimized the resource limit policy.
v2.5.8	Experience optimization	Fixed the issue that removing a directory soft link affects the collection of other directory soft links that point to the same target. Fixed the issue that files in a directory cannot be collected if a soft link of the directory is removed and the same soft link is created again.
v2.5.7	Experience optimization	Fixed the (new) issue that logs will be collected again when the log file size is greater than 2 GB. Fixed the issue where renaming too many files will cause the program to malfunction. Fixed the issue where specified fields cannot be updated under log collection monitoring.
v2.5.6	Experience optimization	Optimized the issue that under specific use cases, the collection program cannot be triggered.
v2.5.5	Experience optimization	Optimized metadata checkpoints for collection to guarantee no data will lose due to restart. Supports resource limit configuration and overrun handling for memory, CPU, and bandwidth.
v2.5.4	New feature	Added support for log collection monitoring.
	Experience optimization	Enhanced memory overrun handling: LogListener will be automatically loaded when memory overrun lasts for a period of time.
v2.5.3	Experience optimization	Optimized LogListener exceptions caused by memory issues.
v2.5.2	New feature	Added support for uploading parsing-failed logs.
	Experience	Optimized the blocklist feature. Now, the blocklist FILE mode supports

	optimization	wildcard filtering.
v2.5.1	Experience optimization	Enhanced the handling when breakpoint metadata could not be found in the collection file.
v2.5.0	New feature	Added support for automatic LogListener upgrade. Added support for automatic LogListener start in Ubuntu operating system.
v2.4.6	Experience optimization	Cleared residual configuration data in the cache after the collection configuration was changed. Optimized the issue where file collection with a soft link pointing to the `realpath` file was affected when an `IN_DELETE` event that deleted the soft link was being processed. Optimized the feature of collecting the same source file via the file's soft link and the directory's soft link at the same time.
v2.4.5	New feature	Added support for `multiline_fullregex_log` log collection.
v2.4.4	Experience optimization	Optimized the issue of inaccurate log time caused by the msec feature.
v2.4.3	New feature	Added support for automatically checking the log format (logFormat).
v2.4.2	Experience optimization	Optimized the issue of cache eviction during configuration pulling in Tencent Cloud container scenarios.
v2.4.1	New feature	Added support for collecting logs in milliseconds.
	Experience optimization	Optimized exceptions due to no line break data in user logs.
v2.4.0	New feature	Added support for instance-level process monitoring by LogListener.
v2.3.9	New feature	Added support for blocklisting collection paths.
	Experience optimization	Optimized the memory leak issue due to outdated Boost library.
v2.3.8	New feature	Added support for multi-path log collection.
v2.3.6	Experience optimization	Fixed the issue where collection stopped due to invalid key value. Fixed the memory leak issue due to request failures with the error code 502 returned.
v2.3.5	New feature	Added support for log context search.
	Experience optimization	Fixed the issue where log collection stopped when logs were uploaded but authentication failed in the static configuration mode.

		Fixed the issue where dynamic configurations were no longer read after the memory exceeded the threshold in the dynamic configuration mode. Fixed the issue where sometimes log collection repeated when the log production speed was too high during log rotation. Fixed the memory leak issue caused by multiple failures to upload logs.
v2.3.1	Experience optimization	Optimized memory limit. When the memory limit was reached, requests lasting over 3s were considered as timed out.
v2.2.6	New feature	Added support for configuring private domain names and public domain names separately.
	Experience optimization	Fixed LogListener exceptions caused by `getip`.
v2.2.5	New feature	Added support for Tencent Cloud COC environment deployment.
	Experience optimization	Fixed the core issue caused by `getip`.
v2.2.4	Experience optimization	Changed the commands for installation and initialization to the subcommands `install` and `init` of `tools/loglistener.sh` respectively. Changed the command for restart to `/etc/init.d/loglistenerd start|stop|restart`.
v2.2.3	Experience optimization	Renaming or creating logs during log rotation will not cause log loss.
v2.2.2	Experience optimization	A log greater than 512 KB will be automatically truncated.
Earlier versions	-	v2.2.2 added support for collection by full regular expression. v2.1.4 added support for full text in multi lines. v2.1.1 added support for log structuring.

Encrypted Storage Of Keys

Last updated：2025-04-18 16:19:02

Overview

LogListener requires key authentication to access the CLS server. By default, LogListener stores the key in plaintext in its configuration file after initialization. If you do not want to save the key in plaintext, this document will guide you on how to enable encryption during the initialization of LogListener and how to modify the key storage mode afterward.

Directions

Specifying the Encrypted Storage Key During Initialization

See LogListener Installation Guideline (for Linux version). When LogListener is initialized, add the `-encryption` parameter. To enable key encryption, set the input parameter to true; if encryption is not required, set the input parameter to false.

Modifying the Key Storage Method After Initialization

Modified to Encrypted Storage

1. Go to the LogListener installation directory.
2. In the LogListener installation directory, run `./bin/encrypt_tool -e {Key ID}` to obtain the encrypted key ID.
3. In the LogListener installation directory, run `./bin/encrypt_tool -e {key}` to obtain the encrypted key.
4. Run `vim ./etc/loglistener.conf` under the LogListener installation path to open the configuration file. Update the secret_id and secret_key fields with the encrypted key ID and key obtained in steps 2 and 3. Finally, set encryption to true.

5. Run the following command to restart LogListener:

Executed By Systemd

Not Executed By Systemd

It is applicable to LogListener 2.8.3 and later versions with the operating system having a systemd.

```
systemctl restart loglistenerd
```

Applicable to LogListener 2.8.3 earlier versions or later versions with the operating system not having a systemd.

```
/etc/init.d/loglistenerd restart
```

Modified to Plaintext Storage

1. Go to the LogListener installation directory.
2. In the LogListener installation directory, run `vim ./etc/loglistener.conf` to replace secret_id and secret_key in the conf file with plaintext key ID and key, and set encryption to false.
3. Run the following command to restart LogListener:

Executed By Systemd

Not Executed By Systemd

It is applicable to LogListener 2.8.3 and later versions with the operating system having a systemd.

```
systemctl restart loglistenerd
```

Applicable to LogListener 2.8.3 earlier versions or later versions with the operating system not having a systemd.

```
/etc/init.d/loglistenerd restart
```

Collecting Text Log
Full Text in a Single Line

Last updated：2024-01-20 17:14:28

Overview

A log with full text in a single line means a line is a full log. When CLS collects logs, it uses the line break `\\n` to mark the end of a log. For easier structural management, a default key value `__CONTENT__` is given to each log, but the log data itself will no longer be structured, nor will the log field be extracted. The time attribute of a log is determined by the collection time.

Prerequisites

Suppose your raw log data is:

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

The log is eventually structured by CLS as follows:

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.e
```

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.
2. In the pop-up dialog box, enter `test_full` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.
2. Click the **Collection Configuration** tab and click the format in which you need to collect logs.

3. On the **Machine Group Management** page, select the server group to which to bind the current log topic and click **Next** to proceed to collection configuration.
For more information, see Machine Group Management.

Configuring collection

Configuring the log file collection path

On the **Collection Configuration** page, set **Collection Path** according to the log collection path format.
Log collection path format: `[directory prefix expression]/**/[filename expression]` .
After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the wildcard characters `\\*` and `?` . `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	Log file name, which supports only the wildcard characters `\\*` and `?` . `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:
[Common directory prefix]/**/[common filename prefix]*
[Common directory prefix]/**/*[common filename suffix]
[Common directory prefix]/**/[common filename prefix]*[common filename suffix]
[Common directory prefix]/**/*[common string]*
Below are examples:

No.	Directory Prefix Expression	Filename Expression	Description
1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log` . LogListener will listen for log files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.
2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log` . LogListener will listen for log files

			suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.

By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

Configuring the "full text in a single line" mode

In the **Collection Configuration** page, select **Full text in a single line** as the **Extraction Mode**.

Configuring the collection policy

Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.

Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

Configuring filter rules

Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.

By default, this "full text in a single line" mode uses `__CONTENT__` as the key name of a log. Assume that a sample log is `Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64`, and you want to collect all logs on Jan 22, then enter `__CONTENT__` in **Key** and `Tue Jan 22.*` in **Filter Rule**.

Note:

The relationship logic between multiple filter rules is "AND". If multiple filter rules are configured for the same key name, previous rules will be overwritten.

Configuring indexes

1. Click **Next** to enter the **Index Configuration** page.
2. On the **Index Configuration** page, set the following information:

Index Status: Select whether to enable it.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: The default value is `@&()='",;:<>[]{}/ \\n\\t\\r` and can be modified as needed.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, you can set

to 


.

Note:

Index configuration must be enabled before you can perform searches.

3. Click **Submit**.

Related Operations

Log search

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to go to the search and analysis page.
3. Select the region, logset, and log topic as needed, and click **Search and Analysis** to search for logs according to the set query rules.

Full Text in Multi Lines

Last updated：2024-01-20 17:14:28

Overview

In "full text in multi lines" mode, a log spans multiple lines (such as a Java program log), and the line break `\\n` cannot be used to mark the end of a log. To help CLS distinguish between logs, a first-line regular expression is used for matching. When a line of a log matches the preset regular expression, it is considered as the beginning of the log, and the log ends before the next matching line.
In "full text in multi lines" mode, a default key `__CONTENT__` is also set, but the log data itself is not structured, and no log fields are extracted. The time attribute of a log is determined by the collection time.

Prerequisites

Assume the raw data of a multi-line log is:

```
10.20.20.10 - - [Tue Jan 22 14:24:03 CST 2019 +0800] GET /online/sample HTTP/1.1 12
Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0 0.310 0.3
```

The log is eventually structured by CLS as follows:

```
__CONTENT__:10.20.20.10 - - [Tue Jan 22 14:24:03 CST 2019 +0800] GET /online/sample
```

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.
2. In the pop-up dialog box, enter `test-mtext` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.

2. Click the **Collection Configuration** tab, click **Add** in **LogListener Collection Configuration**, and select the format in which you need to collect logs.
3. On the **Machine Group Management** page, select the server group to which to bind the current log topic and click **Next** to proceed to collection configuration.
For more information, see Machine Group Management.

Configuring collection

Configuring the log file collection path

On the **Collection Configuration** page, enter the collection rule name and enter the **Collection Path** according to the **log collection path format**.
Log collection path format: `[directory prefix expression]/**/[filename expression]`.
After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the wildcard characters `\\*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	Log file name, which supports only the wildcard characters `\\*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:
[Common directory prefix]/**/[common filename prefix]*
[Common directory prefix]/**/*[common filename suffix]
[Common directory prefix]/**/[common filename prefix]*[common filename suffix]
[Common directory prefix]/**/*[common string]*
Below are examples:

No.	Directory Prefix Expression	Filename Expression	Description
1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log`. LogListener will listen for log files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.

2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log`. LogListener will listen for log files suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.

By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

Configuring the collection policy

Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.

Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

Configuring the "full text in multi lines" mode

1. On the **Collection Configuration** page, select **Full text in multi lines** as the **Extraction Mode**.

2. Define a regular expression according to the following rules.

You can choose **Auto-Generate** or **Enter Manually** to define a first-line regular expression, and the system will verify the regular expression based on the sample content.

Auto-Generate: Enter the sample log in the text box, click **Auto-Generate**, and the system will automatically generate the first-line regular expression in the grayed-out text box.

Enter Manually: Enter the sample log and first-line regular expression in the text box, click **Verify**, and the system will determine whether the expression has passed verification.

Configuring filter rules

Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.

In "full text in multi lines" mode, `__CONTENT__` is used as the key name of a log by default. For example, below is a sample log with full text in multi lines:

```
10.20.20.10 - - [Tue Jan 22 14:24:03 CST 2019 +0800] GET /online/sample HTTP/1.1 12
Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0 0.310 0.3
```

If you want to collect all logs of the machine `10.20.20.10` , enter `__CONTENT__` in **Key** and `10.20.20.10.*` in **Filter Rule**.

Note:

The relationship logic between multiple filter rules is "AND". If multiple filter rules are configured for the same key name, previous rules will be overwritten.

Configuring parsing-failed log upload

We recommend you enable **Upload Parsing-Failed Logs**. After it is enabled, LogListener will upload all types of parsing-failed logs. If it is disabled, such logs will be discarded.

After this feature is enabled, you need to configure the `Key` value for parsing failures (which is `LogParseFailure` by default). All parsing-failed logs are uploaded with the input content as the key name (`Key`) and the raw log content as the key value (`Value`).

Configuring indexes

1. Click **Next** to enter the **Index Configuration** page.
2. On the **Index Configuration** page, set the following information:

Index Status: Select whether to enable it.

Note:

Index configuration must be enabled before you can perform searches.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: The default value is `@&()='",;:<>[]{}/ \\n\\t\\r` and can be modified as needed.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, you can set

to 


.

3. Click **Submit**.

Related Operations

For more information on log search, see Overview and Syntax Rules.

Full Regular Expression (Single-Line)

Last updated：2024-01-20 17:14:28

Overview

The single-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from each log in a log text file in which each line is a raw log based on a regular expression. If you don't need to extract key-value pairs, configure it as instructed in Collecting Logs with Full Text in a Single Line.
When configuring the single-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression.
This document describes how to collect logs in single-line - full regular expression mode.

Prerequisites

Suppose your raw log data is:

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1"
127.0.0.1 200 782 9703 "http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0"  0.354 0.354
```

The custom regex you configure is:

```
(\\S+)[^\\[]+(\\[[^:]+:\\d+:\\d+:\\d+\\s\\S+)\\s"
(\\w+)\\s(\\S+)\\s([^"]+)"\\s(\\S+)\\s(\\d+)\\s(\\d+)\\s(\\d+)\\s"([^"]+)"\\s"
([^"]+)"\\s+(\\S+)\\s(\\S+).*
```

Then CLS extracts key-value pairs based on the `()` capture groups. You can specify the key name of each group.

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0
remote_addr: 10.135.46.111
```

```
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.
2. In the pop-up dialog box, enter `test-whole` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.
2. Click the **Collection Configuration** tab, click **Add** in **LogListener Collection Configuration**, and select the format in which you need to collect logs.
3. On the **Machine Group Management** page, select the server group to which to bind the current log topic and click **Next** to proceed to collection configuration.

For more information, see Machine Group Management.

Configuring collection

Configuring the log file collection path

On the **Collection Configuration** page, enter the collection rule name and enter the **Collection Path** according to the **log collection path format**.

Log collection path format: `[directory prefix expression]/**/[filename expression]` .

After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the

	wildcard characters `\\*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	Log file name, which supports only the wildcard characters `\\*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:

[Common directory prefix]/**/[common filename prefix]*

[Common directory prefix]/*/[common filename suffix]

[Common directory prefix]/**/[common filename prefix]*[common filename suffix]

[Common directory prefix]/**/*[common string]*

Below are examples:

No.	Directory Prefix Expression	Filename Expression	Description
1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log`. LogListener will listen for log files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.
2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log`. LogListener will listen for log files suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

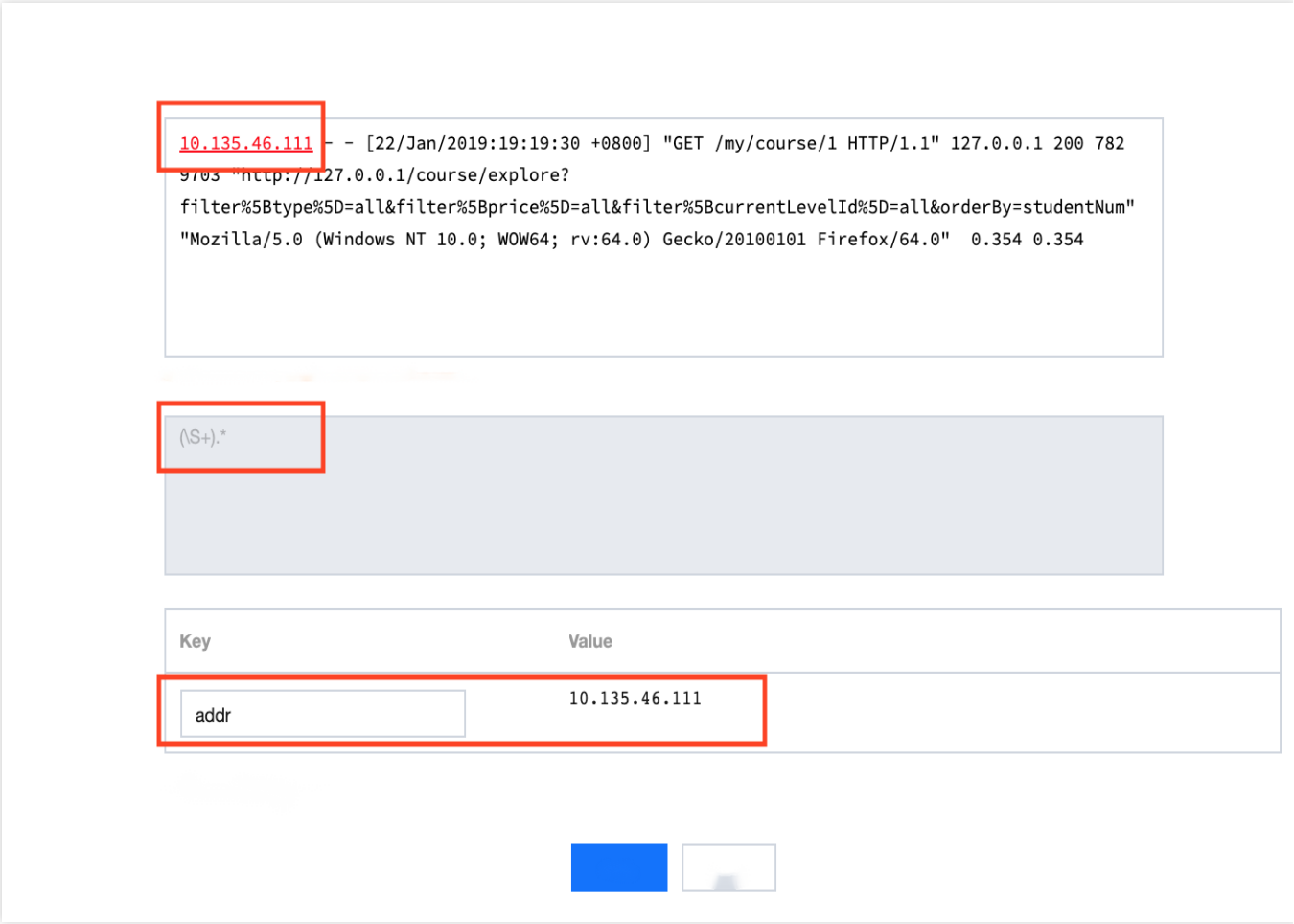
We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.
By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

Configuring the collection policy

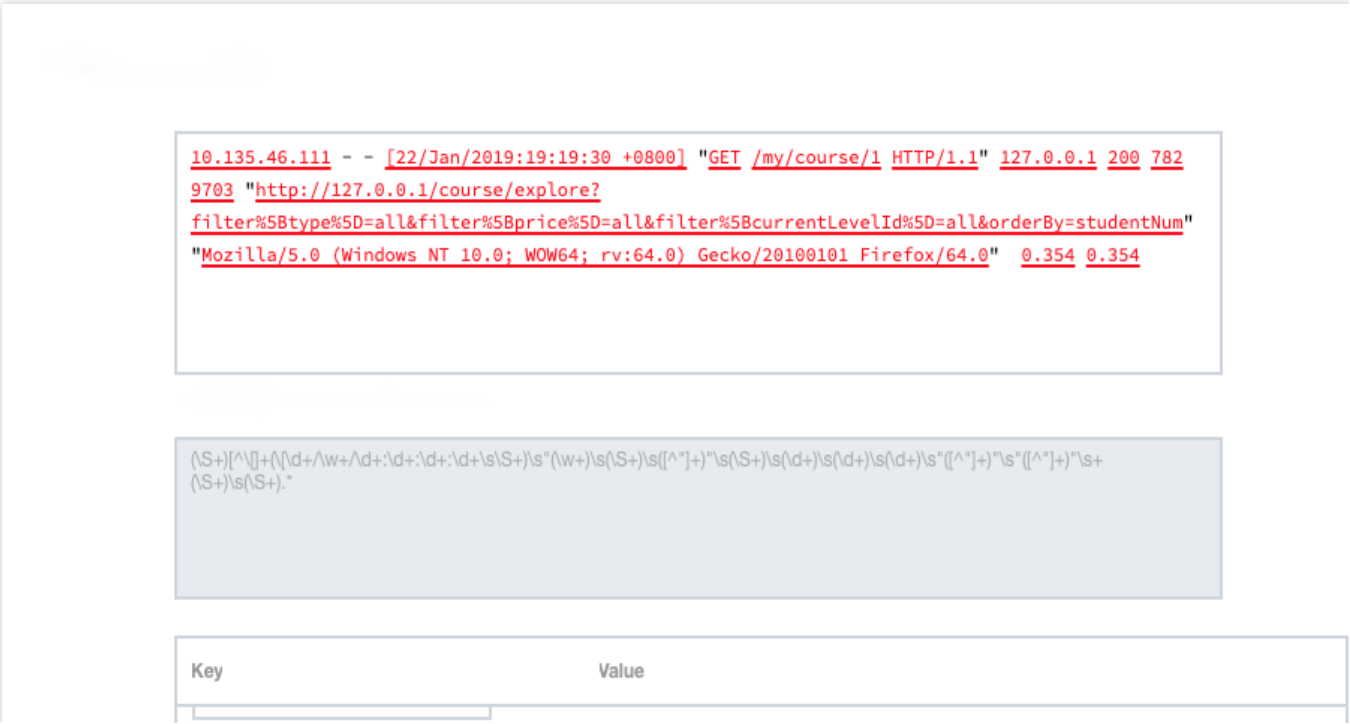
Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.
Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

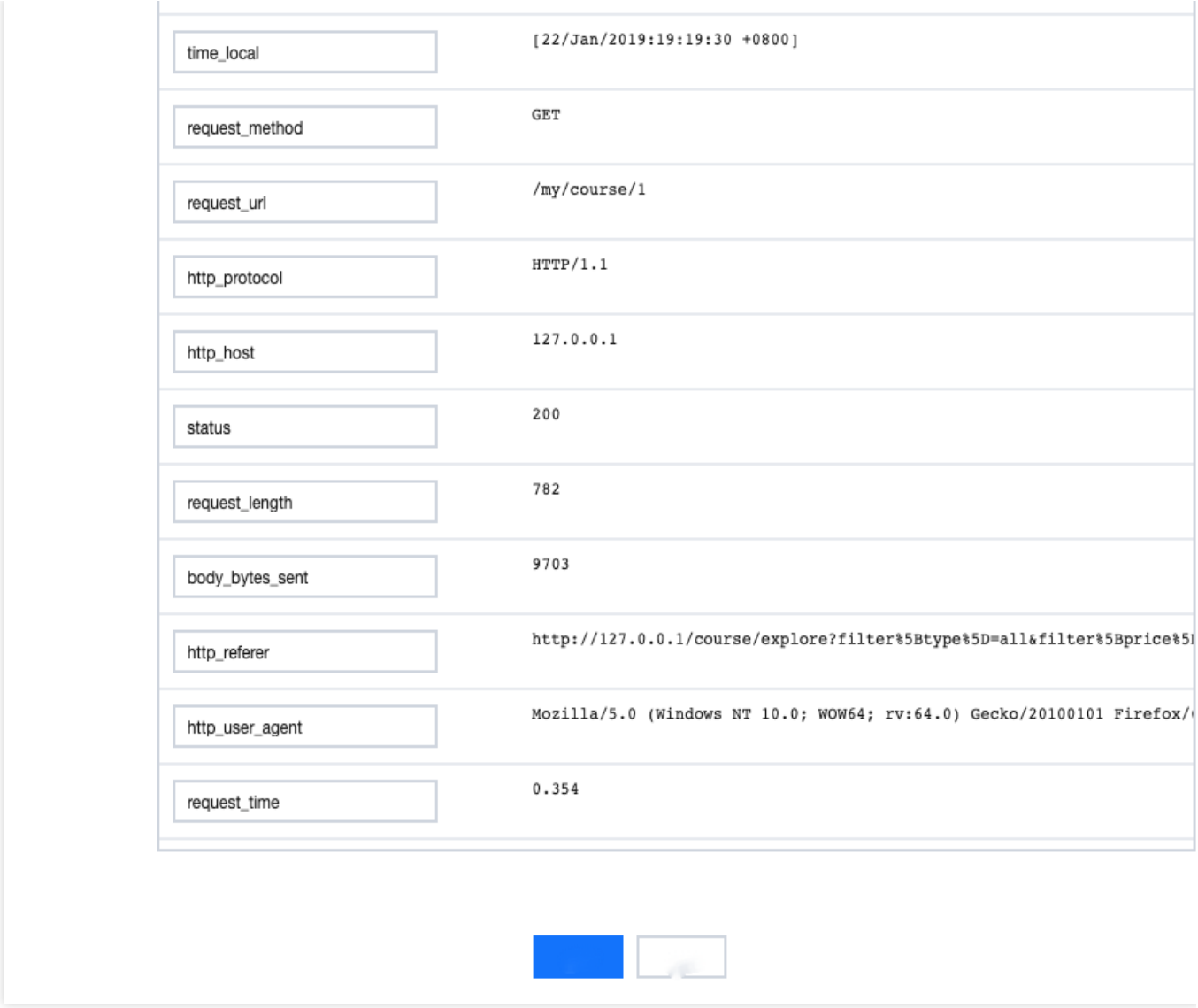
Configuring the single-line - full regular expression mode

1. On the **Collection Configuration** page, set **Extraction Mode** to **Single-line - Full regular expression** and enter a sample log in the **Log Sample** text box.
2. Define a regular expression according to the following rules.
The system offers two ways to define a regular expression: **manual mode** and **auto mode**. You can manually enter the expression to extract key-value pairs for verification or click **Auto-Generate Regular Expression** to switch to auto mode. The system will extract key-value pairs to verify the regular expression according to the mode you selected and the regular expression you defined.
Manual mode:
2.1.1 Enter the regular expression in the **Regular Expression** text box.
2.1.2 Click **Verify**, and the system will determine whether the sample log matches the regular expression.
Auto Mode (click **Auto-Generate Regular Expression** to switch):
2.1.1
In
the **Auto-Generate Regular Expression** pop-up view, select the log content from which to extract key-value pairs based on your actual search and analysis needs, enter the key name in the pop-up text box, and click **Confirm**.
The system will automatically extract a regular expression from the content, and the **Automatic Extraction Result** will appear in the key-value table.



2.1.2 Repeat step a until all key-value pairs are extracted.





2.1.3 Click **OK**, and the system will automatically generate a complete regular expression according to the extracted key-value pairs.

Note:

No matter whether in auto mode or manual mode, the extraction result will be displayed in the **Extraction Result** after the regular mode is defined and verified successfully. You only need to define the key name of each key-value pair for use in log search and analysis.

Performing manual verification

1. If your log data is complex, you can set **Manual Verification** to

to enable manual verification.

2. Enter multiple sample logs, click **Verify**, and the system will verify the pass rate of the regular expression for the logs.

Configuring the collection time

Log time is measured in milliseconds.

The time attribute of a log is defined as follows:

Collection time: It is the default time attribute of a log.

Original timestamp: Set **Use Collection Time** to

and enter the time key of the original timestamp and the corresponding time parsing format.

For more information on the time format, see Configuring the Time Format.

Collection time: The time attribute of a log is determined by the time when CLS collects the log.

Original timestamp: The time attribute of a log is determined by the timestamp in the raw log.

Below are examples of how to enter a time resolution format:

Example 1:

The parsing format of the original timestamp `10/Dec/2017:08:00:00.000` is `%d/%b/%Y:%H:%M:%S.%f`.

Example 2:

The parsing format of the original timestamp `2017-12-10 08:00:00.000` is `%Y-%m-%d %H:%M:%S.%f`.

Example 3:

The parsing format of the original timestamp `12/10/2017, 08:00:00.000` is `%m/%d/%Y, %H:%M:%S.%f`.

Note:

The log time is measured in milliseconds. If the log time is entered in an incorrect format, the collection time is used as the log time.

Configuring filter rules

Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.

To collect logs in full regular expression mode, you need to configure a filter rule according to the defined custom key-value pair. For example, if you want to collect all log data with a `status` field with the value 400 or 500 after the sample log is parsed in full regular expression mode, you need to configure `key` as `status` and the filter rule as `400|500`.

Note:

The relationship between multiple filter rules is logic "AND". If multiple filter rules are configured for the same key name, previous rules will be overwritten.

Configuring parsing-failed log upload

We recommend you enable **Upload Parsing-Failed Logs**. After it is enabled, LogListener will upload all types of parsing-failed logs. If it is disabled, such logs will be discarded.

After this feature is enabled, you need to configure the `Key` value for parsing failures (which is `LogParseFailure` by default). All parsing-failed logs are uploaded with the input content as the key name (`Key`) and the raw log content as the key value (`Value`).

Configuring indexes

1. Click **Next** to enter the **Index Configuration** page.
2. On the **Index Configuration** page, set the following information:

Index Status: Select whether to enable it.

Note:

Index configuration must be enabled before you can perform searches.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: The default value is `@&()='",;:<>[]{}/ \\n\\t\\r` and can be modified as needed.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, you can set

to 


.

3. Click **Submit**.

Related Operations

For more information on log search, see Overview and Syntax Rules.

Full Regular Expression (Multi-Line)

Last updated：2024-01-20 17:14:28

Overview

The multi-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete piece of log data that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. If you don't need to extract key-value pairs, configure it as instructed in Collecting Logs with Full Text in Multi Lines.

When configuring the multi-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression.

This document describes how to collect logs in multi-line - full regular expression mode.

Note:

To collect logs in multi-line - full regular expression mode, you need to upgrade to LogListener 2.4.5 as instructed in LogListener Installation Guide.

Prerequisites

Suppose your raw log data is:

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

The first-line regular expression is:

```
\\[\\d+-\\d+-\\w+:\\d+:\\d+,\\d+]\\s\\[\\w+]\\s.*
```

The custom regex you configure is:

```
\\[(\\d+-\\d+-\\w+:\\d+:\\d+,\\d+)\\]\\s\\[(\\w+)\\]\\s(.*)
```

Then CLS extracts key-value pairs based on the `()` capture groups. You can specify the key name of each group.

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
```

```
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.
2. In the pop-up dialog box, enter `test-multi` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.
2. Click the **Collection Configuration** tab, click **Add** in **LogListener Collection Configuration**, and select the format in which you need to collect logs.
3. On the **Machine Group Management** page, select the server group to which to bind the current log topic and click **Next** to proceed to collection configuration.

For more information, see Machine Group Management.

Configuring collection

Configuring the log file collection path

On the **Collection Configuration** page, enter the collection rule name and enter the **Collection Path** according to the **log collection path format**.

Log collection path format: `[directory prefix expression]/**/[filename expression]`.

After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the wildcard characters `*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	

	Log file name, which supports only the wildcard characters `*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:

[Common directory prefix]/**/[common filename prefix]*

[Common directory prefix]/*/[common filename suffix]

[Common directory prefix]/**/[common filename prefix]*[common filename suffix]

[Common directory prefix]/**/*[common string]*

Below are examples:

No.	Directory Prefix Expression	Filename Expression	Description
1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log`. LogListener will listen for log files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.
2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log`. LogListener will listen for log files suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.

By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

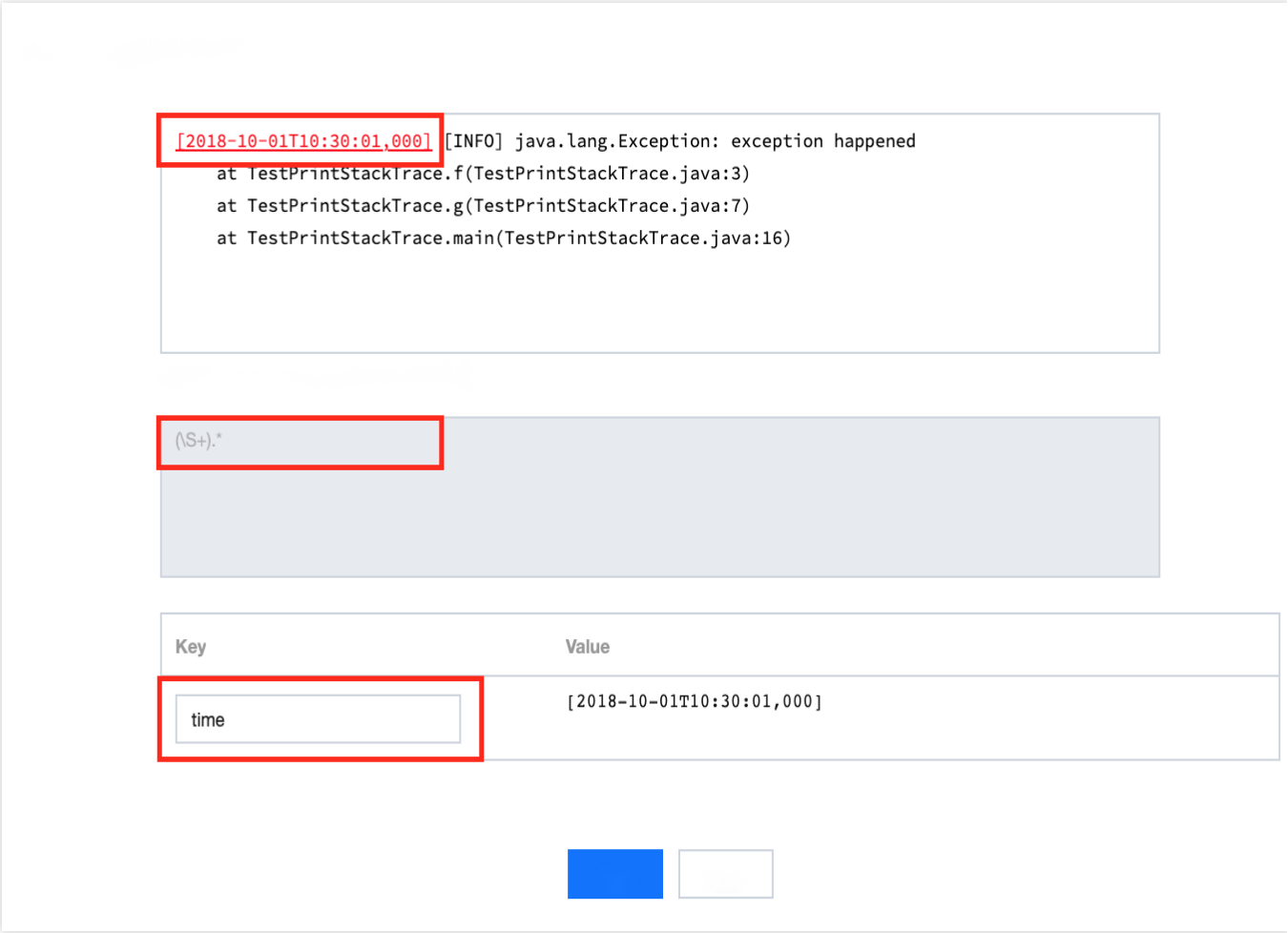
Configuring the collection policy

Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.

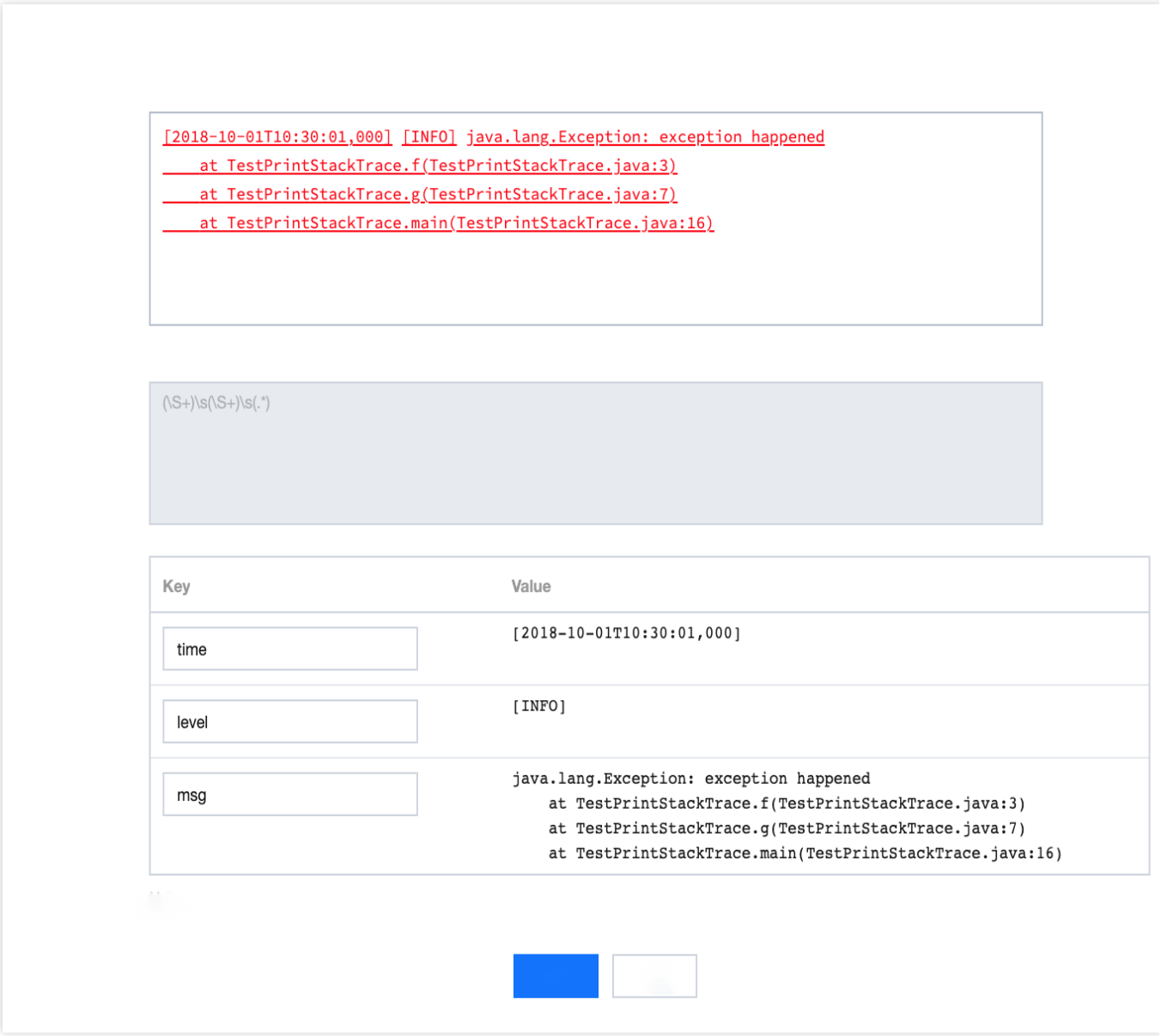
Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

Configuring the multi-line - full regular expression mode

1. On the **Collection Configuration** page, set **Extraction Mode** to **Multi-line - Full regular expression** and enter a sample log in the **Log Sample** text box.
2. Define a regular expression according to the following rules.
You can choose **Auto-Generate** or **Enter Manually** to define a first-line regular expression in order to determine the boundary for multi-line logs. After the expression is verified successfully, the system will determine the number of logs that match the first-line regular expression.
Auto-Generate: Click **Auto-Generate**, and the system will automatically generate the first-line regular expression in the grayed-out text box.
Enter Manually: In the text box, enter the first-line regular expression, click **Verify**, and the system will determine whether the expression has passed.
3. Extract the regular expression.
The system offers two ways to define a regular expression: **manual mode** and **auto mode**. You can manually enter the expression to extract key-value pairs for verification or click **Auto-Generate Regular Expression** to switch to auto mode. The system will extract key-value pairs to verify the regular expression according to the mode you selected and the regular expression you defined.
Manual mode:
3.1.1 Enter the regular expression in the **Regular Expression** text box.
3.1.2 Click **Verify**, and the system will determine whether the sample log matches the regular expression.
Auto Mode (click **Auto-Generate Regular Expression** to switch):
3.1.1
In
the **Auto-Generate Regular Expression** pop-up view, select the log content from which to extract key-value pairs based on your actual search and analysis needs, enter the key name in the pop-up text box, and click **Confirm**.
The system will automatically extract a regular expression from the content, and the **Automatic Extraction Result** will appear in the key-value table.



3.1.2 Repeat step a until all key-value pairs are extracted.



3.1.3 Click **OK**, and the system will automatically generate a complete regular expression according to the extracted key-value pairs.

Note:

No matter whether in auto mode or manual mode, the extraction result will be displayed in the **Extraction Result** after the regular mode is defined and verified successfully. You only need to define the key name of each key-value pair for use in log search and analysis.

Performing manual verification

1. If your log data is complex, you can set **Manual Verification** to

to enable manual verification.

2. Enter multiple sample logs, click **Verify**, and the system will verify the pass rate of the regular expression for the logs.

Configuring the collection time

Note:

The log time is measured in seconds. If the log time is entered in an incorrect format, the collection time is used as the log time.

The time attribute of a log is defined in two ways: collection time and original timestamp.

Collection time: The time attribute of a log is determined by the time when CLS collects the log.

Original timestamp: The time attribute of a log is determined by the timestamp in the raw log.

Using the collection time as the time attribute of logs: Keep **Collection Time** enabled.

Using the original timestamp as the time attribute of logs: Disable **Collection Time** and enter the time key of the original timestamp and the corresponding time parsing format in **Time Key** and **Time Parsing Format** respectively. For more information on the time parsing format, see Configuring Time Format.

Below are examples of how to enter a time parsing format:

Example 1: The parsing format of the original timestamp `10/Dec/2017:08:00:00` is `%d/%b/%Y:%H:%M:%S`.

Example 2: The parsing format of the original timestamp `` `2017-12-10 08:00:00` `` is `%Y-%m-%d %H:%M:%S`.

Example 3: The parsing format of the original timestamp `12/10/2017, 08:00:00` is `%m/%d/%Y, %H:%M:%S`.

Configuring filter rules

Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.

To collect logs in full regular expression mode, you need to configure a filter rule according to the defined custom key-value pair. For example, if you want to collect all log data with a `status` field with the value 400 or 500 after the sample log is parsed in full regular expression mode, you need to configure `key` as `status` and the filter rule as `400|500`.

Note:

The relationship between multiple filter rules is logic "AND". If multiple filter rules are configured for the same key name, previous rules will be overwritten.

Configuring parsing-failed log upload

We recommend you enable **Upload Parsing-Failed Logs**. After it is enabled, LogListener will upload all types of parsing-failed logs. If it is disabled, such logs will be discarded.

After this feature is enabled, you need to configure the `Key` value for parsing failures (which is `LogParseFailure` by default). All parsing-failed logs are uploaded with the input content as the key name

(`Key`) and the raw log content as the key value (`Value`).

Configuring indexes

1. Click **Next** to enter the **Index Configuration** page.
2. On the **Index Configuration** page, set the following information:

Index Status: Select whether to enable it.

Note:

Index configuration must be enabled before you can perform searches.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: The default value is `@&()='",;:<>[]{}/ \\n\\t\\r` and can be modified as needed.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, you can set

to

.

3. Click **Submit**.

Related Operations

For more information on log search, see Overview and Syntax Rules.

JSON Format

Last updated：2024-01-20 17:14:28

Overview

A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to implement structured processing of the entire log. Each complete log ends with a line break `\\n`.

Prerequisites

Suppose your raw JSON log data is:

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34 +0800","body_sent":
```

After being structured by CLS, the log is changed to the following:

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.
2. In the pop-up dialog box, enter `test-json` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.
2. Click the **Collection Configuration** tab, click **Add** in **LogListener Collection Configuration**, and select the format in which you need to collect logs.
3. On the **Machine Group Management** page, select the server group to which to bind the current log topic and click **Next** to proceed to collection configuration.

For more information, see Machine Group Management.

Configuring collection

Configuring the log file collection path

On the **Collection Configuration** page, enter the collection rule name and enter the **Collection Path** according to the **log collection path format**.

Log collection path format: `[directory prefix expression]/**/[filename expression]`.

After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the wildcard characters `*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	Log file name, which supports only the wildcard characters `*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:

[Common directory prefix]/**/[common filename prefix]*

[Common directory prefix]/**/*[common filename suffix]

[Common directory prefix]/**/[common filename prefix]*[common filename suffix]

[Common directory prefix]/**/*[common string]*

Below are examples:

No.	Directory Prefix	Filename Expression	Description

	Expression		
1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log`. LogListener will listen for log files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.
2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log`. LogListener will listen for log files suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.

By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

Configuring the collection policy

Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.

Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

Configuring the JSON mode

On the **Collection Configuration** page, select **JSON** as the **Extraction Mode**.

Configuring the collection time

Note:

The log time is measured in seconds. If the log time is entered in an incorrect format, the collection time is used as the log time.

The time attribute of a log is defined in two ways: collection time and original timestamp.

Collection time: The time attribute of a log is determined by the time when CLS collects the log.

Original timestamp: The time attribute of a log is determined by the timestamp in the raw log.

Using the collection time as the time attribute of logs: Keep **Collection Time** enabled.

Using the original timestamp as the time attribute of logs: Disable **Collection Time** and enter the time key of the original timestamp and the corresponding time parsing format in **Time Key** and **Time Parsing Format** respectively. For more information on the time parsing format, see Configuring Time Format.

Below are examples of how to enter a time parsing format:

Example 1: The parsing format of the original timestamp `10/Dec/2017:08:00:00` is `%d/%b/%Y:%H:%M:%S`.

Example 2: The parsing format of the original timestamp `` `2017-12-10 08:00:00` `` is `%Y-%m-%d %H:%M:%S`.

Example 3: The parsing format of the original timestamp `12/10/2017, 08:00:00` is `%m/%d/%Y, %H:%M:%S`.

Configuring filter rules

Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.

You can configure a filter rule for JSON logs according to the parsed key-value pair. For example, if you want to collect all log data with a `response_code` field with the value 400 or 500 from the original JSON log file, you need to configure `key` as `response_code` and the filter rule as `400|500`.

Note:

The relationship logic between multiple filter rules is "AND". If multiple filter rules are configured for the same key name, previous rules will be overwritten.

Configuring parsing-failed log upload

We recommend you enable **Upload Parsing-Failed Logs**. After it is enabled, LogListener will upload all types of parsing-failed logs. If it is disabled, such logs will be discarded.

After this feature is enabled, you need to configure the `Key` value for parsing failures (which is `LogParseFailure` by default). All parsing-failed logs are uploaded with the input content as the key name (`Key`) and the raw log content as the key value (`Value`).

Configuring indexes

1. Click **Next** to enter the **Index Configuration** page.

2. On the **Index Configuration** page, set the following information:

Index Status: Select whether to enable it.

Note:

Index configuration must be enabled before you can perform searches.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: It is disabled by default and can be enabled as needed.

Key-Value Index: Enabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis as needed. To disable key-value index, you can set

to  
.

3. Click **Submit**.

Related Operations

For more information on log search, see Overview and Syntax Rules.

Separator Format

Last updated：2024-01-20 17:14:28

Overview

In a separator log, the entire log data can be structured according to the specified separator, and each complete log ends with a line break `\\n`. When CLS processes separator logs, you need to define a unique key for each separate field.

Prerequisites

Suppose your raw log data is:

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/
```

If the separator for log parsing is specified as `:::`, the log will be segmented into eight fields, and a unique key will be defined for each of them.

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.
2. In the pop-up dialog box, enter `test-separator` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.
2. Click the **Collection Configuration** tab, click **Add** in **LogListener Collection Configuration**, and select the format in which you need to collect logs.
3. On the **Machine Group Management** page, select the server group to which to bind the current log topic and click **Next** to proceed to collection configuration.

For more information, see Machine Group Management.

Configuring collection

On the **Collection Configuration** page, enter the collection rule name and enter the **Collection Path** according to the **log collection path format**.

Log collection path format: `[directory prefix expression]/**/[filename expression]`.

After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the wildcard characters `*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	Log file name, which supports only the wildcard characters `*` and `?`. `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:

[Common directory prefix]/**/[common filename prefix]*

[Common directory prefix]/**/*[common filename suffix]

[Common directory prefix]/**/[common filename prefix]*[common filename suffix]

[Common directory prefix]/**/*[common string]*

Below are examples:

No.	Directory Prefix Expression	Filename Expression	Description
1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log`. LogListener will listen for log

			files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.
2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log`. LogListener will listen for log files suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.

By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

Configuring the collection policy

Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.

Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

Configuring the separator mode

1. Set **Extraction Mode** to **Separator**.
2. Select **Separator**, enter a sample log in the **Log Sample** text box, and click **Extract**.

The system segments the sample log according to the selected separator and displays it in the extraction result box.

You need to define a unique key for each field. Currently, log collection supports a variety of separators. Common separators include space, tab, comma, semicolon, and vertical bar. If your log data uses other separators such as `:::`, it can also be parsed through custom delimiter.

Configuring the collection time

Note:

The log time is measured in seconds. If the log time is entered in an incorrect format, the collection time is used as the log time.

The time attribute of a log is defined in two ways: collection time and original timestamp.

Collection time: The time attribute of a log is determined by the time when CLS collects the log.

Original timestamp: The time attribute of a log is determined by the timestamp in the raw log.

Using the collection time as the time attribute of logs

Keep **Collection Time** as enabled.

Using the original timestamp as the time attribute of logs

Disable **Collection Time** and enter the time key of the original timestamp and the corresponding time parsing format in **Time Key** and **Time Parsing Format** respectively. For more information on the time parsing format, see Configuring Time Format.

Below are examples of how to enter a time parsing format:

Example 1: The parsing format of the original timestamp `10/Dec/2017:08:00:00` is `%d/%b/%Y:%H:%M:%S`.

Example 2: The parsing format of the original timestamp `` `2017-12-10 08:00:00` `` is `%Y-%m-%d %H:%M:%S`.

Example 3: The parsing format of the original timestamp `12/10/2017, 08:00:00` is `%m/%d/%Y, %H:%M:%S`.

Note:

Second can be used as the unit of log time. If the time is entered in a wrong format, the collection time is used as the log time.

Configuring filter rules

Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.

For separator-formatted logs, you need to configure a filter rule according to the defined custom key-value pair. For example, if you want to collect all log data with a `status` field with the value 400 or 500 after the sample log is parsed in separator mode, you need to configure `key` as `status` and the filter rule as `400|500`.

Note:

The relationship logic between multiple filter rules is "AND". If multiple filter rules are configured for the same key name, previous rules will be overwritten.

Configuring parsing-failed log upload

We recommend you enable **Upload Parsing-Failed Logs**. After it is enabled, LogListener will upload all types of parsing-failed logs. If it is disabled, such logs will be discarded.

After this feature is enabled, you need to configure the `Key` value for parsing failures (which is `LogParseFailure` by default). All parsing-failed logs are uploaded with the input content as the key name (`Key`) and the raw log content as the key value (`Value`).

Configuring indexes

1. Click **Next** to enter the **Index Configuration** page.
2. On the **Index Configuration** page, set the following information:

Index Status: Select whether to enable it.

Note:

Index configuration must be enabled before you can perform searches.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: The default value is `@&()='",;:<>[]{}/ \\n\\t\\r` and can be modified as needed.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, you can set

to 


.

3. Click **Submit**.

Related Operations

For more information on log search, see Overview and Syntax Rules.

Combined Parsing Format

Last updated：2024-01-20 17:14:28

Overview

If your log structure is too complex and involves multiple log parsing modes, and a single parsing mode (such as the NGINX mode, full regex mode, or JSON mode) cannot meet log parsing requirements, you can use LogListener to parse logs in combined parsing mode. You can enter code (in JSON format) in the console to define the pipeline logic for log parsing. You can add one or more LogListener plugins to process configurations, and the LogListener plugins are executed in the configuration processing order.

Prerequisites

Assume that the raw data of a log is as follows:

```
1571394459,http://127.0.0.1/my/course/4|10.135.46.111|200,status:DEAD,
```

The content of a custom plugin is as follows:

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
         "type": "processor_timeformat",
         "detail": {
           "KeepSource": true,
           "TimeFormat": "%s",
           "SourceKey": "time"
          }
       },
       {
         "type": "processor_split_delimiter",
         "detail": {
           "KeepSource": false,
           "Delimiter": "|",
```

```
            "SourceKey": "msg1",
            "ExtractKeys": [ "submsg1","submsg2","submsg3"]
           },
           "processors": []
        },
        {
          "type": "processor_split_key_value",
          "detail": {
            "KeepSource": false,
            "Delimiter": ":",
            "SourceKey": "msg2"
          }
        }
      ]
    }
  ]
}
```

After being structured by CLS, the log is changed to the following:

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

Configuration Instructions

Custom plugin types

Plugin Feature	Plugin Name	Feature Description
Field extraction	processor_log_string	Performs multi-character (line breaks) parsing of fields, typically for single-line logs.
Field extraction	processor_multiline	Performs first-line regex parsing of fields (full regex mode), typically for multi-line logs.
Field extraction	processor_multiline_fullregex	Performs first-line regex parsing of fields (full regex mode), typically for multi-line logs; extracts regexes from multi-line logs.
Field extraction	processor_fullregex	Extracts fields (full regex mode) from single-line logs.

Field extraction	processor_json	Expands field values in JSON format.
Field extraction	processor_split_delimiter	Extracts fields (single-/multi-character separator mode).
Field extraction	processor_split_key_value	Extracts fields (key-value pair mode).
Field processing	processor_drop	Discards fields.
Field processing	processor_timeformat	Parses time fields in raw logs to convert time formats and set parsing results as log time.

Custom plugin parameters

Plugin Name	Support Subitem Parsing	Plugin Parameter	Required	Feature Description
processor_multiline	No	BeginRegex	Yes	Defines the first-line matching regex for multi-line logs.
processor_multiline_fullregex	Yes	BeginRegex	Yes	Defines the first-line matching regex for multi-line logs.
		ExtractRegex	Yes	Defines the extraction regex after multi-line logs are extracted.
		ExtractKeys	Yes	Defines the extraction keys.
processor_fullregex	Yes	ExtractRegex	Yes	Defines the extraction regex.
		ExtractKeys	Yes	Defines the extraction keys.
processor_json	Yes	SourceKey	No	Defines the name of the upper-level processor key processed by the current processor.
		KeepSource	No	Defines whether to retain `SourceKey` in the final key name.
processor_split_delimiter	Yes	SourceKey	No	Defines the name of the upper-level processor key processed

				by the current processor.
		KeepSource	No	Defines whether to retain `SourceKey` in the final key name.
		Delimiter	Yes	Defines the separator (single or multiple characters).
		ExtractKeys	Yes	Defines the extraction keys after separator splitting.
processor_split_key_value	No	SourceKey	No	Defines the name of the upper-level processor key processed by the current processor.
		KeepSource	No	Defines whether to retain `SourceKey` in the final key name.
		Delimiter	Yes	Defines the separator between the `Key` and `Value` in a string.
processor_drop	No	SourceKey	Yes	Defines the name of the upper-level processor key processed by the current processor.
processor_timeformat	No	SourceKey	Yes	Defines the name of the upper-level processor key processed by the current processor.
		TimeFormat	Yes	Defines the time parsing format for the `SourceKey` value (time data string in logs).

Directions

Logging in to the console

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.

Creating a log topic

1. Click **Create Log Topic**.

2. In the pop-up dialog box, enter `define-log` as **Log Topic Name** and click **Confirm**.

Managing the machine group

1. After the log topic is created successfully, click its name to go to the log topic management page.
2. Click the **Collection Configuration** tab, click **Add** in the **LogListener Collection Configuration** area, and select the format in which you need to collect logs.
3. On the **Machine Group Management** page, select the machine group to which to bind the current log topic and click **Next** to proceed to collection configuration.

For more information, see Machine Group Management.

Configuring collection

Configuring the log file collection path

On the **Collection Configuration** page, set **Collection Path** according to the log collection path format.

Log collection path format: `[directory prefix expression]/**/[filename expression]` .

After the log collection path is entered, LogListener will match all common prefix paths that meet the **[directory prefix expression]** rule and listen for all log files in the directories (including subdirectories) that meet the **[filename expression]** rule. The parameters are as detailed below:

Parameter	Description
Directory Prefix	Directory prefix for log files, which supports only the wildcard characters `*` and `?` . `\\*` indicates to match any multiple characters. `?` indicates to match any single character.
/**/	Current directory and all its subdirectories.
File Name	Log file name, which supports only the wildcard characters `*` and `?` . `\\*` indicates to match any multiple characters. `?` indicates to match any single character.

Common configuration modes are as follows:

[Common directory prefix]/**/[common filename prefix]*

[Common directory prefix]/**/*[common filename suffix]

[Common directory prefix]/**/[common filename prefix]*[common filename suffix]

[Common directory prefix]/**/*[common string]*

Below are examples:

No.	Directory Prefix Expression	Filename Expression	Description

1.	/var/log/nginx	access.log	In this example, the log path is configured as `/var/log/nginx/**/access.log`. LogListener will listen for log files named `access.log` in all subdirectories in the `/var/log/nginx` prefix path.
2.	/var/log/nginx	*.log	In this example, the log path is configured as `/var/log/nginx/**/*.log`. LogListener will listen for log files suffixed with `.log` in all subdirectories in the `/var/log/nginx` prefix path.
3.	/var/log/nginx	error*	In this example, the log path is configured as `/var/log/nginx/**/error*`. LogListener will listen for log files prefixed with `error` in all subdirectories in the `/var/log/nginx` prefix path.

Note:

Only LogListener 2.3.9 and later support adding multiple collection paths.

The system does not support uploading logs with contents in multiple text formats, which may cause write failures, such as `key:"{"substream":XXX}"`.

We recommend you configure the collection path as `log/*.log` and rename the old file after log rotation as `log/*.log.xxxx`.

By default, a log file can only be collected by one log topic. If you want to have multiple collection configurations for the same file, add a soft link to the source file and add it to another collection configuration.

Configuring the combined parsing mode

On the **Collection Configuration** page, select **Combined Parsing** as the **Extraction Mode**.

Configuring the collection policy

Full collection: When LogListener collects a file, it starts reading data from the beginning of the file.

Incremental collection: When LogListener collects a file, it collects only the newly added content in the file.

Use Limits

If the combined parsing mode is used for data parsing, LogListener will consume more resources. We recommend you not use overly complex plug-in combinations to process data.

If the combined parsing mode is used, the collection and filter features of the text mode will become invalid, but some of these features can be implemented through relevant user-defined plug-ins.

If the combined parsing mode is used, the feature of uploading logs that fail to be parsed is enabled by default. For logs that fail to be parsed, the input name is the `Key` and the original log content is the `Value` for log uploading.

Related Operations

Log search

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to go to the search and analysis page.
3. Select the region, logset, and log topic as needed, and click **Search and Analysis** to search for logs according to the set query rules.

Configuring the Time Format

Last updated：2024-01-20 17:14:28

CLS requires a time attribute for each log so that the system can manage the data by the time dimension. When logs are collected using LogListener, the time attribute can be configured using two methods:
Default method: use LogListener collection time as the time attribute.
Custom method: use a time field in the log content as the time attribute. In this method, you need to configure a time parsing format.
Note:
The time precision of LogListener collection is millisecond. Therefore, the time parsing format needs to be accurate to milliseconds. If the time specified in the required format is less than 1 millisecond, 0 is automatically filled in.

About Parsing Formats

Parameter Format	Description	Example
%a	Abbreviation for a weekday	Fri
%A	Full name for a weekday	Friday
%b	Abbreviation for a month	Jan
%B	Full name for a month	January
%d	A day of a month (01 to 31)	31
%h	Abbreviation for a month, same as `%b`	Jan
%H	An hour in the 24-hour system (00 to 23)	22
%I	An hour in the 12-hour system (01 to 12)	11
%m	Month (01 to 12), with 01 indicating January	08
%M	Minute (00 to 59), with 01 indicating one minute	59
%n	Line break	Line break
%p	Morning (AM) or afternoon (PM)	AM/PM
%r	Specific 12-hour combined time format, equivalent to `%I:%M:%S %p`	11:59:59 AM

%R	Specific 24-hour combined time format, equivalent to `%H:%M`	23:59
%S	Second (00 to 59)	59
%f	Millisecond	0.123
%t	Tab	Tab
%y	Year, without the century (00 to 99)	19
%Y	Year, with the century, with 2018 indicating the year of 2018	2019
%C	Century (obtained by dividing the year by 100, ranging from 00 to 99)	20
%e	A day of a month (01 to 31)	31
%j	A day of a year (001 to 366)	365
%u	Weekday represented by a digit (1 to 7), with 1 indicating Monday and 7 indicating Sunday	1
%U	A week of a year (00 to 53), with the weeks starting from Sunday, that is, the first Sunday as the first day of the first week	23
%w	Weekday represented by a digit (0 to 6), with 0 indicating Sunday and 6 indicating Saturday	5
%W	A week of a year (00 to 53), with the weeks starting from Monday, that is, the first Monday as the first day of the first week	23
%s	Second-level (10-digit) UNIX timestamp	1571394459
%F	Millisecond-level (13-bit) UNIX timestamp	1571394459123
%z	Supports time zone parsing for time fields, including ISO 8601 time format and GMT time format	UTC/+0800/MST

Configuration Samples

Time Indication Sample	Time Extraction Format
2018-07-16 13:12:57.123	%Y-%m-%d %H:%M:%S.%f
[2018-07-16 13:12:57.012]	[%Y-%m-%d %H:%M:%S.%f]
06/Aug/2019 12:12:19 +0800	%d/%b/%Y %H:%M:%S

Monday, 02-Oct-19 16:07:05 MST	%A, %d-%b-%y %H:%M:%S
1571394459	%s
1571394459123	%F (LogListener 2.6.4 or later)
06/Aug/2019 12:12:19 +0800	%d/%b/%Y %H:%M:%S %z
Monday, 02-Oct-19 16:07:05 MST	%A, %d-%b-%y %H:%M:%S %z

Collecting Logs in Self-Built Kubernetes Cluster
Configuring Log Collection in Self-Built Kubernetes Cluster in the Console

Last updated：2024-01-20 17:14:28

This document describes how to configure log collection rules in a self-built Kubernetes environment and ship logs to CLS.

Use Cases

The self-built Kubernetes log collection feature is a non-Tencent Cloud Kubernetes cluster log collection tool, which can ship logs in the specified paths of cluster services or nodes to CLS. This feature is suitable for users who want to store and analyze service logs in a Kubernetes cluster.
You need to manually enable log collection for each cluster, and configure the collection rules. After log collection is enabled for a cluster, the log collection agent runs as a DaemonSet in the cluster, collects logs from the collection source based on the collection source, CLS log topic, and log parsing method configured in the log collection rules, and sends the collected logs to the consumer. Log collection supports the following operations.

Prerequisites

You have activated CLS.
You have installed LogListener in the self-built Kubernetes cluster and deployed it.
You have configured the target log reporting permission as instructed in Access Policy Templates.

Directions

Creating log collection rules

Configuring self-built Kubernetes collection rules in the CLS console

Step 1. Select a log topic

To select a new log topic, perform the following steps: Log in to the CLS console.

On the left sidebar, click **Overview** to enter the overview page.
In the **Other Logs** section, find self-built Kubernetes cluster collection and click **Access Now**.
On the **Create Log Topic** page, configure the log topic information such as the name and log retention period as needed and click **Next**.
To select an existing log topic, perform the following steps: Log in to the CLS console.
On the left sidebar, click **Log Topic** and select a log topic to be shipped to enter the log topic management page.
On the **Collection Configuration** tab, click **Add** in the **Self-Built Kubernetes Cluster Collection** section.

Step 2. Configure a machine group

On the **Machine Group Management** page, select the target machine group and click **Next** to proceed to collection configuration. For more information, see Machine Group Management.

Step 3. Configure collection in the self-built Kubernetes cluster

Log source configuration
1. Collection rule name: You can enter a custom name for a log collection rule.
1. Select the collection type and configure the log source.
Currently, log collection types include container standard output, container file path, and node file path.
Standard output logs of a container
Log sources include **all containers**, **specified workloads**, and **specified Pod labels**.
Container file logs
Log sources include **specified workloads** and **specified Pod labels**.
You can specify a file path or use wildcards for the collection path. For example, when the container file path is `/opt/logs/*.log`, you can specify the collection path as `/opt/logs` and the filename as `*.log`.
Note:
For **Container file path**, the corresponding path **cannot be a soft link**. Otherwise, the actual path of the soft link will not exist in the collector's container, resulting in log collection failure.
Node file logs
You can specify a file path or use wildcards. For example, when the container file paths for collection are `/opt/logs/service1/*.log` and `/opt/logs/service2/*.log`, you can specify the folder of the collection path as `/opt/logs/service*` and the file name as `*.log`.
Metadata configuration
Besides the original log content, the metadata related to the container or Kubernetes (such as the ID of the container that generated the logs) will also be reported to the CLS. Therefore, when viewing logs, users can trace the log source or search based on the container identifier or characteristics (such as container name and labels). You can select whether to report such metadata as needed.
The metadata related to the container or Kubernetes is as shown below:

Field	Description

container_id	ID of the container to which logs belong
container_name	The name of the container to which logs belong
image_name	The image name IP of the container to which logs belong
namespace	The namespace of the Pod to which logs belong
pod_uid	The UID of the Pod to which logs belong
pod_name	The name of the Pod to which logs belong
pod_lable_{label name}	The labels of the Pod to which logs belong (for example, if a Pod has two labels: app=nginx and env=prod, the reported log will have two metadata entries attached: pod_label_app:nginx and pod_label_env:prod).

Note:

To collect logs with a certain Pod label, you need to manually enter the target label key (or enter multiple ones, each of which ends with a carriage return). Logs will be collected if their label is hit.

Parsing rule configuration

1. Configure the collection policy (**Full** or **Incremental**).

Full: Collecting logs from the beginning of the log file.

Incremental: Collecting logs 1 MB ahead of the end of the log file (for a log file less than 1 MB, incremental collection is equivalent to full collection).

2. Encoding Mode: It can be **UTF-8** or **GBK**.

3. Extraction Mode: The following types of extraction modes are supported:

Parsing Mode	Description	Documentation
Full text in a single line	A log contains only one line of content, and the line break `\\n` to mark the end of a log. Each log will be parsed into a complete string with CONTENT as the key value. When log Index is enabled, you can search for log content via full-text search. The time attribute of a log is determined by the collection time.	Full Text in a Single Line
Full text in multi lines	A log with full text in multi lines spans multiple lines and a first-line regular expression is used for match. When a log in a line matches the preset regular expression, it is considered as the beginning of a log, and the next matching line will be the end mark of the log. A default key value, CONTENT, will be set as well. The time attribute of a log is determined by the collection time. The regular expression can be generated automatically.	Full Text in Multi Lines
Single line - full regex	The single-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete log. When configuring the single-line - full regular expression mode, you need to enter a	Full Regular Format (Single-Line)

	sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	
Multiple lines - full regex	The multi-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete piece of log data that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. When configuring the multi-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	Full Regular Format (Multi-Line)
JSON	A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to implement structured processing of the entire log. Each complete log ends with a line break `\\n`.	JSON Format
Separator	Structure the data in a log with the specified separator, and each complete log ends with a line break `\\n`. Define a unique key for each separate field. Leave the field blank if you don't need to collect it. At least one field is required.	Separator Format

Filter: LogListener collects only logs that meet the filter rules. **Key** supports exact match, and **Filter Rule** supports regular expression matching. For example, you can specify to collect only logs where `ErrorCode` is `404`. You can enable the filter and configure rules as needed.

Note:

Currently, one log topic supports only one collection configuration. Ensure that all container logs that adopt the log topic can accept the log parsing method that you choose. If you create different collection configurations under the same log topic, the earlier collection configurations will be overwritten.

Click **Next**.

Step 4. Configure an index

On the index configuration page, configure the following information:

Index Status: select whether to enable it.

Full-Text Index: Select whether to set it to case-sensitive. Full-Text Delimiter: It is "@&()='",;:<>[]{}/ \\n\\t\\r" by default and can be modified as needed.

Allow Chinese Characters: select whether to enable this feature.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, toggle the switch on.

Note:

Index configuration must be enabled first before you can perform searches.

The modified index rules take effect only for newly written logs. The existing data is not updated.

Configuring Log Collection in Self-Built Kubernetes Cluster via CRD

Last updated：2024-01-20 17:14:28

Overview

CLS supports log collection for self-built Kubernetes clusters. Before performing log collection on a self-built Kubernetes cluster, you need to use a custom resource definition (CRD) to define log collection configuration (LogConfig), and deploy Log-Provisioner, Log-Agent, and LogListener on the cluster. If you are a Tencent Kubernetes Engine (TKE) user, you can quickly access and use the CLS service by referring to Enabling log collection.

Prerequisites

You have created a cluster of Kubernetes 1.10 or above.
You have configured TencentCloud API permissions required for self-built Kubernetes log collection. For more information, see Examples of Custom Access Policies.
You have enabled CLS, created a logset and a log topic, and obtained the log topic ID (`topicId`).
For more information, see Managing Log Topic.
You have obtained the domain name (`CLS_HOST`) of the region of your log topic.
For details of the CLS domain name list, see Available Regions.
You have obtained the API key ID (`TmpSecretId`) and API key (`TmpSecretKey`) required for CLS authentication.
To obtain the API key and API key ID, go to Manage API Key.

Kubernetes Log Collection Principles

Log collection on a Kubernetes cluster requires the following:
LogConfig: CRD of log collection configuration, which defines where logs are collected, how they are parsed, and to which CLS log topic they are shipped after being parsed.
Log-Agent: component used to listen for changes in LogConfig and containers on nodes and dynamically calculate the actual positions of log files in containers on node hosts.
Log-Provisioner: component used to synchronize the log collection configuration defined in LogConfig to CLS.
LogListener: component used to collect log file content from node hosts, parses it, and uploads it to CLS.

Flowchart

1. Install LogListener in a self-built Kubernetes cluster
2. Define the LogConfig object
3. Create a LogConfig object

Directions

Step 1. Install LogListener in a self-built Kubernetes cluster

First, you need to install the LogListener component in the self-built Kubernetes cluster as instructed here to collect logs to CLS.

Step 2. Define the LogConfig object

Define log collection configuration in the LogConfig object via CRD. Run the `wget` command to download the `LogConfig.yaml` CRD declaration file, using the master node path `/usr/local/` as an example.

```
wget https://mirrors.tencent.com/install/cls/k8s/LogConfig.yaml
```

The `LogConfig.yaml` declaration file consists of the following two parts:

`clsDetail` : The configuration for shipping to CLS.

`inputDetail` : Log source configuration.

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig                                    ## Default value
metadata:
  name: test                                                    ## CRD resource nam
spec:
  clsDetail:                                                          ## The conf
    ...
  inputDetail:                                     ## Log source configuration
    ...
```

clsDetail (the configuration for shipping to CLS) field description

```
  clsDetail:
    # You need to specify the logset and topic names to automatically create a log
    logsetName: test                                ## The name of the CLS logset. If t
    topicName: test                             ## The name of the CLS log topic. If the
    # Select an existing logset and log topic. If the logset is specified but the l
    logsetId: xxxxxx-xx-xx-xx-xxxxxxxx            ## The ID of the CLS logset. The l
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx    ## The ID of the CLS log topic. The log to
```

```
    region: ap-xxx                          ## Topic region for cross-region shippin
    # Define the log topic configuration when a log topic is created automatically.
    period: 30                                                  ## Lifecycle in day
    storageType: hot.                             ## Log topic storage class. Valid
    HotPeriod: 7                                ## Transition cycle in days. Value ra
    partitionCount:                       ## The number (an integer) of log topic par
    autoSplit: true                                             ## Whether to enabl
    maxSplitPartitions: 10                        ## The maximum number (an integer)
    tags:                                         ## Tag description list. This param
      - key: xxx                                                        ## Tag key
        value: xxx                                ## Tag value
    # Define collection rules
    logType: json_log                                    ## Log parsing format. Vali
    logFormat: xxx                                ## Log formatting method
    excludePaths:                                 ## Collection path blocklist
      - type: File                                                      ## Type. Va
        value: /xx/xx/xx/xx.log                   ## The value of `type`
    userDefineRule: xxxxxx                  ## Custom collection rule, which is a se
    extractRule: {}                               ## Extraction and filter rule. If `
    AdvancedConfig:                                             ## Advanced collect
      MaxDepth: 1                                               ## Maximum number o
      FileTimeout: 60                                  ## File timeout attribute
    # Define index configuration, which cannot be modified then.
    indexs:                                                                 ## You can
      - indexName:                                              ## The field for wh
        indexType:                                   ## Field type. Valid values
        tokenizer:                                              ## Field delimiter.
        sqlFlag:                                                ## Whether the anal
        containZH:                                   ## Whether Chinese characte
```

extractRule description

Name	Type	Required	Description
timeKey	String	No	The specified field in the log to be used as the log timestamp. If the configuration is empty, the actual log collection time will be used. `time_key` and `time_format` must appear in pairs.
timeFormat	String	No	Time field format. For more information, see the output parameters of the time format description of the `strftime` function in C programming language.
delimiter	String	No	The delimiter for delimited logs, which is valid only if `log_type` is `delimiter_log`.
logRegex	String	No	Full log matching rule, which is valid only if `log_type` is

			`fullregex_log` .
beginningRegex	String	No	First-line matching rule, which is valid only if `log_type` is `multiline_log` or `multiline_fullregex_log` .
unMatchUpload	String	No	Whether to upload the logs failed to be parsed. Valid values: `true` (yes); `false` (no).
unMatchedKey	String	No	The key of the log failed to be parsed.
backtracking	String	No	The size of the data to be rewound in incremental collection mode. Valid values: `-1` (full collection); `0` (incremental collection). Default value: `-1` .
keys	Array of String	No	The key name of each extracted field. An empty key indicates to discard the field. This parameter is valid only if `log_type` is `delimiter_log` , `fullregex_log` , or `multiline_fullregex_log` . `json_log` logs use the key of JSON itself.
filterKeys	Array of String	No	Log keys to be filtered, which correspond to `FilterRegex` by subscript.
filterRegex	Array of String	No	The `regex` of the log keys to be filtered, which corresponds to `FilterKeys` by subscript.
isGBK	String	No	Whether it is GBK-encoded. Valid values: `0` (no); `1` (yes). Note: This field may return null, indicating that no valid values can be obtained.

Log collection rule configuration sample

Full text in a single line
Full text in multi lines
Single line - full regex
Multiple lines - full regex
JSON format
Separator format

In "full text in a single line" mode, a line is a full log. When CLS collects logs, it uses the line break `\\n` to mark the end of a log. For easier structural management, a default key value `\\_\\_CONTENT\\_\\_` is given to each log, but the log data itself will no longer be structured, nor will the log field be extracted. The time attribute of a log is determined by the collection time.

Assume that the raw data of a log is as follows:

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-
6.el7.x86_64
```

A sample of LogConfig configuration is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # Single-line log
    logType: minimalist_log
```

The data collected to CLS is as follows:

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-
1.2.90-6.el7.x86_64
```

In "full text in multi lines" mode, a log may span multiple lines (such as Java stacktrace), and the line break `\\n` cannot be used to mark the end of a log. To help CLS distinguish between logs, a first-line regular expression is used for matching. When a line of a log matches the preset regular expression, it is considered as the beginning of the log, and the log ends before the next matching line.

In the "full text in multi lines" mode, a default key `\\_\\_CONTENT\\_\\_` is also set, but the log data itself is not structured, and no log fields are extracted. The time attribute of a log is determined by the collection time.

Assume that the raw data of a multi-line log is:

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

A sample of LogConfig is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # Multi-line log
    logType: multiline_log
    extractRule:
      # Only a line that starts with a date time is considered the beginning of a n
      beginningRegex: \\d{4}-\\d{2}-\\d{2}\\s\\d{2}:\\d{2}:\\d{2},\\d{3}\\s.+
```

The data collected to CLS is as follows:

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\\njava.lang.NullPointerException\\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

The "single line - full regex" mode is often used to process structured logs. It parses a full log by extracting multiple key-value pairs based on a regex.

Assume that the raw data of a log is as follows:

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1"
127.0.0.1 200 782 9703 "http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0"  0.354 0.354
```

A sample of LogConfig is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # Full Regex
    logType: fullregex_log
    extractRule:
      # Regular expression, in which the corresponding values will be extracted
based on the `()` capture groups
      logRegex: (\\S+)[^\\[]+(\\[[^:]+:\\d+:\\d+:\\d+\\s\\S+)\\s"
(\\w+)\\s(\\S+)\\s([^"]+)"\\s(\\S+)\\s(\\d+)\\s(\\d+)\\s(\\d+)\\s"([^"]+)"\\s"
([^"]+)"\\s+(\\S+)\\s(\\S+).*
      beginningRegex: (\\S+)[^\\[]+(\\[[^:]+:\\d+:\\d+:\\d+\\s\\S+)\\s"
(\\w+)\\s(\\S+)\\s([^"]+)"\\s(\\S+)\\s(\\d+)\\s(\\d+)\\s(\\d+)\\s"([^"]+)"\\s"
([^"]+)"\\s+(\\S+)\\s(\\S+).*
      # List of extracted keys, which are in one-to-one correspondence with the
extracted values
      keys:
['remote_addr','time_local','request_method','request_url','http_protocol','htt
p_host','status','request_length','body_bytes_sent','http_referer','http_user_a
gent','request_time','upstream_response_time']
```

The data collected to CLS is as follows:

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
```

```
By=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

In "multiple lines - full regex" mode, multiple key-value pairs can be extracted from a complete piece of log data that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. If you don't need to extract key-value pairs, please configure it by referring to the "full text in multi lines" mode.

Assume that the raw data of a log is as follows:

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
   at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
   at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
   at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

A sample of LogConfig is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
        # Multiple lines - full regex
        logType: multiline_fullregex_log
        extractRule:
          # The first-line full regular expression: only a line that starts with a
          beginningRegex: \\[\\d+-\\d+-\\w+:\\d+:\\d+,\\d+\\]\\s\\[\\w+\\]\\s.*
          # Regular expression, in which the corresponding values will be extracted
          logRegex: \\[(\\d+-\\d+-\\w+:\\d+:\\d+,\\d+)\\]\\s\\[(\\w+)\\]\\s(.*)
          # List of extracted keys, which are in one-to-one correspondence with the
          keys: ['time','level','msg']
```

Based on the extracted key, the data collected to CLS is as follows:

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg: java.lang.Exception: exception happened
   at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
   at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
   at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to implement structured processing of the entire log. Each complete log ends with a line break `\\n`.

Assume the raw data of a JSON log is as follows:

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34
+0800","body_sent":23,"responsetime":0.232,"upstreamtime":"0.232","upstreamhost
":"unix:/tmp/php-
cgi.sock","http_host":"127.0.0.1","method":"POST","url":"/event/dispatch","requ
est":"POST /event/dispatch HTTP/1.1","xff":"-
","referer":"http://127.0.0.1/my/course/4","agent":"Mozilla/5.0 (Windows NT
10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0","response_code":"200"}
```

A sample of LogConfig is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # JSON log
    logType: json_log
```

The data collected to CLS is as follows:

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

For a log in separator format (separator log), the entire log data can be structured according to the specified separator, and each complete log ends with a line break `\\n`. When CLS processes separator logs, you need to define a unique key for each separate field.

Assume the raw data of a log is as follows:

```
10.20.20.10 ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample
HTTP/1.1 ::: 127.0.0.1 ::: 200 ::: 647 ::: 35 ::: http://127.0.0.1/
```

A sample of LogConfig is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
    # Separator log
    logType: delimiter_log
    extractRule:
      # Separator
      delimiter: ':::'
      # List of extracted keys, which are in one-to-one correspondence to the
separated fields
      keys: ['IP','time','request','host','status','length','bytes','referer']
```

The data collected to CLS is as follows:

```
IP: 10.20.20.10
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

inputDetail (log source) field description

```
  inputDetail:
    type: container_stdout                                  ## Log collection type. Val
    containerStdout:                                                ## Container standa
      namespace: default                                            ## The Kubernetes n
      excludeNamespace: nm1,nm2                             ## The Kubernetes namespace
      nsLabelSelector: environment in (production),tier in (frontend) ## The namesp
      allContainers: false                                          ## Whether to colle
      containerOperator: in                             ## Container selection method. Val
      container: xxx                                                ## The name of the
      includeLabels:                                                ## The labels of th
        key: value1                                                 ## Pods with multip
      excludeLabels:                                                   ## The labels of
        key2: value2                                                ## Pods with multip
      metadataLabels:                                               ## The Pod labels t
```

```
      - label1
      metadataContainer:                                                ## The cont
      - namespace
      customLabels:                                                  ## Custom metadata
        label: l1
      workloads:                                                                ##
      - container: xxx                                               ## The name of the
        containerOperator: in                              ## Container selection method. V
        kind: deployment                                             ## Workload type. V
        name: sample-app                                             ## Workload name
        namespace: prod                                              ## Workload namespa

    containerFile:                                                          ## Containe
      namespace: default                                        ## The Kubernetes namespace
      excludeNamespace: nm1,nm2                        ## The Kubernetes namespace of the
      nsLabelSelector: environment in (production),tier in (frontend) ## The namesp
      containerOperator: in                             ## Container selection method. Val
      container: xxx                                           ## The name of the containe
      logPath: /var/logs                                       ## Log folder. Wildcards ar
      filePattern: app_*.log                                          ## Log filename. Wi
      includeLabels:                                                  ## The labels of th
        key: value1                                                   ## The `metadata` w
      excludeLabels:                                                  ## Pods with the sp
        key2: value2                                                          ## Pods wit
      metadataLabels:                                          ## The Pod labels to be col
      - namespace
      metadataContainer:                                              ## The container en
      customLabels:                                                   ## Custom metadata
        key: value
      workload:
        container: xxx                                                ## The name of the
        containerOperator: in                              ## Container selection method. V
        kind: deployment                                              ## Workload type. V
        name: sample-app                                                 ## Workload name
        namespace: prod                                               ## Workload namespa

    hostFile:                                                         ## Node file path,
      filePattern: '*.log'                                            ## Log filename. Wi
      logPath: /tmp/logs                                              ## Log folder. Wild
      customLabels:                                                   ## Custom metadata
        label1: v1
```

Log source configuration sample

Standard container output

Container file

Host file

Sample 1: collecting the standard output of all containers in the default namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  inputDetail:
    type: container_stdout
    containerStdout:
      namespace: default
      allContainers: true
 ...
```

Sample 2: collecting the container standard output in the Pod that belongs to ingress-gateway deployment in the production namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  inputDetail:
    type: container_stdout
    containerStdout:
      allContainers: false
      workloads:
      - namespace: production
        name: ingress-gateway
        kind: deployment
  ...
```

Sample 3: collecting the container standard output in the Pod whose Pod labels contain "k8s-app=nginx" in the production namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  inputDetail:
    type: container_stdout
    containerStdout:
      namespace: production
      allContainers: false
      includeLabels:
        k8s-app: nginx
  ...
```

Sample 1: collecting the `access.log` file in the `/data/nginx/log/` path in the NGINX container in the Pod that belongs to ingress-gateway deployment in the production namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  inputDetail:
    type: container_file
    containerFile:
      namespace: production
      workload:
        name: ingress-gateway
        kind: deployment
      container: nginx
      logPath: /data/nginx/log
      filePattern: access.log
  ...
```

Sample 2: collecting the `access.log` file in the `/data/nginx/log/` path in the NGINX container in the Pod whose pod labels contain "k8s-app=ingress-gateway" in the production namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  inputDetail:
    type: container_file
    containerFile:
      namespace: production
      includeLabels:
        k8s-app: ingress-gateway
      container: nginx
      logPath: /data/nginx/log
      filePattern: access.log
  ...
```

Sample: collecting all `.log` files in the host path `/data/`

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  inputDetail:
    type: host_file
    hostFile:
      logPath: /data
      filePattern: *.log
  ...
```

Step 3. Create a LogConfig object

As the `LogConfig.yaml` declaration file is defined in Step 2. Define the LogConfig object, you can run the kubectl command to create a LogConfig object based on the file.

```
kubectl create -f /usr/local/LogConfig.yaml
```

Related Operations

After the deployment for cluster log collection is completed, you can go to CLS console > Search and Analysis to view collected logs.

Collecting Syslog

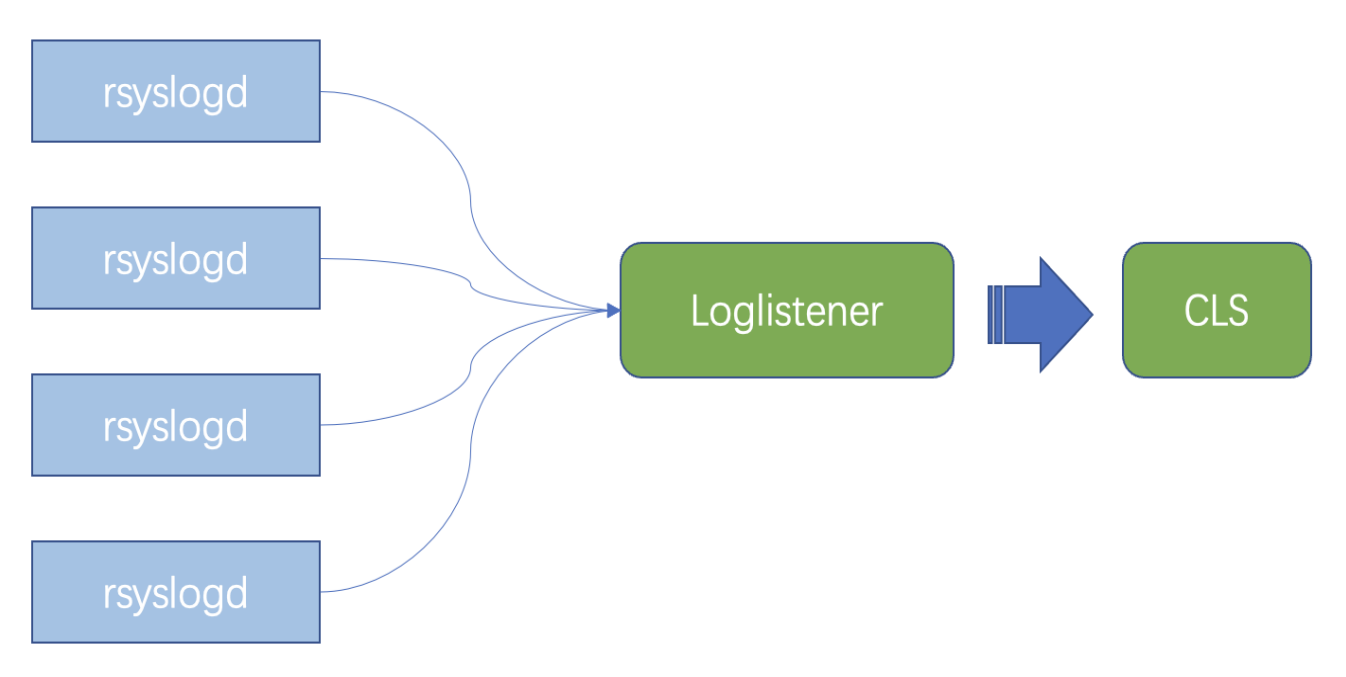
Last updated：2024-01-20 17:14:28

Overview

Syslog refers to system logs or records and is a standard for sending log messages in internet protocols. It is supported by network routers, switches, firewalls, and UNIX/Linux servers. Syslog monitoring and management are important for business operations, helping reduce system downtime, improve network performance, and enhance security policies.

Prerequisites

You have deployed RSyslog.
You have activated CLS.
You have installed LogListener 3.0.1.0 or later on the target server with the RSyslog IP.
The configuration in the console is made available through an allowlist. Submit a ticket for application.
Use `rsyslog/etc/rsyslog.conf` to enable UDP/TCP forwarding:



Note:
For detailed directions on how to install LogListener, see LogListener Installation Guide.

Directions

Configuring RSyslog forwarding

On the syslog server, modify RSyslog's configuration file `/etc/rsyslog.conf` by adding the forwarding rule in the last line. Then, RSyslog will forward syslog to the specified IP and port. If the current server is used to collect local syslog, the forwarding address should be `127.0.0.1`, and the port can be a random idle port.
If another server is used to collect local syslog, the forwarding address should be the public network IP of the server, and the port can be a random idle port.
The following configuration indicates to forward all logs to `127.0.0.1:1000` over TCP. For more information on the configuration file, see RSyslog Documentation.

```
*.* @@127.0.0.1:1000
```

Run the following command to restart RSyslog for the log forwarding rule to take effect.

```
sudo service rsyslog restart
```

Configuring the syslog collection rule in the CLS console

Step 1. Select a log topic

To select a new log topic, perform the following steps: Log in to the CLS console.
On the left sidebar, click **Overview** to enter the overview page.
In the **Other Logs** section, find syslog collection and click **Access Now**.
On the **Create Log Topic** page, configure the log topic information such as the name and log retention period as needed and click **Next**.
To select an existing log topic, perform the following steps: Log in to the CLS console.
On the left sidebar, click **Log Topic** and select the target log topic to enter the log topic management page.
On the **Collection Configuration** tab, click **Add** in the **LogListener Collection Configuration** section.

Step 2. Configure a machine group

On the **Machine Group Management** page, select the machine group to which to bind the current log topic and click **Next** to proceed to collection configuration. For more information, see Machine Group.

Step 3. Configure syslog collection

On the syslog collection configuration page, configure the following information:

Configuration Item	Type	Description
Collection Rule	Input box	Indicates the name of this collection rule.

Name		
Network Type	Radio button	Specifies the syslog transfer protocol: UDP or TCP.
Parsing Protocol	Radio button	Specifies the protocol for log parsing. It is empty by default, indicating no parsing. Valid values: `rfc3164` (RFC 3164), `rfc 5424` (RFC5424), `auto` (automatic selection).
Output Source	Input box	Specifies the protocol, address, and port for LogListener. It is in the format of [tcp/udp]://[ip]:[port]. If it is not specified, tcp://127.0.0.1:10000 will be used by default.
Upload upon Parsing Failure	Toggle	Specifies the operation upon parsing failure. If it is enabled, the full text of the log will be returned based on the input `key` ; otherwise, the log will be discarded.
Key Name of Parsing-Failed Logs	Input box	Specifies the key name of logs that failed to be parsed.

Step 4. Configure an index

1. On the index configuration page, configure the following information:

Index Status: Select whether to enable it.

Full-Text Index: Select whether to set it to case-sensitive. Full-Text Delimiter: It is "@&()='",;:<>[]{}/ \\n\\t\\r" by default and can be modified as needed.

Allow Chinese Characters: Select whether to enable this feature.

Key-Value Index: Disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, toggle the switch on.

Note:

Index configuration must be enabled first before you can perform searches.

The modified index rules take effect only for newly written logs. The existing data is not updated.

2. Click **Submit**.

Viewing syslog

After configuring syslog collection in the current log topic, click **Search** to enter the **Search and Analysis** page to view the syslog.

Field	Description
HOSTNAME	Hostname. The current hostname will be obtained if it is not provided in the log.

program	`tag` field in the protocol.
priority	`priority` field in the protocol.
facility	`facility` field in the protocol.
severity	`severity` field in the protocol.
timestamp	Timestamp of the log.
content	Log content, which will contain all the content of unparsed logs if parsing fails.
SOURCE	IP of the current host.
client_ip	Client IP for log transfer.

Collecting Windows Event Logs

Last updated：2025-04-18 16:20:44

Windows event log record provides a standardized and centralized method for recording critical software and hardware events for both applications and the operating system. When an unexpected error occurs with an application, hardware component, or operating system, Windows event logs enable you to diagnose the root cause of the error. This document will guide you on how to collect Windows event logs and submit them to the CLS using LogListener (Windows edition).

Prerequisites

The target Windows server has LogListener installed. For details, see LogListener Installation Guide (Windows edition).

Directions

Step 1: Selecting a Log Topic

You can either create a log topic or use an existing one as needed.
Using a New Log Topic
Using an Existing Log Topic
To select a new log topic, perform the following steps:
1. Log in to the Cloud Log Service console.
2. In the left sidebar, select **Overview** to enter the overview page.
3. In quick access, find and click **Windows event log** to enter the collection configuration process.

4. On the **Create Log Topic** page, configure the log topic information such as the name and log retention period as needed, and click **Next**.

To select an existing log topic, perform the following steps:
1. Log in to the Cloud Log Service console.
2. In the left sidebar, select **Log Topic**. Then, select the log topic you want to ship and click its name to access the log topic management page.
3. Select the **Collection Configuration** tab, and click **Add** in the **Windows event collection configuration** column.

Step 2: Managing Machine Groups

1. After you have created or selected the log topic, proceed to **Machine Group Management** for configuration.
2. Select the Windows machine group from which you want to collect logs in the list of machine groups.

If you want to select a new machine group, click **Create now**, select **Windows** for **System**, and associate the target Windows server by IP or machine identification. For details, see Machine Group. After completing the creation, select the newly created machine group in the machine group list.

3. Click **Next** to enter the **Collection Configuration** process.

Step 3: Collection Configuration

On the **Collection Configuration** page, configure the rules for Windows event collection. After you have completed the configuration, click **Next**.

Configuring Event Collection Rules

A collection configuration enables you to configure multiple event collection rules, with each rule including the following configuration items:

Configuration Item	Required	Description
Event channel	Yes	It indicates the event channel designated for target collection, with the following configuration options available: Application (application event): Records events generated by applications, such as software crashes, configuration changes, and error messages. System (system event): Records events related to operating system components, such as drivers, system services, and hardware issues. Security (security event): Records events related to security, such as user logins/logouts, permission changes, and audit policy changes. Setup (configuration event): Records events related to system setup and configuration changes. ALL (all events). **Note:** It is recommended that each event channel on a server be dedicated to a single collection configuration. Using the same event channel for multiple collection configurations can result in data duplication.
Start time	Yes	The following two options are supported: Custom time: Event logs will be collected starting from the time you specify. Full collection: All event logs from the server will be collected. Note: If an event exceeds the retention period set by the Windows system, its logs will not be collected.

Custom Time	Yes	It is required to specify the time for collecting event logs when Start time is set to Custom time.
Event ID	No	Support positive filtering for specific values (such as 20) or value ranges (such as 0-20), as well as negative filtering for individual values (such as -20). Multiple filter criteria can be separated by commas. For example, "1-200,-100" indicates that event logs will be collected within the range of 1-200, excluding those with an event ID of 100.

Step 4: Index Configuration

In the **Index Configuration** process, set the following information:

Index Status: Select whether to enable it.

Note:

The index configuration must be enabled for retrieval, otherwise the retrieval cannot be performed.

Full-Text Index: Select whether to set it to case-sensitive.

Full-Text Delimiter: It has a default value of @&()='",;:<>[]{}/ \\n\\t\\r. You can modify it as needed.

Key-Value Index: It is enabled by default and populated based on Windows event log fields. To disable it, set

to

.

Note:

Enabling key-value indexing along with full-text indexing **does not incur any extra fees.**

Step 5: Search and Analysis

At this point, you have completed the collection configuration of Windows event logs. Next, you can go to the Log Search page to view the logs.

Log Field Explanation

Field Name	Description
computer_name	Name of the node that generates the current event.
keywords	Keyword associated with the current event, used for event categorization.
level	Level of the current event.
channel	Channel name of the current event.

event_data	Data related to the current event.
message	Messages associated with the current event.
opcode	Operation code associated with the current event.
process.pid	Process ID of the current event.
type	API used to obtain the current event.
version	Version number of the current event.
record_id	Record number associated with the current event.
event_id	ID of the current event.
task	Task associated with the current event.
provider_guid	Global transaction ID of the current event's source.
activity_id	Global transaction ID of the event's associated activity. All events occurring within this activity will share the same global transaction ID.
process.thread.id	Thread ID of the current event.
provider_name	Source of the current event.
raw_data	Original information of the current event, in XML format.

Uploading Log over Kafka

Last updated：2024-01-20 17:14:28

CLS allows you to upload logs to CLS by using Kafka Producer SDKs or other Kafka related agents.

Overview

Using Kafka as a message pipeline is common in log applications. First, the open source collection client or the producer on the machine directly writes logs to be collected, and then provides them to the downstream, such as Spark and Flink, for consumption through the Kafka message pipeline. CLS has complete upstream and downstream capabilities of the Kafka message pipeline. The following describes the scenarios suitable for you to upload logs using the Kafka protocol. For more Kafka protocol consumption scenarios, see Kafka Real-time Consumption.
Scenario 1: You already have a self-built system based on open source collection and you do not want complex secondary modifications. Then you can upload logs to CLS by modifying configuration files.
For example, if you have set up a log system using ELK, now you only need to modify the Filebeat or Logstash configuration file to configure the output destination (see Filebeat configuration) to CLS to implement convenient and simple log upload to CLS.
Scenario 2: If you want to use Kafka producers to collect and upload logs, you do not need to install collection agents. CLS allows you to use various Kafka producer SDKs to collect logs and upload the logs to CLS via the Kafka protocol. For more information, see SDK call examples in this document.

Use Limits

Supported Kafka protocol versions: 0.11.0.X, 1.0.X, 1.1.X, 2.0.X, 2.1.X, 2.2.X, 2.3.X, 2.4.X, 2.5.X, 2.6.X, 2.7.X, 2.8.X
Supported compression modes: Gzip, Snappy, LZ4
Current authentication mode: SASL_PLAINTEXT
Upload over Kafka requires the `RealtimeProducer` permission. For more information, see Access Policy Templates.

Configuration Methods

To upload logs via Kafka, you need to set the following parameters:

Parameter	Description
LinkType	Currently, SASL_PLAINTEXT is supported.

hosts	Address of the initially connected cluster. For more information, see Service Entries.
topic	Log topic ID. Example: 76c63473-c496-466b-XXXX-XXXXXXXXXXXX
username	Logset ID. Example: 0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX
password	Password in the format of `${SecurityId}#${SecurityKey}` . Example: XXXXXXXXXXXXXX#YYYYYYYY

Service Entries

Region	Network Type	Port Number	Service Entry
Guangzhou	Private network	9095	gz-producer.cls.tencentyun.com:9095
	Public network	9096	gz-producer.cls.tencentcs.com:9096

Note:

This document uses the Guangzhou region as an example. The private and public domain names are identified by different ports. For other regions, replace the address prefixes. For more information, see here.

Examples

Agent call examples

Filebeat/Winlogbeat configuration

```
output.kafka:
  enabled: true
  hosts: ["${region}-producer.cls.tencentyun.com:9095"] # TODO: Service address. Th
  topic: "${topicID}" #  TODO: Topic ID
  version: "0.11.0.2"
  compression: "${compress}"   # Configure the compression method. Valid values: `g
  username: "${logsetID}"
  password: "${SecurityId}#${SecurityKey}"
```

Logstash example

```
output {
  kafka {
    topic_id => "${topicID}"
    bootstrap_servers => "${region}-producer.cls.tencentyun.com:${port}"
    sasl_mechanism => "PLAIN"
    security_protocol => "SASL_PLAINTEXT"
    compression_type => "${compress}"
    sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginModule re
  }
}
```

SDK call examples

Golang SDK call example

```
import (
    "fmt"
    "github.com/Shopify/sarama"
)

func main(){
    config := sarama.NewConfig()

    config.Net.SASL.Mechanism = "PLAIN"
    config.Net.SASL.Version = int16(1)
    config.Net.SASL.Enable = true
    config.Net.SASL.User = "${logsetID}"                          // TODO: Logset ID
    config.Net.SASL.Password = "${SecurityId}#${SecurityKey}"     // TODO: Format: ${
    config.Producer.Return.Successes = true
    config.Producer.RequiredAcks = ${acks}                        // TODO: Select the
    config.Version = sarama.V1_1_0_0
    config.Producer.Compression = ${compress}                     // TODO: Configurat

    // TODO: Service address. The public network port is 9096, and the private netw
    producer, err := sarama.NewSyncProducer([]string{"${region}-producer.cls.tencen
    if err != nil{
        panic(err)
    }

    msg := &sarama.ProducerMessage{
        Topic: "${topicID}", // TODO: Topic ID
        Value: sarama.StringEncoder("goland sdk sender demo"),
    }
    // Send the messages
    for i := 0; i <= 5; i++ {
```

```
        partition, offset, err := producer.SendMessage(msg)
        if err != nil{
            panic(err)
        }
        fmt.Printf("send response; partition:%d, offset:%d\\n", partition, offset)
    }

    _ = producer.Close()

}
```

Python SDK call example

```
from kafka import KafkaProducer

if __name__ == '__main__':
    produce = KafkaProducer(
        # TODO: Service address. The public network port is 9096, and the private n
        bootstrap_servers=["${region}-producer.cls.tencentyun.com:9095"],
        security_protocol='SASL_PLAINTEXT',
        sasl_mechanism='PLAIN',
        # TODO: Logset ID
        sasl_plain_username='${logsetID}',
        # TODO: Format: ${SecurityId}#${SecurityKey}
        sasl_plain_password='${SecurityId}#${SecurityKey}',
        api_version=(0, 11, 0),
        # TODO: Configuration compression mode
        compression_type="${compress_type}",
    )

    for i in range(0, 5):
        # TODO: Topic ID of the sent message
        future = produce.send(topic="${topicID}", value=b'python sdk sender demo')
        result = future.get(timeout=10)
        print(result)
```

Java SDK call example

Maven dependencies:

```
<dependencies>
  <!--https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients-->
  <dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>0.11.0.2</version>
```

```
  </dependency>
</dependencies>
```

Sample code:

```
import org.apache.kafka.clients.producer.*;

import java.util.Properties;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class ProducerDemo {
    public static void main(String[] args) throws InterruptedException, ExecutionEx
        // 0. Set parameters
        Properties props = new Properties();
        // TODO: In use
        props.put("bootstrap.servers", "${region}-producer.cls.tencentyun.com:9095"
        // TODO: Set the following according to the actual business scenario
        props.put("acks", ${acks});
        props.put("retries", ${retries});
        props.put("batch.size", ${batch.size});
        props.put("linger.ms", ${linger.ms});
        props.put("buffer.memory", ${buffer.memory});
        props.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "${compress_type}"); // T
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSe
        props.put("value.serializer", "org.apache.kafka.common.serialization.String

        props.put("security.protocol", "SASL_PLAINTEXT");
        props.put("sasl.mechanism", "PLAIN");
        // TODO: The user name is logsetID, and the password is the combination of
        props.put("sasl.jaas.config",
                "org.apache.kafka.common.security.plain.PlainLoginModule required u

        // 1. Create a producer object.
        Producer<String, String> producer = new KafkaProducer<String, String>(props

        // 2. Call the send method.
        Future<RecordMetadata> meta = producer.send(new ProducerRecord<String, Stri
        RecordMetadata recordMetadata = meta.get(${timeout}, TimeUnit.MILLISECONDS)
        System.out.println("offset = " + recordMetadata.offset());

        // 3. Close the producer.
        producer.close();
    }
}
```

SDK for C call example

```
// https://github.com/edenhill/librdkafka - master
#include <iostream>
#include <librdkafka/rdkafka.h>
#include <string>
#include <unistd.h>

#define BOOTSTRAP_SERVER "${region}-producer.cls.tencentyun.com:${port}"
#define USERNAME "${logsetID}"
#define PASSWORD "${SecurityId}#${SecurityKey}"
#define TOPIC "${topicID}"
#define ACKS "${acks}"
#define COMPRESS_TYPE "${compress_type}"

static void dr_msg_cb(rd_kafka_t *rk, const rd_kafka_message_t *rkmessage, void *op
    if (rkmessage->err) {
        fprintf(stdout, "% Message delivery failed : %s\\n", rd_kafka_err2str(rkme
    } else {
        fprintf(stdout, "% Message delivery successful %zu:%d\\n", rkmessage->len,
    }
}

int main(int argc, char **argv) {
    // 1. Initialize the configuration.
    rd_kafka_conf_t *conf = rd_kafka_conf_new();

    rd_kafka_conf_set_dr_msg_cb(conf, dr_msg_cb);

    char errstr[512];
    if (rd_kafka_conf_set(conf, "bootstrap.servers", BOOTSTRAP_SERVER, errstr, size
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;
    }

    if (rd_kafka_conf_set(conf, "acks", ACKS, errstr, sizeof(errstr)) != RD_KAFKA_C
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;
    }

    if (rd_kafka_conf_set(conf, "compression.codec", COMPRESS_TYPE, errstr, sizeof(
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;
```

```
    }

    // Set the authentication method.
    if (rd_kafka_conf_set(conf, "security.protocol", "sasl_plaintext", errstr, size
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;
    }
    if (rd_kafka_conf_set(conf, "sasl.mechanisms", "PLAIN", errstr, sizeof(errstr))
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;
    }
    if (rd_kafka_conf_set(conf, "sasl.username", USERNAME, errstr, sizeof(errstr))
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;

    }
    if (rd_kafka_conf_set(conf, "sasl.password", PASSWORD, errstr, sizeof(errstr))
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "%s\\n", errstr);
        return -1;
    }

    // 2. Create a handler.
    rd_kafka_t *rk = rd_kafka_new(RD_KAFKA_PRODUCER, conf, errstr, sizeof(errstr));
    if (!rk) {
        rd_kafka_conf_destroy(conf);
        fprintf(stdout, "create produce handler failed: %s\\n", errstr);
        return -1;
    }

    // 3. Send data.
    std::string value = "test lib kafka ---- ";
    for (int i = 0; i < 100; ++i) {
        retry:
        rd_kafka_resp_err_t err = rd_kafka_producev(
                rk, RD_KAFKA_V_TOPIC(TOPIC),
                RD_KAFKA_V_MSGFLAGS(RD_KAFKA_MSG_F_COPY),
                RD_KAFKA_V_VALUE((void *) value.c_str(), value.size()),
                RD_KAFKA_V_OPAQUE(nullptr), RD_KAFKA_V_END);

        if (err) {
            fprintf(stdout, "Failed to produce to topic : %s, error : %s", TOPIC, r
            if (err == RD_KAFKA_RESP_ERR__QUEUE_FULL) {
                rd_kafka_poll(rk, 1000);
```

```
                goto retry;
            }
        } else {
            fprintf(stdout, "send message to topic successful : %s\\n", TOPIC);
        }

        rd_kafka_poll(rk, 0);
    }

    std::cout << "message flush final" << std::endl;
    rd_kafka_flush(rk, 10 * 1000);

    if (rd_kafka_outq_len(rk) > 0) {
        fprintf(stdout, "%d message were not deliverer\\n", rd_kafka_outq_len(rk));
    }

    rd_kafka_destroy(rk);

    return 0;
}
```

SDK for C# call example

```
/*
 * This demo only provides the easiest way of using the feature. The specific produ
 * During use, the TODO items in the demo need to be replaced with actual values.
 *
 * Notes:
 *  1. This demo is verified based on Confluent.Kafka 1.8.2.
 *  2. The maximum value of `MessageMaxBytes` cannot exceed 5 MB.
 *  3. This demo adopts the sync mode for production. You can change to the async m
 *  4. You can adjust other parameters during use as instructed at https://docs.con
 *
 * Confluent.Kafka reference: https://docs.confluent.io/platform/current/clients/co
 */


using Confluent.Kafka;

namespace Producer
{
    class Producer
    {
        private static void Main(string[] args)
        {
```

```
            var config = new ProducerConfig
            {
                // TODO: Domain name. For more information, visit https://intl.clou
                BootstrapServers = "${domain}:${port}",
                SaslMechanism = SaslMechanism.Plain,
                SaslUsername = "${logsetID}", // TODO: Logset ID of the topic
                SaslPassword = "${SecurityId}#${SecurityKey}", // TODO: UIN key of
                SecurityProtocol = SecurityProtocol.SaslPlaintext,
                Acks          = Acks.None, // TODO: Assign a value based on the actu
                MessageMaxBytes = 5242880 // TODO: The maximum size of the request
            };

            // deliveryHandler
            Action<DeliveryReport<Null, string>> handler =
                r => Console.WriteLine(!r.Error.IsError ? $"Delivered message to {r


            using (var produce = new ProducerBuilder<Null, string>(config).Build())
            {
                try
                {
                    // TODO: Test verification code
                    for (var i = 0; i < 100; i++)
                    {
                        // TODO: Replace the log topic ID
                        produce.Produce("${topicID}", new Message<Null, string> { V
                    }
                    produce.Flush(TimeSpan.FromSeconds(10));

                }
                catch (ProduceException<Null, string> pe)
                {
                    Console.WriteLine($"send message receiver error : {pe.Error.Rea
                }
            }
        }
    }
}
```

Uploading Logs via Anonymous Write

Last updated：2024-01-20 17:14:28

CLS allows you to collect logs from the HTML, HTML5, Weixin Mini Programs, iOS, and Android clients.

Use cases

Anonymous write applies to user information collection from browsers, Weixin Mini Programs, iOS apps, and Android apps. The information includes:
Browsers, operating systems, and resolutions used by users.
Users' browsing behavior, such as clicks and purchases, on a website.
The time users spend in an app and whether they are active.

Directions

Step 1. Enable Anonymous write

Note:
Enabling **Anonymous write** grants anonymous internet users the write access to the target log topic. This may generate dirty data because authentication is not performed.
1. Log in to the CLS console.
2. Click **Log Topic** on the left sidebar. On the log topic management page, click **Create Log Topic**, or click **Edit** for an existing log topic.
3. In the pop-up window, toggle on **Anonymous write** to enable anonymous write.

Step 2. Upload logs

After enabling **Anonymous write**, you can upload logs to a log topic using the following methods.

Method 1: Use browser JavaScript SDK

For detailed directions, see Uploading Logs via JavaScript SDK.

Method 2: Use Weixin Mini Programs JavaScript SDK

For detailed directions, see Uploading Logs via Mini Programs JavaScript SDK.

Method 3: Use an HTTP GET request

Use the following command to upload logs. Replace parameters according to your needs.

```
curl --request GET 'http://{host}/track?topic_id=
{topic_id}&key1=val1&key2=val2'
```

Parameter	Required	Description
${host}	Yes	The endpoint of the region where CLS resides. For more information, see Available Regions.
${topic_id}	Yes	topic id
key1=val1&key2=val2	Yes	The key-value pairs you want to upload to CLS. Ensure that the data is less than 16 KB.

Method 4: Use HTML img tag

Note:

The `track.gif` file contains the custom parameters that you want to upload. If you use this method, CLS records the custom parameters as well as the User-Agent HTTP header as log fields.

```
<img src='http://${host}/track.gif?topic_id={topic_id}&key1=val1&key2=val2'/>
```

Method 5: Use an HTTP POST request

If you need to upload a large amount of data, you can use the POST method.

Notes

This method applies to collection of webpage or client logs.

When using Web Tracking to collect logs, only one log can be written to a single request.

You can merge multiple logs into a single request.

This method cannot be used to write log data to multiple log topics.

Request headers

This method only uses common request headers. For more information, see Common Request Headers.

Request syntax

```
POST http://${host}/tracklog?topic_id=${topic_id} HTTP/1.1
```

Parameter	Required	Note
${host}	Yes	The endpoint of the region where CLS resides. For more information, see Available Regions.
${topic_id}	Yes	topic id

Example

```
POST /tracklog?topic_id={topic_id} HTTP/1.1
Host:ap-guangzhou.cls.tencentcs.com
Content-Type:application/json
{
    "logs": [{
        "contents": {
            "key1": "value1",
            "key2": "value2"
        },
        "time": 123456789
    }],
    "source": "127.0.0.1"
}
```

Uploading Logs via Logback Appender

Last updated：2024-01-20 17:14:28

Overview

Currently, CLS allows you to upload logs to CLS by using Logback Appender.

Background

Logback is an open source project of Apache. Logback allows you to deliver logs to various destinations, including consoles, files, GUI components, and even socket servers, NT event loggers, and UNIX Syslog daemons. In addition, you can have the flexibility to configure the logging behavior by editing a configuration file without modifying the application code.

Advantages

Logs are not stored on disks: generated log data is delivered to servers via the network.
No reconstruction is required: for applications that are using Logback, you only need to perform simple configuration to enable log collection.
Logs are delivered in async non-blocking mode: the high concurrency and backend async delivery design make Logback ideal for high write concurrency.
Resources are controllable: you can use parameters to control the size of the memory used by the producer to cache data to be sent and the number of threads used to execute data sending tasks.
Automatic retries: you can configure the number of retries for exceptions that allow retries.
Graceful shutdown: Logback will deliver logs in full mode before exiting.
Log reporting result response: exceptions that occur during Logback running are output via `addError`.

Project Introduction and Configuration

Introducing dependencies into a Maven project

```
<dependency>
    <groupId>com.tencentcloudapi.cls</groupId>
    <artifactId>tencentcloud-cls-logback-appender</artifactId>
    <version>1.0.3</version>
```

```
</dependency>
```

Modifying the Logback configuration file

```
  <appender name="LoghubAppender"
class="com.tencentcloudapi.cls.LoghubAppender">
        <!--Required-->
        <!--Domain Configuration -- Refer to
https://intl.cloud.tencent.com/document/product/614/18940?lang=en&pg=#domain-
name for detailed information.>
        <endpoint><region>.cls.tencentcs.com</endpoint>
        <!--Key Information -- Proceed to
https://console.intl.cloud.tencent.com/cam/capito acquire.>
        <accessKeyId>${SecretID}</accessKeyId>
        <accessKeySecret>${SecretKey}</accessKeySecret>
        <!--Log Topic ID-->
        <topicId>${topicId}</topicId>

        <!-- Optional. For details, see 'Parameter description'-->
        <totalSizeInBytes>104857600</totalSizeInBytes>
        <maxBlockMs>0</maxBlockMs>
        <sendThreadCount>8</sendThreadCount>
        <batchSizeThresholdInBytes>524288</batchSizeThresholdInBytes>
        <batchCountThreshold>4096</batchCountThreshold>
        <lingerMs>2000</lingerMs>
        <retries>10</retries>
        <baseRetryBackoffMs>100</baseRetryBackoffMs>
        <maxRetryBackoffMs>50000</maxRetryBackoffMs>

        <!-- Optional. Set the time format -->
        <timeFormat>yyyy-MM-dd'T'HH:mm:ssZ</timeFormat>
        <timeZone>Asia/Shanghai</timeZone>
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger -
%msg</pattern>
        </encoder>
        <mdcFields>THREAD_ID,MDC_KEY</mdcFields>
  </appender>
```

Parameter Description

Parameter	Description	Example

totalSizeInBytes	Maximum size of cached logs in a single producer instance. The default value is 100 MB.	totalSizeInBytes=104857600
maxBlockMs	If the available space for the producer is insufficient, the maximum blockage time in the **send** method defaults to 60 seconds. To prevent the obstruction of the log printing thread, it is strongly recommended to set this value to 0.	maxBlockMs=0
sendThreadCount	Size of the thread pool for executing log transmission tasks. The default value is the number of available processors.	sendThreadCount=8
batchSizeThresholdInBytes	When the size of the cached logs in a ProducerBatch is greater than or equal to the value of the **batchSizeThresholdInBytes**, the batch will be dispatched. The default value is 512 KB. The maximum value is 5 MB.	batchSizeThresholdInBytes=524288
batchCountThreshold	When the number of cached logs in a ProducerBatch is greater than or equal to the value of the **batchCountThreshold**, the batch will be dispatched. The default value is 4096.The maximum value is 40960.	batchCountThreshold=4096
lingerMs	Linger time of a ProducerBatch from creation to dispatch. The default value is 2 seconds. The minimum value is 100 milliseconds.	lingerMs=2000
retries	In the event of an initial transmisson failure of a particular ProducerBatch, the default value of the **retries** is 10. If the value of the **retries** is less than or equal to 0, the ProducerBatch will directly enter the failure queue following its initial unsuccessful transmission.	retries=10
maxReservedAttempts	You will trace back more information when the value of this parameter	maxReservedAttempts=11

	becomes larger. However, this will also consume more memory.	
baseRetryBackoffMs	Initial backoff time for the first retry. The default value is 100 milliseconds. The Producer uses an exponential backoff algorithm, where the scheduled waiting time for the Nth retries is calculated as baseRetryBackoffMs * 2^(N-1).	baseRetryBackoffMs=100
maxRetryBackoffMs	Maximum backoff time for retries. The default value is 50 seconds.	maxRetryBackoffMs=50000
timeFormat	This parameter is used to set the time format.	Accurate to the second: yyyy-MM-dd'T'HH:mm:ssZ Accurate to the millisecond: yyyy-MM-dd'T'HH:mm:ss.SSSZ

Uploading Logs via Log4j Appender

Last updated：2024-01-20 17:14:28

Overview

CLS allows you to upload logs to CLS by using Log4j Appender.

Background

Log4j Appender is an open source project of Apache. It allows you to deliver logs to various destinations, including consoles, files, GUI components, and even socket servers, NT event loggers, and UNIX Syslog daemons. You can control the output format of each log. By defining the level of each log, you can further control the log generation process. In addition, you can have the flexibility to configure the logging behavior by editing a configuration file without modifying the application code.
Log4j Appender consists of the following components:
Log information priority
From high to low, log message priorities are ERROR, WARN, INFO, and DEBUG, which specify the importance of log messages.
Log information output destination
The output destination of a log specifies whether the log will be printed to the console or to a file.
Log information output format
The output format determines how log information is displayed.

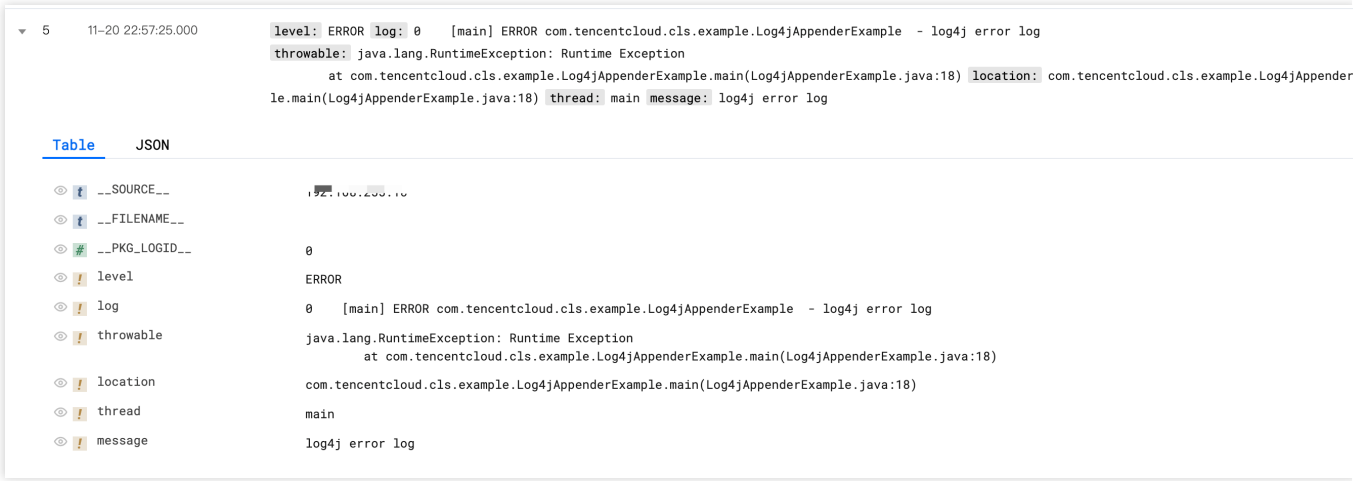
Advantages

Logs are not stored on disks: generated log data is delivered to servers via the network.
No reconstruction is required: for applications that are using Log4j Appender, you only need to perform simple configuration to enable log collection.
Logs are delivered in async non-blocking mode: the high concurrency and backend async delivery design make Logback ideal for high write concurrency.
Resources are controllable: you can use parameters to control the size of the memory used by the producer to cache data to be sent and the number of threads used to execute data sending tasks.
Automatic retries: you can configure the number of retries for exceptions that allow retries.
Graceful shutdown: Log4j Appender will deliver logs in full mode before exiting.

Log reporting result response: exceptions that occur during running are recorded via `org.apache.log4j.helpers.LogLog` and, by default, will be output to the console.

Using Tencent CLS Log4j Appender

With Tencent CLS Log4j Appender, you can specify that logs are output to Tencent Cloud CLS in the format as shown in the figure below.



Field	Description
__SOURCE__	Source IP
__FILENAME__	File name
level	Log level
location	Code location of the log print statement
message	Log content
throwable	Log exception information (This field exists only when exception information is logged.)
thread	Thread name
time	Log print time (You can print the format and time zone via `timeFormat` and `timeZone` respectively.)
log	Custom log format

Project Introduction and Configuration

Introducing dependencies into a Maven project

```
<dependency>
    <groupId>com.tencentcloudapi.cls</groupId>
    <artifactId>tencentcloud-cls-log4j-appender</artifactId>
    <version>1.0.2</version>
</dependency>
```

Modifying the Log4j configuration file

```
#loghubAppender
log4j.appender.loghubAppender=com.tencentcloudapi.cls.LoghubAppender
# CLS HTTP address. Required.
log4j.appender.loghubAppender.endpoint=ap-guangzhou.cls.tencentcs.com
# User ID. Required.
log4j.appender.loghubAppender.accessKeyId=
log4j.appender.loghubAppender.accessKeySecret=
# `log` field format. Required.
log4j.appender.loghubAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.loghubAppender.layout.ConversionPattern=%-4r [%t] %-5p %c %x -
%m%n
# Log topic. Required.
log4j.appender.loghubAppender.topicID =
# Log source. Optional.
log4j.appender.loghubAppender.source =
# Maximum size of logs cached by a single Producer instance. The default value
is 100 MB.
log4j.appender.loghubAppender.totalSizeInBytes=104857600
# Maximum time for blocking a caller from using the `send` method if the
Producer has insufficient free space. The default value is 60 seconds. It is
strongly recommended that this value be set to 0 in order not to block the log
print thread.
log4j.appender.loghubAppender.maxBlockMs=0
# Size of the thread pool for executing log sending tasks. The default value is
the number of available processors.
log4j.appender.loghubAppender.sendThreadCount=8
# When the size of logs cached in ProducerBatch is greater than or equal to
`batchSizeThresholdInBytes`, the batch will be sent. The default value is 512
KB, and the maximum value can be set to 5 MB.
log4j.appender.loghubAppender.batchSizeThresholdInBytes=524288
# When the number of logs cached in ProducerBatch is greater than or equal to
`batchCountThreshold`, the batch will be sent. The default value is 4096, and
the maximum value allowed is 40960.
```

```
log4j.appender.loghubAppender.batchCountThreshold=4096
# Linger time of a ProducerBatch from creation to sending. The default value is
2 seconds, and the minimum value allowed is 100 milliseconds.
log4j.appender.loghubAppender.lingerMs=2000
# Number of retries that a ProducerBatch can be retries if it fails to be sent
for the first time. The default value is 10 retries.
# If `retries` is less than or equal to 0, the ProducerBatch directly enters
the failure queue when it fails to be sent for the first time.
log4j.appender.loghubAppender.retries=10
# A larger parameter value allows you to trace more information, but it also
consumes more memory.
log4j.appender.loghubAppender.maxReservedAttempts=11
# Backoff time for the first retry. The default value is 100 milliseconds.
# The Producer adopts an exponential backoff algorithm. The scheduled wait time
for the Nth retry is baseRetryBackoffMs * 2^(N-1).
log4j.appender.loghubAppender.baseRetryBackoffMs=100
# Maximum backoff time for retries. The default value is 50 seconds.
log4j.appender.loghubAppender.maxRetryBackoffMs=50000
# Time format. Optional.
log4j.appender.loghubAppender.timeFormat=yyyy-MM-dd'T'HH:mm:ssZ
# Set the time zone to the UTC+08:00 time zone. Optional.
log4j.appender.loghubAppender.timeZone=Asia/Shanghai
# Output DEBUG and higher level messages
log4j.appender.loghubAppender.Threshold=DEBUG
```

Log4j Appender SDK

Log4j 1.x: please use tencentcloud-cls-log4j-appender.

Log4j 2.x: please use tencentcloud-cls-log4j2-appender.

Uploading Log via SDK

Last updated：2024-01-20 17:14:28

Overview

To help you use CLS more efficiently, we have created SDKs in multiple programming languages for log upload. You can choose an appropriate version based on your business needs.

Precautions

The SDK encapsulates the data access APIs of CLS uniformly to make log upload much easier.
The SDK implements the encapsulation of CLS logs in the Protobuf format, so you don't need to care about the specific details of this format when writing logs.
The SDK implements the compression methods defined in CLS APIs, so you don't need to care about the details of compression implementation. SDKs for certain programming languages support writing compressed logs by default.
The SDK provides unified features such as async sending, resource control, automatic retry, graceful shutdown, and log perception to make log reporting more comprehensive.

SDK List

The following table lists the source code of CLS SDKs for different programming languages at GitHub:

SDK Language	GitHub Source Code
Java	tencentcloud-cls-sdk-java
C++	tencentcloud-cls-sdk-c++
Go	tencentcloud-cls-sdk-go
NodeJS	tencentcloud-cls-sdk-js
Android	tencentcloud-cls-sdk-android
iOS	tencentcloud-cls-sdk-ios

Uploading Log via API

Last updated：2024-01-20 17:14:28

Feature Description

This API is used to write a log to the specified log topic.
CLS provides the following two modes:
Load balancing mode
Hash routing mode
In this mode, logs will be automatically written to a target partition among all readable/writable partitions under the current log topic based on the load balancing principle. This mode is suitable for scenarios where the sequential consumption is not needed.

Sample

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<`LogGroupList` content packaged as a PB file>
```

Private and public domain names

CLS request domain names divide into private domain names and public domain names:
A private domain name is in the format of `${region}.cls.tencentyun.com`, which is only valid for access requests from the same region, that is, CVM or Tencent Cloud services access the CLS service in the same region through the private domain name.
A public domain name is in the format of `${region}.cls.tencentcs.com`. After the access source is connected to the internet, the public domain name of CLS can be accessed under normal circumstances.
The `region` field is the abbreviation of a CLS service region, such as `ap-beijing` for the Beijing region. For the complete region list, see Available Regions.

```
ap-beijing - Beijing
ap-shanghai - Shanghai
ap-guangzhou - Guangzhou
ap-chengdu - Chengdu
...
```

In this mode, data will be written to a target partition that meets the range requirements based on the hash value (x-cls-hashkey) carried by data. For example, a log source can be bound to a topic partition through `hashkey`, strictly guaranteeing the sequence of the data written to and consumed in this partition.

Sample

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf
x-cls-hashkey: xxxxxxxxxxxxxxxxxxxxxxx

<`LogGroupList` content packaged as a PB file>
```

Note:

For more information on the PB description file format and compilation steps, see PB Compilation Sample.

In addition, CLS allows you to upload logs in the following two modes:

Uploading compressed logs

Uploading original logs

In this mode, logs are compressed in LZ4 format for collection, and then uploaded for retention. This mode reduces the log upload traffic (write traffic) and saves costs.

Sample

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf
x-cls-compress-type:lz4

<`LogGroupList` content packaged as a PB file>
```

In this mode, logs are uploaded in their original size, which incurs higher log write traffic fees.

Sample

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<`LogGroupList` content packaged as a PB file>
```

Request

Request line

```
POST /structuredlog
```

Request headers

The `x-cls-hashkey` request header indicates that logs are written to the CLS topic partitions with a range corresponding to the hashkey route, strictly guaranteeing the write sequence of logs to each topic partition for sequential consumption.

Field Name	Type	Location	Required	Description
x-cls-hashkey	string	header	No	Specifies the topic partition to which the logs will be written based on `hashkey`

Request parameters

Field Name	Type	Location	Required	Description
topic_id	string	query	Yes	ID of the target log topic to which data will be uploaded, which can be viewed on the log topic page
logGroupList	message	pb	Yes	The logGroup list, which describes the encapsulated log groups. No more than five `logGroup` values are recommended.

LogGroup description:

Field Name	Required	Description
logs	Yes	Log array, which is a set consisting of multiple `Log` values. A `Log` indicates a log, and `LogGroup` can contain up to 10,000 `Log` values
contextFlow	No	UID used to maintain context, which does not take effect currently
filename	No	Log filename
source	No	Log source, which is generally the server IP
logTags	No	Tag list of the log

Log description:

Field Name	Required	Description
time	Yes	UNIX timestamp of log time in seconds or milliseconds (recommended)
contents	No	Log content in `key-value` format. A log can contain multiple `key-value` pairs.

Content description:

Field Name	Required	Description
key	Yes	Key of a field group in one log, which cannot start with `_` .
value	Yes	Value of a field group, which cannot exceed 1 MB in one log. The total value cannot exceed 5 MB in `LogGroup` .

`LogTag` description:

Field Name	Required	Description
key	Yes	Key of a custom tag
value	Yes	Value corresponding to the custom tag key

Response

Sample response

```
HTTP/1.1 200 OK
Content-Length: 0
```

Response headers

No special response headers. Only common headers are used.

Response parameters

N/A

Error Codes

For more information, see Error Codes.

PB Compilation Sample

This sample describes how to use the protoc compiler to compile the PB description file into a log upload API in C++.

Note:

Currently, protoc supports compilation in multiple programming languages such as Java, C++, and Python. For more information, see protoc.

1. Install Protocol Buffer

Download Protocol Buffer, then decompress and install it. This document uses protobuf 2.6.1 running on CentOS 7.3 as an example.

Run the following command to decompress the `protobuf-2.6.1.tar.gz` package to `/usr/local` and access this directory:

```
[root@VM_0_8_centos]# tar -zxvf protobuf-2.6.1.tar.gz -C /usr/local/ && cd /usr/loc
```

Run the following commands to start compilation and installation and configure the environment variables:

```
[root@VM_0_8_centos protobuf-2.6.1]# ./configure
[root@VM_0_8_centos protobuf-2.6.1]# make && make install
[root@VM_0_8_centos protobuf-2.6.1]# export PATH=$PATH:/usr/local/protobuf-2.6.1/bi
```

After the compilation succeeds, run the following command to view the version:

```
[root@VM_0_8_centos protobuf-2.6.1]# protoc --version
liprotoc 2.6.1
```

2. Create a PB description file

A PB description file is an agreed-on data exchange format for communication. To upload logs, compile the specified protocol format to an API in the target programming language and add the API to the project code. For more information, see protoc.

Create a PB message description file `cls.proto` based on the PB data format content specified by CLS.

Note:

The PB description file content cannot be modified, and the filename must end with `.proto` .

The content of `cls.proto` (PB description file) is as follows:

```
package cls;

message Log
{
```

```
    message Content
    {
        required string key   = 1; // Key of each field group
        required string value = 2; // Value of each field group
    }
    required int64   time     = 1; // Unix timestamp
    repeated Content contents = 2; // Multiple `key-value` pairs in one log
}

message LogTag
{
    required string key       = 1;
    required string value     = 2;
}

message LogGroup
{
    repeated Log    logs        = 1; // Log array consisting of multiple logs
    optional string contextFlow = 2; // This parameter does not take effect
currently
    optional string filename    = 3; // Log filename
    optional string source      = 4; // Log source, which is generally the
server IP
    repeated LogTag logTags     = 5;
}

message LogGroupList
{
    repeated LogGroup logGroupList = 1; // Log group list
}
```

3. Compile and generate the API

This sample uses the proto compiler to generate a C++ file in the same directory as the `cls.proto` file. Run the following compilation commands:

```
protoc --cpp_out=./ ./cls.proto
```

Note:

`--cpp_out=./` indicates that the file will be compiled in cpp format and output to the current directory.

`./cls.proto` indicates the `cls.proto` description file in the current directory.

After the compilation succeeds, the code file in the corresponding programming language will be generated. This sample generates the `cls.pb.h` header file and `cls.pb.cc` code implementation file.

```
[root@VM_0_8_centos protobuf-2.6.1]# protoc --cpp_out=./ ./cls.proto
[root@VM_0_8_centos protobuf-2.6.1]# ls
cls.pb.cc cls.pb.h cls.proto
```

4. Call

Import the generated `cls.pb.h` header file into the code and call the API for data format encapsulation.

Importing Data
Importing COS Data

Last updated：2024-01-20 17:14:28

Overview

The shipping feature of CLS interconnects the upstream linkages of the product ecosystem to import logs from Tencent Cloud's Cloud Object Storage (COS) to CLS for further operations such as log data query and analysis and processing. You only need to complete simple configurations in the CLS console to import data.

Prerequisites

Activate CLS, create a logset and a log topic, and collect log data.
Activate COS and ensure that the files to be imported have already been uploaded to a COS bucket. For more information, please see Uploading Objects.
Set the COS access permission for the current operation account, that is, authorize CLS to use the CLS_QcsRole role to access COS resources.

Configuration

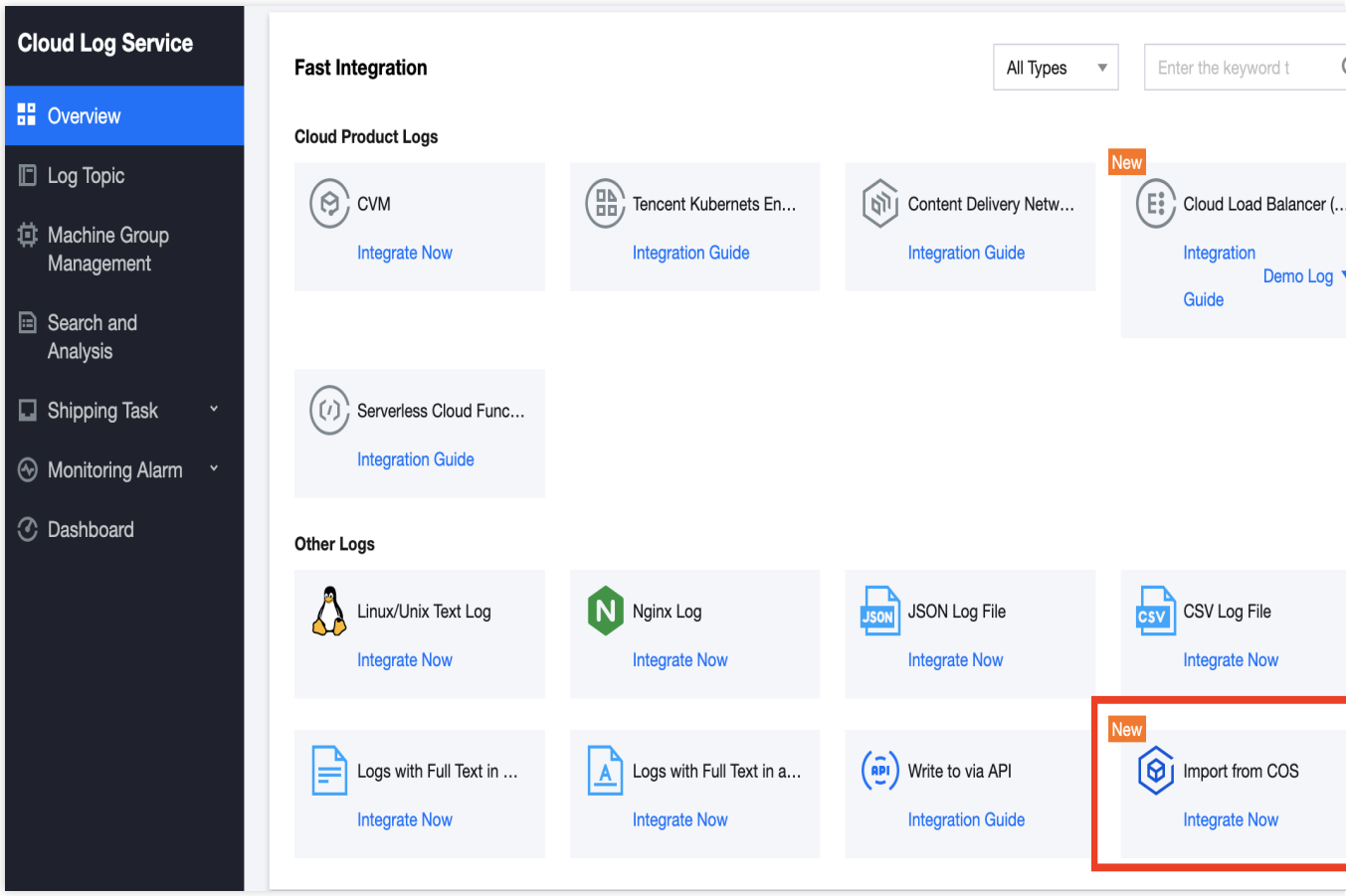
1. **Select a log topic**: select an existing log topic or create a log topic for storing the data to be imported from COS to CLS.
2. **Configure the data source**: path to the COS object to be imported and the corresponding compression mode (GZIP, LZOP, SNAPPY, or no compression).
3. **Configure parsing**: parsing format of the imported file. Currently, the following formats are supported: full text in a single line, JSON, and CSV.
4. **Configure indexes**: you need to configure indexes for the current topic, and enable index configuration before you can perform searches. If you select an existing log topic, the new index configuration takes effect only for the modified data.

Directions

Step 1. Select a log topic

If you want to create a new log topic, perform the following operations:

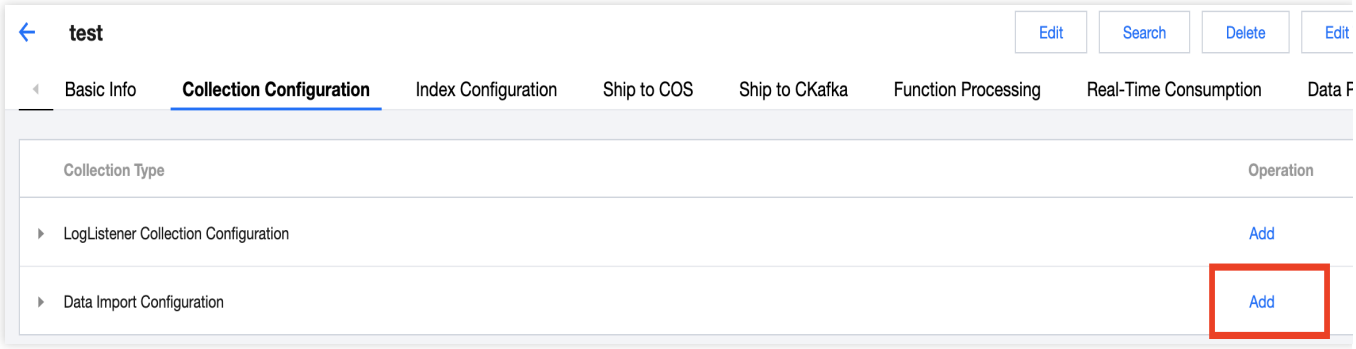
1. Log in to the CLS console.
2. On the left sidebar, click **Overview** to go to the overview page.
3. In the **Other Logs** area, locate **Import from COS** and click **Integrate Now**.



4. On the log topic creation page, configure the log topic information such as the log topic name and log retention period as needed, and click **Next**.

If you want to select an existing log topic, perform the following operations:

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** and select a log topic to be shipped to go to the log topic management page.
3. Click the **Collection Configuration** tab and click **Add** in the **Data Import Configuration** area.



Step 2. Configure the data source

1. On the data source configuration page, configure the following information in sequence:

Configuration Item	Description	Rule	Required
Task Name	Set the name of the import task.	The value can contain letters, numbers, underscores (_), and hyphens (-).	Yes
Bucket Region	Set the region of the bucket where the file to be imported resides. If the file to be imported and the destination log topic are in different regions, public network fees will incur due to cross-region access.	Select an option from the list.	Yes
Bucket	Select the bucket where the file to be imported resides. The drop-down list box provides all buckets in the selected region for you to choose.	Select an option from the list.	Yes
File Prefix	Enter the prefix of the folder where the COS file to be imported resides for accurate locating. You can enter the file prefix **csv/** or the complete file path **csv/object.gz**.	Enter a value.	Yes
Compression Mode	Select the compression mode of the COS file to be imported. CLS decompresses the file and reads data according to the compression mode of the file. Supported compression modes are: GZIP, LZOP, SNAPPY, and no compression.	Select an option from the list.	Yes

2. Click **Preview**. The system selects an eligible path to display and provides the total number of files with the specified file prefix.
3. After confirming that the data for preview is correct, click **Next**.

Step 3. Configure parsing

1. On the parsing configuration page, configure the following information:
Extraction mode: select **Full text in a single line**, **JSON**, or **CSV**.
Full text in a single line: each log will be parsed into a complete string with __CONTENT__ as the key value. If the index feature is enabled, you can search for log content via full-text search. The collection time is the log time.
JSON: you can extract key-value pairs in JSON format.
CSV: you can specify a separator to split each log. You need to define the key name for each split field. Invalid fields (fields that do not need to be collected) can be left empty. However, it's not supported to leave all fields empty.
Filter:
Filters are designed to help you extract valuable log data by adding log collection filter rules based on your business needs. If the filter rule is a Perl regular expression, the created filter rule will be used for matching; in other words, only logs that match the regular expression will be collected and reported.
For separator-formatted logs, you need to configure a filter rule according to the defined custom key-value pair. For example, if you want to collect all log data with a **status** field whose value is 400 or 500 after the sample log is parsed in separator mode, you need to configure `key` as `status` and the filter rule as `400|500`.
Use Collection Time: when this option is toggled on, log time is marked by the collection time. When it is disabled, you need to specify a field as the log time.
Note:
The log time is measured in seconds. If the log time is entered in an incorrect format, the collection time is used as the log time.
The time attribute of a log is defined in two ways: collection time and original timestamp.
Collection time: the time attribute of a log is determined by the time when the log is imported from COS to CLS.
Original timestamp: the time attribute of a log is determined by the timestamp in the raw log.
Use the original timestamp as the time attribute of logs.
Disable **Collection Time** and enter the time key of the original timestamp and the corresponding time parsing format in **Time Key** and **Time Parsing Format** respectively. For more information on the time parsing format, please see Configuring Time Format.
Separator: the system segments the sample log according to the selected separator and displays it in the extraction result box. You need to define a unique key for each field. Currently, log collection supports a variety of separators. Common separators include space, tab, comma, semicolon, and vertical bar. If your log data uses other separators such as `:::`, it can also be parsed through custom delimiter.
2. Click **Next**.

Step 4. Configure indexes

1. On the index configuration page, configure the following information:
Index Status: select whether to enable it.
Full-Text Index: select whether to set it to case-sensitive.
Full-Text Delimiter: the default value is `@&()='",;:<>[]{}/ \\n\\t\\r` and can be modified as needed.
Allow Chinese Characters: select whether to enable this feature.
Key-value Index: disabled by default. You can configure the field type, delimiters, and whether to enable statistical analysis according to the key name as needed. To enable key-value index, you can set

to 


.

Note:
Index configuration must be enabled before you can perform searches.
The modified index rules take effect only for newly written logs. The existing data is not updated.
2. Click **Submit**.

Related Operations

Querying the import progress

1. If the current log topic contains a COS import task, click **Search** to go to the **Search and Analysis** page to view the progress of the import task.
2. The floating balloon in the upper-right corner of the **Search and Analysis** page shows the import progress. Click the balloon to view the details of the import task.
3. On the task details page, click **View Details** to redirect to the collection configuration page to query the detailed configuration of the import task.

Tencent Cloud Service Log Access

Last updated：2024-01-20 17:14:28

CLS supports collecting logs of many Tencent Cloud services, including SCF, CDN, TKE, and CLB. CLS allows you to query operation records, analyze operations data, monitor running status, and set alarms.
Currently, CLS can collect logs of the following Tencent Cloud services:

Service	Collection Configuration Directions	Log Analysis
CLB	Configure log collection in the CLB console. For more information, see Configuring Access Logs.	CLB Access Log Analysis
CDN	Configure log collection in the CDN console. For more information, see Real-time Logs.	CDN Access Log Analysis
ECDN	Configure log collection in the CDN console. For more information, see Real-time Logs.	CDN Access Log Analysis
EdgeOne	Configure log collection in the EdgeOne console. For more information, see Real-time Logs.	-
CVM	Install and configure LogListener. For more information, see Deploying LogListener on CVMs in Batches.	
TKE	Configure log collection in the TKE console. For more information, see Collect container logs to CLS.	TKE Event Log Analysis TKE Audit Log Analysis
SCF	Configure log collection in the SCF console. For more information, see Log Delivery Configuration (Legacy).	-
CloudAudit	Configure log collection in the CloudAudit console. For more information, see Shipping Log with Tracking Set.	-
COS	Configure log collection in the COS console. For more information, see Enabling Real-Time Log Feature on COS.	COS Access Log Analysis
Flow Logs	Configure log collection in the Flow Logs console. For more information, see Create flow logs.	CCN Flow Log Analysis
TI-ONE	Configure log collection in the TI-ONE console.	-
WAF	Configure log collection in the WAF console.	-
CKafka	Configure log collection in the CKafka console.	-

IoT Hub	Configure log collection in the IoT Hub console. For more information, see Cloud Logs.	-

Metric Collection
Metrics Reporting

Last updated：2024-09-20 17:48:27

Metric topics support the Prometheus Remote Write protocol, allowing various collectors compatible with this protocol to collect and report metrics to the metrics topics, such as vmagent and telegraf. With these collectors, you can report various types of monitoring metrics to the metric topics, such as:
Server monitoring metrics, such as CPU, memory, and disk utilization.
Container cluster monitoring metrics, such as Pods memory usage and apiserver request delay.
Middleware monitoring metrics, such as performance metrics for MySQL, Kafka, Nginx, and other middleware.
Application system monitoring metrics, such as API throughput, response time, and the number of error requests.
Metric topics can also serve as remote storage for a local Prometheus instance. Using the Remote Write method, metrics from the local Prometheus can be reported to the metric topics.

Reporting Address

Network Type	address
Public Network	https:// `${region}` .cls.tencentcs.com/prometheus/ `${topicId}` /api/v1/write
Private Network	https:// `${region}` .cls.tencentyun.com/prometheus/ `${topicId}` /api/v1/write

Among them:
Replace `${region}` with the region where the metric topic is located, such as ap-beijing. For more region abbreviations, see Available Regions. Currently, only the regions of Beijing, Shanghai, Guangzhou, and Nanjing are supported.
Replace `${topicId}` with the metric topic ID, such as 0e69453c-0727-4c9c-xxxx-ea51b10d2aba. You can find the topic ID in the Metric Topic List.

Authentication method

Metric reporting uses Basic Auth for authentication. Use the SecretId and SecretKey from the API Key as the username and password, respectively.

username: `${SecretId}`

password: `${SecretKey}`

It is recommended to create a separate sub-account for the collector and use the SecretId and SecretKey of that account. Grant this sub-account only the following permissions to ensure security. For configuration details, see Sub-account Authorization.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsRemoteWrite"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Configuration Example

Telegraf

Add the following output configuration to the Telegraf configuration file:

```
[[outputs.http]]
  ## Reporting address: Replace ${region} and ${topicId}. This example uses the pub
  ## Private network address URL = https://${region}.cls.tencentyun.com/prometheus/
  url = "https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write"

  ## Authentication information: Replace ${SecretId} and ${SecretKey}.
  username = "${SecretId}"
  password = "${SecretKey}"

  ## Do not modify the Telegraf output data format configuration.
  data_format = "prometheusremotewrite"
  [outputs.http.headers]
     Content-Type = "application/x-protobuf"
     Content-Encoding = "snappy"
```

```
      X-Prometheus-Remote-Write-Version = "0.1.0"
```

Vmagent

When starting Vmagent, use the following parameters to configure remoteWrite:

```
./vmagent-prod \\
  -
remoteWrite.url=https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v
1/write \\
  -remoteWrite.basicAuth.username=${SecretId} \\
  -remoteWrite.basicAuth.password=${SecretKey}
```

Note:

The remoteWrite.url is set to the public network address; if network conditions allow, it is recommended to use the private network address.

The private network address is `https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/write`.

Prometheus

Add the following configuration to the Prometheus configuration file:

```
# Reporting address: Replace ${region} and ${topicId}. This example uses the public
# Private network address URL: https://${region}.cls.tencentyun.com/prometheus/${to
url: https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write

# Authentication information: Replace ${SecretId} and ${SecretKey}.
basic_auth:
  username: ${SecretId}
  password: ${SecretKey}

# Data write policy: Including caching and retry mechanisms, the following configur
queue_config:
  capacity: 20480
  min_shards: 100
  max_samples_per_send: 2048
  batch_send_deadline: 20s
  min_backoff: 100ms
  max_backoff: 5s
```

For more configuration details, see the official Prometheus documentation.

Prometheus Operator

Prometheus deployed in Kubernetes using the Prometheus Operator can report metrics in the following way:

1. Use kubectl to create the Secret required for authentication, for example, with the following command:

```
kubectl create secret generic kubepromsecret \\
 --from-literal=username=${SecretId} \\
 --from-literal=password=${SecretKey} \\
 -n monitoring
```

Note:

Replace `${SecretId}` and `${SecretKey}` in the command.

If Prometheus is deployed in another namespace, modify `-n monitoring` to the correct namespace.

You can also create Secret using a manifest file. For more details, see the official Kubernetes documentation.

2. Open the Prometheus manifest configuration file, which is typically located in the GitHub repo file used by the Prometheus Operator. The usual path is `kube-prometheus/manifests/prometheus-prometheus.yaml` .

3. Modify the Prometheus manifest configuration file by adding the following configuration at the end:

```
  remoteWrite:
  - url: "https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/write"
    basicAuth:
      username:
        name: kubepromsecret
        key: username
      password:
        name: kubepromsecret
        key: password
```

Note:

Replace `${region}` and `${topicId}` in the configuration. This example uses the public network address; if network conditions allow, it is recommended to use the private network address.

Private network address `URL:`

`https://${region}.cls.tencentyun.com/prometheus/${topicId}/api/v1/write`

When handling large data volumes, you can add the following data write policy, including caching and retry mechanisms. For more details, see the official Prometheus Operator documentation.

```
    queueConfig:
      capacity: 204800
      minShards: 100
      maxShards: 2048
      maxSamplesPerSend: 4096
      batchSendDeadline: 30s
      minBackoff: 100ms
      maxBackoff: 5s
```

4. Apply the configuration file, for example, with the following command:

```
kubectl apply -f prometheus-prometheus.yaml -n monitoring
```

Note:

Replace `prometheus-prometheus.yaml` with the correct configuration file path.

If Prometheus is deployed in another namespace, modify `-n monitoring` to the correct namespace.

Log Storage
Storage Class Overview

Last updated：2024-01-20 16:46:19

According to users' different requirements for log search latency and log processing capabilities, CLS provides two storage classes: **STANDARD** and **STANDARD_IA**.

Note:

Currently, STANDARD_IA log storage is available only in Beijing, Guangzhou, Shanghai, Hong Kong (China), Nanjing, Singapore, Silicon Valley, and Frankfurt regions. If it is not supported in the region of your log topic, contact smart customer service for application.

STANDARD

STANDARD storage is suitable for users who require statistical analysis and provides log search within seconds, real-time statistical analysis, real-time monitoring, streaming consumption, and other application capabilities.

Use cases

Ops monitoring and troubleshooting: Implements real-time diagnosis of online problems by leveraging the capability of log search within seconds to quickly search the log content scattered on multiple machines for fault cause locating and recovery; calculates quality metrics based on logs in real time and reports alarms when quality metrics exceed thresholds, facilitating development and Ops personnel to discover and rectify faults in the first place.
Streaming processing: Collects the tracking log data of petabyte scale scattered on multiple machines and streams the data to the user-built big data processing cluster in real time for subsequent data lake computing, for example, for the model data calculation business of a recommendation system.

STANDARD_IA

STANDARD_IA is suitable for infrequently accessed logs that do not require statistical analysis, such as archived audit logs. It provides the full-text log search capability, meeting users' requirements for backtracking and archiving historical logs. The overall usage costs of STANDARD_IA storage are 80% lower than those of **STANDARD storage**. For more information, see IA Storage.

Use cases

Historical logs: The explosive growth of log data makes it expensive to store and analyze logs on a large scale over months or even years. This can cause users to delete valuable data and miss out on important insights that long-term

data can yield. **STANDARD_IA** can meet the needs of users to conduct large-scale statistical analysis and backtracking of historical data with low costs.
Non-critical business logs: During troubleshooting, developers need to pay more attention to ERROR and WARN logs and monitor them and generate alarms when necessary. Non-critical business logs, such as INFO logs, are only archived and need to be searched and analyzed in specific scenarios. Common users do not have specific requirements on the search latency of these logs. Using **STANDARD_IA** to store non-critical business logs can significantly reduce user costs and meet users' demands for infrequent search.
Audit logs: Logs for operation and security audits are collected to **STANDARD_IA**, and access behaviors, such as operation records of an account or an object, are analyzed via CLS's infrequently accessed log search capability to determine whether there are illegal operations. In addition, logs can be stored for more than 180 days to meet compliance audit requirements.

Feature Comparison

Feature	STANDARD_IA	STANDARD
Index creation	✓ (supports only full-text indexes)	✓
Context search	✓	✓
Quick analysis	×	✓
Full-text search	✓ (responds in 2 seconds for searches in 100 million records)	✓ (responds in 0.5 second for searches in 100 million records)
Key-value search	×	✓
Log download	✓	✓
SQL analysis	×	✓
Dashboard	×	✓
Monitoring alarm	×	✓
Shipping to COS	✓	✓
Shipping to CKafka	✓	✓

Shipping to ES	✓	✓
Shipping to SCF	✓	✓
Log consumption	✓	✓
Data processing	✓	✓

Data Encryption

Last updated：2023-02-16 17:31:08

Data encryption using KMS

KMS encryption is server-side encryption using a key managed by KMS. KMS is a security management service launched by Tencent Cloud, using a third-party-certified hardware security module (HSM) to generate and protect keys. KMS allows users to easily create and manage keys, meeting their key management needs for multiple applications and services, while satisfying regulatory and compliance requirements.

To use KMS to encrypt CLS log topics, activate KMS and authorize the CLS service role to access KMS resources. If a Tencent Cloud managed CMK for CLS is not available in KMS, the CLS will automatically create a key for you.

Notes:

To use the log topic encryption feature, submit a ticket to apply for it.

Currently, CLS data encryption is available only in Beijing, Shanghai, and Guangzhou regions.

Using KMS encryption will incur an additional cost, which will be charged by KMS. For more information, see KMS Billing Overview.

The encryption feature can be enabled only when you create a log topic, and cannot be disabled once enabled.

Directions

1. Log in to the CLS console, select **Log Topic**, and click **Create Log Topic**.
2. In the pop-up window, click **Advanced Settings** and select **Enable data encryption**.

Metric Storage
Metric Storage Overview

Last updated：2024-09-20 17:48:27

Basic Concepts

Metric

Metrics are measurements used to assess the performance and operation of systems and applications, such as CPU utilization, memory utilization, access throughput, response time, and response success rate. Metrics are typically generated at regular intervals, producing a value at each point in time. Over time, these values form a sequence, which is commonly referred to as a time series.

Cloud Log Service (CLS) is compatible with the Prometheus metrics data model, storing timestamped metric data with the same metric name and labels as time series. Each data point in the time series is referred to as a sample, which consists of a timestamp and a sample value.

For example, the total number of requests for a particular API at 15:35:23.123 on December 30, 2020 would be considered a sample, with the data as follows:

```
requests_total{method="POST", handler="/messages"} 217
```

It is composed of the following parts:

Metric name: requests_total

Label: {method="POST", handler="/messages"} (indicating the API name is messages and the request method is POST.)

Timestamp: 2020/12/30 15:35.123

Sample value: 217

The monitored system often has multiple metrics at the same time, with many different metric names and labels at a given moment. For example, Nginx monitoring metrics might include the following:

```
# HELP nginx_http_requests_total The total number of HTTP requests
# TYPE nginx_http_requests_total counter
nginx_http_requests_total 10234

# HELP nginx_http_requests_duration_seconds The HTTP request duration in
seconds
# TYPE nginx_http_requests_duration_seconds histogram
nginx_http_requests_duration_seconds_bucket{le="0.005"} 2405
nginx_http_requests_duration_seconds_bucket{le="0.01"} 5643
nginx_http_requests_duration_seconds_bucket{le="0.025"} 7890
```

```
nginx_http_requests_duration_seconds_bucket{le="0.05"} 9234
nginx_http_requests_duration_seconds_bucket{le="0.1"} 10021
nginx_http_requests_duration_seconds_bucket{le="0.25"} 10234
nginx_http_requests_duration_seconds_bucket{le="0.5"} 10234
nginx_http_requests_duration_seconds_bucket{le="1"} 10234
nginx_http_requests_duration_seconds_bucket{le="2.5"} 10234
nginx_http_requests_duration_seconds_bucket{le="5"} 10234
nginx_http_requests_duration_seconds_bucket{le="10"} 10234
nginx_http_requests_duration_seconds_bucket{le="+Inf"} 10234
nginx_http_requests_duration_seconds_sum 243.56
nginx_http_requests_duration_seconds_count 10234

# HELP nginx_http_connections Number of HTTP connections
# TYPE nginx_http_connections gauge
nginx_http_connections{state="active"} 23
nginx_http_connections{state="reading"} 5
nginx_http_connections{state="writing"} 7
nginx_http_connections{state="waiting"} 11

# HELP nginx_http_response_count_total The total number of HTTP responses sent
# TYPE nginx_http_response_count_total counter
nginx_http_response_count_total{status="1xx"} 123
nginx_http_response_count_total{status="2xx"} 9123
nginx_http_response_count_total{status="3xx"} 456
nginx_http_response_count_total{status="4xx"} 567
nginx_http_response_count_total{status="5xx"} 65

# HELP nginx_up Is the Nginx server up
# TYPE nginx_up gauge
nginx_up 1
```

The meanings of the metrics are as follows:

nginx_http_requests_total: The total number of HTTP requests processed by Nginx.

nginx_http_requests_duration_seconds: The duration of HTTP requests, provided using the Histogram type, which shows the number of requests within different time intervals.

nginx_http_connections: The current number of HTTP connections in Nginx, categorized into active, reading, writing, and waiting status.

nginx_http_response_count_total: The total number of HTTP responses returned by Nginx, categorized by status code.

nginx_up: The operation status of the Nginx server. 1 indicates that it is running; 0 indicates that it is not running.

Metric Topic

Refers to the fundamental unit for collecting, storing, searching, and analyzing metric data on the Cloud Log Service platform. The collected metric data is managed within metric topics, including configurations such as retention period

and retrieval analysis. Metric topics are compatible with the Prometheus metrics data model and metric query API, functioning similarly to a Prometheus instance. As long as the metric names do not conflict and the data volume does not exceed product specifications and limits, metrics from different applications or services can be stored in the same metric topic. In practice, metric data from production, testing, and development environments of a business system are typically stored in separate metric topics.

Note:

The metric topics service ended its public beta on July 1, 2024, and is now officially a paid service. For more details, see the Billing Overview.

Features

Metric collection:

Metric reporting: Supports the Prometheus Remote Write protocol, allowing various collectors compatible with this protocol, such as vmagent and telegraf, to collect and report metrics to the metric topic.

Log to metric: Logs from a log topic can be converted into metrics using scheduled SQL queries. This approach is suitable for long-term, low-cost storage of key system metrics, and it often provides better performance when conducting visualization analysis based on these metrics.

Cloud product metric subscription: Supports the proactive subscription to cloud product metrics from the TCOP, allowing centralized storage and querying within CLS. This enables a more flexible statistical analysis of cloud product metrics.

Metric query: Use PromQL to query metrics.

Metric visualization: You can use dashboards to visualize metric data in formats such as tables, time series charts, single value charts, and gauges. Additionally, you can use Grafana to display metric data directly.

Monitoring and alarms: You can configure alarm policies for metric topics, notifying users via SMS, WeChat, phone calls, emails, and WeCom when anomalies in the metrics occur.

Advantages

Metric topics are compatible with the Prometheus metric data model and query API, enabling seamless integration with various Prometheus-compatible open-source projects, such as Grafana.

Compared to a self-built Prometheus, it eliminates the need for deployment and maintenance, significantly reducing labor costs.

It can be used in combination with logs to centrally collect, store, and analyze metrics and log data, enabling the construction of a unified monitoring platform and improving Ops efficiency.

Fee Description

For more details, see the Billing Overview.

Specifications and Limits

Restriction Item	Description
Metric name	Supports English letters, numbers, underscores, and colons. It should conform to the regular expression `[a-zA-Z_:][a-zA-Z0-9_:]*`.
Label name	Supports English letters, numbers, and underscores. It should conform to the regular expression `[a-zA-Z_][a-zA-Z0-9_]*`.
Label value	No special restrictions, supporting all types of Unicode characters.
Sample value	A float64 type value
Sample timestamp	Millisecond precision
Query Concurrency	A single metric topic supports up to 15 concurrent queries.
Query data volume	A single query can involve up to 200,000 time series, with a maximum of 11,000 data points per time series in the query results.
Metric upload frequency control	25000QPS
Metric upload flow control	250MB/s

Compatible with Prometheus API

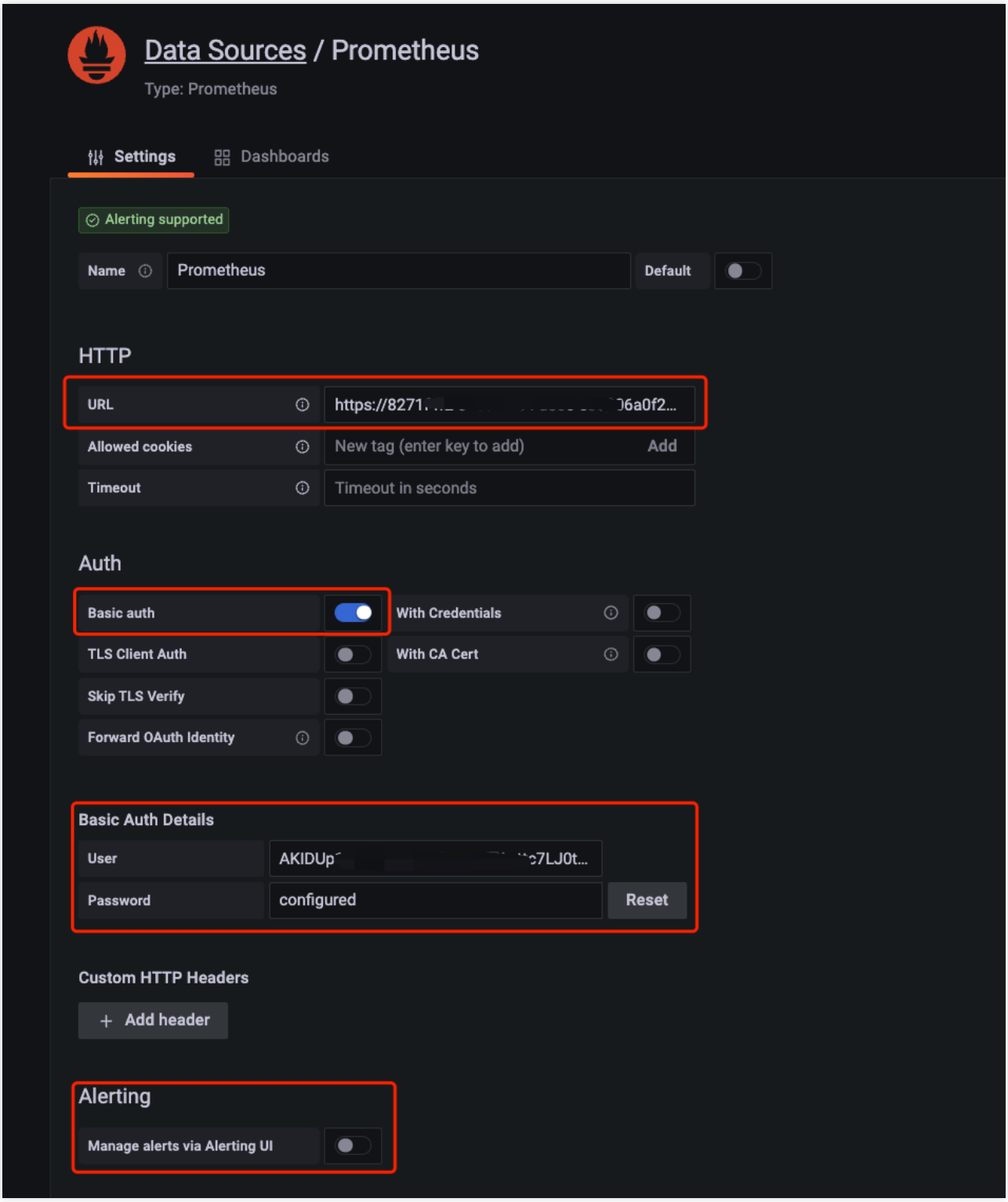
Last updated：2024-09-20 17:48:27

Metric topics are compatible with the Prometheus metrics data model and the following APIs:

Metrics Query API

Prometheus HTTP API

Compatible with the following HTTP APIs and commonly used for integration with Grafana.
Instant queries : /api/v1/query
Range queries : /api/v1/query_range
Finding series by label matchers : /api/v1/series
Getting label names : /api/v1/labels
Querying label values : /api/v1/label/<label_name>/values
Querying exemplars : /api/v1/query_exemplars
Not compatible with APIs used for native Prometheus collection configurations, alarm rules, and cluster management features, such as Targets, Rules, Alerts, Querying target metadata, Querying metric metadata, Alertmanagers, Status, TSDB Admin APIs.
Directions for integration with Grafana:
1. On the **Grafana Data Sources** page, click **Add data source**.
2. Select **Prometheus,** and configure the following information in the form:
Search for the required CAM policy as needed, and click to complete policy association.



URL: Replace ${region} and ${topicId} where ${region} represents the region abbreviation, and ${topicId} represents the metric topic ID.

Public network address: https://${region}.cls.tencentcs.com/prometheus/${topicId}

Private network address: https://${region}.cls.tencentyun.com/prometheus/${topicId}

Basic auth: Turn on this switch.

Basic Auth Details: The API uses Basic Auth for authentication. Use the SecretId and SecretKey from the API Key as the username and password, respectively.

username: `${SecretId}`

password: `${SecretKey}`

Note:

It is recommended to create a separate sub-account and use the SecretId and SecretKey of that account. Grant this account only the following permissions to ensure security. For configuration details, see Sub-account Authorization.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsSeries",
                "cls:MetricsQueryExemplars",
                "cls:MetricsLabelValues",
                "cls:MetricsQueryRange",
                "cls:MetricsLabels",
                "cls:MetricsQuery"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Manage alerts via Alerting UI: Turn off this switch. CLS is not compatible with Prometheus Alerts-related APIs and does not support this feature.

3. Click **Save & Test** at the bottom to test whether the configuration is correct and to save the settings.

Prometheus Remote Read API

This is commonly used to read metric topic data with a self-built Prometheus instance. Add the following configuration to the Prometheus configuration file (for more configuration details, see the Prometheus Official Documentation):

```
# Reading address: Replace ${region} and ${topicId}. This example uses the public n
# Private network address URL: https://${region}.cls.tencentyun.com/prometheus/${to
url: https://${region}.cls.tencentcs.com/prometheus/${topicId}/api/v1/read

# Authentication information: Replace ${SecretId} and ${SecretKey}.
basic_auth:
  username: ${SecretId}
  password: ${SecretKey}
```

In the URL, ${region} represents the region abbreviation, and ${topicId} represents the metric topic ID.
The API uses Basic Auth for authentication. Use the SecretId and SecretKey from the API Key as the username and password, respectively.
username: `${SecretId}`
password: `${SecretKey}`
Note:
It is recommended to create a separate sub-account and use the SecretId and SecretKey of that account. Grant this account only the following permissions to ensure security. For configuration details, see Sub-account Authorization.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cls:MetricsRemoteRead"
            ],
            "resource": [
                "*"
            ]
        }
    ]
}
```

Metrics Reporting API

Prometheus Remote Write API

This API is commonly used to collect and report metrics to the metric topics by using various collectors compatible with the Prometheus Remote Write protocol, such as vmagent and telegraf. Metrics from a local Prometheus instance can also be reported to metric topics via this API. For detailed instructions, see Metric Reporting.

Metric Pre-aggregation

Last updated：2024-09-20 17:48:27

When large volumes of metric data are monitored or complex queries are executed, real-time queries can become slow, and writing query statements may become challenging. To address this, the metric storage system supports a pre-aggregation feature. Pre-aggregation allows for the pre-computation of complex queries (using PromQL) and stores the results as new metrics. These precomputed data can be directly used in subsequent queries and alarms, reducing real-time computation costs, improving query performance, and simplifying and streamlining monitoring configuration. Pre-aggregation is compatible with Prometheus Recording Rules, allowing native YAML configuration files to be directly imported.

For example, the original metric `prometheus_http_requests_total` records the number of requests made to various Prometheus APIs with different response status codes.

```
# HELP prometheus_http_requests_total Counter of HTTP requests.
# TYPE prometheus_http_requests_total counter
prometheus_http_requests_total{code="200",handler="/api/v1/label/:name/values"}
7
prometheus_http_requests_total{code="200",handler="/api/v1/query"} 19
prometheus_http_requests_total{code="200",handler="/api/v1/query_range"} 27
prometheus_http_requests_total{code="200",handler="/graph"} 11
prometheus_http_requests_total{code="200",handler="/metrics"} 8929
prometheus_http_requests_total{code="200",handler="/static/*filepath"} 52
prometheus_http_requests_total{code="302",handler="/"} 1
prometheus_http_requests_total{code="400",handler="/api/v1/query_range"} 6
```

If you need to regularly query the total number of requests for each status code, you can use the following configuration to create a pre-aggregation task:

```
groups:
  - name: example
    rules:
    - record: code:prometheus_http_requests_total:sum
      expr: sum by (code) (prometheus_http_requests_total)
```

Among them:

`sum by (code) (prometheus_http_requests_total)` is the metric query statement (PromQL), which calculates the sum of request counts grouped by status code.

`code:prometheus_http_requests_total:sum` is the generated metric name, which can be customized. You can use this name in subsequent queries to directly retrieve the precomputed metric.

Application Scenario

Improve metric query performance: When some complex queries need to be executed frequently, pre-aggregation can be used to precompute and store these queries as new metrics. This can significantly enhance query performance and reduce query response time.
Simplify query statements: Complex query statements can be lengthy and difficult to read. By using pre-aggregation, you can store the results of complex queries as simplified metrics. This makes subsequent queries easier to write and maintain.
Store core metrics separately: All data in the metric topic is retained for the same duration and is automatically cleared after expiration. If you need to store a subset of core metrics for a longer period, you can use a pre-aggregation task to store this portion of data in another metric topic.

Prerequisites

A metric topic has already been created.

Directions

Creating Task

1. Log in to the Cloud Log Service console.
2. In the left sidebar, click **Metric Topic**.
3. Click the metric topic ID/name for which you want to create a pre-aggregation task to enter the metric topic management page.
4. Click the **Metric Pre-aggregation** tab to enter the pre-aggregation task list page, then click **Creating Task**. The main configuration items are described as follows:

Configuration Item	Description
Service Log	Saves the task running logs in the cls_service_log log topic, making it easier to monitor the task's operation status. This log topic is free, and it is recommended to enable it.
Query Statement	The PromQL statement to be executed. The pre-aggregation task will run this statement at scheduled intervals to retrieve the execution results.
Indicator Name	The result of the execution statement will be stored under this metric name, which can be used for subsequent data queries. It supports English letters, numbers, underscores, and colons, and should conform to the regular expression `[a-zA-Z_:][a-zA-Z0-9_:]*` .

Custom Dimension	Adds dimensions to the metric. If there is a conflict between the custom dimension and the dimension names in the execution statement results, the custom dimension takes precedence.
Scheduling cycle	The execution interval for the pre-aggregation task, with the range from 1 to 1440 minutes. It is recommended to use a 1-minute interval.
Advanced Settings	Target Metric Topic: Specifies where the pre-aggregated metric data will be stored. By default, it is stored in the current topic. If you want to store this data separately (e.g., to set a different retention period for this portion of data), you can store it in another metric topic. Delayed Execution: Since there may be delays in metric data collection, you can set a delayed execution to ensure the data is fully collected before the pre-aggregation task runs. The default delay is 30 seconds.

5. Once the configuration is complete, click **Submit**.

Import Configuration File

1. Log in to the Cloud Log Service console.
2. In the left sidebar, click **Metric Topic**.
3. Click the metric topic ID/name for which you want to create a pre-aggregation task to enter the metric topic management page.
4. Click the **Metric Pre-aggregation** tab to enter the pre-aggregation task list page, then click **Import Configuration File**. The main configuration items are described as follows:

Configuration Item	Description
Enabling Status	Indicates whether the task needs to run. Tasks that are not running will not generate pre-aggregated result data.
Service Log	Saves the task running logs in the cls_service_log log topic, making it easier to monitor the task's operation status. This log topic is free, and it is recommended to enable it.
Execution Interval	The execution interval for the pre-aggregation task, with the range from 1 to 1440 minutes. It is recommended to use a 1-minute interval.

YAML Configuration	Compatible with Prometheus Recording Rule YAML. The interval only supports a range from 1 to 1440 minutes. For example:
	`groups:`
	`  - name: example`
	`  rules:`
	`  - record: code:prometheus_http_requests_total:sum`
	`  expr: sum by (code) (prometheus_http_requests_total)`

Advanced Settings	Target Metric Topic: Specifies where the pre-aggregated metric data will be stored. By default, it is stored in the current topic. If you want to store this data separately (e.g., to set a different retention period), you can store it in another metric topic. Delayed Execution: Since there may be delays in metric data collection, you can set a delayed execution to ensure the data is fully collected before the pre-aggregation task runs. The default delay is 30 seconds.

5. Click **Submit** to save the configuration file. A pre-aggregation task will be automatically generated, with the task name consisting of the group name and record from the YAML file.

Note:

Pre-aggregation tasks automatically generated through a configuration file should be modified via the configuration file; direct modification is not supported.

Manage the Configuration File

1. Log in to the Cloud Log Service console.
2. In the left sidebar, click **Metric Topic**.
3. Click the metric topic ID/name for which you want to create a pre-aggregation task to enter the metric topic management page.
4. Click the **Metric Pre-aggregation** tab to enter the pre-aggregation task list page, then click **Manage the Configuration File**.
5. In the list, you can view the existing configuration files. In the **Operation** column, you can manage the configuration files:

Edit: Edit the configuration file, with the configuration items consistent with Import Configuration File. After modification, the system will automatically adjust the pre-aggregation tasks based on the latest configuration file (deleting tasks that no longer exist in the file, adding new tasks, and updating existing tasks).

Pause: Suspend the pre-aggregation tasks associated with the configuration file. During the suspension, these tasks will no longer be executed.

Delete: Remove the pre-aggregation tasks associated with the configuration file.

Search and Analysis (Log Topic)
Overview of Search and Analysis

Last updated：2024-12-20 16:11:32

Basic Concepts

Search and analysis support filtering, searching and statistically analyzing log data, such as querying logs containing `error` , counting the number of logs by URL grouping, calculating PV change trend, etc. It is the most commonly used function in the Cloud Log Service.
Search and analysis are based on Word Segmentation and Indexing. If you are unfamiliar with the two concepts, it is recommended that you consult this document in advance.

Features

Basic Features

Full-text search: Search the full text of logs. The log can be found when any field in the log meets the search conditions, such as using `error` to search for all logs with errors.
Key-value search: Search for the specified field in the log, and the log will be found when the specified field meets the search criteria. For example, use `level:error AND timeCost:>1000` to search for logs whose level is error and that consume time (timeCost) greater than 1,000 ms.
Statistical analysis: You can perform statistical analysis on the logs meeting specified search criteria, and returns the results. For example, you can enter `status:404 | select count(*) as logCounts` to get the number of logs whose response status code is 404.
The above functions require the use of search and analysis statements. See Syntax Rules for details.
Note:
Infrequent storage only supports full-text search, not key-value retrieval or statistical analysis.

Advanced Features

Log Search
Multi-topic retrieval: Retrieve multiple log topics simultaneously, but statistical analysis is not supported. It can span log sets but not regions.
Contextual search: Retrieve several logs before (previous logs) or after (subsequent logs) the target log data in the original file. By viewing the context of specified logs, you can quickly find fault information during troubleshooting, making it easier to locate issues.

Custom redirect: Click the log field value to perform a custom redirect, facilitating further operations, such as querying user information on the internal user management platform based on user_id.
Download logs: Download logs that meet the search criteria to the local machine, with a maximum of 50 million logs per download.
Log statistical analysis
Rapid analysis: Click the log field name to quickly analyze the distribution of field values, trends over time, and numerical statistics.
Sampling analysis: When using the statistical analysis feature, if the number of original logs is very large or the statistical analysis statement (SQL) is complex, the analysis may be slow or even timeout. In this case, you can use the sampling analysis feature to randomly sample the original logs and then perform statistical analysis.
Associate external data: When performing statistical analysis on logs, you can associate the logs with external data for analysis. For example, you can query the user gender in the user information database based on the user ID in the logs, and then perform statistical analysis on the logs by gender.

Advantages

Based on pre-built index data, the search and analysis performance is much higher than that of queries with general text scanning methods (such as Linux grep commands and query functions in text editor), making the product suitable for real-time query of massive log data.
Statistical analysis supports SQL and conforms to the SQL-92 specification. In addition, flexible SQL syntax and 400+ SQL functions can be used for log statistics, meeting most statistical analysis requirements.

Fee Description

Search and analysis itself do not incur fees, but since it relies on reporting logs to the CLS platform and enabling indexing, it will generate log and index traffic fees, storage fees, and topic partition fees. Please see Billing Overview.
Additionally, using the API to call the search and analysis interface will incur service request fees, and using the log download feature will incur public network read traffic fees.
To reduce product usage costs, please see Saving Product Use Costs.

Specifications and Limits

Please see Specifications and Limits for specifications and limitations of search and analysis.

Syntax and Rules

Last updated：2025-05-26 17:07:44

Overview

A search and analysis statement is composed of two parts, search condition and SQL statement, which are separated by a vertical bar `|`. To search for logs only without statistical analysis, omit the vertical bar `|` and SQL statement.

```
[Search condition] | [SQL statement]
```

Search condition: Specifies the condition for log search. Only logs that meet the condition are returned. For example, you can use `status:404` to search for application request logs with response status code 404. If the search condition is empty or `*`, it indicates there is no search condition, and all logs are searched for.
SQL statement: Performs statistical analysis on logs that meet the search condition and returns the statistical analysis result. For example, you can use `status:404 | select count(*) as logCounts` to count the number of logs with response status code 404.

Note:

Search is based on log segments. Raw logs can be matched after segmentation only if they contain the segment specified in the search condition. For example, `errorMessage` cannot be matched with `error`, as they are different segments. In this case, you need to add a wildcard and search for it with `error*`. For more information on segments and examples, see Segment and Index.

Prerequisites

Using a search condition:

Full-text search: Full-text index is enabled during index configuration.

Key-value search:

Log access supports standard storage. Infrequent storage does not support key-value search. For details, see Log Storage Overview.

Log fields have been extracted (Log Structuring) during log collection.

Key-value index is enabled for the field to search for during index configuration.

Using a SQL statement:

Logs are connected to STANDARD storage. STANDARD_IA does not support SQL statements for statistical analysis. For more information, see Storage Class Overview.

Log fields have been extracted (Log Structuring) during log collection.

Key-value index and statistics are enabled for the field to search for during index configuration.

Search Condition Syntax

CQL：It is CLS query language (CQL) designed for log search in CLS. We recommend that you use it as it is easy to learn and use.
Lucene：Open-source Lucene syntax. Since this syntax is not specifically designed for log retrieval, it has considerable limitations on special symbols, case sensitivity, wildcards, etc., is relatively cumbersome to use, and is prone to syntax errors. Not recommended for use.
CQL(Recommended)
Lucene
Lucene and CQL Comparison

Syntax	Description
key:value	Key-value search, which indicates to query logs with a `key` field whose value contains the `value`, such as `level:ERROR`.
value	Full-text search, which indicates to query logs with the full text containing the `value`, such as `ERROR`.
AND	Logical AND operator, which is case-insensitive, such as `level:ERROR AND pid:1234`.
OR	Logical OR operator, which is case-insensitive, such as `level:ERROR OR level:WARNING` `level:(ERROR OR WARNING)`.
NOT	Logical NOT operator, which is case-insensitive, such as `level:ERROR NOT pid:1234` `level:ERROR AND NOT pid:1234`.
()	Parentheses, which control the precedence of logical operations, such as `level:(ERROR OR WARNING) AND pid:1234`. **Note:** When parentheses are not used, `AND` has a higher priority than `OR`.
" "	Phrase search, which encloses a string in double quotation marks to match logs that contain all the words in the string in the same sequence, such as `name:"john Smith"`. A phrase search has no logical operators, and the phrase used is equivalent to the query character, such as `name:"and"`.
' '	Phrase search, which encloses a string in single quotation marks and is equivalent to `""`. When the phrase to be searched for contains double quotation marks, single quotation marks can be used to enclose the phrase to avoid syntax errors, such as `body:'user_name:"bob"'`.
*	Fuzzy search, which is used to match zero, one, or multiple characters, such as `host:www.test*.com`. Fuzzy prefix search is not supported.
>	Range operator, which indicates the left operand is greater than the right operand, such as

	`status>400` or `status:>400` .
>=	Range operator, which indicates the left operand is greater than or equal to the right operand, such as `status>=400` or `status:>=400` .
<	Range operator, which indicates the left operand is less than the right operand, such as `status<400` or `status:<400` .
<=	Range operator, which indicates the left operand is less than or equal to the right operand, such as `status<=400` or `status:<=400` .
=	Range operator, which indicates the left operand is equal to the right operand, such as `status=400` (equivalent to `status:400`).
\\	Escape symbol, the escaped character represents the symbol itself. When the retrieved value contains spaces, `:` , `(` , `)` , `>` , `=` , `<` , `"` , `'` , or `*` , it needs to be escaped. For example: `body:user_name\\:bob` When using double quotes for phrase search, only `"` and `*` need to be escaped. When using single quotes for phrase search, only `'` and `*` need to be escaped. Unescaped `*` represents fuzzy retrieval.
key:*	Field of the `text` type: Queries logs containing the field (`key`), no matter whether the value is empty, such as `url:*` . Field of the `long` / `double` type: Queries logs containing the field (`key`) whose value is not empty, such as `response_time:*` .
key:""	Field of the `text` type: Queries logs containing the field (`key`) whose value is empty (the value is also empty if it contains only delimiters), such as `url:""` . Field of the `long` / `double` type: Queries logs not containing the field (`key`) or containing the field whose value is empty (equivalent to `NOT key:*`).

Sample	Statement
Logs from a specified server	`__SOURCE__:127.0.0.1` or `__SOURCE__:192.168.0.*`
Logs from a specified file	`__FILENAME__:"/var/log/access.log"`
Logs containing `ERROR`	`ERROR`
Logs of failures (with a status code greater than 400)	`status>400`
Logs of failed `GET` requests (with a status code greater than 400)	`method:GET AND status>400`
Logs at `ERROR` or `WARNING` level	`level:(ERROR OR WARNING)`

Logs except those at `INFO` level	`NOT level:INFO`

Phrase search

A
string is enclosed in double or single quotation marks for search, such as `name:"john Smith"` and `filepath:"/var/log/access.log"` , Compared with searches without quotation marks, a phrase search means that the matched logs should contain all the words in the string and in the same sequence as required in the search condition.
Below are two sample logs with the delimiter of `/` :

```
#1 filepath:"/var/log/access.log"
#2 filepath:"/log/var/access.log"
```

When you use `filepath:/var/log/access.log` for search, the above two logs will be matched, as it does not involve the sequence of words.
When you use `filepath:"/var/log/access.log"` for search, only the first log will be matched.
Phrase searches have stricter search conditions and are recommended when long strings are searched for.
Note:
Phrase searches support wildcards such as `filepath:"/var/log/acc*.log"` but not in the beginning of words such as `filepath:"/var/log/*cess.log"` .
Wildcards in phrase searches can only match the first 128 words meeting the search condition and return all logs containing these words. The more specific the words, the more accurate the results. This is not the case for non-phrase searches.

Fuzzy search

To
perform a fuzzy search, you need to add wildcards to the middle or end of words. You can use the asterisk `*` to match zero, one, or multiple characters, for example:
`IP:192.168.1.*` can be used to match `192.168.1.1` and `192.168.1.34` .
`host:www.te*t.com` can be used to match `www.test.com` and `www.telt.com` .
Note:
The asterisk `*` cannot be used at the beginning of a word; that is, fuzzy prefix search is not supported.
Fields of the `long` or `double` type support a value range but not the asterisk `*` for a fuzzy search, such as `status>400 and status<500` .
If you need to use fuzzy search with prefix specified, you can use the following methods.

Adding a prefix: for example, if the logs are `host:www.test.com` , `host:m.test.com` , and you need to query logs containing `test` in the middle, you can add the prefix `.` to search for logs with `host:test` .
Use the strpos function in SQL: for example `* | select * where strpos(host,'test')>0` , but this approach has poorer performance compared with retrieval conditions and is not suitable for scenarios with large log data volumes.
Phrase searches support wildcards such as `filepath:"/var/log/acc*.log"` but not in the beginning of words such as `filepath:"/var/log/*cess.log"` . In addition, wildcards in phrase searches can only match the first 128 words meeting the search condition and return all logs containing these 128 words. The more specific the words, the more accurate the results. This restriction is not applicable to non-phrase searches.

Syntax rules

Syntax	Description
AND	Logical AND operator, such as `level:ERROR AND pid:1234` .
OR	Logical OR operator, such as `level:ERROR OR level:WARNING` .
NOT	Logical NOT operator, such as `level:ERROR NOT pid:1234` .
()	Grouping operator, which controls the precedence of logical operations, such as `(ERROR OR WARNING) AND pid:1234` .
:	Colon, which is used for key-value search, such as `level:ERROR` .
""	Double quotation marks, which quote a phrase to match logs that contain all the words in the phrase and in the same sequence, such as `name:"john Smith"` .
*	Wildcard, which is used to replace zero, one, or more characters, such as `host:www.test*.com` . Prefix fuzzy queries are not supported. You can also use `key:*` to query logs where the specified field (`key`) exists. `key:*` is equivalent to `_exists_:key` .
?	Wildcard, which can match one single character, such as `host:www.te?t.com` . Similar to `*` , it does not support prefix fuzzy queries.
>	Range operator, which indicates the left operand is greater than the right operand, such as `status:>400` .
>=	Range operator, which indicates the left operand is greater than or equal to the right operand, such as `status:>=400` .
<	Range operator, which indicates the left operand is less than the right operand, such as `status:<400` .
<=	Range operator, which indicates the left operand is less than or equal to the right operand, such as

	`status:<=400` .
TO	Logical TO operator, such as `request_time:[0.1 TO 1.0]` .
[]	Range operator, which includes the upper and lower boundary values, such as `age:[20 TO 30]` .
{}	Range operator, which excludes the upper and lower boundary values, such as `age:{20 TO 30}` .
\\	Escape character. An escaped character represents the literal meaning of the character, such as `url:\\/images\\/favicon.ico` . You can also use `""` to wrap special characters as a whole, e.g., `url:"/images/favicon.ico"` . Note that the characters in the double quotation marks are considered as a phrase to match logs that contain all the words in the phrase and in the same sequence.
exists	`\\_exists\\_:key` returns logs that contains `key` . For example, `_exists_:userAgent` means to return logs that contains the `userAgent` field.

Note:

The syntax is case-sensitive. For example, `AND` and `OR` represent logical search operators, while `and` and `or` are regarded as common text.

When multiple search conditions are connected with spaces, they are regarded as in the `OR` logic. For example, `warning error` is equivalent to `warning OR error` .

The following special characters must be escaped: +, -, &&, ||, !, ( ), { }, [ ], ^, ", ~, *, ?, :, \\

Use `()` to group search conditions and clarify the precedency when using the "AND" and "OR" operators, such as `(ERROR OR WARNING) AND pid:1234` .

Sample	Statement
Logs from a specified server	`__SOURCE__:127.0.0.1` or `__SOURCE__:192.168.0.*`
Logs from a specified file	`__FILENAME__:"/var/log/access.log"` or `__FILENAME__:\\/var\\/log\\/*.log`
Logs containing `ERROR`	`ERROR`
Logs of failures (with a status code greater than 400)	`status:>400`
Failed logs in the `GET` request (with a status code greater than 400)	`method:GET AND status:>400`

Logs at `ERROR` or `WARNING` level	`level:ERROR OR level:WARNING`
Logs except those at `INFO` level	`NOT level:INFO`
Logs from `192.168.10.10` but except those at `INFO` level	`__SOURCE__:192.168.10.10 NOT level:INFO`
Logs from the `/var/log/access.log` file on `192.168.10.10` but except those at `INFO` level	`(__SOURCE__:192.168.10.10 AND __FILENAME__:"/var/log/access.log.*") NOT level:INFO`
Logs from `192.168.10.10` and at `ERROR` or `WARNING` level	`__SOURCE__:192.168.10.10 AND (level:ERROR OR level:WARNING)`
Logs with a status code of `4XX`	`status:[400 TO 500}`
Logs with the container name `nginx` in the metadata	`__TAG__.container_name:nginx`
Logs with the container name `nginx` in the metadata, and request latency greater than 1s	`__TAG__.container_name:nginx AND request_time:>1`
Logs containing the `message` field	`message:*` or `_exists_:message`
Logs that do not contain the `message` field	`NOT _exists_:message`

Fuzzy search

To
perform a fuzzy search via CLS, you need to add wildcards to the middle or end of words, either by using the asterisk `*` to match zero, single, or multiple characters, or using the question mark `?` to match a single character. The following are examples:

`IP:192.168.1.*` can be used to match `192.168.1.1` and `192.168.1.34`.

`host:www.te*t.com` can be used to match `www.test.com` and `www.telt.com`.

Note:

The asterisk `*` or question mark `?` cannot be used at the beginning of a word, i.e. prefix fuzzy searches are not supported.

Data of `long` or `double` type does not support an asterisk `*` or question mark `?` for fuzzy search, but it supports a value range for fuzzy search, such as `status:[400 TO 500}`.

If you need to use fuzzy search with prefix specified, you can use the following methods.

Adding a prefix: for example, if the logs are `host:www.test.com`, `host:m.test.com`, and you need to query logs containing `test` in the middle, you can add the prefix `.` to search for logs with `host:test`.

Using the `LIKE` syntax: For example, you can use `* | select * where host like '%test%'`. However, this method delivers lower performance than the search condition method and is not suitable for scenarios with large volume of log data.

The CQL syntax is much easier to use than the Lucene syntax, and some features not commonly used are simplified in the former. The differences between the two are as follows:

Feature	Lucene	CQL
Logical operator	Only uppercase letters are supported, such as `AND`, `NOT`, and `OR`.	Both uppercase and lowercase letters are supported, such as `AND`, `and`, `NOT`, `not`, `OR,` and `or`.
Symbol escape	Many symbols need to be escaped. For example, to search for `/book/user/login/`, you need to escape it as `\\/book\\/user\\/login\\/`.	Few symbols need to be escaped, and you can search for `/book/user/login/` directly.
Keyword search	The logical relationship between segments in a keyword is `OR`. For example, if the delimiter is `/`, `/book/user/login/` is equivalent to `book OR user OR login`, and many irrelevant logs will be matched.	The logical relationship between segments in a keyword is `AND`. For example, if the delimiter is `/`, `/book/user/login/` is equivalent to `book AND user AND login`, which is in line with search habits.
Phrase search	Phrase searches do not support wildcards. For example, `"/book/user/log*/"` cannot match `/book/user/login/` and `/book/user/logout/`.	Phrase searches support wildcards. For example, `"/book/user/log*/"` can match `/book/user/login/` and `/book/user/logout/`.
Numeric range search	Use the syntax in the form of `timeCost:[20 TO 30]` for retrieval.	Use the syntax in the form of `timeCost>=20 AND timeCost<=30` for retrieval.
Logs with existing fields	Search using `_exists_:key`, where key is the field name.	Search using `key:*`, where key is the field name.

SQL Statement Syntax

Syntax rules

Syntax	Description
SELECT	Selects data from a table. It selects eligible data from the current log topic by default. Example:

	`level:ERROR
AS	Specifies an alias for a column (KEY). Example: `level:ERROR
GROUP BY	Combines aggregate functions to group results based on one column (KEY) or more. Example: `level:*
ORDER BY	Sorts results according to the specified `KEY`. Example: `level:*
LIMIT	Limits the amount of data returned by the `SELECT` statement. Example: `level:*
WHERE	Filters the original data found. Example: `level:ERROR
HAVING	Filters grouped and aggregated data. The difference between `HAVING` and `WHERE` is that `HAVING` is executed on data after grouping (`GROUP BY`) and before ordering (`ORDER BY`) while `WHERE` is executed on the original data before aggregate. Example: `level:*
Nested Subquery	In some complex statistical analysis scenarios, you need to perform statistical analysis on the original data first and then perform secondary statistical analysis on the analysis results. In this case, you need to nest a `SELECT` statement into another `SELECT` statement. This query method is called nested subquery. Example: `*

Note:

SQL statements are case-insensitive, so `SELECT` is equivalent to `select`.

Strings must be included in single quotation marks `''`, while characters that are unsigned or included in double quotation marks `""` indicate field or column names. For example, `'status'` indicates the string `status`, while `status` or `"status"` indicates the log field `status`.

When a string contains a single quotation mark `'`, you need to use `''` (two single quotation marks) to represent the single quotation mark itself. For example `'{''version'': ''1.0''}'` indicates the raw string {'version': '1.0'}. No special processing is required if the string itself contains a double quotation mark `"`.

You don't need to add a semicolon at the end of a SQL statement to indicate the end.

SQL functions

CLS supports a large number of SQL functions. For all the SQL functions, see SQL Functions. The following lists some common functions.

Syntax	Description
String function	String concatenation, splitting, length calculation, case conversion, and more.
Date and time functions	Time format conversion, statistics by time, time interval calculation, and more.
IP geographic function	Parsing IPs to obtain geographic information and more.

URL function	Obtaining domain names and parameters from URLs, encoding/decoding URLs, and more.
General aggregate function	Calculating the log count, maximum value, minimum value, average value, and more.
Estimation function	Calculating the number of unique values, percentile values (e.g., p95/p90), and more.
Type conversion function	Variable type conversion; often used in functions that have special requirements on the variable types of parameters.
Logical function	AND, OR, NOT, and other logical operations.
Operator	Mathematical operators (+, -, *, /, etc.) and comparison operators (>, <, etc.).
Conditional expression	Condition determination expressions such as CASE WHEN and IF.
Array function	Getting the elements in an array, and more.
Interval-valued comparison and periodically-valued comparison functions	Comparing the calculation result of the current time period with the calculation result of a time period n seconds before.
JSON function	Getting JSON objects, converting JSON types, and more.

Sample

Sample	Statement
Number of logs of failed `GET` requests (with a status code greater than 400)	`method:GET AND status:>400 | select count(*) as errorCount`
Number of logs of failed `GET` requests (with a status code greater than 400) per minute	`method:GET AND status:>400 | select histogram(__TIMESTAMP__, interval 1 minute) as analytic_time_minute, count(*) as errorCount group by analytic_time_minute limit 1000`
Top five URLs with the largest number of requests	`* | select URL, count(*) as log_count group by URL order by log_count desc limit 5`
Count the proportion of ERROR logs	`* | select round((count_if(upper(Level) = 'ERROR'))*100.0/count(*),2) as "ERROR log percentage (%)"`
Number of requests of each province	`* | select ip_to_province(client_ip) as`

	`province , count(*) as PV group by province order by PV desc limit 1000`

Use limits

Metric	Limit	Remarks
Number of SQL results	Each SQL execution can return up to 10,000 results.	The default value is `100` . You can adjust the limit by using the LIMIT syntax.
Memory usage	Each SQL execution can occupy up to 3 GB of server memory.	Usually, this limit can be triggered when `group by` , `distinct()` , or `count(distinct())` is used, because the fields with statistics collected have too many values after deduplication via `group by` or `distinct()` . We recommend that you optimize the query statement and use fields with fewer values for group statistics, or use `approx_distinct()` instead of `count(distinct())` .

Directions

1. Log in to the CLS console.
2. On the left sidebar, select **Search and Analysis** to go to the search and analysis page.
3. Select the **logset** and **log topic** for log search.
4. Enter a **search and analysis statement** and select a **time range** (choose the last hour, last 4 hours, last day, or last 3 days, or set a custom a time range).
5. Click **Search and Analysis**.

If the search and analysis statement **contains only search conditions**, you can view the logs found on the **Raw Data** tab page. The logs are sorted in descending order based on the log time by default.

If the search and analysis statement **contains SQL statements**, you can view the analysis result on the **Chart Analysis** tab page and change the chart type as needed to view the statistical result more intuitively. You can compare the analysis result and the raw log by switching between the **Chart Analysis** and **Raw Data** tab pages.

Statistical Analysis (SQL)
Quick Analysis

Last updated：2024-01-20 17:25:15

Overview

CLS provides quick statistical analysis of fields. You can quickly analyze field value distribution, trends over time, and numerical statistics without writing query statements. For example, you can quickly collect statistics on the top 5 request APIs and API response time trends.

Prerequisite

You have enabled quick analysis feature for fields in index configuration.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to go to the search and analysis page.
3. Click the drop-down lists of **Logset** and **Log Topic** to select the log topic to be searched.
4. Click **Log Time** and select a log time for search.
You can select **Recent Time** or **Relative Time** or customize the time range.
5. On the left sidebar, click the field name.
Note:
The quick analysis feature for fields of the string type (text) is slightly different from that for fields of the numeric type (long and double):
String: supports field value distributed statistics, intelligent aggregated time line charts and top 5 values.
Numeric: supports intelligent aggregated time line charts and top 5 values.
Gray fields, such as `__PKG_LOGID__` in the above figure, cannot be clicked because statistics are not enabled for them.
6. In the pop-up quick analysis window, click a statistics chart to automatically generate the corresponding search statement and chart. You can query chart details here, and you can further modify the search statement and chart configuration to get an analysis chart that better fits your specific needs.

SQL Syntax
AS Syntax

Last updated：2024-01-20 17:25:15

The `AS` clause is used to specify an alias for a column (KEY).

Syntax Format

```
* | SELECT column name (KEY) AS  alias
```

Syntax Sample

Counting access requests

```
* | SELECT COUNT(*) AS PV
```

GROUP BY Syntax

Last updated：2024-01-20 17:25:15

The `GROUP BY` syntax, together with an aggregate function, is used to group analysis results by one or more columns.

Syntax Format

```
* | SELECT column, aggregate function GROUP BY [ column name | alias | serial number ]
```

Note:

When executing a `SELECT` statement containing the `GROUP BY` syntax, you can select only the `GROUP BY` column or an aggregate calculation function, but not a non-GROUP BY column. For example, `* | SELECT status, request_time, COUNT(*) AS PV GROUP BY status` is an invalid analysis statement because **request_time** is not a `GROUP BY` column.

The `GROUP BY` syntax supports grouping by column name, alias, or serial number, as described in the following table:

Parameter	Description
Column name	Group data by log field name or aggregate function calculation result column. The syntax supports grouping data by one or multiple columns.
Alias	Group data by alias of the log field name or aggregate function calculation result.

Serial number	Serial number (starting from 1) of a column in the `SELECT` statement.
	For example, the serial number of the `status` column is 1, and therefore the following statements are equivalent:
	`* | SELECT status, count(*) AS PV GROUP BY status`
	`* | SELECT status, count(*) AS PV GROUP BY 1`

Aggregate function	The `GROUP BY` syntax is usually used together with aggregate functions such as `MIN`, `MAX`, `AVG`, `SUM`, and `COUNT`. For more information, please see Aggregate Function.

Syntax Example

Count the number of access requests with different status codes:

```
* | SELECT status, count(*) AS pv GROUP BY status
```

Calculate PV by the time granularity of 1 minute:

```
* |
SELECT
date_trunc(
 'minute',
 cast(__TIMESTAMP__ as timestamp)
) AS dt,
count(*) AS pv
GROUP BY
dt
ORDER BY
dt
limit
10
```

The `\\_\\_TIMESTAMP\\_\\_` field is the reserved field in CLS and indicates the time column. **dt** is the alias of `date_trunc('minute', cast(\\_\\_TIMESTAMP\\_\\_ as timestamp))`. For more information on the date_trunc() function, see Time Truncation Function.

Note:

`limit 10` indicates up to 10 rows of results are obtained. If the `LIMIT` syntax is not used, CLS obtains 100 rows of results by default.

If you enable the statistics feature for any field during index configuration, CLS will automatically enable the statistics feature for the `\\_\\_TIMESTAMP\\_\\_` field.

Calculate PV and UV by the time granularity of 5 minutes:

The date_trunc() function collects statistics only at fixed time intervals. You can use the histogram function to collect statistics at custom time intervals.

```
* |
SELECT
histogram(
  cast(TIMESTAMP as timestamp),
  interval 5 minute
) as dt,
count(*) as pv,
count(
  distinct(remote_addr)
) as uv
group by
dt
order by
dt
```

LIMIT Syntax

Last updated：2024-01-20 17:25:15

The `LIMIT` syntax is used to limit the number of rows in the output result.

Syntax Format

Read the first N rows:

```
limit N
```

Read N rows starting from row S:

```
offset S limit N
```

Syntax Example

Get the first 10 rows of results:

```
* | select status, count(*) as pv group by status limit 10
```

Get the results for rows 3 to 42 (40 rows in total):

```
* | select status, count(*) as pv group by status offset 2 limit 40
```

Restrictions

Metric	Restriction	Remarks
Number of SQL results	Each SQL returns up to 10,000 results.	Default: 100; Maximum: 10,000

ORDER BY Syntax

Last updated：2024-01-20 17:25:15

The `ORDER BY` syntax is used to sort analysis results by a specified column name.

Syntax Format

```
ORDER BY column name [DESC | ASC]
```

Note:

You can specify multiple column names to sort analysis results in different sorting modes, for example, `ORDER BY column name 1[DESC | ASC], column name 2[DESC | ASC]`.

If you do not specify the keyword `DESC` or `ASC`, the system sorts the analysis results in ascending order.

If the target column for sorting contains duplicated values, the sorting result may be different each time. If you want the sorting result to remain the same in each sorting, you can specify multiple columns for sorting.

Parameter descriptions:

Parameter	Description
Column name	Group data by log field name or aggregate function calculation result column.
DESC	Sort data in descending order.
ASC	Sort data in ascending order.

Syntax Example

Count the number of requests in different states and sort them in descending order by the number of requests:

```
* |
SELECT
status,
count(*) AS pv
GROUP BY
status
ORDER BY
pv DESC
```

Calculate the average request time for each server and sort them in ascending order:

```
* |
SELECT
remote_addr,
avg(request_time) as request_time
group by
remote_addr
order by
request_time ASC
LIMIT
10
```

SELECT Syntax

Last updated：2024-01-20 17:25:15

The `SELECT` statement is used to select data from a table. It selects eligible data from the current log topic by default.

Syntax Format

```
* | SELECT [Column name(KEY)]
```

Syntax Example

Select values with columns (KEY) being `remote_addr` and `method` from the log data:

```
* | SELECT remote_addr, method
```

Select all columns (KEY) from the log data:

```
* | SELECT *
```

`SELECT` can also be followed by arithmetic expressions; for example, you can query the download speed of log data:
Download speed (`speed`) = total number of bytes sent (`body_bytes_sent`) / request time (`request_time`)

```
* | SELECT body_bytes_sent / request_time AS speed
```

Column Naming Conventions

In SQL specifications, a column name consists of letters, digits, and underscores and starts with a letter, such as `remote_addr` . If a field in a log has a non-compliant name, you need to surround the name by `""` . You can also specify an alias for the field with AS syntax in SQL.
If a log field is named `remote_addr` , which conforms to SQL's column naming conventions, it can be queried by SELECT:

```
* | SELECT remote_addr
```

If a log field is named `__TAG__.pod_label_qcloud-app`, which does not conform to SQL's column naming conventions, it needs to be surrounded by `""`:

```
* | SELECT "__TAG__.pod_label_qcloud-app"
```

If a log field is named `__TIMESTAMP__`, which does not conform to SQL's column naming conventions, it needs to be surrounded by `""` and specified with an alias through the AS syntax:

```
* | SELECT "__TIMESTAMP__" AS log_time
```

WHERE Syntax

Last updated：2024-01-20 17:25:15

The `WHERE` statement is used to extract the logs that meet the specified conditions.

Syntax Format

```
* | SELECT column (KEY) WHERE column (KEY) operator value
```

The operator can be `=` , `<>` , `>` , `<` , `>=` , `<=` , `BETWEEN` , `IN` , or `LIKE` .

Note:

In SQL, search conditions deliver higher performance than filters. You are advised to use search conditions to meet data filtering requirements. For example, you can use `status:>400 | select count(*) as logCounts` instead of `* | select count(*) as logCounts where status>400` to get the statistical result faster.

The `WHERE` statement does not allow the `AS` clause. For example, if `level:* | select level as log_level where log_level='ERROR'` is run, an error will be reported because the statement does not comply with the SQL-92 specifications.

Syntax Example

Query logs with status code greater than 400 in the log data:

```
* | SELECT * WHERE status > 400
```

Query the number of logs whose request method is GET and client IP is 192.168.10.101 in the log data:

```
* | SELECT count(*) as count WHERE method='GET' and remote_addr='192.168.10.101'
```

Count the average size of requests with the URL suffix of .mp4:

```
* | SELECT round(sum(body_bytes_sent) / count(body_bytes_sent), 2) AS avg_size  WHE
```

HAVING Syntax

Last updated：2024-01-20 17:25:15

The `HAVING` syntax is used to filter grouped and aggregated data. The difference between `HAVING` and WHERE is that `HAVING` is executed on data after grouping (`GROUP BY`) and before ordering (`ORDER BY`) while `WHERE` is executed on the original data before aggregation.

Syntax Format

```
* | SELECT column, aggregate function GROUP BY [ column name | alias | serial numbe
```

The operator can be `=` , `<>` , `>` , `<` , `>=` , `<=` , `BETWEEN` , `IN` , or `LIKE` .

Syntax Example

Get URLs whose average response time is greater than 1,000 ms in descending order:

```
* |
select
  avg(responseTime) as time_avg,
  URL
group by
  URL
having
  avg(responseTime)> 1000
order by
  avg(responseTime) desc
limit
  10000
```

The filter condition is the average response time of each URL, which is the aggregate result, and therefore, WHERE cannot be used for data filtering.

Nested Subquery

Last updated：2024-01-20 17:25:15

This document introduces the basic syntax and examples of nested subqueries.

Syntax Format

In some complex statistical analysis scenarios, you need to perform statistical analysis on the original data first and then perform secondary statistical analysis on the analysis results. In this case, you need to nest a `SELECT` statement into another `SELECT` statement. This query method is called nested subquery.

```
* | SELECT key FROM (subquery)
```

subquery: subquery, which needs to be enclosed in parentheses.
key: fields that need to be obtained from a subquery for secondary statistical analysis.
Two or more levels of nesting is supported.

Syntax Example

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 for search and analysis to 1 hour and execute the following search and analysis statement, where `86400` indicates the current time minus 86400 seconds (1 day).

Search and analysis statement

```
* | SELECT compare(PV, 86400) FROM (SELECT count(*) AS PV)
```

`SELECT count(*) AS PV` is level-1 statistical analysis: analyze the website PV based on raw logs.
`SELECT compare(PV, 86400) FROM` is level-2 statistical analysis: perform secondary statistical analysis based on the PV result of the level-1 statistical analysis. Use the compare function to obtain the website PV of the day before.

Search and analysis result

1860: indicates the PVs of the current 1 hour.
1656: indicates the PVs of the same time period the day before.
1.1231884057971016: indicates the ratio of the PVs of the current hour to the PVs of the same time period the day before.
To display the search and analysis result in multiple columns, perform a further nested search:

Search and analysis statement

```
* |
SELECT compare[1] AS today, compare[2] AS yesterday, compare[3] AS ratio
FROM (
    SELECT compare(PV, 86400) AS compare
    FROM (
        SELECT COUNT(*) AS PV
    )
)
```

`SELECT compare[1] AS today, compare[2] AS yesterday, compare[3] AS ratio FROM` is to get the value of a specified position in the result of the `compare` function based on the array subscript.

Search and analysis result

SQL Functions
String Function

Last updated：2024-03-20 11:47:49

This document introduces the basic syntax and examples of string functions.

Note:

Strings must be included in single quotation marks `''`, while characters that are unsigned or included in double quotation marks `""` indicate field or column names. For example, `'status'` indicates the string `status`, while `status` or `"status"` indicates the log field `status`.

When a string contains a single quotation mark `'`, you need to use `''` (two single quotation marks) to represent the single quotation mark itself. For example `'{''version'': ''1.0''}'` indicates the raw string {'version': '1.0'}. No special processing is required if the string itself contains a double quotation mark `"`.

In the following functions, all the `key` parameters indicate log field names.

Function	Description	Example
chr(number)	Returns characters that match the ASCII code point (bit) specified by the input parameter. The return value is of the VARCHAR type.	Return characters that match ASCII code bit 77: `* | SELECT chr(77)`
codepoint(string)	Converts ASCII field values to BIGINT values. The return value is of the integer type.	Convert character values in ASCII code to their corresp `* | SELECT codepoint('M')`
concat(key1, ..., keyN)	Concatenates strings key1, key2, ...keyN. The concatenation effect is consistent with that of the || connectors. The return value is of the VARCHAR type. Note that when the random string is `null`, the return value is	Concatenate multiple strings into one: `* | SELECT concat(remote_addr, host, ti`

	`null` . To skip `null` , use `concat_ws` .	
concat_ws(split_string,key0, ..., keyN)	Concatenates strings key1, key2, ...keyN using `split_string` as the separator. `split_string` can be a string or variable. If `split_string` is null, null values in key1, key2, ...keyN are skipped. The return result is of the VARCHAR type.	Concatenate multiple strings using / as the separator: `* | SELECT concat_ws('/', remote_addr,host,time_local)`
concat_ws(split_string, array(varchar))	Concatenates elements in an array into a string using `split_string` as the separator. If `split_string` is `null` , the result is null and null values in the array are skipped. The return result is of the VARCHAR type. Note: In this function, the `array(varchar)` parameter is an array, not a string.	Concatenate elements in an array into a string using # a (in this example, the output of the `split` function is a `* | select concat_ws('#',split('cloud.tencent.com/p '/'))`
format(format, args...)	Formats the output of the `args` parameter using the `format` format. The return value is of the VARCHAR type.	Format the output of the `remote_addr` and `host` using the format of `IP address: %s, Domain nar` `* | SELECT format('IP address: %s, Doma %s', remote_addr, host)`
hamming_distance(key1, key2)	Returns the Hamming distance between the	Return the Hamming distance between the `remote_a` `remote_addr` strings:

	`key1` and `key2` strings. Note that the two strings must have the same length. The return value is of the BIGINT type.	`* | SELECT hamming_distance(remote_addr remote_addr)`
length(key)	Returns the length of a string. The return value is of the BIGINT type.	Return the length of the `http_user_agent` string: `* | SELECT length(http_user_agent)`
levenshtein_distance(key1, key2)	Returns the Levenshtein distance between the `key1` and `key2` strings. The return value is of the BIGINT type.	Return the Levenshtein distance between the `remote` `http_protocol` strings: `* | SELECT levenshtein_distance(remote_ http_protocol)`
lower(key)	Converts a string to lowercase. The return value is of the VARCHAR type in lowercase.	Convert the `http_protocol` string to lowercase: `* | SELECT lower(http_protocol)`
lpad(key, size, padstring)	Left pads `padString` to a string to `size` characters. If `size` is less than the length of `key` , the result is truncated to `size` characters. `size` must be non-negative, and `padstring` must be non-empty. The return value is of the VARCHAR type.	Left pad the '0' to the `remote_addr` string to 32 cha `* | SELECT lpad(remote_addr, 32, '0')`
ltrim(key)	Removes all leading whitespace characters from a string. The return value is of the VARCHAR type.	Remove all leading whitespace characters from the `http_user_agent` string: `* | SELECT ltrim(http_user_agent)`

position(substring IN key)	Returns the position of `substring` in a string. Positions start with 1. If the position is not found, `0` is returned. This function takes the special syntax `IN` as a parameter. For other information, see strpos(). The return value is of the BIGINT type.	Return the position of the 'G' characters in `http_meth` `* | select position('G' IN http_method)`
replace(key, substring)	Removes all `substring` from the `key` string. The return value is of the VARCHAR type.	Remove all 'Oct' from the `time_local` string: `* | select replace(time_local, 'Oct')`
replace(key, substring, replace)	Replaces all `substring` in a string with the `replace` string. The return value is of the VARCHAR type.	Replace all 'Oct' in the `time_local` string with '10': `* | select replace(time_local,'Oct','10`
reverse(key)	Reverses the `key` string. The return value is of the VARCHAR type.	Reverse the `host` string: `* | select reverse(host)`
rpad(key, size, padstring)	**Right** pads `padstring` to a string to `size` characters. If `size` is less than the length of `key`, the result is truncated to `size` characters. `size` must be non-negative, and `padstring` must be non-empty. The return value is of the VARCHAR type.	Right pad '0' to the `remote_addr` string to 32 chara `* | select rpad(remote_addr, 32, '0')`

rtrim(key)	Removes all trailing whitespace characters from a string. The return value is of the VARCHAR type.	Remove all trailing whitespace characters from the `http_user_agent` string: `* | select rtrim(http_user_agent)`
split(key, delimiter)	Splits a string using a specified delimiter and returns a string array.	Split the `http_user_agent` string using the '/' delin a string array: `* | SELECT split(http_user_agent, '/')`
split(key, delimiter, limit)	Splits a string using a specified delimiter and returns a string array with the maximum length specified by `limit`. The last element in the string array always contains all the remaining part of `key`. `limit` must be a positive integer.	Split the `http_user_agent` string using the '/' delin a string array with the length of 10 characters: `* | SELECT split(http_user_agent, '/',`
split_part(key, delimiter, index)	Splits a string using a specified delimiter and returns the string at the `index` position in the array. Indexes start with 1. If the value of `index` is greater than the length of the array, `null` is returned. The return value is of the VARCHAR type.	Split the `http_user_agent` string using the '/' delin the string at position 1: `* | SELECT split_part(http_user_agent,`
strpos(key, substring)	Returns the position of `substring` in a string. Positions start with 1. If the position is not found, `0` is returned. The return	Return the position of 'org' in the `host` string: `* | SELECT strpos(host, 'org')`

	value is of the BIGINT type.	
strpos(key, substring, instance)	Returns the position of the N-th `instance` of `substring` in the string. If `instance` is a negative number, the position is counted starting from the end of the string. Positions start with 1. If the position is not found, `0` is returned. The return value is of the BIGINT type.	Return the position of the first instance of 'g' in the `hos` `* | SELECT strpos(host, 'g', 1)`
substr(key, start)	Returns the rest of a string from the starting position `start` . Positions start with 1. A negative starting position is interpreted as being relative to the end of the string, for example, [...]. The return value is of the VARCHAR type.	Return the rest of the `remote_user` string from the s character: `* | SELECT substr(remote_user, 2)`
substr(key, start, length)	Returns a substring from a string of `length` length from the starting position `start` . Positions start with 1. A negative starting position is interpreted as being relative to the end of the string. The return value is of the VARCHAR type.	Return the 2nd to 5th characters of the `remote_user` `* | SELECT substr(remote_user, 2, 5)`
translate(key, from, to)	Replaces all	Replace the '123' characters in the `remote` string wi

	characters in `key` that appear in `from` with characters at the corresponding position in `to`. If `from` contains repeated characters, only the first character is counted. If the characters in `from` do not exist in the source, the source is copied directly. If the length of `from` is greater than that of `to`, the corresponding characters will be deleted. The return value is of the VARCHAR type.	characters: `* | SELECT translate(remote_user, '123'`
trim(key)	Removes leading and trailing whitespace characters from a string. The return value is of the VARCHAR type.	Remove leading and trailing whitespace characters from `http_cookies` string: `* | SELECT trim(http`
upper(key)	Converts a string to uppercase. The return value is of the VARCHAR type in uppercase.	Convert the lowercase characters in the `host` string t characters: `* | SELECT upper(host)`
word_stem(word)	Returns the stem of `word` in the English language. The return value is of the VARCHAR type.	Return the English word of 'Mozilla': `* | SELECT word_stem('Mozilla')`
word_stem(word, lang)	Returns the stem of `word` in the `lang` language.	Return the stem of `selects` in English: `* | SELECT word_stem('selects', 'en')`

	The return value is of the VARCHAR type.	

Unicode functions

Function	Description
normalize(string)	Converts a string to the NFC standard format. The return value is of the VARCHAR type.
normalize(string, form)	Converts `string` to the `form` format. The `form` parameter must be keywords (NFD, NFC, NFKD, or NFKC) instead of a string. The return value is of the VARCHAR type.
to_utf8(string)	Converts `string` to a UTF-8 binary string varbinary. The return value is of the VARCHAR type.
from_utf8(binary)	Converts a binary string to a UTF-8 string. Invalid UTF-8 characters will be replaced with "U+FFFD". The return value is of the VARCHAR type.
from_utf8(binary, replace)	Converts a binary string to a UTF-8 string. Invalid UTF-8 characters will be replaced with `replace` . The return value is of the VARCHAR type.

Examples

This section provides samples of the query and analysis statements based on the following log example.

Raw log data sample:

```
10.135.46.111 - - [05/Oct/2015:21:14:30 +0800] "GET /my/course/1 HTTP/1.1"
127.0.0.1 200 782 9703 "http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0"  0.354 0.354
```

After configuring the "Single-line - Full regular expression" collection mode, custom key names are as follows:

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
```

```
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [05/Oct/2015:21:14:30 +0800]
upstream_response_time: 0.354
```

Analysis statement sample:

Split the value of the **request_url** field using the question mark (?), return the first string (the file path part), and then count the number of requests corresponding to different paths:

```
* | SELECT count(*) AS pv, split_part(request_url, '?', 1) AS Path GROUP BY Path OR
```

Extract the first 4 characters (the HTTP part) of the value of the **http_protocol** field, and then count the number of requests corresponding to the HTTP protocol:

```
* | SELECT substr(http_protocol,1,4) AS http_protocol, count(*) AS count group by h
```

Replace the '123' characters in the **remote_user** string with the `ABC` characters and return a result of the VARCHAR type:

```
* | SELECT translate(remote_user, '123', 'ABC')
```

Extract the 2nd to 5th characters from the value of the **remote_user** field:

```
* | SELECT substr(remote_user, 2, 5)
```

Return the position of the 'H' letter in the value of the **http_protocol** field:

```
* | SELECT strpos(http_protocol, 'H')
```

Split the value of the **http_protocol** field into 2 substrings using a slash (/) and return the collection of the substrings:

```
* | SELECT split(http_protocol, '/', 2)
```

Replace 'Oct' in the value of the **time_local** field with '10':

```
* | select replace(time_local, 'Oct', '10')
```

Date and Time Functions

Last updated：2024-03-20 11:47:49

CLS provides time grouping, time truncation, time interval, and time sequence completion functions, and supports format conversion, grouping and aggregation, and other processing of date and time values in logs.

Note:

Among date and time functions, except the `histogram` and `time_series` functions that adopt the UTC+8 time zone, other Unix timestamp (`unixtime`) conversion functions adopt the UTC+0 time zone. To use another time zone, you need to use a function with the specified time zone feature, for example, such as `from_unixtime(__TIMESTAMP__/1000, 'Asia/Shanghai')` , or manually add the time zone offset for `unixtime` , for example, `date_trunc('second', cast(__TIMESTAMP__+8*60*60*1000 as timestamp))` .

Basic Functions

Function	Description	Example
current_date	Returns the current date. Return value format: YYYY-MM-DD, such as `2021-05-21` Return value type: DATE	`* | select curr`
current_time	Returns the current time. Return value format: HH:MM:SS.Ms Time zone, such as `17:07:52.143+08:00` Return value type: TIME	`* | select curr`
current_timestamp	Returns the current timestamp. Return value format: YYYY-MM-DDTHH:MM:SS.Ms Time zone, such as `2021-07-15T17:10:56.735+08:00[Asia/Shanghai]` Return value type: TIMESTAMP	`* | select curr`
current_timezone()	Returns the time zone defined by IANA (America/Los_Angeles) or the offset from UTC (+08:35). Return value type: VARCHAR, such as `Asia/Shanghai`	`* | select curr`

localtime	Returns the local time. Return value format: HH:MM:SS.Ms, such as `19:56:36` Return value type: TIME	`* | select loca`
localtimestamp	Returns the local date and time. Return value format: YYYY-MM-DD HH:MM:SS.Ms, such as `2021-07-15 19:56:26.908` Return value type: TIMESTAMP	`* | select loca`
now()	Returns the current date and time. This function is used in the same way as the current_timestamp function. Return value format: YYYY-MM-DDTHH:MM:SS.Ms Time zone, such as `2021-07-15T17:10:56.735+08:00[Asia/Shanghai]` Return value type: TIMESTAMP	`* | select now(`
last_day_of_month(x)	Returns the last day of a month. Return value format: YYYY-MM-DD, such as `2021-05-31` Return value type: DATE	`* | select last_day_of_mont as timestamp))`
from_iso8601_date(string)	Parses an ISO 8601 formatted string into a date. Return value format: YYYY-MM-DD, such as `2021-05-31` Return value type: DATE	`* | select from 03-21')`
from_iso8601_timestamp(string)	Parses an ISO 8601 formatted string into a timestamp with a time zone. Return value format: HH:MM:SS.Ms Time zone, such as `17:07:52.143+08:00` Return value type: TIMESTAMP	`* | select from_iso8601_tim 13')`
from_unixtime(unixtime)	Parses a Unix formatted string into a timestamp. Return value format: YYYY-MM-DD HH:MM:SS.Ms, such as `2017-05-17 01:41:15.000` Return value type: TIMESTAMP	Example 1: `* | sel from_unixtime(14` Example 2: `* | sel from_unixtime(__`
from_unixtime(unixtime, zone)	Parses a Unix formatted string into a timestamp with a time zone. Return value format: YYYY-MM-DD HH:MM:SS.Ms Time zone, such as `2017-05-`	Example 1: `* | sel from_unixtime(14 'Asia/Shanghai')`

	`17T09:41:15+08:00[Asia/Shanghai]` Return value type: TIMESTAMP	Example 2: `* | sel from_unixtime(__ 'Asia/Shanghai')`
to_unixtime(timestamp)	Parses a timestamp formatted string into a Unix timestamp. Return value type: LONG, such as `1626347592.037`	`* | select to_unixtime(cast timestamp))`
to_milliseconds(interval)	Returns a time interval in milliseconds. Return value type: BIGINT, such as `300000`	`* | select to_m 5 MINUTE)`
to_iso8601(x)	Parses a date and time expression of the DATE or TIMESTAMP type into a date and time expression in the ISO8601 format.	`* | select to_iso8601(curre`
timezone_hour(timestamp)	Returns the hour offset of the timestamp's time zone.	`* | SELECT curr timezone_hour(cu`
timezone_minute(timestamp)	Returns the minute offset of the timestamp's time zone.	`* | SELECT curr timezone_minute(`

Time Grouping Function

The time grouping function can be used to group and aggregate the log data at a given interval. For example, you can use it to count page views (PV) every 5 minutes.

Function format

```
histogram(time_column, interval)
```

Parameter description

Parameter	Description
time_column	Time column (KEY), such as `\\_\\_TIMESTAMP\\_\\_` . The value in this column must be a UNIX timestamp of the LONG type or a date and time expression of the TIMESTAMP type in milliseconds. If a value does not meet the requirement, use the `cast` function to convert the ISO 8601 formatted time string into the TIMESTAMP type, for example, `cast('2020-08-19T03:18:29.000Z' as timestamp)` , or use the `[date_parse](#date_parse)` function to convert a time string of another custom type. If the time column adopts the TIMESTAMP type, the corresponding date and time expression must be in the UTC+0 time zone. If the date and time expression itself is in a different time zone, adjust it to UTC+0 by calculation. For example, if the time zone of the original time is UTC+8,

	use `cast('2020-08-19T03:18:29.000Z' as timestamp) - interval 8 hour` to adjust the time zone.
interval	Time interval. The following time units are supported: SECOND, MINUTE, HOUR, and DAY. For example, `INTERVAL 5 MINUTE` indicates an interval of 5 minutes.

Sample

Count the PV value every 5 minutes:

```
* | select histogram(__TIMESTAMP__, INTERVAL 5 MINUTE) AS dt, count(*) as PV
group by dt order by dt limit 1000
```

Time Completion Function

The `time_series()` function can be used to group and aggregate the log data at a given interval. Its main difference from the `histogram()` function is that it can complete missing data in your query time window.

Note:

The time_series() function must be used with the GROUP BY and ORDER BY syntax, and ORDER BY syntax does not support `desc` sorting.

Function format

```
time_series(time_column, interval, format, padding)
```

Parameter description

Parameter	Description
time_column	Time column (KEY), such as `\\_\\_TIMESTAMP\\_\\_` . The value in this column must be a UNIX timestamp of the LONG type or a date and time expression of the TIMESTAMP type in milliseconds. If a value does not meet the requirement, use the `cast` function to convert the ISO 8601 formatted time string into the TIMESTAMP type, for example, `cast('2020-08-19T03:18:29.000Z' as timestamp)` , or use the `[date_parse](#date_parse)` function to convert a time string of another custom type. If the time column adopts the TIMESTAMP type, the corresponding date and time expression must be in the UTC+0 time zone. If the date and time expression itself is in a different time zone, adjust it to UTC+0 by calculation. For example, if the time zone of the original time is UTC+8, use `cast('2020-08-19T03:18:29.000Z' as timestamp) - interval 8 hour` to adjust the time zone.
interval	Time interval. Valid values are `s` (second), `m` (minute), `h` (hour), and `d` (day). For example, `5m` indicates 5 minutes.

format	Time format of the return result.
padding	Value used to complete missing data. Valid values include: 0: Complete a missing value with `0` null: Complete a missing value with `null` last: Complete a missing value with the value of the previous point in time next: Complete a missing value with the value of the next point in time avg: Complete a missing value with the average value of the previous and next points in time

Sample

Complete data with a time unit of 2 minutes:

```
* | select time_series(__TIMESTAMP__, '2m', '%Y-%m-%dT%H:%i:%s+08:00', '0')  as tim
```

Time Truncation Function

The date_trunc() function truncates a date and time expression based on the date part you specify, supporting alignment by second, minute, hour, day, month, and year. This function is often used in scenarios that require statistical analysis based on time.

Function	Description	Example
date_trunc(unit,x)	Truncates `x` to `unit`. `x` is of the TIMESTAMP type.	`* | SELECT date_trunc('second', cast(__TIMESTAMP__ as timestamp))`

The date_trunc() function supports the following units:

Unit	Example Truncated Value	Description
second	2021-05-21 05:20:01.000	-
minute	2021-05-21 05:20:00.000	-
hour	2021-05-21 05:00:00.000	-
day	2021-05-21 00:00:00.000	Returns the zero o'clock of a specified date.
week	2021-05-19 00:00:00.000	Returns the zero o'clock on Monday of a specified week.
month	2021-05-01 00:00:00.000	Returns the zero o'clock on the first day of a specified month.
quarter	2021-04-01 00:00:00.000	Returns the zero o'clock on the first day of a specified quarter.
year	2021-01-01 00:00:00.000	Returns the zero o'clock on the first day of a specified year.

Time Extraction Functions

Time extraction functions are used to extract the specified time fields, such as the year and month, from date and time expressions.

Function	Description	Example
extract(field FROM x)	Extracts the specified fields from the date and time expression (x).	`* |select extract(hour from cast('2021-05-21 05:20:01.100' as timestamp))`

`field` supports the following values: year, quarter, month, week, day, day_of_month, day_of_week, dow, day_of_year, doy, year_of_week, yow, hour, minute, second.
`extract(field FROM x)` can be simplified to `field()` ; for example, `extract(hour from cast('2021-05-21 05:20:01.100' as timestamp))` can be simplified to `hour(cast('2021-05-21 05:20:01.100' as timestamp))` .

Field	Extraction Result	Description	Simplified Format
year	2021	Extracts the year from the target date.	year(x)
quarter	2	Extracts the quarter from the target date.	quarter(x)
month	5	Extracts the month from the target date.	month(x)
week	20	Calculates the week of the year the target date is in.	week(x)
day	21	Extracts the day from the target date by month, which is equivalent to `day_of_month` .	day(x)
day_of_month	21	Equivalent to `day` .	day(x)
day_of_week[]	5	Calculates the day of the week for the target date, which is equivalent to `dow` .	day_of_week(x)
dow[]	5	Equivalent to `day_of_week` .	day_of_week(x)
day_of_year	141	Calculates the day of the year for the target date, which is equivalent to `doy` .	day_of_year(x)
doy	141	Equivalent to `day_of_year` .	day_of_year(x)
year_of_week	2021	Returns the year for the target date in ISO week date,	year_of_week(x)

		which is equivalent to `yow` .	
yow	2021	Equivalent to `year_of_week` .	year_of_week(x)
hour	5	Extracts the hour from the target date.	hour(x)
minute	20	Extracts the minute from the target date.	minute(x)
second	1	Extracts the second from the target date.	second(x)

Time Interval Functions

Time interval functions perform time period-related operations, such as adding or subtracting a specified interval from a date or counting the time between two dates.

Function	Description	Example
date_add(unit,value,timestamp)	Adds N time units (`unit`) to timestamp. If `value` is a negative value, subtraction is performed.	`* | SELECT date_add('day', -1, TIMESTAMP '2020-03-03 03:01:00')` The return value is the date and time one day earlier than `2020-03-03 03:01:00` , i.e., `2020-03-02 03:01:00` .
date_diff(unit, timestamp1, timestamp2)	Returns the time difference between two time expressions, for example, calculates the number of time units (`unit`) between `timestamp1` and `timestamp2` .	`* |SELECT date_diff('hour', TIMESTAMP '2020-03-01 00:00:00', TIMESTAMP '2020-03-02 00:00:00')` The return value is the time difference between 2020-03-01 and 2020-03-02, i.e., one day.

The following units (`unit`) are supported:

unit	Description
millisecond	Millisecond
second	Second
minute	Minute
hour	Hour

day	Day
week	Week
month	Month
quarter	Quarter of a year
year	Year

Sample

Return the interval value in seconds between '2020-03-01 00:00:00' and '2020-03-02 00:00:00':

```
* | SELECT date_diff('second', TIMESTAMP '2020-03-01 00:00:00', TIMESTAMP '2020-03-
```

Duration Functions

Function	Description	Example
parse_duration(string)	Parses a unit value string into a duration expression. Return value type: INTERVAL, such as `0 00:00:00.043 (D HH:MM:SS.Ms)`	`* | SELECT parse_duration('3.81 d')`
human_readable_seconds(double)	Parses a unit value string into a duration expression. Return value type: VARCHAR, such as `1 minutes` and `36 seconds`	`* | SELECT human_readable_seconds(96)`

The following units are supported:

Unit	Description
ns	Nanosecond
us	Microsecond
ms	Millisecond
s	Second
m	Minute

h	Hour
d	Day

Sample

Parse the unit value string '3.81 d' into a duration string:

```
* | SELECT parse_duration('3.81 d')
```

Time Formatting Function

Function	Description	Example
date_format(timestamp, format)	Parses a date and time string of the `timestamp` type into a string in the `format` format.	`* | select date_format(cast(__TIMESTAMP__ as timestamp), '%Y-%m-%d')`
date_parse(string, format)	Parses a date and time string in the `format` format into the `timestamp` type.	`* | select date_parse('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%s')`

The following formats (`format`) are supported:

Format	Description
%a	Abbreviated names of the days of the week, such as Sun and Sat
%b	Abbreviated month name, such as Jan and Dec
%c	Month, numeric. Value range: 1-12
%d	Day of the month, decimal. Value range: 01-31
%e	Day of the month, decimal. Value range: 1-31
%f	Millisecond. Value range: 0-000000
%H	Hour, in the 24-hour time system
%h	Hour, in the 12-hour time system
%I	Hour, in the 12-hour time system

%i	Minute, numeric. Value range: 00-59
%j	Day of the year. Value range: 001-366
%k	Hour. Value range: 0-23
%l	Hour. Value range: 1-12
%M	Month name in English, such as January and December
%m	Month name in digits, such as 01 and 02
%p	AM or PM
%r	Time, in the 12-hour time system. Format: `hh:mm:ss AM/PM`
%S	Second. Value range: 00-59
%s	Second. Value range: 00-59
%T	Time, in the 24-hour time system. Format: `hh:mm:ss`
%v	Week of the year, where Monday is the first day of the week. Value range: 01-53
%W	Names of the days of the week, such as Sunday and Saturday
%Y	Year (4-digit), such as 2020
%y	Year (2-digit), such as 20
%%	Escape character of %

Sample

Parse the time string '2017-05-17 09:45:00' in `format` into a date and time expression of the TIMESTAMP type, i.e., '2017-05-17 09:45:00.0':

```
* | SELECT date_parse('2017-05-17 09:45:00','%Y-%m-%d %H:%i:%s')
```

IP Geographic Function

Last updated：2024-03-20 11:47:49

IP geographic functions can be used to determine whether an IP address belongs to a private or public network or analyze the country, province, or city to which an IP address belongs. This document introduces the basic syntax and examples of IP geographic functions.

IP Address Function

Note:

The `KEY` field in the following functions indicates the log field (for example, `ip`) and its value is an IP address. If the value is an internal IP address or an invalid field, the value cannot be parsed and is displayed as `NULL` or `Unknown`.

Currently, only IPv4 addresses are supported.

Due to the limitations of the IP address assignment mechanism, the IP address database cannot accurately cover all the geographic information of IP addresses. For a few IP addresses, the detailed geographic information may fail to be queried or the geographic information found may be incorrect.

Function	Description	Example
ip_to_domain(KEY)	Determines whether an IP address belongs to a private or public network. Valid values are `intranet` (private network IP address), `internet` (public network IP address), and `invalid` (invalid IP address).	`* | SELECT ip_to_domain(ip)`
ip_to_country(KEY)	Analyzes the country or region to which an IP address belongs. The country's or region's name is returned.	`* | SELECT ip_to_country(ip)`
ip_to_country_code(KEY)	Analyzes the code of the country or region to which an IP address belongs. The country's or region's code is returned.	`* | SELECT ip_to_country_code(ip)`
ip_to_country_geo(KEY)	Analyzes the latitude and longitude of the country or region to which an IP address belongs. The country's or region's latitude and longitude are returned.	`* | SELECT ip_to_country_geo(ip`)
ip_to_province(KEY)	Analyzes the province to which an IP	`* | SELECT`

	address belongs. The province's name is returned.	`ip_to_province(ip)`
ip_to_province_code(KEY)	Analyzes the code of the province to which an IP address belongs. The province's administrative zone code is returned.	`* | SELECT ip_to_province_code(ip)`
ip_to_province_geo(KEY)	Analyzes the latitude and longitude of the province to which an IP address belongs. The province's latitude and longitude are returned.	`* | SELECT ip_to_province_geo(ip)`
ip_to_city	Analyzes the city to which an IP address belongs. The city's name is returned.	`* | SELECT ip_to_city(ip)`
ip_to_city_code	Analyzes the code of the city to which an IP address belongs. The city's administrative zone code is returned. Currently, cities in Taiwan (China) and outside China are not supported.	`* | SELECT ip_to_city_code(ip)`
ip_to_city_geo	Analyzes the latitude and longitude of the city to which an IP address belongs. The city's latitude and longitude are returned. Currently, cities in Taiwan (China) and outside China are not supported.	`* | SELECT ip_to_city_geo(ip)`
ip_to_provider(KEY)	Analyzes the ISP to which an IP address belongs. The ISP's name is returned.	`* | SELECT ip_to_provider(ip)`

IP Range Function

Note:

The `KEY` field in the following functions indicates the log field (for example, `ip`) and its value is an IP address. In the `ip_subnet_min` , `ip_subnet_max` , and `ip_subnet_range` functions, the value of the `KEY` field is an IP address with a subnet mask (for example, 192.168.1.0/24). If the value field is a general IP address, you need to use the `cancat` function to convert it to an IP address with a subnet mask.

Function	Description	Example
ip_prefix(KEY,prefix_bits)	Gets the prefix of an IP address. An IP address with a subnet mask is returned, for	`* | SELECT ip_prefix(ip,24)`

	example, `192.168.1.0/24`.	
ip_subnet_min(KEY)	Gets the smallest IP address in an IP range. The return value is an IP address, for example, `192.168.1.0`.	`* | SELECT ip_subnet_min(concat(ip,'/24')`
ip_subnet_max(KEY)	Gets the largest IP address in an IP range. The return value is an IP address, for example, `192.168.1.255`.	`* | SELECT ip_subnet_max(concat(ip,'/24')`
ip_subnet_range(KEY)	Gets the range of an IP range. The return value is an IP address of the Array type, for example, `[[192.168.1.0, 192.168.1.255]]`.	`* | SELECT ip_subnet_range(concat(ip,'/24`
is_subnet_of	Determines whether an IP address is in a specified IP range. The return value is of the Boolean type.	`* | SELECT is_subnet_of('192.168.0.1/24',`
is_prefix_subnet_of	Determines whether an IP range is a subnet of a specified IP range. The return value is of the Boolean type.	`* | SELECT is_prefix_subnet_of('192.168.0.1/24',conca '/24'))`

Examples

The following are query and analysis statement examples of IP geographic functions. After performing such query and analysis operations, you can select appropriate statistics charts to display the query and analysis results.

Note:

In the following examples, the log field is `ip`.

Count the total number of requests that are not from the private network:

```
* | SELECT count(*) AS PV where ip_to_domain(ip)!='intranet'
```

Find the top 10 provinces with the largest total number of requests:

```
* | SELECT ip_to_province(ip) AS province, count(*) as PV GROUP BY province ORDER B
```

If the result includes requests from the private network and you want to exclude them, use the following query and analysis statement:

```
* | SELECT ip_to_province(ip) AS province, count(*) as PV where ip_to_domain(ip)!='
```

Collect the longitude and latitude statistics of IP addresses to determine client distribution:

```
* | SELECT ip_to_geo(ip) AS geo, count(*) AS pv GROUP BY geo ORDER BY pv DESC
```

URL Function

Last updated：2024-03-20 11:47:49

This document introduces the basic syntax and examples of URL functions.

Syntax Format

URL functions can extract fields from standard HTTP URLs. The following is an example of a standard URL:

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

Note:

The extracted fields do not include URL delimiters `:` and `?`.

Common URL Functions

Function	Description	Example
url_extract_fragment(url)	Extracts `fragment` from the URL. The result is of the varchar type.	`* | select url_extract_fragment('https://console.intl.clou demo/categoryList')`
url_extract_host(url)	Extracts `host` from the URL. The result is of the varchar type.	`* | select url_extract_host('https://console.i`
url_extract_parameter(url, name)	Extracts the value of `query` from the URL. The result is of the varchar type.	`* | select url_extract_parameter('https://cons region=ap-chongqing','region')`
url_extract_path(url)	Extracts `path` from the URL. The	`* | select url_extract_path('https://console.i region=ap-chongqing')`

	result is of the varchar type.	
url_extract_port(url)	Extracts `port` from the URL. The result is of the bigint type.	`* | select url_extract_port('https://console.i region=ap-chongqing')`
url_extract_protocol(url)	Extracts `protocol` from the URL. The result is of the varchar type.	`* | select url_extract_protocol('https://conso region=ap-chongqing')`
url_extract_query(url)	Extracts the key of `query` from the URL. The result is of the varchar type.	`* | select url_extract_query('https://console region=ap-chongqing')`
url_encode(value)	Escapes `value` so that it can be used in `URL_query`. Letters will not be decoded. .-*_ will not be encoded. Spaces are decoded as +. Other characters are decoded into the UTF-8 format.	`* | select url_encode('https://console.intl.cl chongqing')`
url_decode(value)	Decodes the URL.	`* | select url_decode('https%3A%2F%2Fconsole.cloud.tencent chongqing')`

Mathematical Calculation Functions

Last updated：2024-03-20 11:47:49

This document introduces the basic syntax and examples of mathematical calculation functions.
CLS's log analysis feature allows you to analyze logs by analyzing fields of int, long, and double types via mathematical calculation functions and mathematical statistical functions.

Note:

Mathematical calculation functions support operators +-*/%.

`x` and `y` in the following functions can be numbers, log fields, or expressions with numerical calculation results.

Basic Syntax

Function	Description
abs(x)	Returns the absolute value of `x`.
cbrt(x)	Returns the cube root of `x`.
sqrt(x)	Returns the square root of `x`.
cosine_similarity(x,y)	Returns the cosine similarity between the vectors `x` and `y`. For example, `* | SELECT cosine_similarity(MAP(ARRAY['x','y'], ARRAY[1.0,0.0]), MAP(ARRAY['x','y'], ARRAY[0.0,1.0]))` returns 0.
degrees(x)	Converts angle `x` in radians to degrees.
radians(x)	Converts angle `x` in degrees to radians.
e()	Returns the natural logarithm of the number.
exp(x)	Returns the exponent of the natural logarithm.
ln(x)	Returns the natural logarithm of `x`.
log2(x)	Returns the base-2 logarithm of `x`.
log10(x)	Returns the base-10 logarithm of `x`.
log(x,b)	Returns the base-b logarithm of `x`.
pi()	Returns the value of Pi, accurate to 14 decimal places.
pow(x,b)	Returns `x` raised to the power of `b`.

rand()	Returns a random value.
random(0,n)	Returns a random number within the [0,n) range.
round(x)	Returns the rounded value of `x`.
round(x, N)	Returns `x` rounded to `N` decimal places.
floor(x)	Returns `x` rounded down to the nearest integer.
ceiling(x)	Returns `x` rounded up to the nearest integer.
from_base(varchar, bigint)	Converts a string into a number based on BASE encoding.
to_base(x, radix)	Converts a number into a string based on BASE encoding.
truncate(x)	Returns `x` rounded to an integer by dropping digits after the decimal point.
acos(x)	Returns the arc cosine of `x`.
asin(x)	Returns the arc sine of `x`.
atan(x)	Returns the arc tangent of `x`.
atan2(y,x)	Returns the arc tangent of the result of dividing `y` by `x`.
cos(x)	Returns the cosine of `x`.
sin(x)	Returns the sine of `x`.
cosh(x)	Returns the hyperbolic cosine of `x`.
tan(x)	Returns the tangent of `x`.
tanh(x)	Returns the hyperbolic tangent of `x`.
infinity()	Returns the constant representing positive infinity.
is_nan(x)	Determines if the target value is Not a Number (NaN).
nan()	Returns a "Not a Number" (NaN) value.
mod(x, y)	Returns the remainder when `x` is divided by `y`.
sign(x)	Returns the sign of `x` represented by 1, 0, or -1.
width_bucket(x,	Returns the bucket number of `x` in an equi-width histogram, with `n` buckets within

bound1, bound2, n)	bounds of bound1 and bound2. For example, `* | select timeCost,width_bucket(timeCost,10,1000,5)`
width_bucket(x, bins)	Returns the bin number of `x` with specific bins specified by the array `bins` . For example, `* | select timeCost,width_bucket(timeCost,array[10,100,1000])`

Examples

To compare the PV values of today and yesterday and express the result as a percentage, the query and analysis statement is as follows:

```
* | SELECT diff [1] AS today, round((diff [3] -1.0) * 100, 2) AS growth FROM (SELEC
```

Mathematical Statistical Function

Last updated：2024-03-20 11:47:49

This document introduces the basic syntax and examples of mathematical statistical functions.

Basic syntax

Function	Description
corr(key1, key2)	Returns the correlation coefficient of two columns. The calculation result range is [0,1].
covar_pop(key1, key2)	Returns the population covariance of two columns.
covar_samp(key1, key2)	Returns the sample covariance of two columns.
regr_intercept(key1, key2)	Returns linear regression intercept of input values. `key1` is the dependent value. `key2` is the independent value.
regr_slope(key1, key2)	Returns linear regression slope of input values. `key1` is the dependent value. `key2` is the independent value.
stddev(key)	Returns the sample standard deviation of the `key` column. This function is equivalent to the `stddev_samp` function.
stddev_samp(key)	Returns the sample standard deviation of the `key` column.
stddev_pop(key)	Returns the population standard deviation of the `key` column.
variance(key)	Returns the sample variance of the `key` column. This function is equivalent to the `var_samp` function.
var_samp(key)	Returns the sample variance of the `key` column.
var_pop(key)	Returns the population variance of the `key` column.

Example

Returns the correlation between data in the `timeCost` (response time) and `SamplingRate` (sampling rate) columns:

```
* | select corr(timeCost,SamplingRate)
```

General Aggregate Function

Last updated：2024-03-20 11:47:49

This document describes the basic syntax and examples of aggregate functions.
An aggregate function calculates a set of values and returns the calculation result. CLS supports the following aggregate functions.

Note:
In CLS analysis statements, strings must be included in single quotes (''), and field names and column names are unsigned or included in double quotes (""). For example, 'status' indicates the string **'status'**, and **status** or **"status"** indicates the log field `status` .

Function	Description	Example
arbitrary(KEY)	Returns an arbitrary non-null value of the `KEY` column.	`* | SELECT arbitrary(request_method) AS request_method`
avg(KEY)	Returns the average (arithmetic mean) of the KEY column.	`* | SELECT AVG(request_time)`
bitwise_and_agg(KEY)	Returns the bitwise AND result of all input values of the KEY column.	`* | SELECT bitwise_and_agg(status)`
bitwise_or_agg(KEY)	Returns the bitwise OR result of all input values of the KEY column.	`* | SELECT bitwise_or_agg(request_length)`
checksum(KEY)	Returns the checksum of the KEY column. The return result is of Base64 encoding type.	`* | SELECT checksum(request_method) AS request_method`
count(*)	Returns the number of input rows.	`* | SELECT COUNT(*) WHERE http_status >200`
count(1)	Returns the number of input rows. This function is equivalent to count(*).	`* | SELECT COUNT(1)`
count(KEY)	Returns the number of non-null input values of the KEY column.	`* | SELECT COUNT(request_time) WHERE request_time >5.0`
count_if(boolean)	Returns the number of logs that meet specified conditions.	`* | select count_if(returnCode>=400) as errorCounts`
geometric_mean(KEY)	Returns the geometric mean of KEY,	`* | SELECT`

	which cannot contain negative numbers; otherwise, the result will be NaN.	`geometric_mean(request_time) AS request_time`
max(KEY)	Returns the maximum value of KEY.	`* | SELECT MAX(request_time) AS max_request_time`
max_by(x,y)	Returns the value of x associated with the maximum value of y over all input values.	`* | SELECT MAX_BY(request_method, request_time) AS method`
max_by(x,y,n)	Returns n values of x associated with the n largest of all input values of y in descending order of y.	`* | SELECT max_by(request_method, request_time, 3) AS method`
min(KEY)	Returns the minimum value of KEY.	`* | SELECT MIN(request_time) AS min_request_time`
min_by(x,y)	Returns the value of x associated with the minimum value of y over all input values.	`* | SELECT min_by(request_method, request_time) AS method`
min_by(x,y,n)	Returns n values of x associated with the n smallest of all input values of y in descending order of y.	`* | SELECT min_by(request_method, request_time, 3) AS method`
sum(KEY)	Returns the sum of the KEY column.	`* | SELECT SUM(body_bytes_sent) AS sum_bytes`
bool_and(boolean)	Returns TRUE if all logs meet the specified condition or FALSE otherwise.	`* | select bool_and(returnCode>=400)`
bool_or(boolean)	Returns TRUE if any log meets the specified condition or FALSE otherwise.	`* | select bool_or(returnCode>=400)`
every(boolean)	Equivalent to `bool_and(boolean)` .	`* | select every(returnCode>=400)`

Field description

Parameter	Description
KEY	Name of the log field.

x	The parameter value can be of any data type.
y	The parameter value can be of any data type.
n	An integer greater than 0.

Geospatial Function

Last updated：2024-03-20 11:47:49

This document introduces the basic syntax and examples of geospatial functions.

Concepts

Geospatial functions support Well-Known Text (WKT) and Well-Known Binary (WKB) forms of geometries. For related concepts, see Well-known text representation of geometry - Wikipedia.

Geometry	WKT Format
Point	POINT (0 0)
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, -2 -2, -2 -1, -1 -1)))
GeometryCollection	GEOMETRYCOLLECTION (POINT(2 3), LINESTRING (2 3, 3 4))

Geometries default to plane geometries, in which the shortest distance between two points is a straight line.
Geometries also support spherical geometries, where the shortest distance between two points is a great circle arc.
You can use `to_spherical_geography()` to convert a plane geometry into a spherical geometry.
For example:
`ST_Distance(ST_Point(-71.0882, 42.3607), ST_Point(-74.1197, 40.6976))` calculates the distance between two points on a plane, and the result is 3.4577.
`ST_Distance(to_spherical_geography(ST_Point(-71.0882, 42.3607)), to_spherical_geography(ST_Point(-74.1197, 40.6976)))` calculates the distance between two points on a sphere, and the result is 312822.179.
When a length is calculated (for example, `ST_Distance()` and `ST_Length()`), the unit is meter. When an area is calculated (for example, `ST_Area()`), the unit is square meter.

Constructing a Geometry

Function	Return Value Type	Description
ST_Point(double, double)	Point	Constructs a point.
ST_LineFromText(varchar)	LineString	Constructs a LineString based on WKT text.
ST_Polygon(varchar)	Polygon	Constructs a polygon based on WKT text.
ST_GeometryFromText(varchar)	Geometry	Constructs a geometry based on WKT text.
ST_GeomFromBinary(varbinary)	Geometry	Constructs a geometry based on WKB representation.
ST_AsText(Geometry)	varchar	Converts a geometry into WKT format.
to_spherical_geography(Geometry)	SphericalGeography	Converts a plane geometry into a spherical geometry.
to_geometry(SphericalGeography)	Geometry	Converts a spherical geometry into a plane geometry.

Relationship Tests

Function	Return Value Type	Description
ST_Contains(Geometry, Geometry)	boolean	Returns `true` if and only if no points of the second geometry lie in the exterior of the first geometry, and at least one point of the interior of the first geometry lies in the interior of the second geometry. Returns `false` if the second geometry lies exactly on the boundary of the first geometry.
ST_Crosses(Geometry, Geometry)	boolean	Returns `true` if the given geometries have some, but not all, interior points in common.
ST_Disjoint(Geometry, Geometry)	boolean	Returns `true` if the given geometries do not spatially intersect.
ST_Equals(Geometry, Geometry)	boolean	Returns `true` if the given geometries represent the same geometry.

ST_Intersects(Geometry, Geometry)	boolean	Returns `true` if the given geometries spatially intersect in two dimensions.
ST_Overlaps(Geometry, Geometry)	boolean	Returns `true` if the given geometries are of the same dimension, but are not completely contained by each other.
ST_Relate(Geometry, Geometry)	boolean	Returns `true` if the first geometry is spatially related to the second geometry.
ST_Touches(Geometry, Geometry)	boolean	Returns `true` if a geometry spatially touches another geometry, but their interiors do not intersect.
ST_Within(Geometry, Geometry)	boolean	Returns `true` if first geometry is completely inside the second geometry. Return `false` if their boundaries intersect.

Operations

Function	Return Value Type	Description
geometry_nearest_points(Geometry, Geometry)	row(Point, Point)	Returns the two closest points between two geometries.
geometry_union(array(Geometry))	Geometry	Combines multiple geometries into one.
ST_Boundary(Geometry)	Geometry	Returns the boundary (closure) of a geometry.
ST_Buffer(Geometry, distance)	Geometry	Returns a geometric object that represents the union of all points whose distance from a geometry is less than or equal to a specified value.
ST_Difference(Geometry, Geometry)	Geometry	Returns the geometry value that represents the point set difference of the two given geometries.
ST_Envelope(Geometry)	Geometry	Returns the bounding rectangular polygon of the geometry.
ST_ExteriorRing(Geometry)	Geometry	Returns the exterior ring of the input polygon.
ST_Intersection(Geometry, Geometry)	Geometry	Returns the geometry value that represents the point set intersection of two given geometries.
ST_SymDifference(Geometry, Geometry)	Geometry	Returns the geometry value that represents the point set symmetric difference of two geometries.

Accessors

Function	Return Value Type	Description
ST_Area(Geometry)	double	Returns the area of a polygon in a plane geometry.
ST_Area(SphericalGeography)	double	Returns the area of a polygon in a spherical 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 the geometry.
ST_Dimension(Geometry)	bigint	Returns the inherent dimension of this geometry, which must be less than or equal to the coordinate dimension.
ST_Distance(Geometry, Geometry)	double	Returns the minimum distance between two geometries.
ST_Distance(SphericalGeography, SphericalGeography)	double	Returns the smallest distance between two spherical geographies.
ST_IsClosed(Geometry)	boolean	Returns `true` if the start and end points of the given geometry are the same.
ST_IsEmpty(Geometry)	boolean	Returns `true` if the geometry is an empty GeometryCollection, polygon, point etc.
ST_IsRing(Geometry)	boolean	Returns `true` if and only if the geometry is a closed and simple line.
ST_Length(Geometry)	double	Returns the length of a LineString or MultiLineString on a plane geometry.
ST_Length(SphericalGeography)	double	Returns the length of a LineString or MultiLineString on a spherical geometry.
ST_XMax(Geometry)	double	Returns the X maxima of a bounding box of a geometry.
ST_YMax(Geometry)	double	Returns the Y maxima of a bounding box of a geometry.
ST_XMin(Geometry)	double	Returns the X minima of a bounding box of a geometry.
ST_YMin(Geometry)	double	Returns the Y minima of a bounding box of a geometry.

ST_StartPoint(Geometry)	point	Returns the first point of a LineString geometry.
ST_EndPoint(Geometry)	point	Returns the last point of a LineString geometry.
ST_X(Point)	double	Returns the X coordinate of the point.
ST_Y(Point)	double	Returns the Y coordinate of the point.
ST_NumPoints(Geometry)	bigint	Returns the number of points in a geometry.
ST_NumInteriorRing(Geometry)	bigint	Returns the number of the interior rings of a polygon.

Binary String Function

Last updated：2024-03-20 11:47:49

This document introduces the basic syntax and examples of binary string functions.

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

Statement	Description
Concatenation function ||	The result of `a || b` is `ab`.
length(binary) → bigint	Returns a binary length.
concat(binary1, ..., binaryN) → varbinary	Concatenates binary strings. This function provides the same functionality as ||.
to_base64(binary) → varchar	Converts a binary string into a base64 string.
from_base64(string) → varbinary	Converts a base64 string into a binary string.
to_base64url(binary) → varchar	Converts a binary string into a base64 string with a URL safe alphabet.
from_base64url(string) → varbinary	Converts a base64 string with a URL safe alphabet into a binary string.
to_hex(binary) → varchar	Converts a binary string into a hexadecimal string.
from_hex(string) → varbinary	Converts a hexadecimal string into a binary string.
to_big_endian_64(bigint) → varbinary	Converts a bigint number into a big endian binary string.
from_big_endian_64(binary) → bigint	Converts a big endian binary string into a number.
md5(binary) → varbinary	Computes the MD5 hash of a binary string.
sha1(binary) → varbinary	Computes the SHA1 hash of a binary string.
sha256(binary) → varbinary	Computes the SHA256 hash of a binary string.
sha512(binary) → varbinary	Computes the SHA512 hash of a binary string.
xxhash64(binary) → varbinary	Computes the xxHash64 hash of a binary string.

Estimation Function

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of estimation functions.

Function	Syntax	Description
approx_distinct	approx_distinct(x)	Returns the approximate number of distinct input values (column x).
approx_percentile	approx_percentile(x,percentage)	Sorts the values in the x column in ascending order and returns the value approximately at the given `percentage` position.
	approx_percentile(x,array[percentage01, percentage02...])	Sorts the values in the x column in ascending order and returns the values approximately at the given `percentage` positions (percentage01, percentage02...).

approx_distinct

The `approx_distinct` function is used to get the approximate number of distinct input values of a field. The standard result deviation is 2.3%.

Syntax

```
approx_distinct(x)
```

Field description

Parameter	Description
x	The parameter value can be of any data type.

Return value type

Bigint

Sample

Use the `count` function to calculate the PV value and use the `approx_distinct` function to get the approximate number of distinct input values of the `client_ip` field and use it as the UV value.

```
* | SELECT count(*) AS PV, approx_distinct(ip) AS UV
```

approx_percentile

The `approx_percentile` function is used to sort values of the target field in ascending order and return the value in the position around `percentage`. It uses the T-Digest algorithm for estimation, which has a low deviation and can meet the most statistical analysis requirements. If needed, you can use `* | select count_if(x<(select approx_percentile(x,percentage)),count(*))` to accurately count the number of field values below `percentage` and the total number of field values respectively and then verify the statistical deviation.

Syntax

Return the value (double) approximately at the given `percentage` position

```
approx_percentile(x, percentage)
```

Return the value (array) approximately at the given `percentage` positions (percentage01,percentage02...)

```
approx_percentile(x, array[percentage01,percentage02...])
```

Field description

Parameter	Description
x	Value type: double
percentage	Value range: [0,1]

Return value type

double or array.

Sample

Sample 1

Sort the values of the **resTotalTime** column and return the value of **resTotalTime** approximately at the 50% position.

```
* | select approx_percentile(resTotalTime,0.5)
```

Sample 2

Sort the values of the **resTotalTime** column and return the values of **resTotalTime** approximately at the 10%, 20%, and 60% positions.

```
* | select approx_percentile(resTotalTime, array[0.2,0.4,0.6])
```

Type Conversion Function

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of type conversion functions.
If you need to distinguish more detailed data types when querying and analyzing data, you can use type conversion functions for data type conversion in query and analysis statements.

Function	Syntax	Description
cast	cast(x as type)	Parses the data type of `x`. During `cast` execution, if a value fails to be parsed, the system terminates the entire query and analysis operation.
try_cast	try_cast(x as type)	Parses the data type of `x`. During `try_cast` execution, if a value fails to be parsed, the system returns `NULL` and continues processing by skipping the value.
typeof	typeof(x)	Returns the data type of `x`.

Note:
When dirty data may exist in logs, you are advised to use the `try_cast` function to avoid query and analysis failures caused by dirty data.

Function cast

The `cast` function is used to parse the data type of `x`. During `cast` execution, if a value fails to be parsed, the system terminates the entire query and analysis operation.

Syntax

```
cast(x as type)
```

Parameter description

Parameter	Description
x	The parameter value can be of any type.
type	SQL data type. Valid values: `bigint`, `varchar`, `double`, `boolean`, `timestamp`, `decimal`, `array`, or `map`.

	For the mappings between index and SQL data types, please see Appendix: Data Type Mappings.

If `type` is `timestamp`, `x` must be a timestamp in milliseconds (such as 1597807109000) or a time string in the ISO 8601 time format (such as 2019-12-25T16:17:01+08:00).

Return value type

The return value type is determined by the `type` parameter.

Example

1. Parse the numeric value 0.01 into a BIGINT value:

```
* | select cast(0.01 as bigint)
```

2. Convert the CLS log collection time `__TIMESTAMP__` to `TIMESTAMP`.

```
* | select cast(TIMESTAMP as timestamp)
```

Function try_cast

The `try_cast` function is used to parse the data type of `x`. During `try_cast` execution, if a value fails to be parsed, the system returns `NULL` and continues processing by skipping the value.

Syntax

```
try_cast(x as type)
```

Parameter description

Parameter	Description
x	The parameter value can be of any type.
type	SQL data type. Valid values: `bigint`, `varchar`, `double`, `boolean`, `timestamp`, `decimal`, `array`, or `map`. For the mappings between index and SQL data types, please see Appendix: Data Type Mappings.

Return value type

The return value type is determined by the `type` parameter.

Example

Parse the **remote_user** field value into a VARCHAR value:

```
* | select try_cast(remote_user as varchar)
```

Appendix: Data Type Mappings

The mappings between index and SQL data types are as follows:

Index Data Type	SQL Data Type
long	bigint
text	varchar
double	double
json	varchar

Logical Function

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of logic functions.

Logical operators

Operator	Description	Example
AND	The result is TRUE only if both the left and right operands are TRUE.	a AND b
OR	The result is TRUE if either of the left and right operands is TRUE.	a OR b
NOT	The result is FALSE only if the right operand is FALSE.	NOT a

NULL-related logical operations

The following truth tables demonstrate the processing of cases where `a` and `b` are TRUE, FALSE, and NULL:

AND and OR truth table

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

NOT truth table

a	NOT a
TRUE	FALSE

FALSE	TRUE
NULL	NULL

Operators

Last updated：2024-01-22 10:52:48

An operator is a reserved word or a character used primarily to specify conditions in a SQL statement and to serve as conjunctions for multiple conditions in a SQL statement.

Mathematical Operators

Mathematical operators are symbols used to process four arithmetic operations. They are the simplest and most commonly used operators, especially for number processing. Almost all number processing involves mathematical operators.

Assume variable `a` holds 1 and variable `b` holds 2, then:

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	a + b
- (Subtraction)	Subtracts the right hand operand from the left hand operand.	a - b
* (Multiplication)	Multiplies values on either side of the operator.	a * b
/ (Division)	Divides the left hand operand by the right hand operand.	b / a
% (Modulus)	Divides the left hand operand by the right hand operand and returns the remainder.	b % a

Comparison Operators

Comparison operators are used to determine the size relationships of values and support any value type that can be compared, such as int, long, double, and text.

Assume variable `a` holds 1 and variable `b` holds 2, then:

Operator	Description	Example
=	Checks if the values of two operands are equal or not. If yes, the condition is true.	a = b

!=	Checks if the values of two operands are equal or not. If no, the condition is true.	a != b
<>	Checks if the values of two operands are equal or not. If no, the condition is true.	a <> b
>	Checks if the value of the left operand is greater than the value of the right operand. If yes, the condition is true.	a > b
<	Checks if the value of the left operand is less than the value of the right operand. If yes, the condition is true.	a < b
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, the condition is true.	a >= b
<=	Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, the condition is true.	a <= b
IN	The IN operator is used to compare a value with a specified list of values.	status IN (200,206,404)
NOT IN	The NOT IN operator is used to compare a value with values that are not in a specified list. It is the opposite of the IN operator.	status NOT IN (200,206,404)
BETWEEN AND	The BETWEEN operator tests if a value is within a specified range (BETWEEN min AND max).	status between 200 AND 400
LIKE	The LIKE operator is used to compare a value with a similar value using the wildcard operator. The percent sign (%) represents zero, one, or multiple characters. The underscore (_) represents a single digit or character.	url LIKE '%.mp4'
IS NULL	The NULL operator compares a value with NULL. If the value is null, the condition is true.	status IS NULL
IS NOT NULL	The NULL operator compares a value with NULL. If the value is not null, the condition is true.	status IS NOT NULL
DISTINCT	Syntax: `x IS DISTINCT FROM y` or `x IS NOT DISTINCT FROM y`. The DISTINCT operator checks if x equals to y. Unlike <>, it can compare nulls. For more information, see Differences between <> and DISTINCT.	NULL IS NOT DISTINCT FROM NULL

LEAST	Syntax: `LEAST(x, y...)` . Returns the minimum value among x,y...	LEAST(1,2,3)
GREATEST	Syntax: `GREATEST(x, y...)` . Returns the maximum value among x,y...	GREATEST(1,2,3)
ALL	Syntax: `x expression operator ALL ( subquery )` Returns `true` if x meets all conditions. Supported operators are `<, >, <=, >=, =, <>, !=` .	Example 1: 21 < ALL (VALUES 19, 20, 21) Example 2: * | SELECT 200 = ALL(SELECT status)
ANY / SOME	Syntax: `x expression operator ANY ( subquery )` or `x expression operator SOME ( subquery )` . Returns `true` if x meets any condition. Supported operators are `<, >, <=, >=, =, <>, !=` .	Example 1: 'hello' = ANY (VALUES 'hello', 'world') Example 2: * | SELECT 200 = ANY(SELECT status)

Differences between <> and DISTINCT:

x	y	x = y	x <> y	x IS DISTINCT FROM y	x IS NOT DISTINCT FROM y
1	1	true	false	false	true
1	2	false	true	true	false
1	null	null	null	true	false
null	null	null	null	false	true

Bitwise Operation

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of bitwise operation functions.

Function	Syntax	Description
bit_count	bit_count(x, bits)	Returns the number of ones in `x` in binary representation.
bitwise_and	bitwise_and(x, y)	Returns the result of the bitwise AND operation on `x` and `y` in binary representation.
bitwise_not	bitwise_not(x)	Inverts all bits of `x` in binary representation.
bitwise_or	bitwise_or(x, y)	Returns the result of the bitwise OR operation on `x` and `y` in binary representation.
bitwise_xor	bitwise_xor(x, y)	Returns the result of the bitwise XOR operation on `x` and `y` in binary representation.

bit_count

The `bit_count` function is used to return the number of ones in `x` .

Syntax

```
bit_count(x, bits)
```

Parameter description

Parameter	Description
x	The parameter value is of the bigint type.
bits	Number of bits, for example, 64 bits.

Return value type

Bigint

Example

Compute the binary representation of the number 24 and return the number of ones in the binary number.

Query and analysis statement

```
* | SELECT bit_count(24, 64)
```

Query and analysis result

```
2
```

bitwise_and

The `bitwise_and` function is used to return the result of the bitwise AND operation on `x` and `y` in binary representation.

Syntax

```
bitwise_and(x, y)
```

Parameter description

Parameter	Description
x	The parameter value is of the bigint type.
y	The parameter value is of the bigint type.

Return value type

Bigint

Example

Perform an AND operation on numbers 3 and 5 in binary form.

Query and analysis statement

```
* | SELECT bitwise_and(3, 5)
```

Query and analysis result

```
1
```

bitwise_not

The `bitwise_not` function is used to invert all bits of `x` in binary representation.

Syntax

```
bitwise_not(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the bigint type.

Return value type

Bigint

Example

Invert all bits of the number 4 in binary form.

Query and analysis statement

```
* | SELECT bitwise_not(4)
```

Query and analysis result

```
-5
```

bitwise_or

The `bitwise_or` function is used to return the result of the bitwise OR operation on `x` and `y` in binary representation.

Syntax

```
bitwise_or(x, y)
```

Parameter description

Parameter	Description

x	The parameter value is of the bigint type.
y	The parameter value is of the bigint type.

Return value type

Bigint

Example

Perform an OR operation on numbers 3 and 5 in binary representation.

Query and analysis statement

```
* | SELECT bitwise_or(3, 5)
```

Query and analysis result

```
7
```

bitwise_xor

The `bitwise_xor` function is used to return the result of the bitwise XOR operation on `x` and `y` in binary representation.

Syntax

```
bitwise_xor(x, y)
```

Parameter description

Parameter	Description
x	The parameter value is of the bigint type.
y	The parameter value is of the bigint type.

Return value type

Bigint

Example

Perform an XOR operation on numbers 3 and 5 in binary representation.

Query and analysis statement

```
* | SELECT bitwise_xor(3, 5)
```

Query and analysis result

```
6
```

Regular Expression Function

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of regular expression functions.

CLS supports the following regular expression functions:

Function	Syntax	Description
regexp_extract_all	regexp_extract_all(x, regular expression)	Extracts the substrings that match a specified regular expression from a specified string and returns a collection of all matched substrings.
	regexp_extract_all(x, regular expression, n)	Extracts the substrings that match a specified regular expression from a specified string and returns a collection of substrings that match the target capture group.
regexp_extract	regexp_extract(x, regular expression)	Extracts and returns the first substring that matches a specified regular expression from a specified string.
	regexp_extract(x, regular expression, n)	Extracts the substrings that match a specified regular expression from a specified string and returns the first substring that matches the target capture group.
regexp_like	regexp_like(x, regular expression)	Checks whether a specified string matches a specified regular expression.
regexp_replace	regexp_replace(x, regular expression)	Deletes the substrings that match a specified regular expression from a specified string and returns the substrings that are not deleted.
	regexp_replace(x, regular expression, replace string)	Replaces the substrings that match a specified regular expression in a specified string and returns the new string after the replacement.
regexp_split	regexp_split(x, regular expression)	Splits a specified string into multiple substrings by using a specified regular expression and returns a collection of the substrings.

regexp_extract_all

The `regexp_extract_all` function is used to extract the substrings that match a specified regular expression from a specified string.

Syntax

Extract the substrings that match a specified regular expression from a specified string and return a collection of all matched substrings.

```
regexp_extract_all(x, regular expression)
```

Extract the substrings that match a specified regular expression from a specified string and return a collection of substrings that match the target capture group.

```
regexp_extract_all(x, regular expression, n)
```

Parameter description

Parameter	Description
x	The parameter value is of the varchar type.
regular expression	The regular expression that contains capture groups. For example, `(\\d)(\\d)(\\d)` indicates three capture groups.
n	The nth capture group. `n` is an integer that starts from 1.

Return value type

Array

Example

Extract all numbers from the value of the `http_protocol` field.

Query and analysis statement

```
* | SELECT regexp_extract_all(http_protocol, '\\d+')
```

Query and analysis result

```
[1,1]
```

regexp_extract

The `regexp_extract` function is used to extract the first substring that matches a specified regular expression from a specified string.

Syntax

Extract and return the first substring that matches a specified regular expression from a specified string.

```
regexp_extract(x, regular expression)
```

Extract the substrings that match a specified regular expression from a specified string and return the first substring that matches the target capture group.

```
regexp_extract(x, regular expression, n)
```

Parameter description

Parameter	Description
x	The parameter value is of the varchar type.
regular expression	The regular expression that contains capture groups. For example, `(\\d)(\\d)(\\d)` indicates three capture groups.
n	The nth capture group. `n` is an integer that starts from 1.

Return value type

Varchar

Example

Example 1. Extract the first number from the value of the `http_protocol` field

Query and analysis statement

```
* | SELECT regexp_extract_all(http_protocol, '\\d+')
```

Query and analysis result

```
1
```

Example 2. Extract the file information from the value of the `request_uri` field and count the number of times each file is accessed

Query and analysis statement

```
* | select regexp_like(server_protocol, '\\d+')
```

Query and analysis result

regexp_like

The `regexp_like` function is used to check whether a specified string matches a specified regular expression.

Syntax

```
regexp_like (x, regular expression)
```

Parameter description

Parameter	Description
x	The parameter value is of the varchar type.
regular expression	Regular expression.

Return value type

Boolean

Example

Check whether the value of the `server_protocol` field contains digits.

Query and analysis statement

```
* | select regexp_like(server_protocol, '\\d+')
```

Query and analysis result

```
TRUE
```

regexp_replace

The `regexp_replace` function is used to delete or replace the substrings that match a specified regular expression in a specified string.

Syntax

Delete the substrings that match a specified regular expression from a specified string and return the substrings that are not deleted.

```
regexp_replace (x, regular expression)
```

Replace the substrings that match a specified regular expression in a specified string and return the new string after the replacement.

```
regexp_replace (x, regular expression, replace string)
```

Parameter description

Parameter	Description
x	The parameter value is of the varchar type.
regular expression	Regular expression.
replace string	Substring that is used to replace the matched substring.

Return value type

String

Example

Delete the version number in the value of the `server_protocol` field and calculate the number of requests for each communication protocol.

Query and analysis statement

```
* | select regexp_replace(server_protocol, '.\\d+') AS server_protocol, count(*) AS
```

Query and analysis result

server_protocol	count
HTTP	357

regexp_split

The `regexp_split` function is used to split a specified string into multiple substrings and return a collection of the substrings.

Syntax

```
regexp_split (x, regular expression)
```

Parameter description

Parameter	Description
x	The parameter value is of the varchar type.
regular expression	Regular expression.

Return value type

Array

Example

Split the value of the `server_protocol` field with forward slashes (/).

Query and analysis statement

```
* | select regexp_split(server_protocol, '/')
```

Query and analysis result

```
["HTTP","1.1"]
```

Lambda Function

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of Lambda functions.
CLS allows you to define Lambda expressions in SQL analysis statements and pass them to specified functions to enrich the expressions of functions.

Syntax

Lambda expressions need to be used together with functions such as filter, reduce, transform, and zip_with. The syntax of a Lambda expression is as follows:

```
parameter -> expression
```

Parameter	Description
parameter	Identifier used to pass the parameter.

expression	Expression. Most MySQL expressions can be used in Lambda expressions, such as:
	`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)`

Example

Example 1. Using the Lambda expression "x-> x is not null"

Return non-null elements in the [5, null, 7, null] array.
Query and analysis statement

```
* | SELECT filter(array[5, null, 7, null], x -> x is not null)
```

Query and analysis result

```
[5,7]
```

Example 2. Using the Lambda expression "0, (s, x) -> s + x, s -> s"

Return the sum of the elements in array [5, 20, 50].

Query and analysis statement

```
* | SELECT reduce(array[5, 20, 50], 0, (s, x) -> s + x, s -> s)
```

Query and analysis result

```
75
```

Example 3. Using the Lambda expression "(k, v) -> v> 10"

Map two arrays to a map with a key value greater than 10.

Query and analysis statement

```
* | SELECT map_filter(map(array['class01', 'class02', 'class03'], array[11, 10, 9])
```

Query and analysis result

```
{"class01":11}
```

Example 4. Using the Lambda expression "(x, y) -> (y, x)"

Swap the positions of two elements in an array and extract the elements with the same index to form a new two-dimensional array.

Query and analysis statement

```
| SELECT zip_with(array['a', 'b', 'c'], array['d', 'e', 'f'], (x, y) -> concat(x, y))
```

Query and analysis result

Example 5. Using the Lambda expression "x -> coalesce (x, 0) +1"

Increment each element in the [5, NULL, 6] array by 1 and return. If the array contains a null element, the null element is converted to 0 and then incremented by 1.

Query and analysis statement

```
* | SELECT transform(array[5, NULL, 6], x -> coalesce(x, 0) + 1)
```

Query and analysis result

```
[6,1,7]
```

Other 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 i
            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 ->
```

Conditional Expressions

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of conditional expressions.

Expression	Syntax	Description
CASE WHEN	CASE WHEN condition1 THEN result1 [WHEN condition2 THEN result2] [ELSE result3] END	Classifies data according to specified conditions.
IF	IF(condition, result1)	If `condition` is `true`, returns `result1`. Otherwise, returns `null`.
	IF(condition, result1, result2)	If `condition` is `true`, returns `result1`. Otherwise, returns `result2`.
NULLIF	NULLIF(expression1, expression2)	Determines whether the values of two expressions are equal. If the values are equal, returns `null`. Otherwise, returns the value of the first expression.
TRY	TRY(expression)	Captures exception information to enable the system to continue query and analysis operations.
COALESCE	COALESCE(expression1, expression2...)	Gets the first non-null value in multiple expressions.

CASE WHEN

The `CASE WHEN` expression is used to classify data.

Syntax

```
CASE WHEN condition1 THEN result1
            [WHEN condition2 THEN result2]
            [ELSE result3]
END
```

Parameter description

Parameter	Description
condition	Conditional expression

result	Return result

Example

Example 1

Extract browser information from the `http_user_agent` field, classify the information into the Chrome, Safari, and unknown types, and calculate the PVs of the three types.

```
* |
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
```

Example 2

Get the statistics on the distribution of different request times.

```
* |
SELECT
CASE
  WHEN request_time < 0.001 then 't0.001'
  WHEN request_time < 0.01 then 't0.01'
  WHEN request_time < 0.1 then 't0.1'
  WHEN request_time < 1 then 't1'
  ELSE 'overtime'
END AS request_time,
count(*) AS pv
GROUP BY
request_time
```

IF

The `IF` expression is used to classify data. It is similar to the `CASE WHEN` expression.

Syntax

If `condition` is `true`, return `result1`. Otherwise, return `null`.

```
IF(condition, result1)
```

If `condition` is `true`, return `result1`. Otherwise, return `result2`.

```
IF(condition, result1, result2)
```

Parameter description

Parameter	Description
condition	Conditional expression
result	Return result

Example

Calculate the proportion of requests with status code 200 to all requests.

```
* | SELECT sum(IF(status = 200, 1, 0)) * 1.0 / count(*) AS status_200_percentag
```

NULLIF

The `NULLIF` expression is used to determine whether the values of two expressions are equal. If the values are equal, return `null`. Otherwise, return the value of the first expression.

Syntax

```
NULLIF(expression1, expression2)
```

Parameter description

Parameter	Description
expression	Any valid scalar expression

Example

Determine whether the values of the `server_addr` and `http_host` fields are the same. If the values are different, return the value of the `server_addr`.

```
* | SELECT NULLIF(server_addr,http_host)
```

TR

The `TRY` expression is used to capture exception information to enable the system to continue query and analysis operations.

Syntax

```
TRY(expression)
```

Parameter description

Parameter	Description
expression	Expression of any type

Example

When an exception occurs during the `regexp_extract` function execution, the `TRY` expression captures the exception information, continues the query and analysis operation, and returns the query and analysis result.

```
* |
SELECT
TRY(regexp_extract(uri, './(index.)', 1))
  AS file, count(*)
  AS count
GROUP BY
file
```

COALESCE

The `COALESCE` expression is used to get the first non-null value in multiple expressions.

Syntax

```
COALESCE(expression1, expression2...)
```

Parameter description

Parameter	Description
expression	Any valid scalar expression

Example

```
* | select COALESCE(null, 'test')
```

Array Functions

Last updated：2024-01-22 10:57:48

This document introduces the basic syntax and examples of array functions.

Function	Syntax	Description
Subscript operator: []	[x]	Returns an element from an array. Equivalent to the `element_at` function.
array_agg	array_agg(x)	Returns all values in `x` as an array.
array_distinct	array_distinct(x)	Deduplicates an array and returns unique values from the array.
array_except	array_except(x, y)	Returns the difference between the `x` and `y` arrays.
array_intersect	array_intersect(x, y)	Returns the intersection between the `x` and `y` arrays.
array_join	array_join(x, delimiter)	Concatenates the elements in an array using the specified delimiter. If the array contains a null element, the null element is ignored.Note: For the `array_join` function, the maximum return result supported is 1 KB, and data exceeding 1 KB will be truncated.
	array_join(x, delimiter, null_replacement)	Concatenates the elements in an array using `delimiter` and uses `null_replacement` to replace null values.Note: For the `array_join` function, the maximum return result supported is 1 KB, and data exceeding 1 KB will be truncated.
array_max	array_max(x)	Returns the maximum value of an array.
array_min	array_max(x)	Returns the minimum value of an array.
array_position	array_position(x, element)	Returns the subscript (starting from 1) of a specified element. If the specified element does not exist, return `0`.
array_remove	array_remove(x, element)	Removes a specified element from an array.
array_sort	array_sort(x)	Sorts elements in an array in ascending order. If there are null elements, the null elements will be placed at the end of the returned array.
array_union	array_union(x, y)	Returns the union of two arrays.
cardinality	cardinality(x)	Calculates the number of elements in an array.

concat	concat(x, y...)	Concatenates multiple arrays into one.
contains	contains(x, element)	Determines whether an array contains a specified element and returns `true` if the array contains the element.
element_at	element_at(x, y)	Returns the yth element of an array.
filter	filter(x, lambda_expression)	Filters elements in an array and returns only the elements that comply with the Lambda expression.
flatten	flatten(x)	Converts a two-dimensional array to a one-dimensional array.
reduce	reduce(x, lambda_expression)	Adds the elements in the array as defined by the Lambda expression and returns the result.
reverse	reverse(x)	Reverses the elements in an array.
sequence	sequence(x, y)	Returns an array of consecutive and increasing values within the specified starting value range. The increment interval is the default value 1.
	sequence(x, y, step)	Returns an array of consecutive and increasing values within the specified starting value range. The increment interval is `step`.
shuffle	shuffle(x)	Randomizes the elements in an array.
slice	slice(x, start, length)	Returns a subset of an array.
transform	transform(x, lambda_expression)	Applies a Lambda expression to each element of an array.
zip	zip(x, y)	Combines multiple arrays into a two-dimensional array (elements with the same subscript in each original array form a new array).
zip_with	zip_with(x, y, lambda_expression)	Merges two arrays into one as defined by a Lambda expression.

Subscript Operator []

The subscript operator ([]) is used to return the xth element in an array. It is equivalent to the `element_at` function.

Syntax

```
[x]
```

Parameter description

Parameter	Description
x	Array subscript, starting from 1. The parameter value is of the bigint type.

Returned value type

Return the data type of the specified element.

Sample

Return the second element of the `number` field value.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT cast(json_parse(array) as array(bigint)) [2]
```

Search and analysis result

```
23
```

array_agg

The `array_agg` function is used to return all values in `x` as an array.

Syntax

```
array_agg(x)
```

Parameter description

Parameter	Description
x	The parameter value can be of any data type.

Returned value type

Array

Sample

Return the values of the `status` field as an array.

Search and analysis statement

```
* | SELECT array_agg(status) AS array
```

Search and analysis result

```
[200,200,200,404,200,200]
```

array_distinct

The `array_distinct` function is used to delete duplicate elements from an array.

Syntax

```
array_distinct(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Array

Sample

Delete duplicate elements from the `array` field.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT array_distinct(cast(json_parse(array) as array(bigint)))
```

Search and analysis result

```
[12,23,26,48]
```

array_except

The `array_except` function is used to calculate the difference between two arrays.

Syntax

```
array_except(x, y)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
y	The parameter value is of the array type.

Returned value type

Array

Sample

Calculate the difference between arrays [1,2,3,4,5] and [1,3,5,7].

Search and analysis statement

```
* | SELECT array_except(array[1,2,3,4,5],array[1,3,5,7])
```

Search and analysis result

```
[2,4]
```

array_intersect

The `array_intersect` function is used to calculate the intersection between two arrays.

Syntax

```
array_intersect(x, y)
```

Parameter description

Parameter	Description

x	The parameter value is of the array type.
y	The parameter value is of the array type.

Returned value type

Array

Sample

Calculate the difference between arrays [1,2,3,4,5] and [1,3,5,7].
Search and analysis statement

```
* | SELECT array_intersect(array[1,2,3,4,5],array[1,3,5,7])
```

Search and analysis result

```
[1,3,5]
```

array_join

The `array_join` function is used to concatenate the elements in an array into a string using the specified delimiter.

Syntax

Concatenate the elements in an array into a string using the specified delimiter. If the array contains null elements, the null elements are ignored.

```
array_join(x, delimiter)
```

Concatenate the elements in an array into a string using the specified delimiter. If the array contains null elements, the null elements are replaced with `null_replacement` .

```
array_join(x, delimiter, null_replacement)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
delimiter	Connector, which can be a string.
null_replacement	String used to replace a null element.

Returned value type

Varchar

Sample

Use spaces to concatenate the elements in the [null,'China','sh'] array as a string, and replace the null element with `region`.

Search and analysis statement

```
* | SELECT array_join(array[null,'China','sh'],'/','region')
```

Search and analysis result

```
region/China/sh
```

array_max

The `array_max` function is used to get the maximum value of an array.

Syntax

```
array_max(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Same as the element data type of the parameter value.

Sample

Get the maximum value 48 from the [12,23,26,48,26] array.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT array_max(try_cast(json_parse(array) as array(bigint))) AS max_number
```

Search and analysis result

```
48
```

array_min

The `array_min` function is used to get the minimum value of an array.

Syntax

```
array_min(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Same as the element data type of the parameter value.

Sample

Get the minimum value 12 from the [12,23,26,48,26] array.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT array_min(try_cast(json_parse(array) as array(bigint))) AS min_number
```

Search and analysis result

```
12
```

array_position

The `array_position` function is used to get the subscript (starting from 1) of a specified element. If the specified element does not exist, return `0` .

Syntax

```
array_position(x, element)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
element	Element in an array.

Returned value type

Bigint

Sample

Return the subscript of 46 in the [23,46,35] array.

Search and analysis statement

```
* | SELECT array_position(array[23,46,35],46)
```

Search and analysis result

```
2
```

array_remove

The `array_remove` function is used to delete a specified element from an array.

Syntax

```
array_remove(x, element)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

element	Element in an array.

Returned value type

Array

Sample

Delete 23 from the [23,46,35] array.

Search and analysis statement

```
* | SELECT array_remove(array[23,46,35],23)
```

Search and analysis result

```
[46,35]
```

array_sort

The `array_sort` function is used to sort elements in an array in ascending order.

Syntax

```
array_sort(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Array

Sample

Sort elements in the ['b', 'd', null, 'c', 'a'] array in ascending order.

Search and analysis statement

```
* | SELECT array_sort(array['b','d',null,'c','a'])
```

Search and analysis result

```
["a","b","c","d",null]
```

array_union

The `array_union` function is used to calculate the union of two arrays.

Syntax

```
array_union(x, y)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
y	The parameter value is of the array type.

Returned value type

Array

Sample

Calculate the union of the arrays [1,2,3,4,5] and [1,3,5,7].

Search and analysis statement

```
* | SELECT array_union(array[1,2,3,4,5],array[1,3,5,7])
```

Search and analysis result

```
[1,2,3,4,5,7]
```

cardinality

The `cardinality` function is used to calculate the number of elements in an array.

Syntax

```
cardinality(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Bigint

Sample

Calculate the number of elements in the `number` field value.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT cardinality(cast(json_parse(array) as array(bigint)))
```

Search and analysis result

```
5
```

concat

The `concat` function is used to concatenate multiple arrays into one.

Syntax

```
concat(x, y...)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
y	The parameter value is of the array type.

Returned value type

Array

Sample

Concatenate the arrays ['red','blue'] and ['yellow','green'] into one array.

Search and analysis statement

```
* | SELECT concat(array['red','blue'],array['yellow','green'])
```

Search and analysis result

```
["red","blue","yellow","green"]
```

contains

The `contains` function is used to determine whether an array contains a specified element and return `true` if the array contains the element.

Syntax

```
contains(x, element)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
element	Element in an array.

Returned value type

Boolean

Sample

Determine whether the `array` field value contains 23.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT contains(cast(json_parse(array) as array(varchar)),'23')
```

Search and analysis result

```
TRUE
```

element_at

The `element_at` function is used to return the yth element in an array.

Syntax

```
element_at(x, y)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
element	Array subscript, starting from 1. The parameter value is of the bigint type.

Returned value type

Any data type

Sample

Return the second element of the `number` field value.

Field sample

```
array:[12,23,26,48,26]
```

Search and analysis statement

```
* | SELECT element_at(cast(json_parse(number) AS array(varchar)), 2)
```

Search and analysis result

```
23
```

filter

The `filter` function is used to filter elements in an array and return only the elements that comply with a specified Lambda expression

Syntax

```
filter(x, lambda_expression)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
lambda_expression	Lambda expression. For more information, see Lambda Function.

Returned value type

Array

Sample

Return elements greater than 0 in the [5,-6,null,7] array, where `x -> x > 0` is the Lambda expression.

Search and analysis statement

```
* | SELECT filter(array[5,-6,null,7],x -> x > 0)
```

Search and analysis result

```
[5,7]
```

flatten

The `flatten` function is used to convert a two-dimensional array to a one-dimensional array.

Syntax

```
flatten(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Array

Sample

Convert the two-dimensional array "array[1,2,3,4],array[4,3,2,1]" into a one-dimensional array.

Search and analysis statement

```
* | SELECT flatten(array[array[1,2,3,4],array[4,3,2,1]])
```

Search and analysis result

```
[1,2,3,4,4,3,2,1]
```

reduce

The `reduce` function is used to add the elements in an array as defined by the Lambda expression and return the result.

Syntax

```
reduce(x, lambda_expression)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
lambda_expression	Lambda expression. For more information, see Lambda Function.

Returned value type

Bigint

Sample

Return the sum of the elements in array [5, 20, 50].

Search and analysis statement

```
* | SELECT reduce(array[5,20,50],0,(s, x) -> s + x, s -> s)
```

Search and analysis result

```
75
```

reverse

The `reverse` function is used to reverse the elements in an array.

Syntax

```
reverse(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Array

Sample

Reverse the elements in array [1,2,3,4,5].

Search and analysis statement

```
* | SELECT reverse(array[1,2,3,4,5])
```

Search and analysis result

```
[5,4,3,2,1]
```

sequence

The `sequence` function is used to return an array of consecutive and increasing values within the specified starting value range.

Syntax

The increment interval is the default value `1` .

```
sequence(x, y)
```

The increment interval is custom.

```
sequence(x, y, step)
```

Parameter description

Parameter	Description
x	The parameter value is of the bigint or timestamp type (UNIX timestamp or date and time expression).
y	The parameter value is of the bigint or timestamp type (UNIX timestamp or date and time expression).
step	Value interval.If the parameter value is a date and time expression, the format of `step` is as follows: interval 'n' year to month: the interval is `n` years. interval 'n' day to second: the interval is `n` days.

Returned value type

Array

Sample

Example 1. Return even numbers between 0 and 10.

Search and analysis statement

```
* | SELECT sequence(0,10,2)
```

Search and analysis result

```
[0,2,4,6,8,10]
```

Example 2. Return dates between 2017-10-23 and 2021-08-12 at an interval of one year.

Search and analysis statement

```
* | SELECT sequence(from_unixtime(1508737026),from_unixtime(1628734085),interval '1
```

Search and analysis result

```
["2017-10-23 05:37:06.0","2018-10-23 05:37:06.0","2019-10-23 05:37:06.0","2020-10-2
```

Example 3. Return UNIX timestamps between 1628733298 and 1628734085 at an interval of 60 seconds.

Search and analysis statement

```
* | SELECT sequence(1628733298,1628734085,60)
```

Search and analysis result

```
[1628733298,1628733358,1628733418,1628733478,1628733538,1628733598,1628733658,16287
```

shuffle

shuffle

The `shuffle` function is used to randomize the elements in an array.

Syntax

```
shuffle(x)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.

Returned value type

Array

Sample

Randomize the elements in the [1,2,3,4,5] array.

Search and analysis statement

```
* | SELECT shuffle(array[1,2,3,4,5])
```

Search and analysis result

```
[5,2,4,1,3]
```

slice

The `slice` function is used to return a subset of an array.

Syntax

```
slice(x, start, length)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
start	Index start position. If `start` is negative, start from the end. If `start` is positive, start from the beginning.
length	Number of elements in a subset.

Returned value type

Array

Sample

Return a subset of the [1,2,4,5,6,7,7] array, starting from the third element and consisting of two elements.

Search and analysis statement

```
* | SELECT slice(array[1,2,4,5,6,7,7],3,2)
```

Search and analysis result

```
[4,5]
```

transform

The `transform` function is used to apply a Lambda expression to each element of an array.

Syntax

```
transform(x, lambda_expression)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
lambda_expression	Lambda expression. For more information, see Lambda Function.

Returned value type

Array

Sample

Increment each element in the [5,6] array by 1 and return.

Search and analysis statement

```
* | SELECT transform(array[5,6],x -> x + 1)
```

Search and analysis result

```
[6,7]
```

zip

The `zip` function is used to combine multiple arrays into a two-dimensional array, and elements with the same subscript in each original array form a new array.

Syntax

```
zip(x, y)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
y	The parameter value is of the array type.

Returned value type

Array

Sample

Combine the arrays [1,2] and [3,4] into a two-dimensional array.

Search and analysis statement

```
* | SELECT zip(array[1,2], array[3,4])
```

Search and analysis result

```
["{1, 3}","{2, 4}"]
```

zip_with

The `zip_with` function is used to merge two arrays into one as defined by a Lambda expression.

Syntax

```
zip_with(x, y, lambda_expression)
```

Parameter description

Parameter	Description
x	The parameter value is of the array type.
y	The parameter value is of the array type.
lambda_expression	Lambda expression. For more information, see Lambda Function.

Returned value type

Array

Sample

Use Lambda expression `(x, y) -> x + y` to add the elements in arrays [1,2] and [3,4], respectively, and return the sum results as an array.

Search and analysis statement

```
* | SELECT zip_with(array[1,2], array[3,4],(x,y) -> x + y)
```

Search and analysis result

```
[4,6]
```

Interval-Valued Comparison and Periodicity-Valued Comparison Functions

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of interval-valued comparison and periodicity-valued comparison functions.
CLS supports the interval-valued comparison and periodicity-valued comparison functions listed in the table below.

Function	Syntax	Description
compare	compare(x,n)	Compares the calculation result of the current time period with the calculation result of a time period n seconds before.
	compare(x,n1,n2,n3...)	Compares the calculation result of the current time period with the calculation results of time periods n1, n2, and n3 seconds before.

compare

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.

Syntax

Compare the calculation result of the current time period with the calculation result of a time period n seconds before:

```
compare (x, n)
```

Compare the calculation result of the current time period with the calculation results of time periods n1, n2, and n3 seconds before:

```
compare (x, n1, n2, n3...)
```

Parameter description

Parameter	Description
x	The parameter value is of the double or long type.

n	Time window. Unit: seconds. Example: 3600 (1 hour), 86400 (1 day), 604800 (1 week), or 31622400 (1 year).

Return value type

JSON array in the following format: [the current calculation result, the calculation result n seconds before, the ratio of the current calculation result to the calculation result of n seconds before].

Example

Example 1. 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 for query and analysis to 1 hour and execute the following query and analysis statement, where `86400` indicates the current time minus 86400 seconds (1 day).

Query and analysis statement

```
* | SELECT compare(PV, 86400) FROM (SELECT count(*) AS PV)
```

Query and analysis result

```
[1860,1656,1.1231884057971016]
```

1860: indicates the PVs of the current 1 hour.

1656: indicates the PVs of the same time period the day before.

1.1231884057971016: indicates the ratio of the PVs of the current hour to the PVs of the same time period the day before.

To display the query and analysis result in multiple columns, execute the following query and analysis statement:

```
* |
SELECT compare[1] AS today, compare[2] AS yesterday, compare[3] AS ratio
FROM (
    SELECT compare(PV, 86400) AS compare
    FROM (
        SELECT COUNT(*) AS PV
    )
)
```

Example 2. Calculate the trend of the PVs of every 5 minutes today to the trend of the PVs of the same time period the day before.

Set the time range for query and analysis to Today and execute the following query statement. `86400` indicates the current time minus 86400 seconds (1 day), and `current_date` indicates the date of the current day.

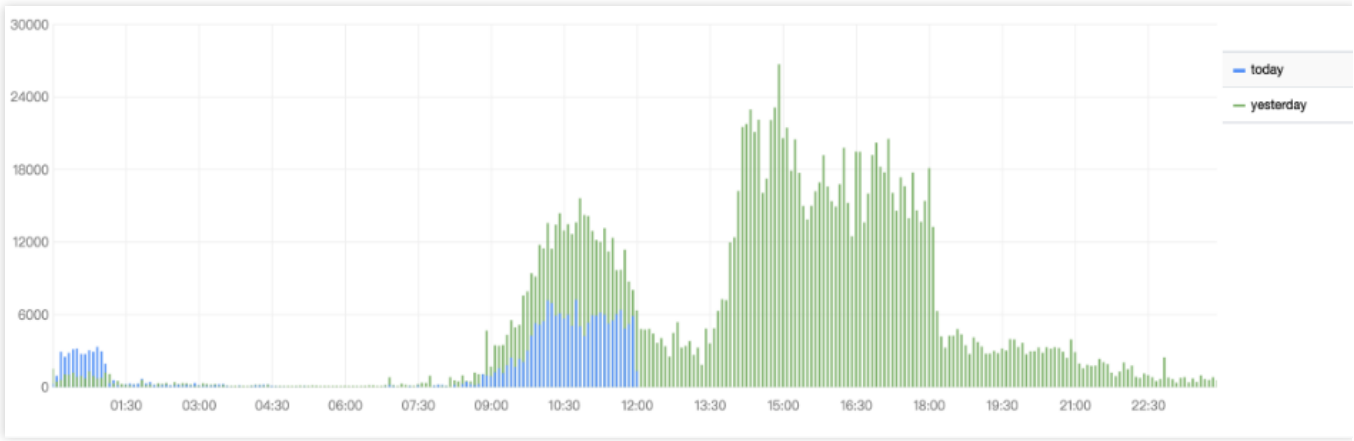
Note:

The time range cannot span days. The reason is as follows: in this example, the log time is truncated to `%H:%i:%s`, which contains only the hour, minute, and second but does not contain the date. If the time range spans days, data statistics errors will occur.

Query and analysis statement

```
* |
select concat(cast(current_date as varchar),' ',time) as time,compare[1] as today,c
    select time,compare(pv, 86400) as compare from (
        select time_series(__TIMESTAMP__, '5m', '%H:%i:%s', '0')  as time, count(*)
    limit 1000)
order by time limit 1000
```

Query and analysis result



JSON Functions

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of JSON functions.

Function	Syntax	Description
json_array_contains	json_array_contains(x, value)	Determines whether a JSON array contains a given value.
json_array_get	json_array_get(x, index)	Returns the element with the specified index in a given JSON array.
json_array_length	json_array_length(x)	Returns the number of elements in a given JSON array. If `x` is not a JSON array, `null` will be returned.
json_extract	json_extract(x, json_path)	Extracts a set of JSON values (array or object) from a JSON object or array.
json_extract_scalar	json_extract_scalar(x, json_path)	Extracts a set of scalar values (strings, integers, or Boolean values) from a JSON object or array. Similar to the `json_extract` function.
json_format	json_format(x)	Converts a JSON value into a string value.
json_parse	json_parse(x)	Converts a string value into a JSON value.
json_size	json_size(x, json_path)	Calculates the number of elements in a JSON object or array.

json_array_contains

The `json_array_contains` function is used to determine whether a JSON array contains a specified value.

Syntax

```
json_array_contains(x, value)
```

Parameter description

Parameter	Description
x	The parameter value is a JSON array.
value	Value.

Return value type

Boolean

Example

Determine whether the JSON array [1, 2, 3] contains 2.

Search and analysis statement

```
* | SELECT json_array_contains('[1, 2, 3]', 2)
```

Search and analysis result

```
TRUE
```

json_array_get

The `json_array_get` function is used to get the element with a specified index in a JSON array.

Syntax

```
json_array_get(x, index)
```

Parameter description

Parameter	Description
x	The parameter value is a JSON array.
index	JSON subscript (index), starting from 0.

Return value type

Varchar

Example

Return the element with index 1 in the JSON array ["a", [3, 9], "c"].

Search and analysis statement

```
* | SELECT json_array_get('["a", [3, 9], "c"]', 1)
```

Search and analysis result

```
[3,9]
```

json_array_length

The `json_array_length` function is used to calculate the number of elements in a JSON array. If `x` is not a JSON array, `null` will be returned.

Syntax

```
json_array_length(x)
```

Parameter description

Parameter	Description
x	The parameter value is a JSON array.

Return value type

Bigint

Example

Example 1. Calculate the number of JSON elements in the **apple.message** field

```
apple.message:[{"traceName":"StoreMonitor"},
{"topicName":"persistent://apache/pulsar/test-partition-17"},
{"producerName":"pulsar-mini-338-36"},{"localAddr":"pulsar://pulsar-mini-
broker-5.pulsar-mini-broker.pulsar.svc.cluster.local:6650"},{"sequenceId":826},
{"storeTime":1635905306062},{"messageId":"19422-24519"},{"status":"SUCCESS"}]
```

Search and analysis statement

```
* | SELECT json_array_length(apple.message)
```

Search and analysis result

```
8
```

json_extract

The `json_extract` function is used to extract a set of JSON values (array or object) from a JSON object or array.

Syntax

```
json_extract(x, json_path)
```

Parameter description

Parameter	Description
x	The parameter value is a JSON object or array.
json_path	JSONPath, such as `$.store.book[0].title`. Note: JSON syntax requiring array element traversal is not supported, such as the following: `$.store.book[*].author`, `$..book[(@.length-1)]`, `$..book[?(@.price<10)]`.

Return value type

JSON string

Example

Get the value of **epochSecond** in the **apple.instant** field.

Field sample

```
apple.instant:{"epochSecond":1635905306,"nanoOfSecond":63001000}
```

Search and analysis statement

```
* | SELECT json_extract(apple.instant, '$.epochSecond')
```

Search and analysis result

```
1635905306
```

json_extract_scalar

The `json_extract_scalar` function is used to extract a set of scalar values (strings, integers, or Boolean values) from a JSON object or array.

Syntax

```
json_extract_scalar(x, json_path)
```

Parameter description

Parameter	Description
x	The parameter value is a JSON array.
json_path	JSONPath, such as `$.store.book[0].title`. Note: JSON syntax requiring array element traversal is not supported, such as the following: `$.store.book[*].author`, `$..book[(@.length-1)]`, `$..book[?(@.price<10)]`.

Return value type

Varchar

Example

Get the value of **epochSecond** from the **apple.instant** field and convert the value into a bigint value for summation.

Field sample

```
apple.instant:{"epochSecond":1635905306,"nanoOfSecond":63001000}
```

Search and analysis statement

```
* | SELECT sum(cast(json_extract_scalar(apple.instant,'$.epochSecond') AS bigint) )
```

Search and analysis result

```
1635905306
```

json_format

The `json_format` function is used to convert a JSON value into a string value.

Syntax

```
json_format(x)
```

Parameter description

Parameter	Description
x	The parameter value is of JSON type.

Return value type

Varchar

Example

Convert the JSON array [1,2,3] into a string [1, 2, 3].

Search and analysis statement

```
* | SELECT json_format(json_parse('[1, 2, 3]'))
```

Search and analysis result

```
[1, 2, 3]
```

json_parse

The `json_parse` function is used to convert a string value into a JSON value and determine whether it complies with the JSON format.

Syntax

```
json_parse(x)
```

Parameter description

Parameter	Description

x	The parameter value is a string.

Return value type

JSON

Example

Convert the JSON array [1,2,3] into a string [1, 2, 3].

Search and analysis statement

```
* | SELECT json_parse('[1, 2, 3]')
```

Search and analysis result

```
[1, 2, 3]
```

json_size

The `json_size` function is used to calculate the number of elements in a JSON object or array.

Syntax

```
json_size(x, json_path)
```

Parameter description

Parameter	Description
x	The parameter value is a JSON object or array.
json_path	JSON path, in the format of `$.store.book[0].title` .

Return value type

Bigint

Example

Convert the JSON array [1,2,3] into a string [1, 2, 3].

Search and analysis statement

```
* | SELECT json_size(json_parse('[1, 2, 3]'),'$')
```

Search and analysis result

```
3
```

Window Functions

Last updated：2024-01-22 10:52:48

This document introduces the basic syntax and examples of window functions.
A window function calculates the data of multiple rows and returns the calculation result. Unlike GROUP BY, it only appends the calculation result to each row of data and does not merge the rows.

Syntax

```
window_function (expression) OVER (
   [ PARTITION BY part_key ]
   [ ORDER BY order_key ]
   [ { ROWS | RANGE } BETWEEN frame_start AND frame_end ] )
```

Parameters

Parameter	Description
window_function	Specifies the window value calculation method. Aggregate functions, ranking functions and value functions are supported.
PARTITION BY	Specifies how a window is partitioned.
ORDER BY	Specifies how the rows in each window partition are ordered.
{ ROWS | RANGE } BETWEEN frame_start AND frame_end	Window frames, that is, the data range (rows) used when calculating the value of each row within the window partition. If not specified, it represents all rows within the window partition. Example: rows between current row and 1 following: The current row and the subsequent row rows between 1 preceding and current row: The current row and the preceding row rows between 1 preceding and 1 following: From the preceding row to the subsequent row (a total of three rows) rows between current row and unbounded following: The current row and all subsequent rows rows between unbounded preceding and current row: The current row and all preceding rows

General Aggregate Functions

All general aggregate functions are supported, such as sum() and avg(), can be used to calculate the statistics of each row of data in window frames.

Ranking Functions

Ranking functions cannot use window frames.

Function	Description
rank()	Returns the rank of each row in a window partition. Rows that have the same field value are assigned the same rank, and therefore ranks may not be consecutive. For example, if two rows have the same rank of 1, the rank of the next row is 3.
dense_rank()	Similar to rank(). The difference is that the ranks in this function are consecutive. For example, if two rows have the same rank of 1, the rank of the next row is 2.
cume_dist()	Returns the cumulative distribution of each value in a window partition, that is, the proportions of rows whose field values are less than or equal to the current field value to the total number of rows in the window partition.
ntile(n)	Divides the rows for a window partition into `n` groups. If the number of rows in the partition is not divided evenly into `n` groups, the remaining values are distributed one per group, starting with the first group. For example, if there are 6 rows of data, and they need to be divided into 4 groups, the numbers of each row of data are: 1, 1, 2, 2, 3, 4.
percent_rank()	Calculates the percentage ranking of each row in a window partition. The calculation formula is: (r - 1) / (n - 1), where `r` is the rank value obtained via rank() and `n` is the total number of rows in the window partition.
row_number()	Calculates the rank of each row (after ranking based on ranking rules) in a window partition. The ranks are unique and start from 1.

Value Functions

Function	Description
first_value(key)	Returns the first value of `key` of the window partition.
last_value(key)	Returns the last value of `key` of the window partition.

nth_value(key, offset)	Returns the value of `key` in the row at the specified offset of the window partition. Offsets start from 1 and cannot be 0 or negative. If `offset` is `null` or exceeds the number of rows in the window partition, `null` is returned.
lead(key[, offset[, default_value]])	Returns the value of `key` in the row that is at the specified offset after the current row of the window partition. Offsets start from 0, indicating the current row. `offset` is 1 by default. If `offset` is `null`, `null` is returned. If the offset row exceeds the window partition, `default_value` is returned. If `default_value` is not specified, `null` is returned. When using this function, you must specify the ranking rule (ORDER BY) within the window partition and cannot use window frames.
lag(key[, offset[, default_value]])	Similar to `lead(key[, offset[, default_value]])`. The only difference is that this function returns the value at `offset` rows before the current row.

Example

Example 1. Query the 5 slowest requests and their IDs of each API in the last hour

Select the last hour as the time range and run the following query and analysis statement, where `action` indicates the API name, `timeCost` indicates the API response time, and `seqId` indicates the request ID.

Query and analysis statement

```
* | select * from (select action,timeCost,seqId,rank() over (partition by action or
```

Query and analysis result

action	timeCost	seqId	ranking
ModifyXXX	151	d75427b3-c562-6d7a-354f-469963aab689	1
ModifyXXX	104	add0d353-1099-2c73-e9c9-19ad02480474	2
CreateXXX	1254	c7d591f0-2da6-292c-8abf-98a0716ff8c6	1
CreateXXX	970	d920cf7a-7e7b-524b-68e9-a957c454c328	2
CreateXXX	812	16357f6d-33b3-83ea-0ae3-b1a2233d4858	3
CreateXXX	795	0efdab5e-af5f-4a4a-0618-7961420d17a1	4
CreateXXX	724	fb0481f2-dcfc-9500-cb44-a139b774aceb	5
DescribeXXX	55242	4129dcda-46d7-9213-510e-f58cba29daf5	1
DescribeXXX	17413	e36cdeb0-cbc5-ce2b-dec7-f485818ab6c7	2

DescribeXXX	10171	cd6228f7-4644-ba45-f539-0fce7b09455b	3
DescribeXXX	9475	48b6f6e3-6d08-5a31-cd68-89006a346497	4
DescribeXXX	9337	940b5398-e2ae-9141-801b-b7f0ca548875	5

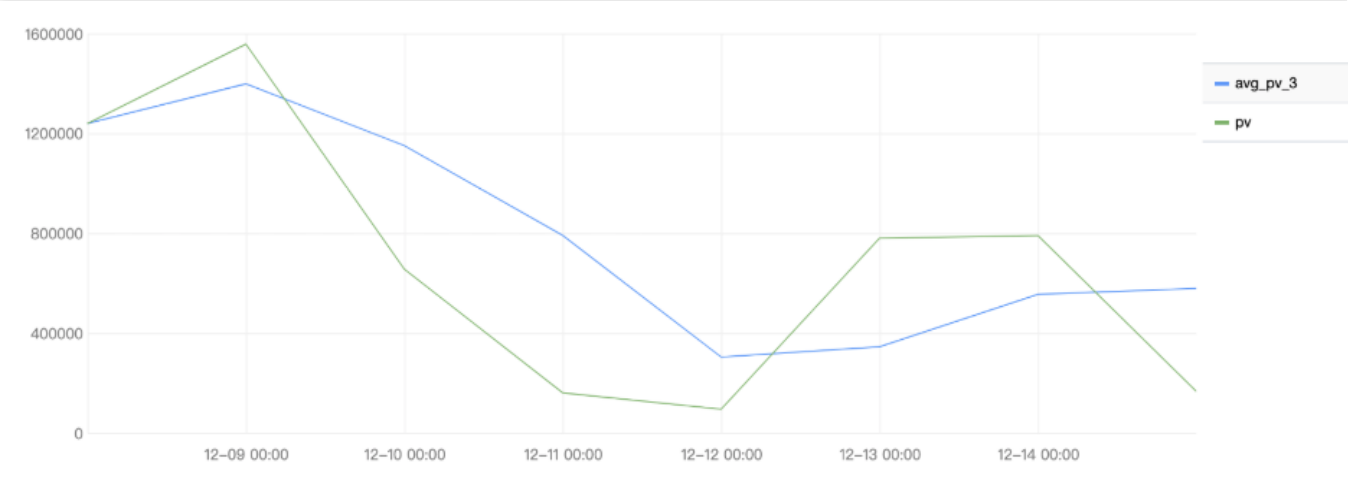
Example 2. Query the 3-day moving average trend of the application throughput

Select the last 7 days as the query and analysis time range and run the following query and analysis statement, where `pv` indicates the daily application throughput and `avg_pv_3` indicates the application throughput after 3-day moving average.

Query and analysis statement

```
* | select avg(pv) over(order by analytic_time rows between 2 preceding and
current row) as avg_pv_3,pv,analytic_time from (select histogram(
cast(__TIMESTAMP__ as timestamp),interval 1 day) as analytic_time, count(*) as
pv group by analytic_time order by analytic_time)
```

Query and analysis result



Sampling Analysis

Last updated：2024-01-20 17:25:15

Overview

When using the statistical analysis feature as described in Overview and Syntax Rules, if the number of raw logs is very high or the search statement (SQL) is complex, the analysis may be slow or even time out. In this case, you can use the sampling analysis feature to sample raw logs first and then perform statistical analysis.
CLS samples raw logs randomly. According to relevant statistical principles, under a reasonable sample rate and a large number of samples, the statistical results after sampling are very close to the accurate full statistical results, which can satisfy the needs of big data analysis in most cases.

Sampling Method and Accuracy

Sampling method

Statistical analysis adopts the "precise analysis" method by default, i.e., no sampling. If sampling is required, you can choose either of the following two sampling methods:
Automatic sampling: The sample rate is automatically determined based on the number of raw log entries, so that the number of samples is about 1 million, and statistical analysis is performed on such samples. For example, if raw logs contain 10 billion entries, and automatic sampling is used to get 1 million samples, then the sample rate is 1:10,000.
Fixed-rate sampling: You can manually set the sample rate to get samples from raw logs accordingly. For example, if the raw logs contain 10 billion entries and are sampled at the rate of 1:100,000, then the number of samples is 100,000. The sample rate can range from 1:1,000,000 to 1 (no sampling).
Regardless of the sampling method used, the final actual sample rate can be viewed in the statistical analysis results.

Estimating the true value

Sampling analysis results can largely reflect the true value. Based on the value calculation method, the true value can be estimated in the following ways:
For calculation methods related to averages such as `avg` and `geometric_mean`, the sample statistical result can directly represent the true value.
For calculation methods related to values such as `count(*)`, `sum`, and `count_if`, the sample statistical result divided by the sample rate can represent the true value. For example, if the sample rate for pv(`count(*)`) is 1:10,000 and the statistical result is 232, then the true value is about 232 / (1:10000) = 2320000.

Examples

To sample at 1:10,000, the search statement is:

```
* | select count(*)/(1.0/10000) as pv,avg(response_time) as response_time_avg
```

`pv` calculation: `count(*)` is to calculate the number of log entries and involves sum calculation. You can divide the sample statistical result by the sample rate to get the true value. Here, `(1.0/10000)` is the sample rate, and the calculated `pv` is the estimated true value of `pv` .

`response_time_avg` calculation: `avg(response_time)` is to calculate the average of `response_time` and involves average calculation. The result can be directly used as the estimated true value of `response_time_avg` .

Sampling analysis accuracy

In statistics, the confidence level and confidence interval are used to measure the accuracy of the sampling results. The former represents the reliability of the sampling results, and the latter represents the interval of the true value at the specified confidence level. For example, if the confidence level of a certain sample statistical result is 95%, and the confidence interval is [212,478], then the true value has a 95% probability of being between 212 and 478.

The higher the confidence level, the larger the confidence interval. The commonly used confidence level is 95% (in statistics, a probability below 5% is generally considered as a small probability). The confidence interval is calculated as follows:

Here, $\bar{x}$ is the sample average, s is the sample standard deviation, and n is the number of samples.

In case of statistical analysis using automatic sampling:

```
* | select avg(response_time) as x,count(response_time) as
n,stddev_samp(response_time) as s
```

The statistical analysis result of the samples is:

x: The sample average, which is 544.4656932942215.

n: The number of samples, which is 995,097.

s: The sample standard deviation, which is 1,382.618439585749.

Then, if the confidence level is 95%, the true value of `avg(response_time)` is within the confidence interval of [541.75,547.18]. The `avg(response_time)` value obtained by accurate statistical calculation in this case is 545.16, which is within the confidence interval.

Note:

The above is a strict sampling accuracy measurement method. In actual use, when there are more than 1,000 samples, the results obtained by sampling generally are highly accurate.

During statistical analysis by dimension (i.e., `group by`), as the samples will be divided into multiple groups by the specified dimension, and the statistical value will be calculated in each group, the number of samples in a single group will be lower than the total number of samples. This will result in less accurate statistics for groups with a small sample

size.

In case of sampling statistical analysis:

```
* | select avg(response_time) as response_time,count(*) as sampleCount,url
group by url order by count(*) desc
```

The statistical analysis result of the samples is:

url	response_time	sampleCount
/user	45.23	7845
/user/list	78.45	6574
/user/login	45.85	5235
/user/logout	45.48	1245
/book/new	125.78	987
/book/list	17.23	658
/book/col	10.21	23
/order	12.13	2

Here, the number of samples of the two URLS `/book/col` and `/order` is too low, so the statistical results are less accurate. To get more accurate statistical results for these two URLs, you can increase the overall sample rate or perform statistical analysis on them separately, for example:

```
url:"/book/col" OR url:"/order" | select avg(response_time) as
response_time,count(*) as sampleCount,url group by url order by count(*) desc
```

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to enter the log topic search and analysis page.
3. At the top of the page, select the log topic for search and analysis.
4. After entering the search statement (with SQL included), set whether sampling analysis is required and select the corresponding sample rate below the **Search and Analysis** button on the right.
5. Click **Search and Analysis**.

After the search and analysis is completed, the sample rate used for this analysis will be displayed at the top of the chart.

Configuring Indexes

Last updated：2025-03-21 16:06:48

Overview

Index configuration is a necessary condition for using CLS for log search and analysis; that is, log search and analysis can be performed only after index is enabled. In addition, different index rules can lead to different search and analysis results. This document describes how to configure an index and how it works.

The core of index configuration is to segment raw logs so that logs can be quickly and conveniently searched based on specific search conditions. In addition, you can enable statistics for specific fields in index configuration to facilitate statistical analysis of logs using SQL. CLS provides the following three types of indexes:

Category	Description	Configuration Method
Full-Text Index	A raw log is split into multiple segments, and indexes are created based on the segments. You can query logs based on keywords (full-text search). For example, entering error means to search for logs that contain the keyword error.	Console: Enable full-text index on the index configuration page.
Key-Value Index	A raw log is split into multiple segments based on a field (key:value), and indexes are created based on the segments. You can query logs based on key-value (key-value search). For example, entering level:error means to search for logs with a level field whose value contains error.	Console: On the index configuration page, enable key-value index and enter the field name (`key`), such as level.
Metadata Index	A metadata index is also a key-value index, but the field name is prefixed with __TAG__. Metadata indexes are usually used to classify logs.For example, entering __TAG__.region:"ap-beijing" means to search for logs with a region field whose value is ap-beijing.	Console: On the index configuration page, enable key-value index and enter the metadata field name (`key`), such as __TAG__.region.

Note:

Index configuration will incur any index traffic and index storage fees. For more billing information, see Billing Overview. For how to save product use costs, see Saving Product Use Costs.

Log data collected cannot be searched when index is disabled. It takes about one minute for the log search and analysis feature to become available after index is enabled.

Only fields for which **Enable Statistics** is toggled on in the key-value index configuration support SQL statistical analysis.

Index rule changes (including adding, modifying, and deleting fields, and adjusting delimiter configuration) take effect only for newly written logs. Existing data is not updated.

Prerequisites

For log collection using LogListener, if the extraction mode in the collection configuration is set to full text in a single line or full text in multi lines, the original log is stored under the `__CONTENT__` field and supports only full-text index configuration. If you need to configure key-value indexes for some content in the log or enable statistics, you need to perform log structuring and use log extraction modes other than full text in a single line or full text in multi lines.

Full-Text Index

A raw log is split into multiple segments, and indexes are created based on the segments. You can query logs based on keywords (full-text search).

Configuration Item	Description
Full-Text Delimiter	A set of characters that split the raw log into segments. Only English symbols are supported. Default delimiters in the console are @&?|#()='",;:<>[]{}/ \\n\\t\\r. **Note:** If a segment is too long, an index will be created only for the first 10,000 characters, and the excessive part cannot be found. However, the complete log will be stored.
Case Sensitivity	Specifies whether log search is case-sensitive.For example, if a log is Error and log search is case-sensitive, the log cannot be matched by error.
Allow Chinese Characters	This feature can be enabled when logs contain Chinese characters and the Chinese characters need to be searched. For example, if the original text of a log is in Chinese, and this feature is disabled, you cannot query the log by using a Chinese keyword contained in the original text. The query can be successful only if you use the exact raw log text to query the log. However, if you enable this feature, you can query the log by using a Chinese keyword contained in the raw log text.

For example, a complete log is as shown below:

```
10.20.20.10;[2018-07-16 13:12:57];GET /online/sample HTTP/1.1;200
```

If you use the Separator Format to extract log fields, the structured log uploaded to CLS will be as follows:

```
IP: 10.20.20.10
request: GET /online/sample HTTP/1.1
status: 200
time: [2018-07-16 13:12:57]
```

If the full-text delimiter is `@&()='",;:<>[]{}/ \\n\\t\\r` (including space), all field values in the raw log will be segmented into the following keywords (each line denotes a keyword):

```
10.20.20.10
GET
online
sample
HTTP
1.1
200
2018-07-16
13
12
57
```

Under the above index configuration, the following results are obtained using the following search conditions:

Search condition A:

```
\\/online\\/login
```

`\\` is used to escape the `/` symbol (this symbol is a reserved symbol of the search syntax and therefore needs to be escaped).

The escaped `/` symbol is a delimiter, so the actual search condition is `online OR login`. A log containing `online` **or** `login` is considered to meet the search condition.

The example log provided above **meets** the search condition.

Search condition B:

```
"/online/login"
```

Being enclosed by double quotation marks, the `/` symbol does not need to be escaped.

The content in the double quotation marks is also divided into two words. However, the double quotation marks indicate that only a log that **contains both the two words in the exact order** as that in the search condition is considered to meet the search condition.

The example log provided above does not contain `login` and therefore **does not meet** the search condition.

Search condition C:

```
"/online/sample"
```

The example log provided above contains both `online` and `sample` in the exact order as that in the search condition and therefore is considered to **meet** the search condition.

Key-Value Index

A raw log is split into multiple segments based on a field (key:value), and indexes are created based on the segments. You can query logs based on key-value (key-value search). To perform a key-value search, you must specify the field name in the syntax format of `key:value`, for example, `status:200`. If no field name is specified, a full-text search will be performed.

To meet basic log search requirements, CLS automatically creates key-value indexes for some built-in reserved fields, which does not incur index traffic fee. Details are as follows:

Built-in Reserved Field	Description
`__FILENAME__`	Filename for log collection, which can be used to search for logs in a specified file. For example, you can use `__FILENAME__:/"var/log/access.log"` to search for logs from the `/var/log/access.log` file.
`__SOURCE__`	Source IP for log collection, which can be used to search for logs of a specified server. For example, you can use `__SOURCE__:192.168.10.10` to search for the logs of the server whose IP is `192.168.10.10`.
`__HOSTNAME__`	The server name of the log, which can be used to search for logs of a specified server. Only LogListener 2.7.4 or later can collect this field.
`__TIMESTAMP__`	Log timestamp (UNIX timestamp in milliseconds). When a log is searched by time range, the system automatically searches for the log by this time and displays the time as the log time on the console.
`__PKG_LOGID__`	Log ID in a log group. This ID is used for context search. You are not recommended to use this ID alone.

Configuration Item	Description	Remarks
Field Name	Field name in Collection Overview **Note:** You can add up to 300 fields for a key-value index of a log topic.	-
Data Type	Data type of the field. There are three types: `text`, `long`, and `double`. The `text` type supports fuzzy search by wildcard, while the `long` and `double` types support range search. **Note:** 1. Fields of the `long` type support a data range of -1E15 to 1E15. Data out of the range may lose certain decimal places or not be matched. In the case of index configuration for a super long numeric field, we recommend that you:	long - Integer type (Int 64) double - Floating point (64 bit) double text - String

	store the field as the `text` type if you don't need to search for it by comparing it with the numeric range. store the field as the `double` type if you need to do so, which may lose certain decimal places. 2. Fields of the `double` type support a data range of -1.79E+308 to +1.79E+308. If the number of code characters of the floating-point number exceeds 64, decimal places will be lost.	
Delimiter	A set of characters that split the field value into segments. Only English symbols are supported. **Note:** If a segment is too long, an index will be created only for the first 10,000 characters, and the excessive part cannot be found. However, the complete log will be stored.	Default delimiters in the console: `@&?|#()='",;:<>[]{}/ \\n\\t\\r`
Allow Chinese Characters	This feature can be enabled when fields contain Chinese characters and the Chinese characters need to be searched. For example, if the original text of a log is in Chinese, and this feature is disabled, you cannot query the log by using a Chinese keyword contained in the original text. The query can be successful only if you use the exact raw log text to query the log. However, if you enable this feature, you can query the log by using a Chinese keyword contained in the raw log text.	-
Enable Statistics	After it is toggled on, SQL statistical analysis can be performed on the field, such as `group by ${key}` and `sum(${key})`. **Note:** If it is toggled on for a field of the `text` type and the value is too long, only the first 32,766 characters will be included in the statistical calculation (SQL). If the field contains Chinese characters, the log will be lost if the value contains more than 32,766 characters. We recommend that you toggle the feature off in this case.	This feature is part of the key-value index feature and therefore is not billed separately.
Case Sensitivity	Specifies whether log search is case-sensitive.For example, if a log is level:Error and log search is case-sensitive, the log cannot be matched by `level:error`.	-

For example, a complete log is as shown below:

```
10.20.20.10;[2018-07-16 13:12:57];GET /online/sample HTTP/1.1;200
```

If you use the Separator Format to extract log fields, the structured log uploaded to CLS will be as follows:

```
IP: 10.20.20.10
request: GET /online/sample HTTP/1.1
status: 200
time: [2018-07-16 13:12:57]
```

Assume that the key-value index configuration is as follows:

Field Name	Field Type	Delimiter	Allow Chinese Characters	Enable Statistics
IP	text	`@&()='",;:<>[]{}/ \\n\\t\\r`	No	Yes
request	text	`@&()='",;:<>[]{}/ \\n\\t\\r`	No	Yes
status	long	None	No	Yes
time	text	`@&()='",;:<>[]{}/ \\n\\t\\r`	No	Yes

Under the above index configuration, the following results are obtained using the following search conditions:

Search condition A:

```
request:\\/online\\/login
```

`\\` is used to escape the `/` symbol (this symbol is a reserved symbol of the search syntax and therefore needs to be escaped).

The escaped `/` symbol is a delimiter, so the actual search condition is `online OR login`. A log containing `online` **or** `login` is considered to meet the search condition.

The example log provided above **meets** the search condition.

Search condition B:

```
request:"/online/login"
```

Being enclosed by double quotation marks, the `/` symbol does not need to be escaped.

The content in the double quotation marks is also divided into two words. However, the double quotation marks indicate that only a log that **contains both the two words in the exact order** as that in the search condition is considered to meet the search condition.

The example log provided above does not contain `login` and therefore **does not meet** the search condition.

Search condition C:

```
request:"/online/sample"
```

The example log provided above contains both `online` and `sample` in the exact order as that in the search condition and therefore is considered to **meet** the search condition.

For fields with the statistics feature enabled, you can also use SQL to perform statistical analysis on logs.

Search and analysis statement A:

```
request:"/online/login" | select count(*) as logCounts
```

Count the number of logs whose value of `request` is "/online/login".

Search and analysis statement B:

```
* | select count() as logCounts,request group by request order by count() desc li
```

Get the top 10 requests with the largest number of log entries.

Automatic Configuration

After **Automatic Configuration** is enabled, fields in a log will be automatically added to the key-value index field list, including newly added fields in the log.

If a field is added, the field type will be automatically determined based on the field value. The field type will be set to text, long, or double, and the following configuration will be used. If the following configuration does not meet usage requirements (for example, if the field type is incorrect), it can be manually modified on the added field.

Field Type	Delimiter	Chinese Characters	Statistics
text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled
long	Not involved	Not involved	Enabled
double	Not involved	Not involved	Enabled

Note:

Log field names only support letters, digits, underscores, and -./@, and cannot start with an underscore. Fields that do not meet this rule will not be added to a key-value index field list.

If a log field is a json string, the json will not be parsed by default. If you need to add subfields in the json format to an key-value index, manually add any of the subfields. Subsequently, fields at the same level as the subfields will be automatically added to a key-value index field list.

A sample raw log that has 3 fields is shown as follows:

```
key1:textValue
key2:123
key3:{"ip":"123.123.123.132","url":"class/132.html","detail":{"status_code":"500","
```

The following table shows an automatically added key-value index field list.

Field Name	Field Type	Delimiter	Chinese Characters	Statistics
key1	text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled
key2	long	Not involved	Not involved	Enabled
key3	text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled

If the name of a field is manually modified from key3 to key3.ip, the fields url and detail at the same level as ip will be automatically added to the key-value index field list. The following table shows the final list.

Field Name	Field Type	Delimiter	Chinese Characters	Statistics
key1	text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled
key2	long	Not involved	Not involved	Enabled
key3.ip	text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled
key3.url	text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled
key3.detail	text	@&?|#()='",;:<>[]{}/ \\n\\t\\r\\\\	Included	Enabled

Metadata Index

When a log is uploaded to CLS, its metadata is passed through the `LogTag` field (for more information, see the `LogTag` field in Uploading Log via API), while the raw log content is passed through the `Log` field. A metadata index needs to be configured for all data which is passed via `LogTag` . A metadata index is a key-value index in essence, adopting the same indexing rules and configuration methods as key-value indexes. The only difference is that the metadata field in a metadata index is identified by the specific prefix `__TAG__.` . For example, the `region` metadata field is indexed as `__TAG__.region` .

For example, a complete log is as shown below:

```
10.20.20.10;[2018-07-16 13:12:57];GET /online/sample HTTP/1.1;200
```

If you use the Separator Format to extract log fields with the metadata `region:ap-beijing`, the structured log uploaded to CLS will be as follows:

```
IP: 10.20.20.10
request: GET /online/sample HTTP/1.1
status: 200
time: [2018-07-16 13:12:57]
__TAG__.region:ap-beijing
```

If the rules for metadata indexing are as follows:

Field Name	Delimiter
__TAG__.region	`@&()='",;:<>[]{}/ \\n\\t\\r`

Sample search: if you enter `__TAG__.region:"ap-beijing"`, the sample log can be returned.

Advanced Settings

To meet the needs in some special use cases, the index configuration feature provides the following advanced settings. We recommend you adopt the recommended configurations in practical uses, which will also be used by default when an index configuration is created in the console.

Configuration Item	Description	Recommended Configuration
Include built-in reserved fields in full-text index	**Contain**: The full-text index contains built-in fields `__FILENAME__`, `__HOSTNAME__`, and `__SOURCE__`, and full-text search and key-value search are supported, such as `"/var/log/access.log"` and `__FILENAME__:"/var/log/access.log"`. **Not contain**: The full-text index does not contain the aforementioned built-in fields, and only key-value search can be used, such as `__FILENAME__:"/var/log/access.log"`.	Contain
Include metadata fields in full-text index	**Contain**: The full-text index contains all metadata fields (those prefixed with `__TAG__`), and log fields can be searched for directly with full-text search, such as `ap-beijing`. **Not contain**: The full-text index does not contain any metadata fields, and log fields can be searched for only with key-value search, such as `__TAG__.region:ap-beijing`. Key-value search is not supported for STANDARD_IA log topics, and fields cannot be searched for in this case. **Contain only metadata fields with key-value index enabled**: The full-text index contains metadata fields with key-value index enabled but not	Contain

	metadata fields with key-value index disabled. This option is not available for STANDARD_IA log topics.	
Log storage rule in case of index creation exception	In case of any exception during index creation for logs, CLS will store raw logs in `__RAWLOG__` to avoid log loss. If index creation fails only for certain fields, the failed part can be stored in the specified field (which is `RAWLOG_FALL_PART` by default). For more information, see Handling rule for a log index creation exception.	Enable

Directions

Modifying index configuration

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to enter the log topic list page.
3. Click the target log topic ID/name to enter the log topic management page.
4. Click the **Index Configuration** tab and click **Edit** to enter the index configuration page.
5. Modify the index configuration as needed and click **OK**.

When modifying index configuration, you can also click **Auto Configure** to enable the system to automatically get the latest log collected as a sample and parse the fields in it into key-value indexes. You can perform fine tuning on the basis of automatic configuration to quickly obtain the final index configuration information.

Importing the index configuration

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to enter the log topic list page.
3. Click the target log topic ID/name to enter the log topic management page.
4. Click the **Index Configuration** tab and click **Import Configuration Rule**.
5. In the dialog box, select the log topic whose index configuration is to be imported and click **OK**. The index configuration of the selected log topic is automatically filled into the index configuration of the current log topic.
6. After confirming that everything is correct, click **OK**.

Appendix

JSON field parsing rules

During the storage process, logs may fail to be stored because there are too many JSON field levels, objects, or object types. To solve this problem, CLS will parse nested JSON fields only to the levels for which indexes are configured (existing log topics whose index configuration remains unchanged will not be affected), and the rest levels will be stored and displayed as strings. This feature upgrade affects only the search results of raw logs (search

statements contain only search criteria but not SQL statements) and does not affect the statistical analysis results (SQL results).

The following uses an actual log as an example to describe the parsing rule in detail.

Sample log:

The log contains three fields, where `kye1` is a common field, and `kye2` and `kye3` are nested JSON fields.

```
{
    "kye1": "http://www.example.com",
    "kye2": {
        "address": {
            "country": "China",
            "city": {
                "name": "Beijing",
                "code": "065001"
            }
        },
        "contact": {
            "phone": {
                "home": "188xxxxxxxx",
                "work": "187xxxxxxxx"
            },
            "email": "xxx@xxx.com"
        }
    },
    "kye3": {
        "address": {
            "country": "China",
            "city": {
                "name": "Beijing",
                "code": "065001"
            }
        },
        "contact": {
            "phone": {
                "home": "188xxxxxxxx",
                "work": "187xxxxxxxx"
            },
            "email": "xxx@xxx.com"
        }
    }
}
```

Index configuration

The following figure shows the key-value index configuration. Key-value index is configured for the `kye1` and `kye2.address` fields but not the `kye3` field.

Search and analysis effect on the console

`kye2.address` is displayed as a string, and its attributes and objects are not further expanded.

Although `kye2.contact` is not configured with a key-value index, because `kye2.address` is configured with an index, `kye2.contact` as an object at the same level as `kye2.address` is also displayed as a string.

`kye3` is not configured with a key-value index, and therefore its attributes and objects are not expanded.

The console provides the JSON formatting feature to display string-type JSON fields in a hierarchical manner, but this feature is only a display effect on the console, and the actual fields are still strings, as shown in the **API-based log search and analysis effect** section below.

API-based log search and analysis effect

If you use the SearchLog API, the value of the `Results` parameter in the output parameters is as follows (other parameters are not affected and remain unchanged):

```
{
  "Time": 1645065742008,
  "TopicId": "f813385f-aee0-4238-xxxx-c99b39aabe78",
  "TopicName": "TestJsonParse",
  "Source": "172.17.0.2",
  "FileName": "/root/testLog/jsonParse.log",
  "PkgId": "5CB847DA620DB3D4-10D",
  "PkgLogId": "65536",
  "HighLights": [],
  "Logs": null,
  "LogJson": "{\\"kye1\\":\\"http://www.example.com\\",\\"kye2\\":{\\"address\\":\\
}
```

`kye2.address` is a string, so its value is escaped as a string.

`kye2.contact` is an object at the same level as `kye2.address` , and although `kye2.contact` is not configured with a key-value index, its value is also escaped as a string.

`kye3` is not configured with a key-value index and is escaped as a string as a whole.

If you use code to process the output parameters of the SearchLog API, note the escape characters to avoid processing logic exceptions. To help you compare the API output parameters before and after the feature upgrade, the API output parameters before the upgrade are provided below:

```
{
  "Time": 1645065742008,
  "TopicId": "f813385f-aee0-4238-xxxx-c99b39aabe78",
  "TopicName": "zhengxinTestJsonParse",
  "Source": "172.17.0.2",
  "FileName": "/root/testLog/jsonParse.log",
  "PkgId": "25D0A12F620DBB64-D3",
  "PkgLogId": "65536",
  "HighLights": [],
  "Logs": null,
```

```
    "LogJson": "{\\"kye1\\":\\"http://www.example.com\\",\\"kye2\\":{\\"address\\":\\
}
```

Handling rule for a log index creation exception

If the raw log format is abnormal or the index configuration and raw log do not match, index creation may fail. In this case, CLS will store the raw log in `__RAWLOG__` for exception handling. This avoids log loss. `__RAWLOG__` supports only full-text search (full-text index needs to be enabled) but not key-value search, key-value index, and statistical analysis. After full-text index is enabled, index traffic, index storage, and fees will still be calculated according to the full text of the raw log for the abnormal log, without additional fees.

Index creation exceptions divide into two cases:

1. Index creation according to the index configuration fails for all fields in the log. In this case, the log has only the `__RAWLOG__` field, and only full-text search can be used.
2. Index creation fails for some fields and succeeds for the others in the log. In this case, the log has both the `__RAWLOG__` field and certain fields with a successfully created index (these fields support properly configuring key-value index and statistical analysis). In **Index Configuration** > **Advanced Settings**, you can also store abnormal fields in the specified field (which is `RAWLOG_FALL_PART` by default and supports configuring key-value index and statistical analysis).

CLS will keep optimizing the compatibility of index configuration with raw logs to avoid the abovementioned log exceptions. As the product iterates, specific exception handling rules may change.

Reindexing

Last updated：2024-01-20 17:25:15

Overview

As an edited index rule takes effect only for logs written after the rule is edited, if you want to reindex historical data according to the latest index rule, you need to use the reindexing feature. Below are common reindexing scenarios:
A field is added to a key-value index, and historical data needs to be searched for by this field.
The delimiter configuration is adjusted, and historical data needs to be searched for by the new delimiter.
Statistics collection wasn't enabled for a field. After it is enabled in the index configuration, the statistics of this field in historical data needs to be collected (i.e., SQL).
During reindexing, the index of raw logs in the specified time range will be rebuilt, which will incur index traffic fees (write traffic fees and index storage fees won't be incurred). If the data volume is high, reindexing will be time-consuming and incur high fees. We recommend you avoid frequently modifying the index configuration and reindexing as much as possible.

Prerequisites

The index has been enabled, and the latest index configuration can meet future requirements for search and analysis.

Notes

Only one reindexing task can be executed for a single log topic at any time, and each log topic can have up to ten reindexing task records. If there are already ten records, you can create a new reindexing task only after deleting an unwanted record.
Logs for the same time range can be reindexed only once. You need to delete historical task records to reindex such logs again.
After a reindexing task record is deleted, the index data before reindexing will be restored.
You can only reindex logs for the time range with a write traffic below 500 MB in the console. If the limit is exceeded, we recommend you narrow down the time range or submit a ticket to increase the limit.
The reindexing time range uses the log time. If the difference between the log upload time and the time range of logs to be reindexed is greater than one hour (for example, a log generated at 2:00 AM is uploaded at 4:00 PM, and logs between 12:00 AM and 12:00 PM need to be reindexed), the log won't be reindexed and cannot be searched for

subsequently. If a new log is reported within the time range for which logs have been reindexed, it won't be reindexed and cannot be searched for subsequently either.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to enter the log topic list page.
3. Click the target log topic ID/name to enter the log topic management page.
4. Select the **Index Configuration** tab and click **Reindex** at the bottom.
5. Select the time range of logs to be reindexed and click **Start Reindexing**.

Note:

Here, the write traffic and time of the log data to be reindexed is estimated according to the selected time range. If the write traffic is too high and the time is too long, we recommend you narrow down the time range accordingly.

6. After reindexing is completed, you can view the completed task in the list. You can also delete the specified task. Once a task is deleted, the index data before reindexing will be restored.

Multi-Topic Search

Last updated：2024-12-20 16:15:18

Overview

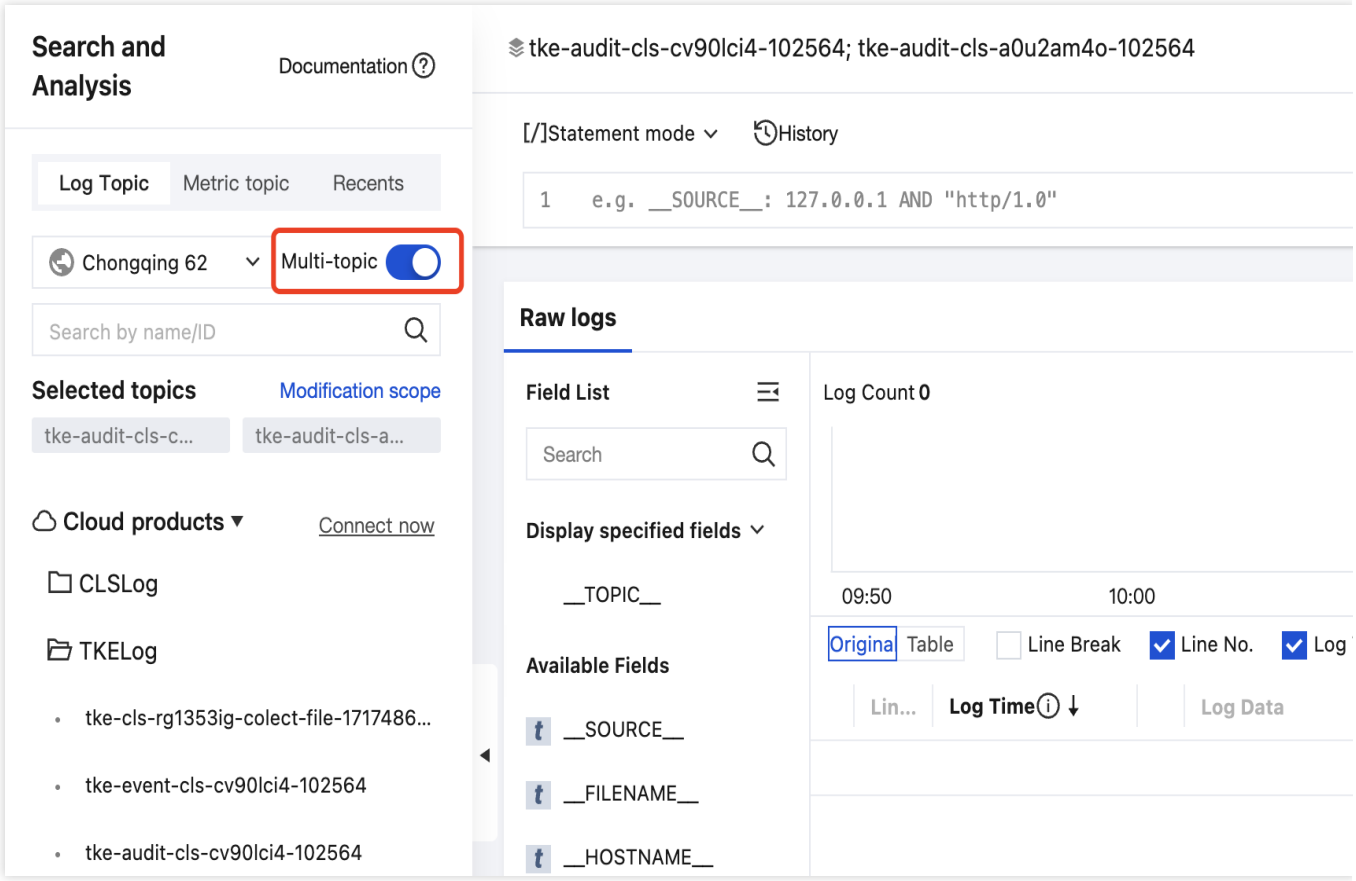
Multi-topic retrieval is used to centrally retrieve logs distributed across multiple log topics, for example, by RequestID to retrieve related logs across log topics. This document shows you how to use multi-topic retrieval.

Use Limits

Supports selecting log topics across log sets, but does not support selecting log topics across regions.
Up to 50 log topics can be searched simultaneously.
Only supports retrieval of raw logs, SQL-based statistical analysis is not supported yet.

Operation Steps

1. Log in to the CLS console.
2. In the left sidebar, click **Search and Analysis** to enter the retrieval analysis page.
3. Enable **Multi-topic** in the upper left corner to enter multi-topic retrieval mode, then select the log topics to be searched and click **OK**.



4. Enter the retrieval and analysis statement, select the time range on the right side, and then click

to execute the retrieval.

Notes:

When the index configuration among multiple topics is inconsistent, the retrieval statement may report errors under some log topics. At this time, only the data and error information of log topics without error reporting are displayed. For example, Topic A has the field KeyA and enables the key-value index, but Topic B does not have this field. When using the retrieval statement `KeyA:xxx` to perform retrieval, Topic B execution will report an error, and only logs related to Topic A can be viewed.

Context Search and Analysis

Last updated：2024-01-20 17:25:15

This document describes how to view the context of a log in the original file in the CLS console.

Overview

Log context search refers to searching for the log's context, that is, several logs before or after the log itself. This feature allows you to troubleshoot quickly whenever an error occurs.

Advantages

Log context search frees you from the hassles associated with machine logins. On the log search page, you can quickly view the log context of any file/machine.
Knowing the actual error occurrence time, you can specify a time range on the log search page to quickly locate the suspicious log and query its context for troubleshooting.
Storage capacity of the server or data loss caused by log rotation will not be a problem. The data history can be viewed on the log search page anytime.

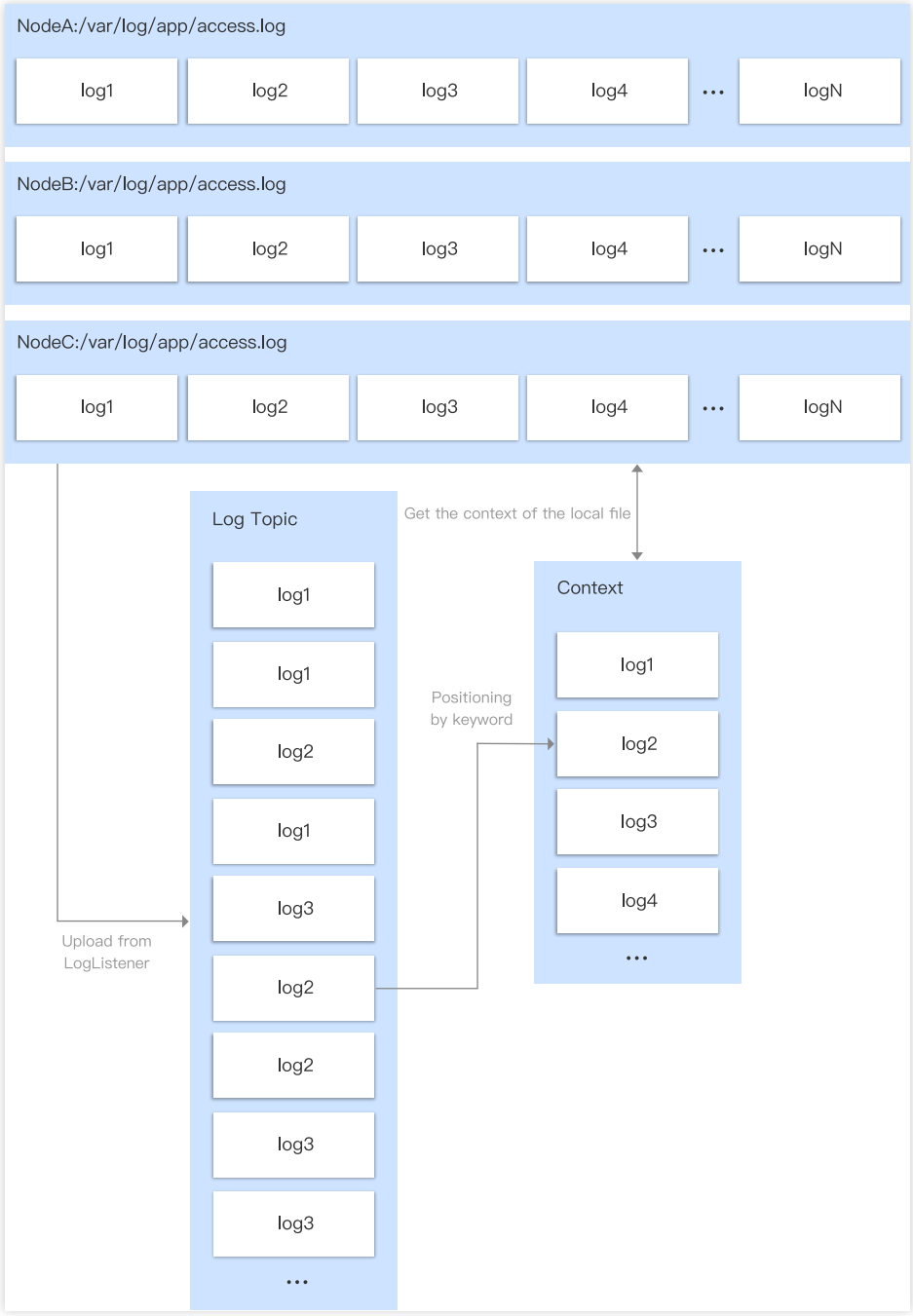
Prerequisites

Log context search and analysis are available only for version 2.3.5 and above. You are advised to install or upgrade to the latest version.
Context search is supported for only logs collected by LogListener.
You have enabled and configured index. For more information, please see Configuring Index.

Example

The common process of an order is as follows: log in > browse merchandise > select merchandise> add merchandise to the shopping cart > place an order > pay for the order > make a deduction > generate the order.
Use case: if one of the orders of a user fails, you can query the error log using the order number. Then, locate the log's context to find out the cause (for example, order deduction failed).
You can troubleshoot in CLS as follows:

1. Log in to the CLS console and go to the **Search and Analysis** page. Then, specify a time range based on the error occurrence time and enter the **keyword** (order number) to locate the error log of the order.
2. Scroll up/down based on the error log until you locate the desired context of the log.



Directions

1. Log in to the CLS console.

2. On the left sidebar, click **Log Search** to go to the **Search and Analysis** page.
3. Select the **Region**, **Logset**, and **Log Topic** as needed.
4. On the **Raw Data** tab, find the time of the error log and click

to go to the context search and analysis page.
5. On the context search and analysis page, 10 logs before and after the error log will be displayed.

On this page, you can:

Scroll up/down to view the context.

Click **Show Earlier** to page up. Up to 20 previous logs can be displayed at a time.

Click **Show More** to page down. Up to 20 later logs can be displayed at a time.

Enter the keyword in the **Highlight** text box to fill the keyword in yellow.

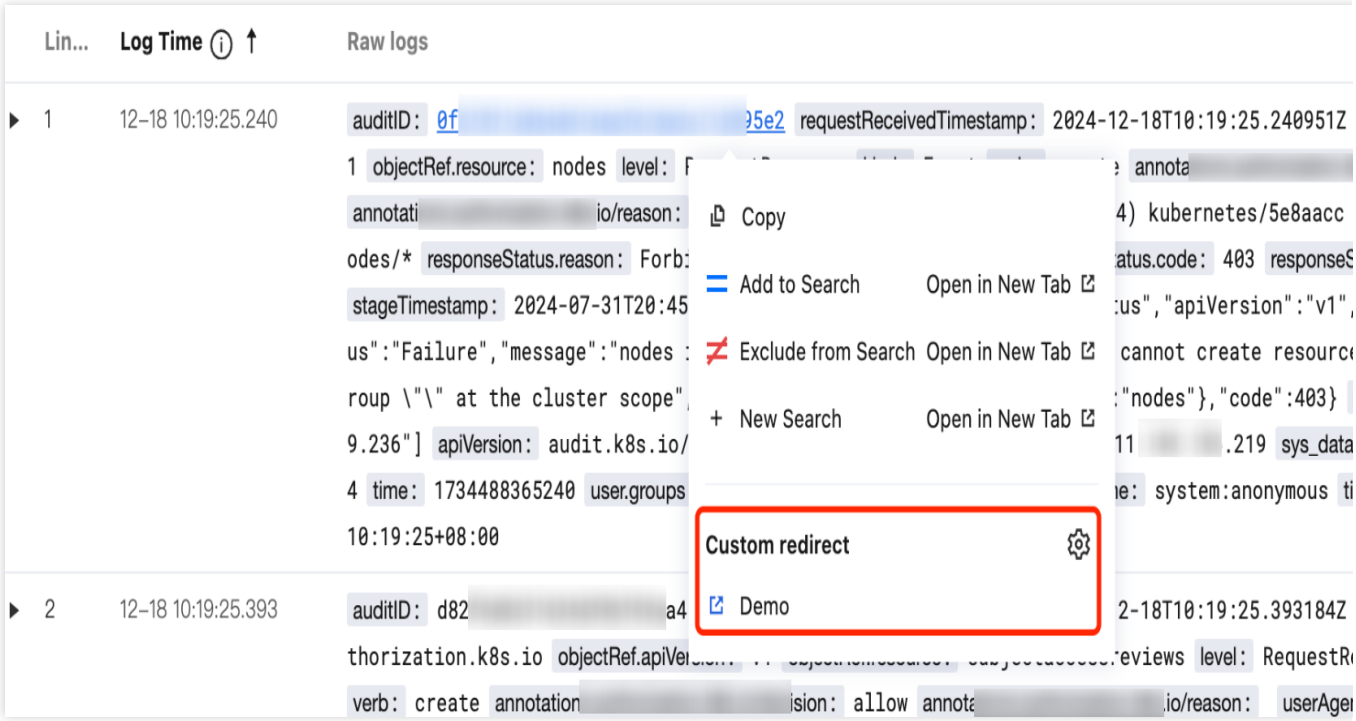
Enter a string in the **Filter Logs** text box to highlight the string.

Custom Redirect

Last updated：2024-12-20 16:17:33

Overview

For the raw log entries found on the retrieval analysis page, CLS supports custom redirection by clicking the field value for further operations, such as:
Search related logs in other log topics according to request_id.
Query user information in the internal user management platform using user_id.



Custom redirection supports two types of operations:

Redirection type	Applicable Scenario
Open External URL	Open the specified URL and carry the designated fields from the log as parameters in the URL, for example, query user information on the internal user management platform based on user_id.
Search for other log topic	Retrieve the specified log topic and carry the designated fields from the log as retrieval conditions, for example, retrieve related logs in other log topics based on request_id.

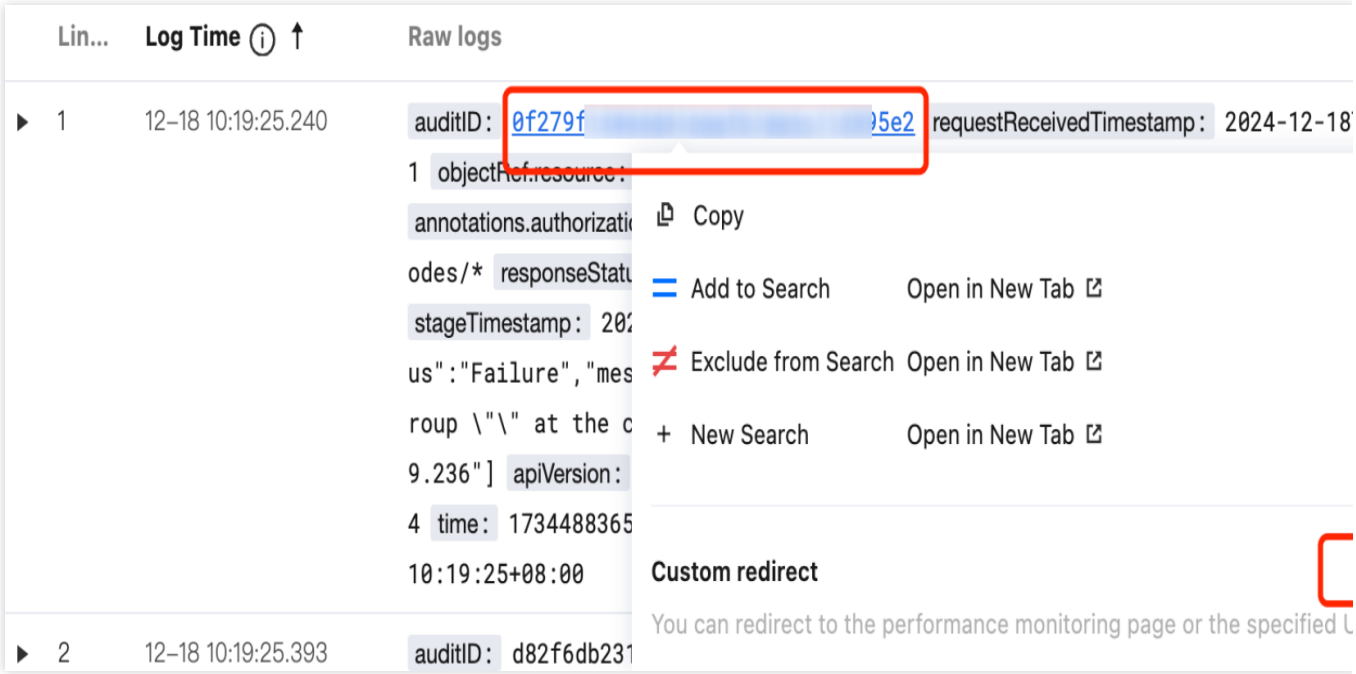
Operation Steps

Adding custom redirection

1. Log in to the CLS console.
2. In the left sidebar, click **Search and Analysis** to enter the retrieval analysis page.
3. After querying the log, click the field value that needs redirection. In the pop-up menu, click

+

to add a new redirection.



4. Enter the following configuration information in the pop-up box:

Jump type: Select the corresponding jump type as needed.

Trigger field: click a field value to trigger the jump, for example, "stgw_request_id" in the above figure.

Configuration Name: Custom name for the jump, such as "View downstream service logs".

Other Configuration Items: Related to the jump type, see the table below for details:

Open External URL

Search for other log topic

URL: The URL address that needs to be opened, starting with http:// or https://. You can use variables in the URL to dynamically obtain log field values and other information. For example, `{{__currentValue__}}` represents the value currently being clicked.

Open in a new window: Whether to redirect to a new window.

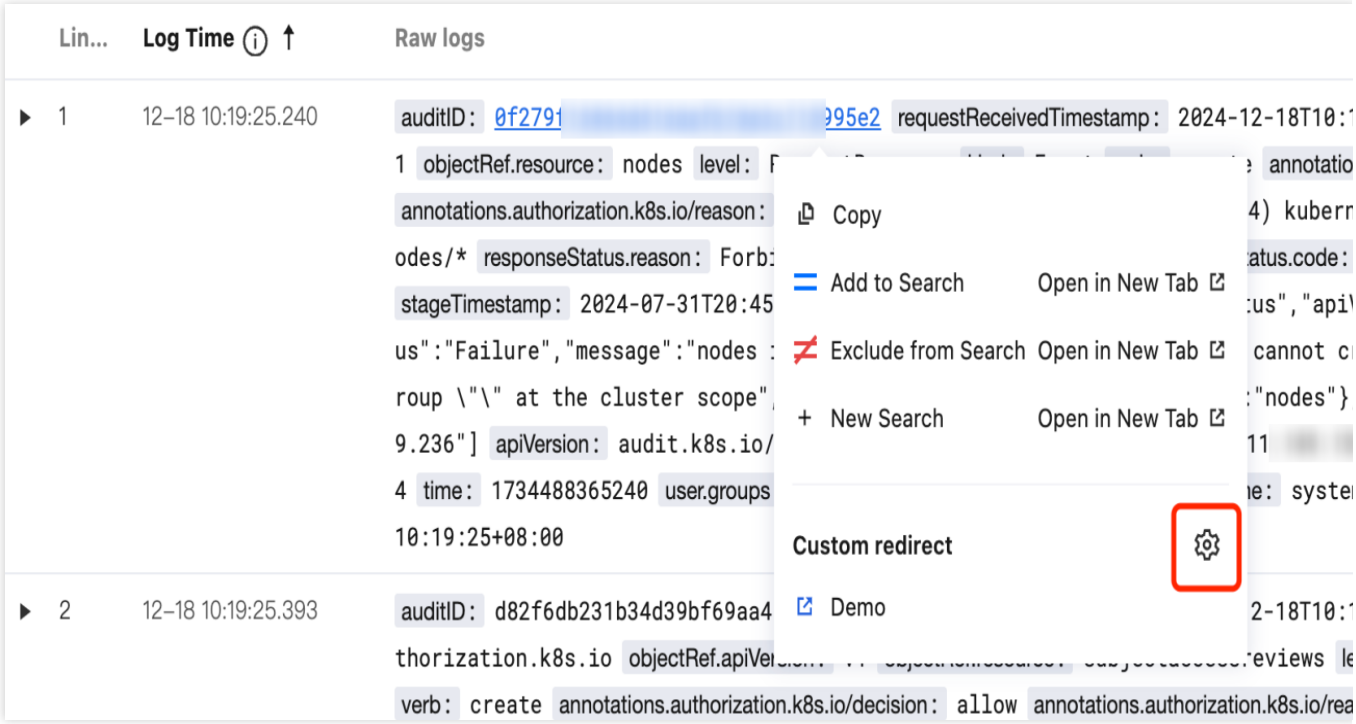
Log Topic: The log topic to be retrieved.

Time Range: The time range to be retrieved. It is recommended to fuse the current time range or customize it.
Continue to Use Current Search Criteria: Whether to continue to use the current search criteria.
Add Search Criteria: Whether to add new search criteria. It can use variables to dynamically acquire log field values and other information. For example, using `stgw_request_id:{{__currentValue__}}` to use the currently clicked value as the key-value search criteria. When searching for other log topics, it will automatically convert to `stgw_request_id:"8da469b42947445891cc10fc55d75471"` appearing in the search statement.
Open in a new window: Whether to redirect to a new window.

Editing/Deleting Custom Redirection

1. Log in to the CLS console.
2. In the left sidebar, click **Search and Analysis** to enter the retrieval analysis page.
3. After inquiring the logs, click to edit/delete the redirected field value. In the pop-up menu, click


Manage Custom Redirection.

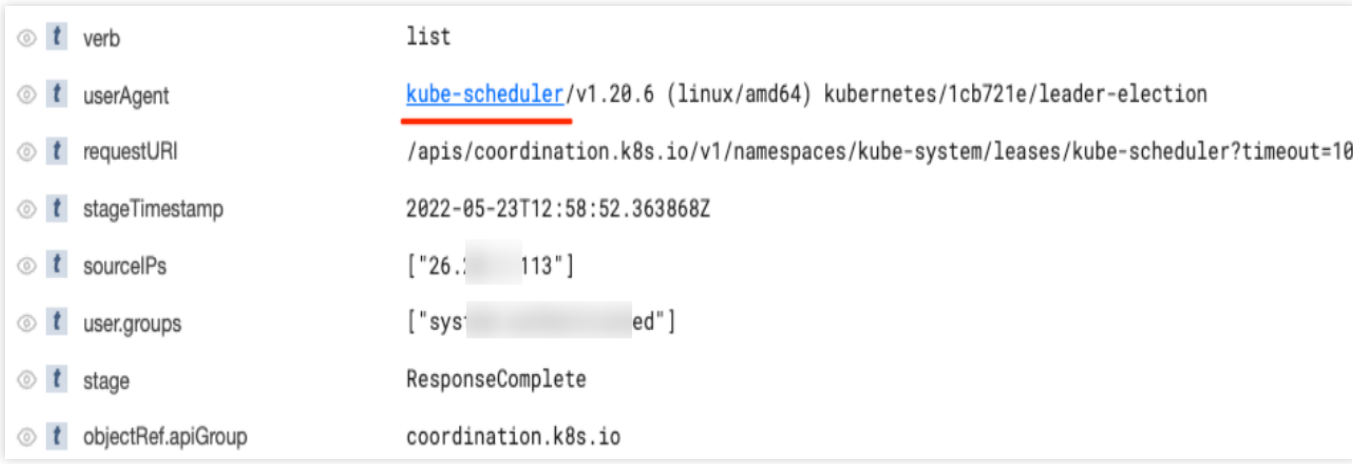


4. In the pop-up window, click the custom redirection configuration name. Click


on the right side of the name to delete the configuration, and update the configuration using the form on the right side. Finally, click **Application** to save.

Variable Description

`${__CurrentValue}` indicates the currently clicked field value. When the field is word-segmented, this variable refers to the words after the word segmentation. For example, in the following figure, the separator `/` is behind kube-scheduler. When the mouse pointer is hovered, only kube-scheduler is highlighted. When a custom redirection is triggered by clicking, the corresponding `${__CurrentValue}` is the kube-scheduler.



`${__TopicId}` indicates the current topic ID, such as `a85bbd1c-233f-xxxx-aeda-70cbd9f8715a`. `${__StartTime}` and `${__EndTime}` indicate the start and end Unix timestamps of the current query time range.

You can use `${}` to enclose the field name as a variable to represent the complete value of that field. For example, in the above image, you can use `${userAgent}` to represent the value of the userAgent field, `${userAgent}` = `kube-scheduler/v1.20.6 (linux/amd64) kubernetes/1cb721e/leader-election`.

Downloading Log

Last updated：2024-01-20 17:25:15

Overview

CLS allows you to export the collected log data, so you can download the data to view it locally.
The log download feature is as described below:
Up to 50 million logs can be downloaded at a time.
Search conditions and search time range can be specified.
JSON and CSV export formats are supported.
Note:
Only raw logs can be downloaded. To download statistical analysis results (SQL execution results), click

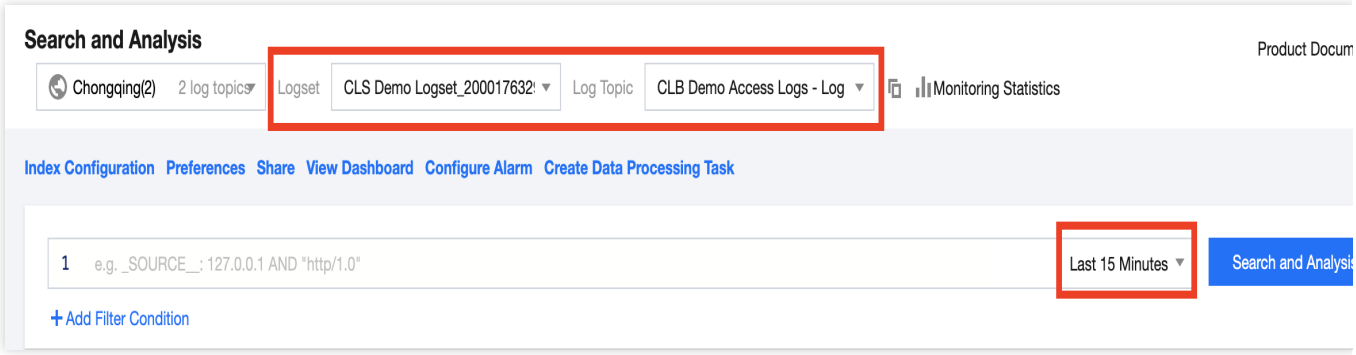
in the upper-right corner.

Billing description

Log download incurs public network read traffic fees, which CLS charges based on the size after gzip compression. If your raw log is 100 GB, around 40–70 GB will be billable after compression (due to raw log differences). As the public network read traffic price is 0.141 USD/GB, the fees will be 5.64–9.87 USD.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to go to the search and analysis management page.
3. Click the drop-down lists of **Logset** and **Log Topic** to select the content to be searched.



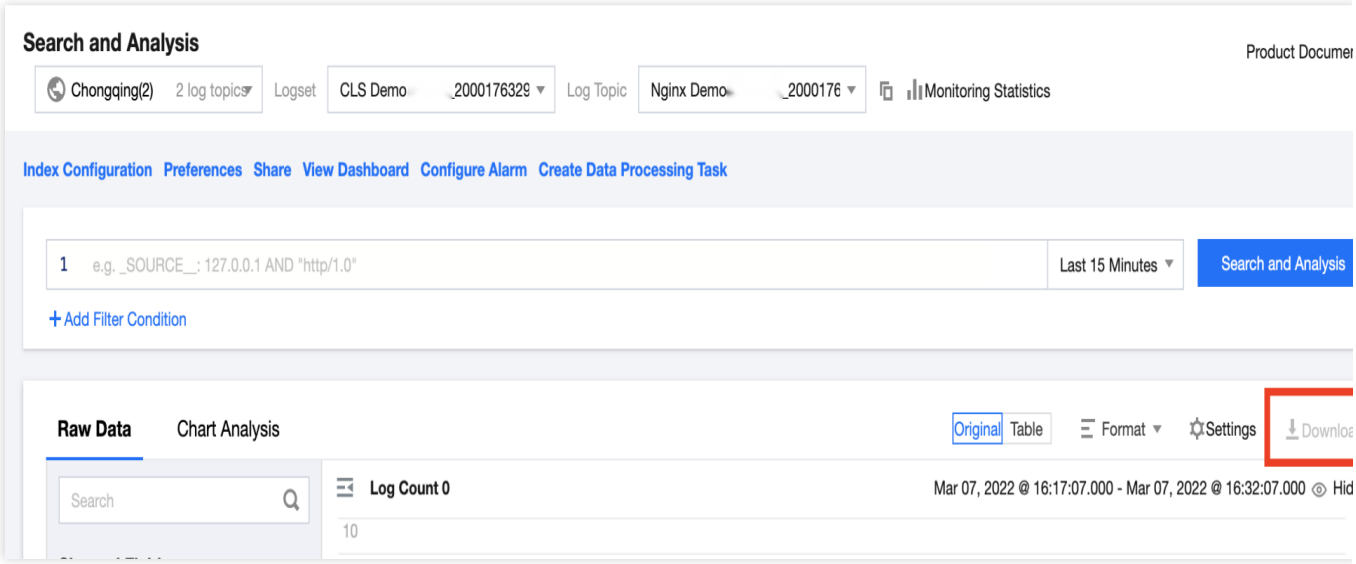
4. Click **Log Time** and select a log time for search.

You can select **Recent Time** or **Relative Time** or customize the time range.

5. Click **Search and Analysis**.

6. After the log data is returned, click

to **download logs**.



Note:

If no log data is returned, you cannot download logs.

7. In the pop-up window, confirm the time range of the logs to be downloaded and the search statement and download log data according to the following settings:

Data format: "CSV format" or "JSON format" can be selected as needed.

Log sorting: Logs can be sorted in "ascending" or "descending" (default) order by time.

Log quantity: "All logs" are exported by default. You can also select a "custom log quantity".

Note:

For the CSV format, only fields with configured indexes can be exported, and the downloaded logs may be empty. > -

For the JSON format, all fields can be exported, regardless of index configuration. > - If the number of logs to be

downloaded exceeds 50 million, we recommend you narrow the search scope (for example, use more precise search conditions or narrow the search time range) or download only a specified number of logs. > - If you do need to download more than 50 million logs, you can set the log search time range to create multiple download tasks and download logs in batches.

8. Click **Export** to switch to the **Export Logs** page.

Note:

A download task can be in any of the following states: **Waiting**, **File generating**, and **File generated**. A newly created download task waits for a while before entering the **File generating** state and starts generating data files in the background. If multiple download tasks are created at the same time, the system executes them in sequence based on the creation time. Tasks that are created later are in the **Waiting** state.

9. After the **File Name**/**Task ID** status changes to **File generated successfully**, click **Download** to export the logs.

Note:

The exported log file is retained for only 3 days.

Search and Analysis (Metric Topic) Syntax Rules

Last updated：2024-09-20 17:48:27

Metric topics use Prometheus' PromQL syntax to query metric data. For detailed syntax, see the Prometheus official documentation. This document mainly describes some common syntax and directions to help you quickly understand the basic usage.

There are two types of metric queries:

Range queries: Also known as Prometheus Range queries. This method queries metric data at each time point within a specified time range based on the step. It is commonly used to observe trends in metric changes.

Note:

When the query step is less than the metric collection interval, the query results may contain duplicate data. For example, if the CPU utilization is reported every 15 seconds, and the query interval is set to 1 second, you will see 15 identical duplicate data.

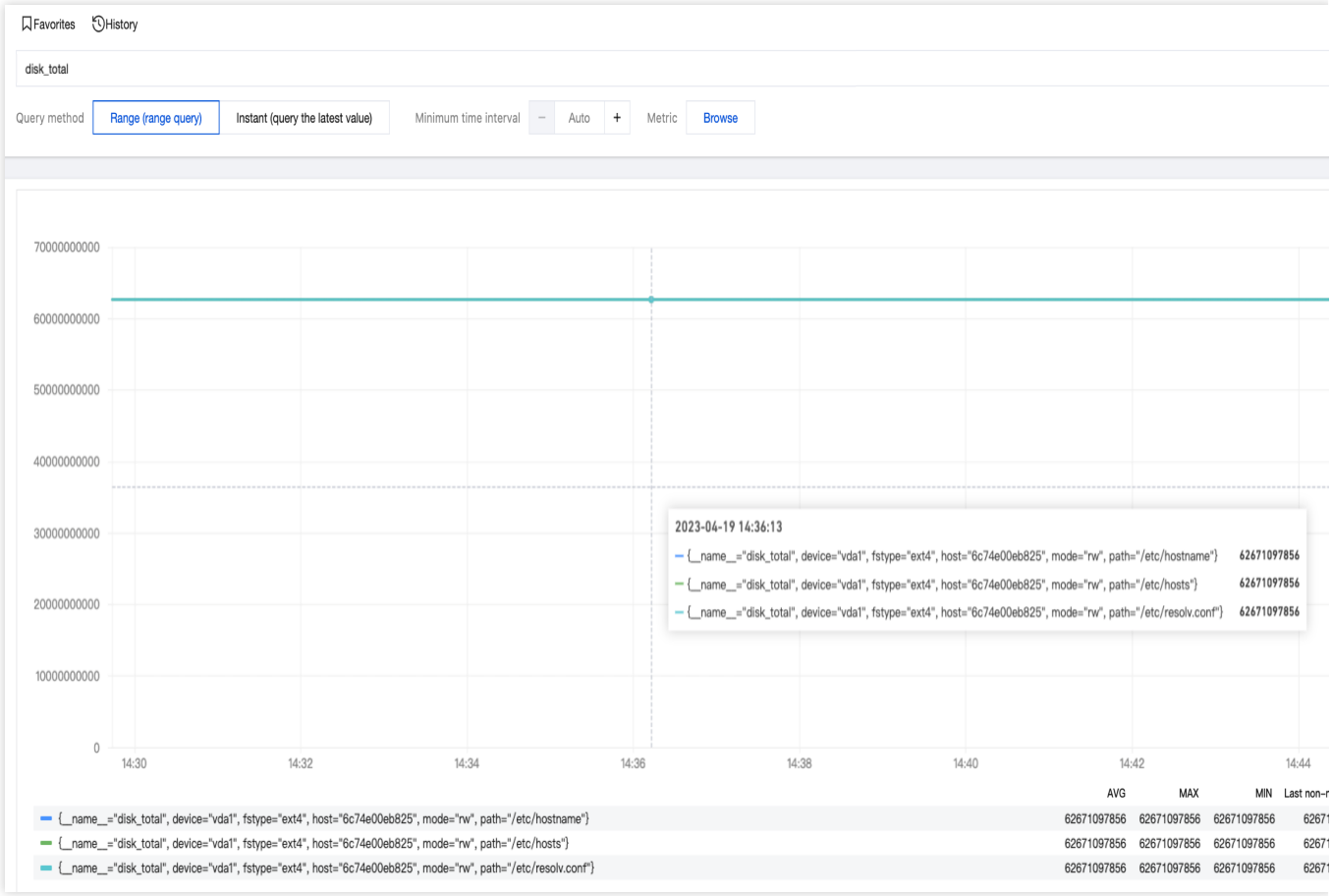
Instant queries: Also known as Prometheus Instant queries, this method returns the latest metric data within the query time range. It is commonly used to retrieve the current value of a metric.

Common syntax

1. Specifies the metric name to query the trend of metric values for each dimension under that metric name. For example, querying the disk capacity (disk_total):

```
disk_total
```

Search for the required CAM policy as needed, and click to complete policy association.

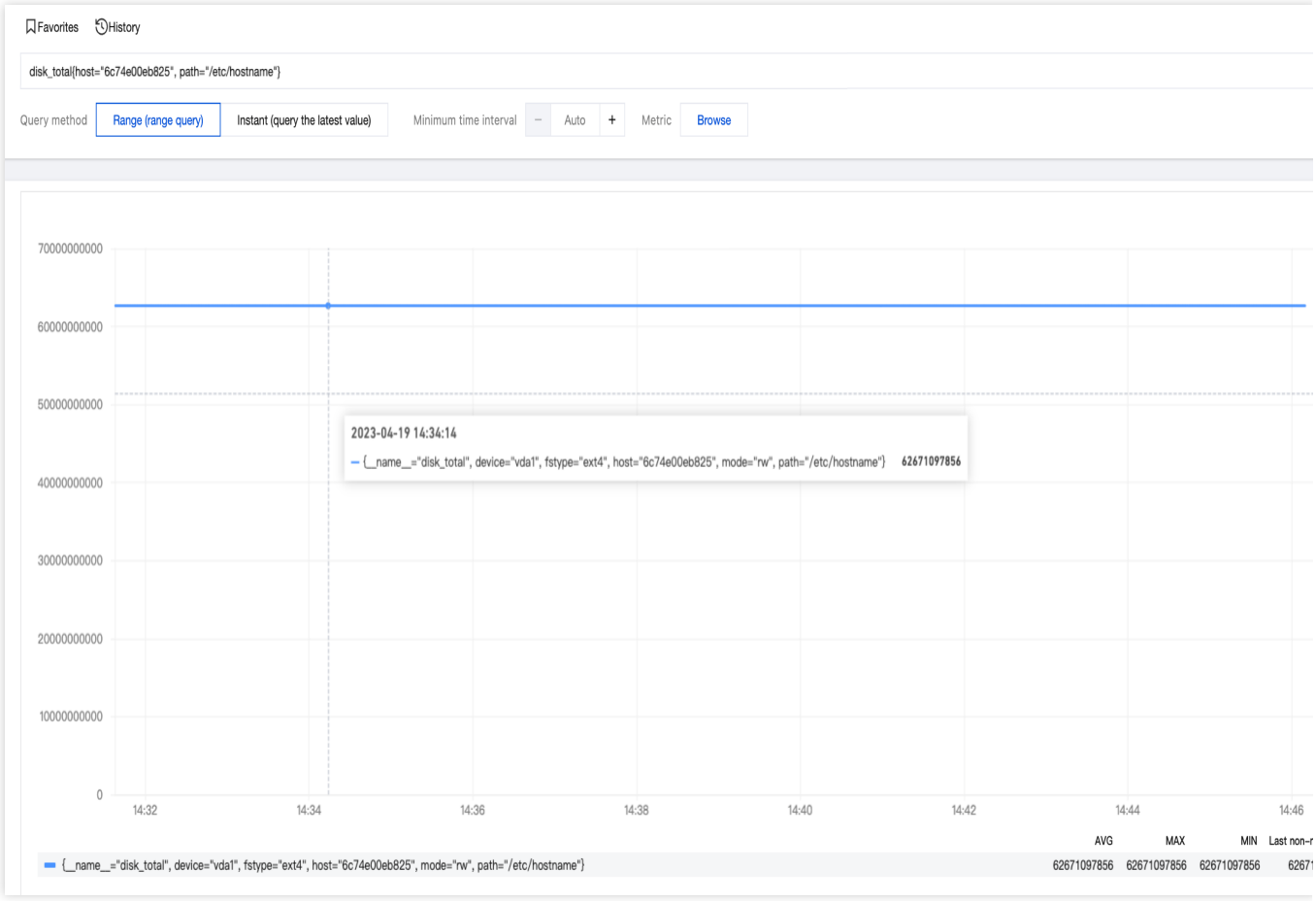


2. Specifies the metric name and dimensions to query the trend of metric values for a specific dimension combination.

For example, querying the disk capacity where the hostname is 6c74e00eb825 and the path is /etc/hostname:

```
disk_total{host="6c74e00eb825", path="/etc/hostname"}
```

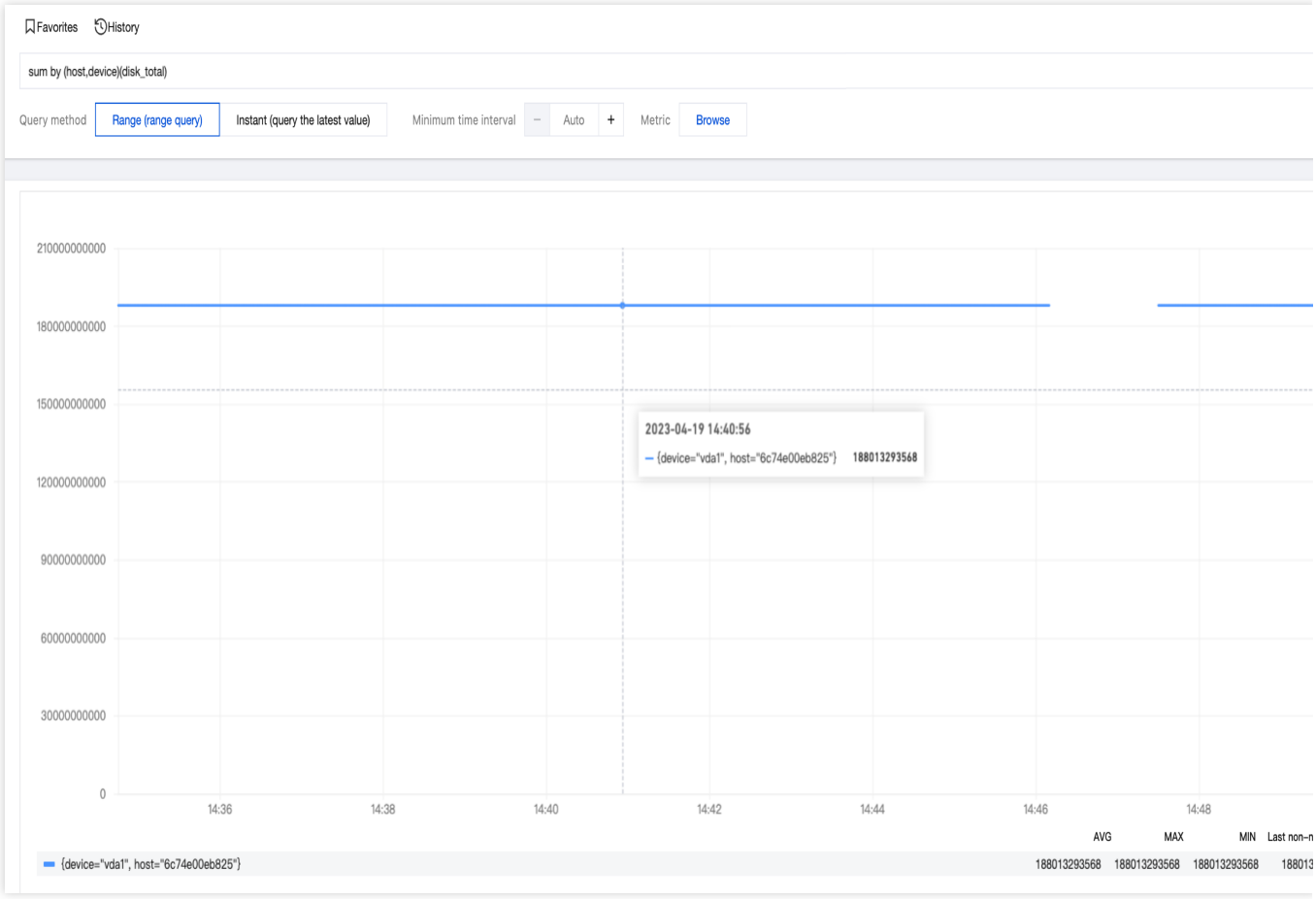
Search for the required CAM policy as needed, and click to complete policy association.



3. Aggregates metric data by specific dimensions. For example, querying the total disk capacity for each host and each device:

```
sum by (host,device)(disk_total)
```

Search for the required CAM policy as needed, and click to complete policy association.



Note:

For more PromQL syntax, see the Prometheus official documentation, or see the query examples.

Directions

1. Log in to the Cloud Log Service console.
2. In the left navigation bar, select **Search and Analysis**, at the top, select the region where the metric topic is located, choose "Metric topic" as the subject type, and then select the required metric topic.
3. Enter the query statement in the input box, then click the

on the right to query metrics. You can also click **Browse** to query the names of existing metrics.

Dashboard
Chart Overview

Last updated：2024-12-25 14:41:47

CLS provides a great variety of chart types. You can use SQL to flexibly collect system and business metrics in logs and display them in charts. You can also save chart analysis results to dashboards for long-term monitoring.

Features

Enter Path	Feature	Description	Details
	Folder Creation	Dashboards support folder creation for classifying and collapsing dashboards, making them easier to manage.	-
	Create a Dashboard	CLS supports various ways to create a dashboard: Blank Dashboard: Create a blank dashboard that requires users to add charts manually and save them. Import Dashboard: Import a Json file of an existing CLS dashboard. The Json file is exported from an existing dashboard. Create from Dashboard Template (Recommended): CLS offers a wide range of dashboard templates. You can choose the right template according to your scenario and log content, enable it with one click, and a dashboard will be automatically generated.	Creating a Dashboard
	Filters and Variables	Filters: All chart data in the dashboard can be filtered by the specified field value. Variables: Users can set a variable value via static input or dynamic query and apply it to search statements, titles, and text.	Filters and Variables
	Table	A table is the most common type of data display, where data is organized for comparison and counting. It is suitable for most scenarios.	Table
	Sequence Diagram	A sequence diagram requires statistics to have a sequence field so that it can organize and aggregate the metrics in chronological order. It visually reflects the change trend of a metric over time. It is suitable for trend	Sequence Diagram

		analysis scenarios, for example, analyzing the trend of the daily number of 404 errors in the past week.	
	Bar Chart	A bar chart describes categorical data. It visually reflects the comparison of each category in size. It is suitable for category statistics scenarios, for example, collecting the numbers of each type of error code in the last 24 hours.	Bar Chart
	Pie Chart	A pie chart describes the proportions of different types. It measures the proportion of each type by the slice size. It is suitable for proportion statistics scenarios, for example, analyzing the proportions of different error codes.	Pie Chart
	Individual Value Plot	An individual value plot describes a single metric, typically a key metric with business value. It is suitable for collecting daily, weekly, or monthly metrics such as PV and UV.	Individual Value Plot
	Gauge Chart	A gauge chart describes a single metric. Unlike an individual value plot, it is generally used with a threshold to measure the metric status. It is suitable for rating scenarios, such as system health monitoring.	Gauge Chart
	Map	A map shows the geographic location of data through the position of graphics. It is generally used to display the distribution of data in different geographic locations. It is suitable for geographic statistics scenarios, such as the geographic distribution of attacker IPs.	Map
	Sankey Diagram	A Sankey diagram is a special type of flow diagram used to describe the flow of one set of values to another set. It is suitable for directional statistics scenarios, such as firewall source and destination IP traffic.	Sankey Diagram
	Word Cloud	A word cloud is a visual representation of the frequency of words. It is suitable for audit statistics scenarios, such as high-frequency personnel statistics.	Word Cloud
	Funnel Chart	A funnel chart is suitable for business processes with one single flow direction and path. It collects the statistics of each stage and uses a trapezoidal area to represent the business volume difference between two stages.	Funnel Chart
	Log	Log charts allow you to save raw logs to the dashboard. You can quickly view the analysis result and the	log

		associated log content on the dashboard page, with no need to redirect to the search and analysis page.	
	Text	Text type charts support the MarkDown syntax. You can insert text, image links, hyperlinks, etc. on the dashboard page.	Text
	Heat Map	Heatmaps display statistical charts by coloring the blocks. For statistical indicators, higher values are represented by darker colors and lower values are represented by lighter colors. Heatmaps are suitable for viewing the overall situation, detecting anomalies, displaying differences among multiple variables, and checking whether there is any correlation between them.	Heat Map
	Data Conversion	Data conversion allows you to perform further processing of search results, including modifying data types, selecting fields for chart creation, and merging groups. This satisfies your chart creation needs without modifying SQL statements.	Data Conversion
	Unit Configuration	Automatic unit conversion is available in charts. When you select an original unit, the value is automatically converted to the next higher unit if it meets the conversion factor. Units can be configured to display decimal places.	Unit Configuration
Other Feature Configuration	Interaction Event	The chart has its interaction event feature, which allows clicking the chart content to trigger interactions such as opening up the search and analysis page, dashboard page, and third-party URL.	Interaction Event
	Add a Group	The dashboard supports chart grouping. Grouping allows you to categorize and collapse dashboard chart contents.	-
	Subscribing to Dashboard	CLS allows you to subscribe to a dashboard and export it as an image. Daily, weekly, and monthly reports can be sent to specified recipients regularly via email or WeCom. It is available for those dashboards that need to send daily, monthly, and reports to the team.	Subscribing to Dashboard
	Chart Time Configuration	After turning off the use of global time, the chart time is controlled independently and no longer changes with the change in global time. Different charts in the dashboard can use different time ranges, supporting more varied comparison scenarios.	Chart Time Configuration

Specifications and Limits

For the specifications and limits of the dashboard, see LogListener Limits.

Creating a Dashboard

Last updated：2024-01-20 17:31:30

The dashboard provides a global view of data analysis and monitoring. You can view multiple statistical charts based on query and analysis results in the dashboard. This document describes how to create a dashboard and related operations.

Directions

1. Log in to the CLS console.
2. Select **Dashboard** on the left sidebar to enter the dashboard list page.
3. On the dashboard list page, click **Create**.
4. In the pop-up window, enter the dashboard name and click **OK**.
5. Click **Save**.

Related Operations

After creating a dashboard, you can perform the following operations in the dashboard:

Operation	Description
Adding a chart	A dashboard also provides an entry for creating statistical charts, supporting simple chart creation and custom chart creation. For more information, please see Adding a Chart.
Deleting a chart	You can delete an existing chart.
Editing a chart	You can edit a chart on the chart editing page.
Copying a chart	You can copy a chart to the current or another dashboard.
Exporting chart data	You can export chart data in CSV format.
Viewing data on the search and analysis page	You can quickly add search statements, log topics, and other information for the current chart on the search and analysis page.
Quickly adding	You can quickly add search statements, log topics, and other information for the current chart

an alarm	on the alarm editing page.
Full-screen browsing	Full-screen browsing of a single chart or the entire dashboard is supported.
Refreshing	Automatically periodic data refreshing and manual data refreshing are supported.
Adding a template variable	Template variables allow you to define and modify data query and filtering parameters in the dashboard more flexibly, which improves the reuse rate of the dashboard and the granularity of analysis. For more information, please see Template Variables.
Viewing/Editing a dashboard	You can view a dashboard and edit the dashboard layout.

On the dashboard management page, you can perform the following operations:

Operation	Description
Creating a dashboard	You can create a dashboard on the dashboard creation page.
Deleting a dashboard	You can delete an existing dashboard.
Modifying dashboard tags	You can modify a single dashboard tag or multiple dashboard tags at a time.
Opening a dashboard	You can open a dashboard and go to the dashboard viewing/editing page.
Searching for or filtering dashboards	You can search for or filter dashboards by a combination of dashboard attributes such as the dashboard ID, name, region, and tag.

Creating Statistical Charts
Adding Chart

Last updated：2024-01-20 17:31:30

CLS supports visual chart statistics of search and analysis results, and saves statistical charts to dashboards for continuous monitoring and viewing. A dashboard also provides an entry for creating statistical charts, supporting simple chart creation and custom chart creation.

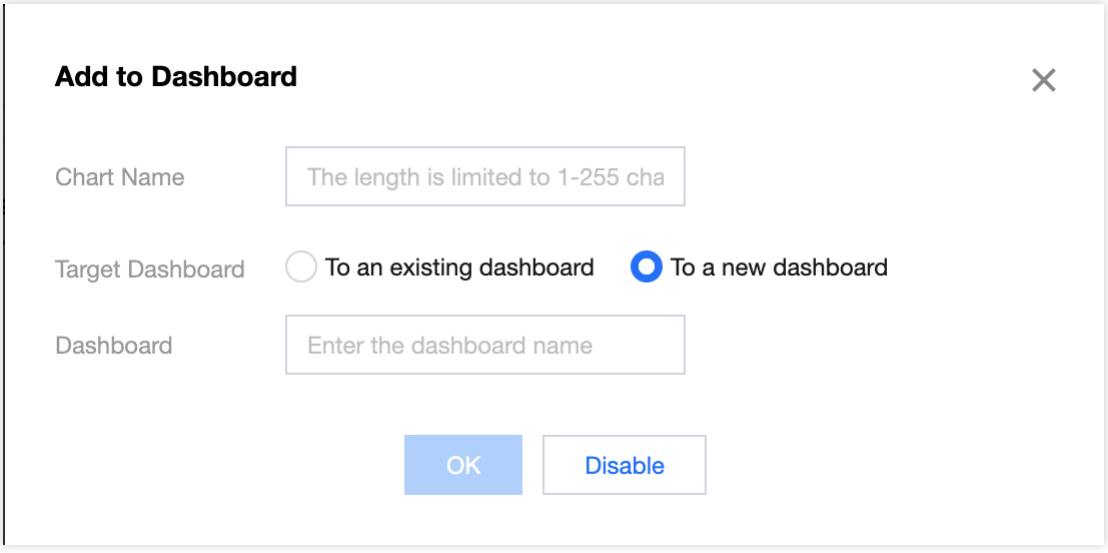
Directions

Adding a chart via the Search and Analysis page

Note:

In this mode, you need to search for data to generate chart content. For more information, see here.

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to go to the search and analysis management page.
3. Click the **Chart Analysis** tab and click **Add to Dashboard**.
4. In the pop-up window, enter information and click **OK**.



Form Element	Description
Chart Name	The chart name.
Target Dashboard	The dashboard type of the chart. If you select **To an existing dashboard**, the chart will be added to an existing dashboard. If you select **Create Dashboard**, you need to create a

	dashboard and add the chart to it.
Dashboard	The dashboard name.

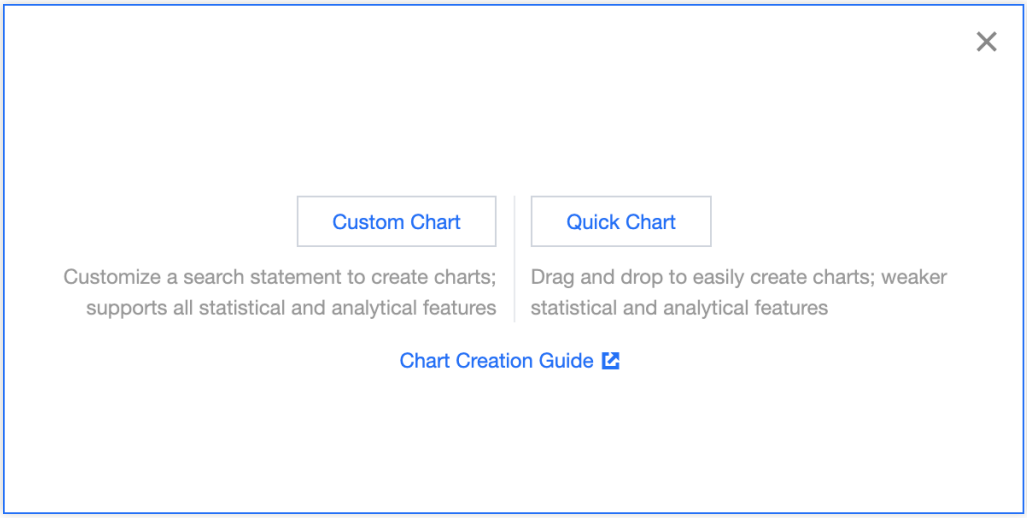
Adding a chart via the Dashboard page

1. Log in to the CLS console.
2. On the left sidebar, click **Dashboard** to enter the dashboard management page.
3. Click the ID/name of the target dashboard to enter the dashboard details page.

Note:

If you do not have a dashboard, create one.

4. Click **Create** and select a chart type as needed to create a chart.



Quick Chart: You can drag and drop to easily create charts. However, this mode supports weaker statistical and analytical features.

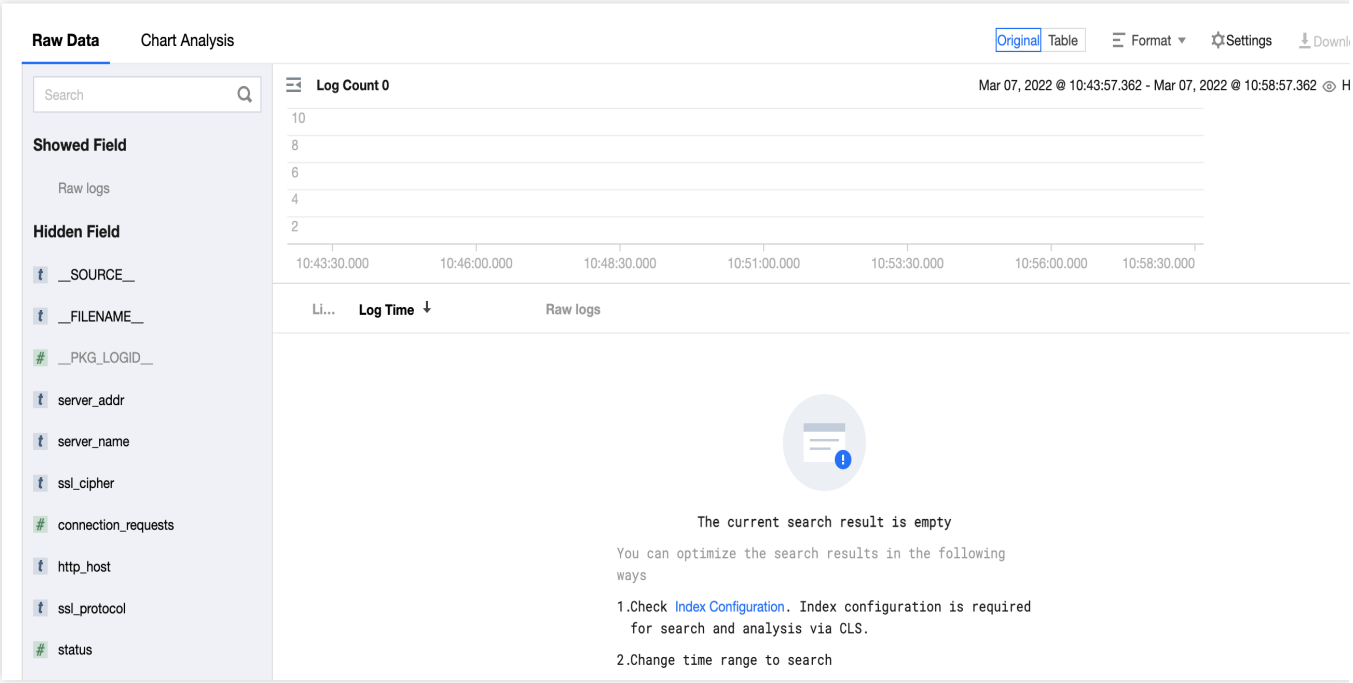
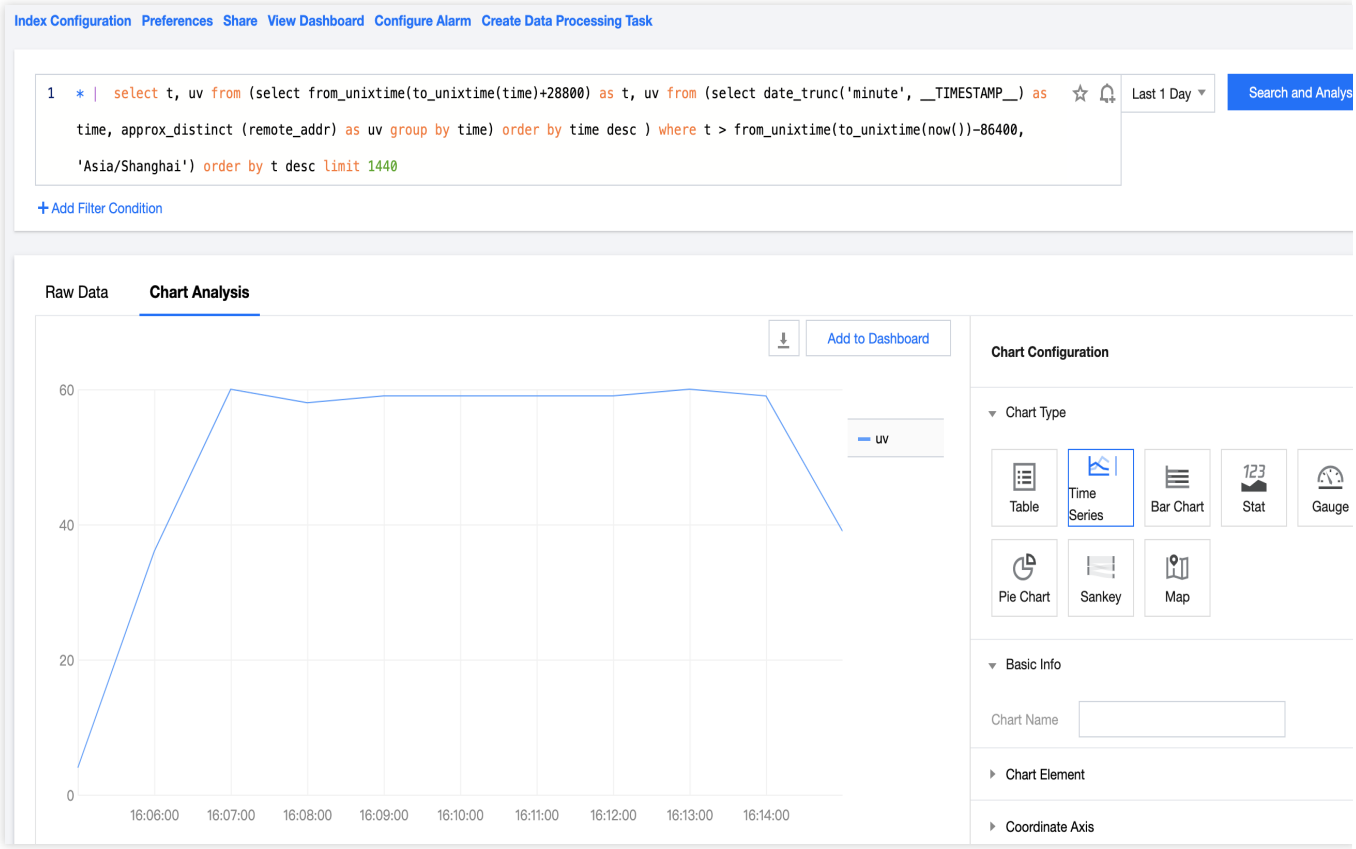


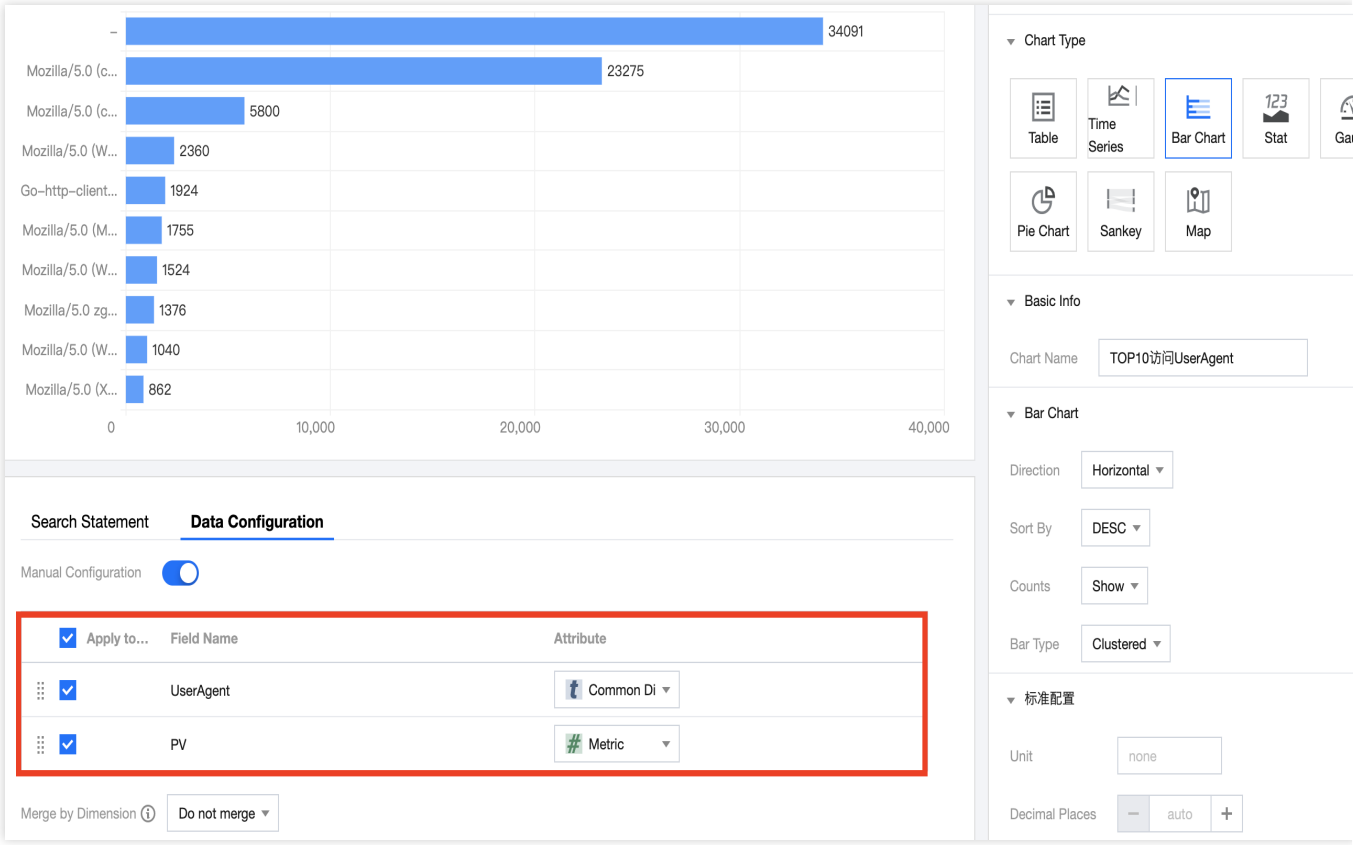
Chart Element	Description
Field	The list of current log topic fields. You can click or drag and drop a field to the condition input box on the right.
Metric	A metric measures a certain characteristic of an item, generally a numeric field. You can drag and drop a field from the field list on the left or click the "+" icon to display a drop-down list to select a field. After adding a field, you can click the settings icon next to the field to modify the field's aggregate calculation mode, which is "AVG" by default.
Dimension	A dimension is the perspective for analyzing a metric. It is generally a string-type field describing the attributes of an item. You can drag and drop a field from the field list on the left or click the "+" icon to display a drop-down list to select a field.
Filter	A filter field filters data attributes. You can drag and drop a field from the field list on the left or click the "+" icon to display a drop-down list to select a field. After adding a field, you can click the settings icon next to the field to modify the field filter mode, which is "Exist" by default.
Sort	A sorting field sorts statistical results. Only specified condition fields can be sorted. We recommend you click the "+" icon to display a drop-down list to select a field. After adding a field, you can click the settings icon next to the field to modify the sorting mode, which is "Ascending" by default.
Row quantity limit	Row quantity limit filters the number of statistical results. After it is set, a certain number of statistical results will be displayed in reverse order. The valid range is 1–1,000, and the default value is `1000`.

Custom Chart: You can customize a search statement to create charts. This mode supports all statistical and analytical features.

Enter the search statement to automatically generate a chart.



5. Check whether the data configuration meets the requirements. The system will automatically fill in data based on the chart data structure. If the data type doesn't match, you need to **convert the data** manually and adjust the configuration. For more information, see Data conversion.


34091
Mozilla/5.0 (c...
23275
Mozilla/5.0 (c...
5800
Mozilla/5.0 (W...
2360
Go-http-client...
1924
Mozilla/5.0 (M...
1755
Mozilla/5.0 (W...
1524
Mozilla/5.0 zg...
1376
Mozilla/5.0 (W...
1040
Mozilla/5.0 (X...
862
0
10,000
20,000
30,000
40,000
Search Statement
Data Configuration
Manual Configuration
Apply to...
Field Name
Attribute
UserAgent
Common Di
PV
Metric
Merge by Dimension
Do not merge
Chart Type
Table
Time Series
Bar Chart
Stat
Pie Chart
Sankey
Map
Basic Info
Chart Name
TOP10访问UserAgent
Bar Chart
Direction
Horizontal
Sort By
DESC
Counts
Show
Bar Type
Clustered
标准配置
Unit
none
Decimal Places
auto

6. Click **Save**.

Table

Last updated：2024-01-20 17:31:30

A table is the most common type of data display, where data is organized for comparison and counting. It is suitable for most scenarios.

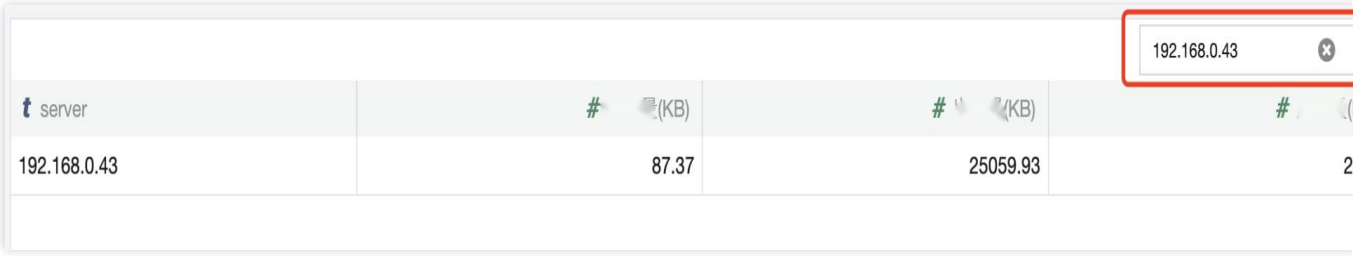
Chart Configuration

Configuration Item	Description
Basic information	Chart name: Set the display name of the table, which can be left empty.
Table	Alignment: Set the alignment mode of table content in the cells. By default, metric-type fields are aligned to the right, and dimension-type fields are aligned to the left.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Chart Operations

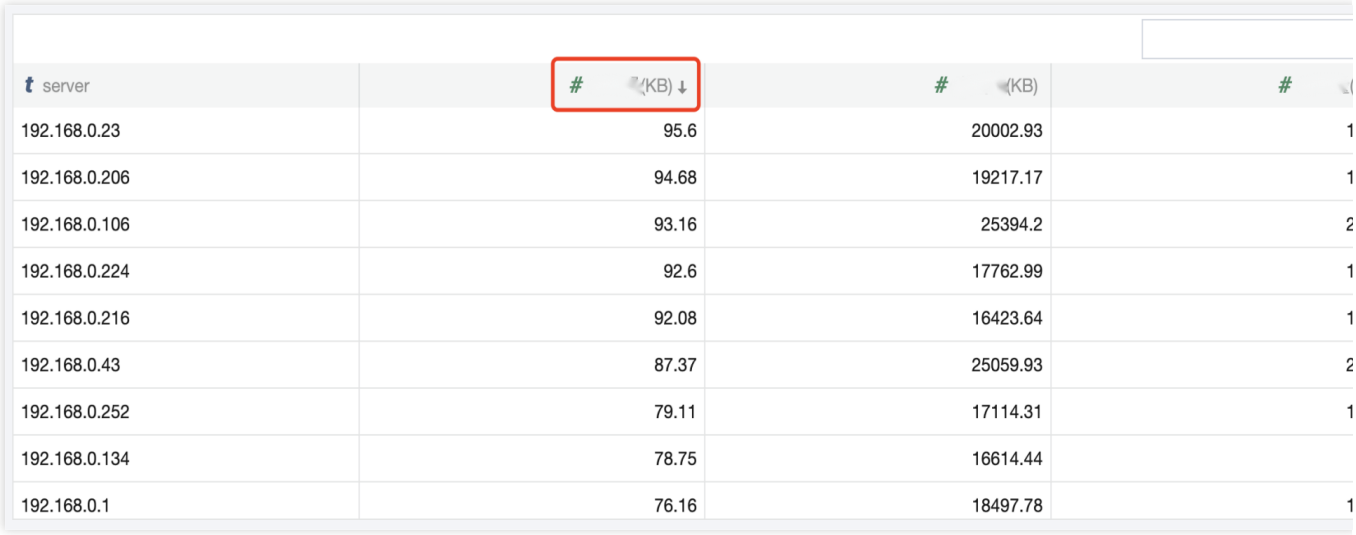
Searching for table content

Enter a keyword in the **search box** at the top of the table and click **Search**.



Sorting by column

Click the **header** of the target column to sort rows in the column in ascending order. Click it again to sort rows in descending order.



server	# (KB) ↓	# (KB)	#
192.168.0.23	95.6	20002.93	1
192.168.0.206	94.68	19217.17	1
192.168.0.106	93.16	25394.2	2
192.168.0.224	92.6	17762.99	1
192.168.0.216	92.08	16423.64	1
192.168.0.43	87.37	25059.93	2
192.168.0.252	79.11	17114.31	1
192.168.0.134	78.75	16614.44	
192.168.0.1	76.16	18497.78	1

Sequence Diagram

Last updated：2024-01-20 17:31:30

A sequence diagram requires statistics to have a time series field, so that it can organize and aggregate the metric in chronological order. It visually reflects the change trend of a metric over time. It is suitable for trend analysis scenarios, for example, analyzing the trend of the daily number of 404 errors in the past week.

Chart Configuration

General configuration

Configuration Item	Description
Chart Name	Set the display name of the table, which can be left empty.
Legend	Set the chart legends. You can control the legend styles and positions and add comparison data to legends.
Tooltip	Control the content style of the bubble tip displayed when the mouse is hovered over.
Unit	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Changes

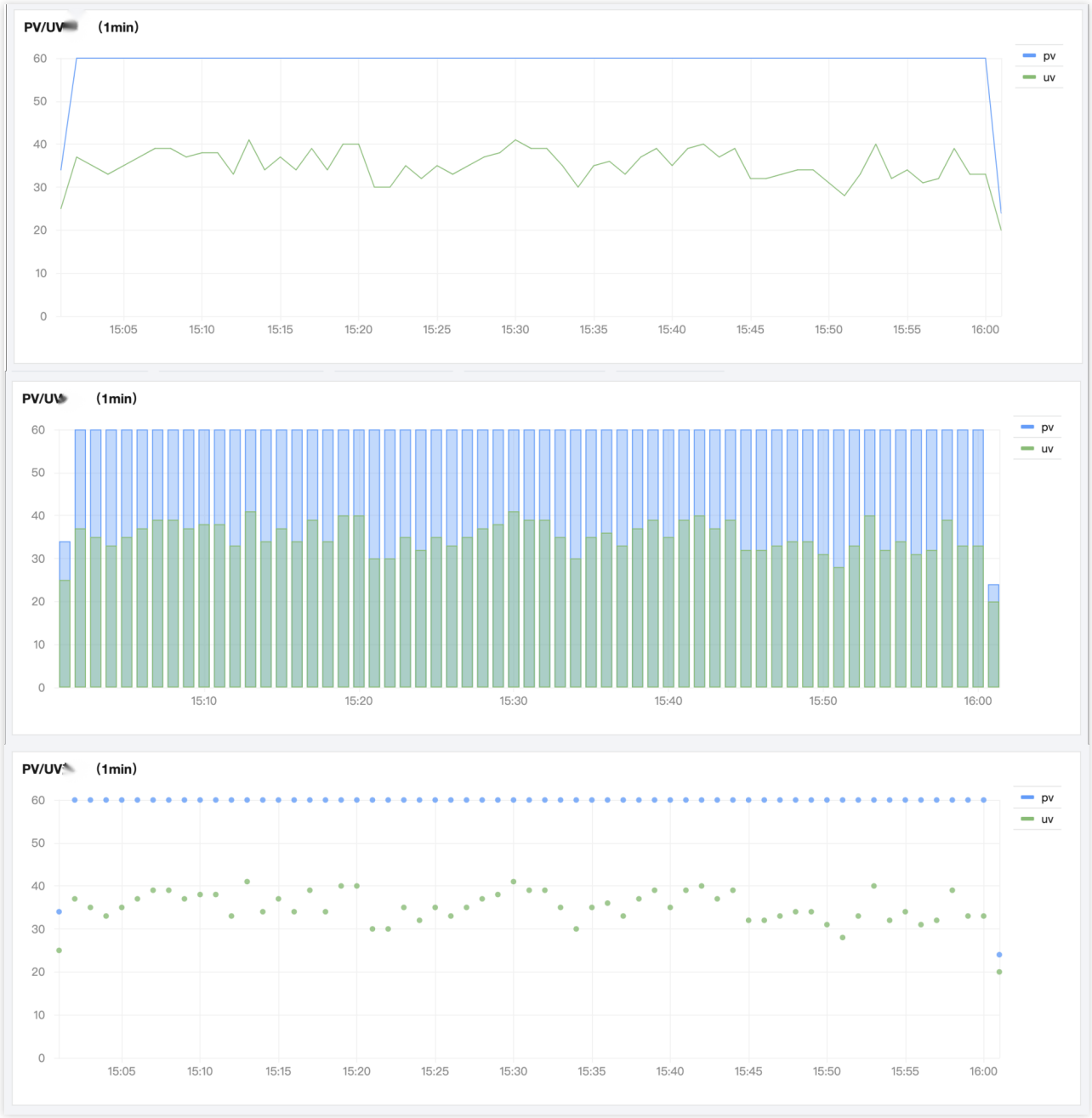
Configuration Item	Description
Changes	After the changes feature is enabled, you can compare the data in a time period with the data in the same period X hours, days, months, or years ago. The comparison data is displayed as dotted lines in the chart.

Sequence diagram configuration

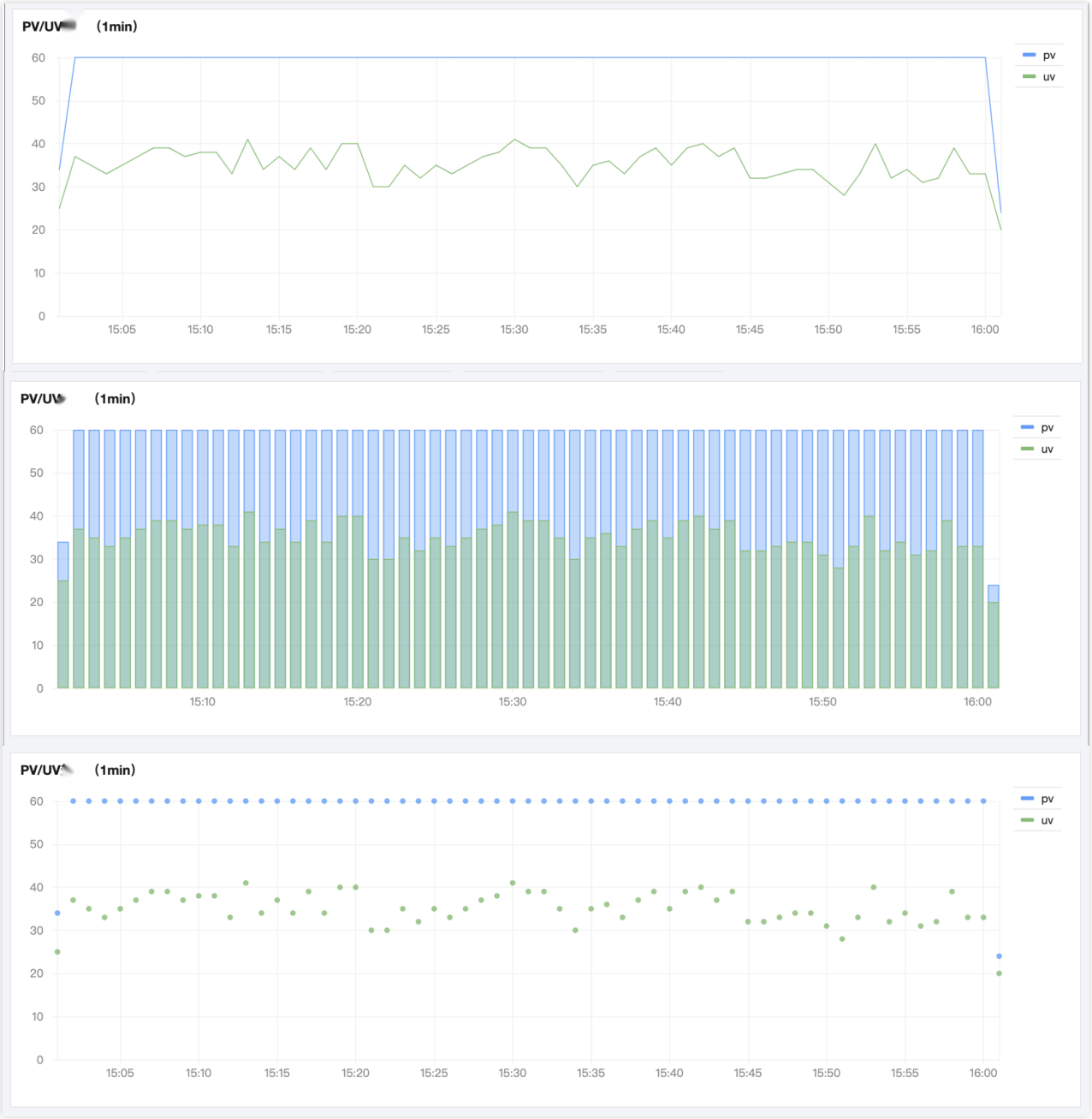
Configuration Item	Description
Sequence diagram	Drawing Style: Set the display style of data on coordinate axes. If you select a line, column, or dot, it will be a line chart, histogram, or scatter plot respectively. Linear: Set whether to smooth the connections between points. Line Width: Control the thickness of lines. Fill: Control the transparency of the fill area. If this value is 0, there will be no fill.

	Display Point: Display data points. If there is no data, no points will be displayed. Null: Control the processing of a sequence point if there is no data on the point. This value is 0 by default. Stack: Control whether to display data in a stack.
Axes	Show: Show/Hide axes. MAX/MIN: Control the maximum and minimum values displayed on coordinate axes. Coordinate areas greater than the maximum value or smaller than the minimum value will not be displayed.

Sample drawing style:



Sample fill:



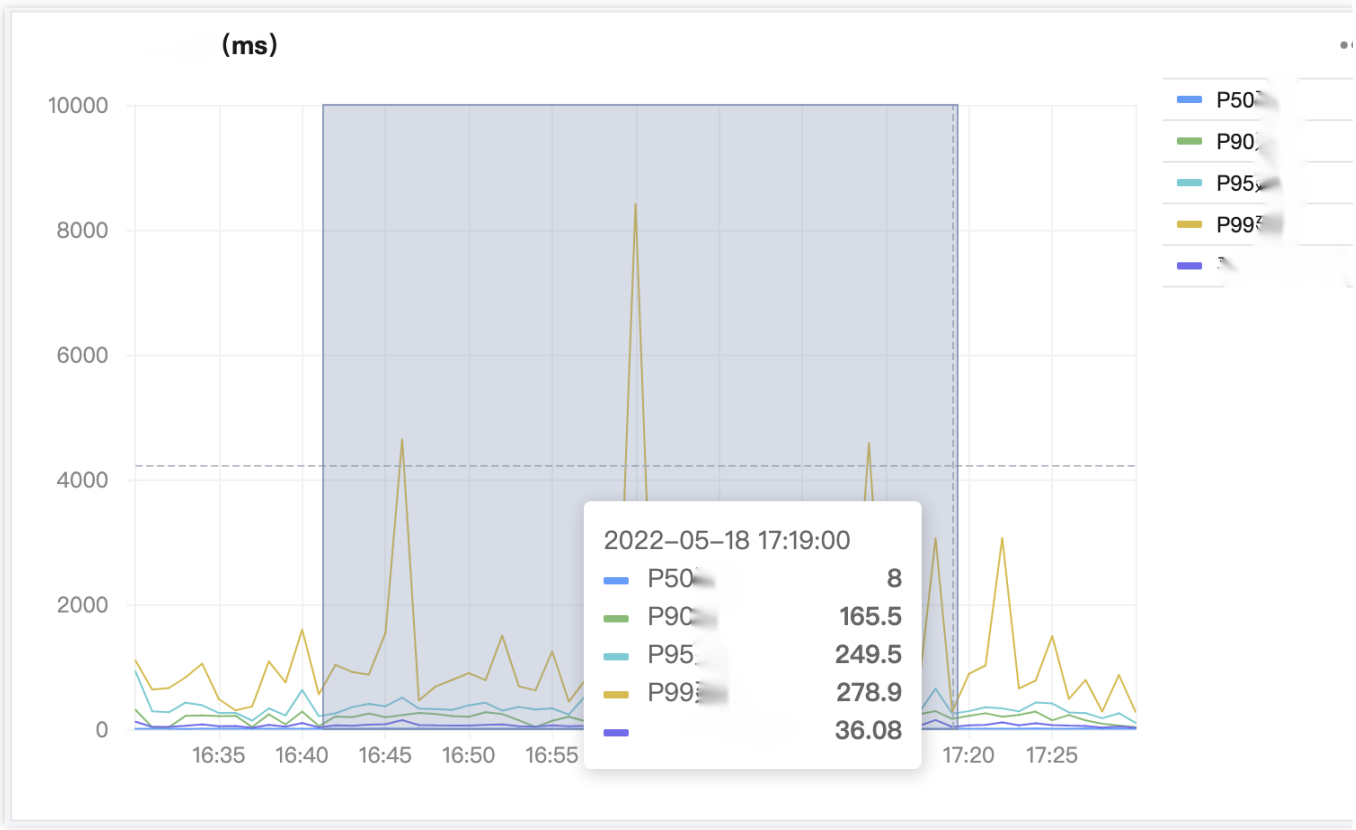
Sample null value:

Threshold configuration

Configuration Item	Description
Threshold configuration	Threshold Point: Set the threshold points. You can add multiple threshold intervals. You can click a threshold color to open the color picker to customize the color. Threshold Display: Control the style of threshold display, including three modes: threshold line, area filling, and both. If this option is disabled, no threshold will be used.

Chart Operations

Time range

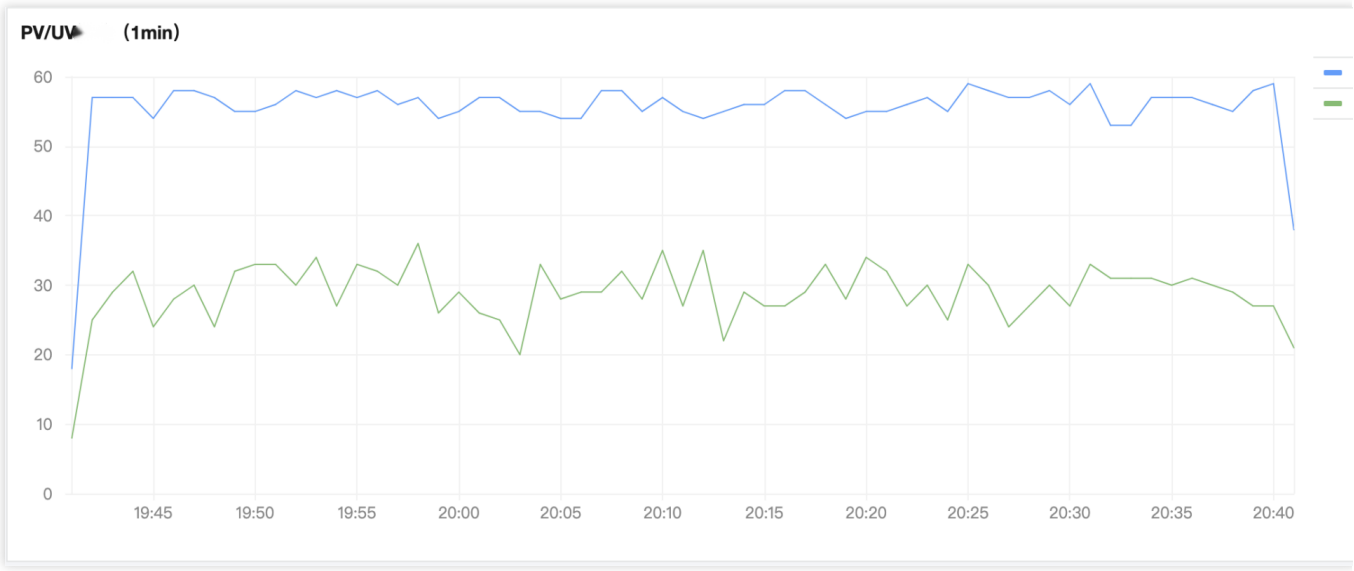


Hover over the chart, press and hold the left mouse button, and drag to trigger the selector and use time in the selected area as the time range. This is suitable for scenarios such as drilling down into the time range of abnormal points.

Use Cases

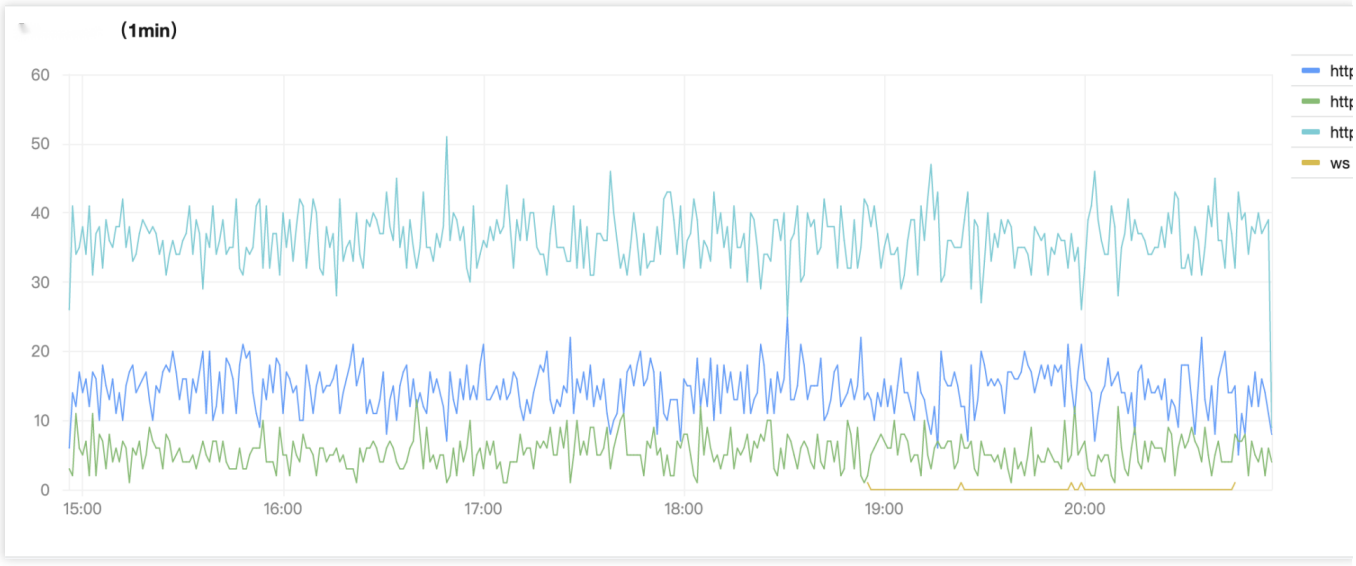
A sequence diagram requires fields of time type during creation and depends on various functions to process time fields during use. For more time functions for sequence diagram, see Time and Date Functions.
Calculate the PV and UV per minute:

```
* | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as time, count(*) as pv,count( distinct remote_addr) as uv group by time order by time desc limit 10000
```



Calculate the PV for each protocol type per minute:

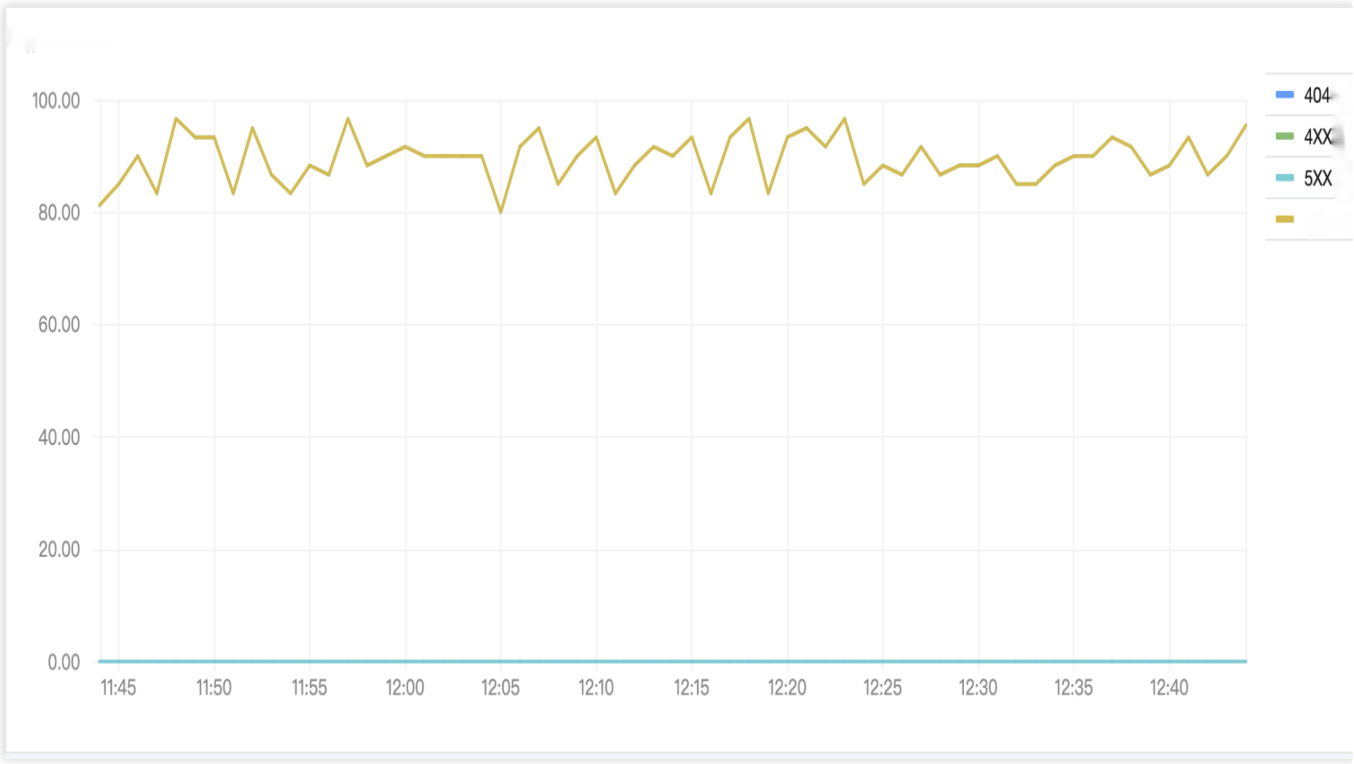
```
* | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as
time, protocol_type, count(*) as pv group by  time, protocol_type order by time
desc limit 10000
```



Calculate the request failure rate (%) per minute:

```
* | select date_trunc('minute', __TIMESTAMP__) as time, round(sum(case when
status = 404 then 1.00 else 0.00 end)/ cast(count(*) as double)*100,3) as "404
proportion", round(sum(case when status >= 500 then 1.00 else 0.00
end)/cast(count(*) as double)*100,3) as "5XX proportion", round(sum(case when
status >= 400 and status < 500 then 1.00 else 0.00 end)/cast(count(*) as
double)*100,3) as "4XX proportion", round(sum(case when status >= 400  then
```

```
1.00 else 0.00 end)/cast(count(*) as double)*100,3) as "total failure rate"
group by time order by time limit 10000
```



Bar Chart

Last updated：2024-01-20 17:31:30

A bar chart describes categorical data. It visually reflects the comparison of each category in size. It is suitable for category statistics scenarios, for example, collecting the numbers of each type of error codes in the last 24 hours.

Chart Configuration

General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Bar chart configuration

Configuration Item	Description
Bar chart	Direction: Control the bar/column direction. A bar chart is horizontal, while a column chart is vertical. Sort By: Control the bar/column sorting order, which can be ascending or descending by metric. If there are multiple metrics, you need to select one for sorting. Sorting is disabled by default. Display Value: Control whether to display the value label of each bar/column. Bar/Column Mode: Grouped and stacked display modes are supported.

Bar/Column mode example:

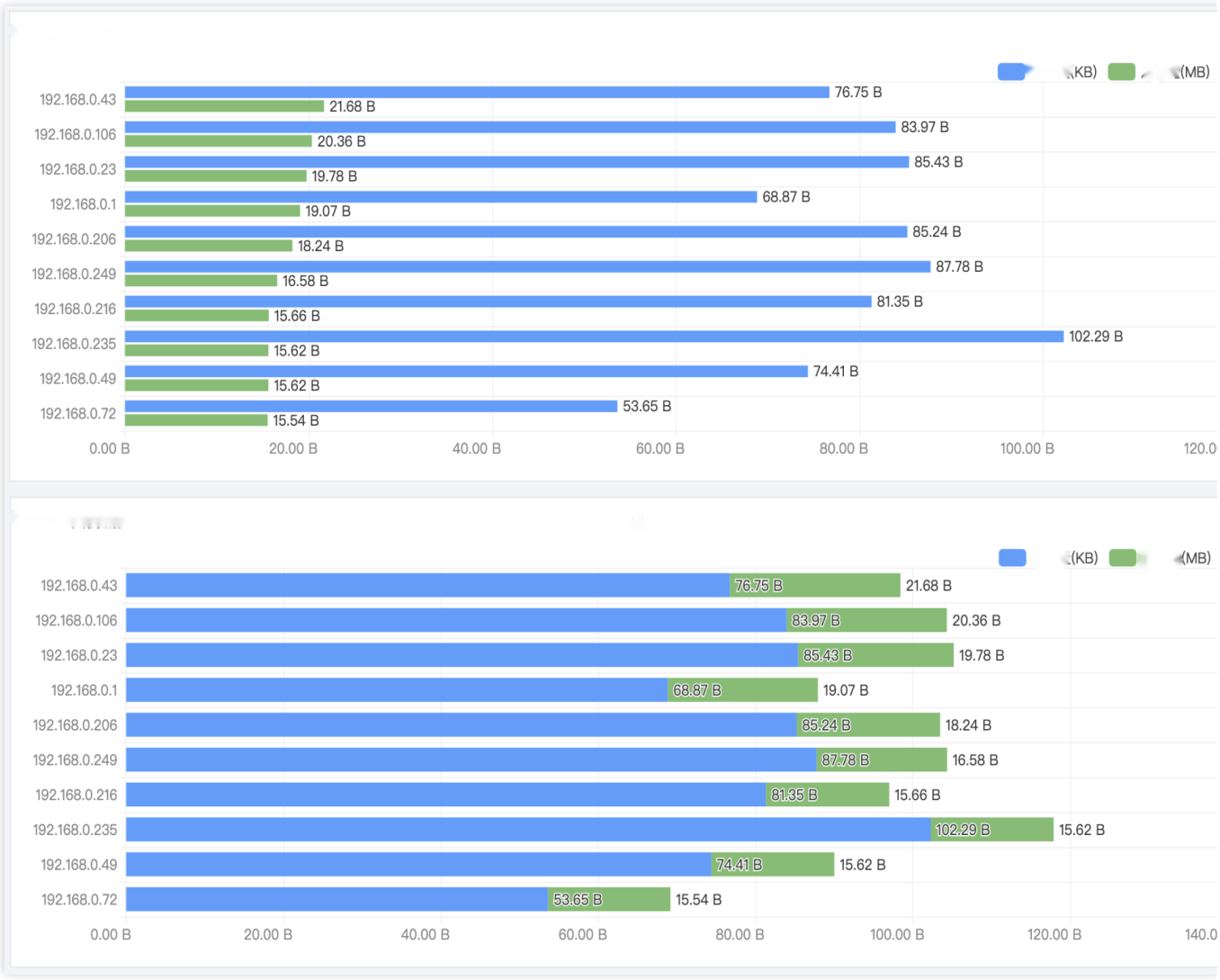
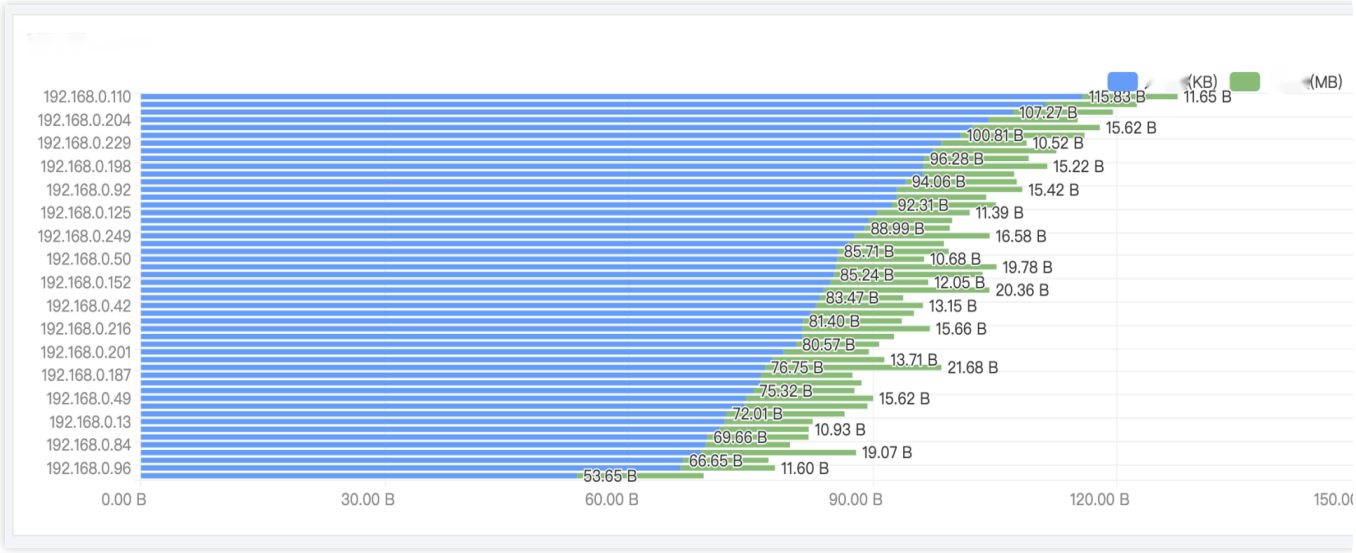


Chart Operations

Local zoom-out



If there are too many statistical results, the bars/columns will be too dense and labels will overlap each other as shown above, which will affect the analysis. In this case, you can hover over the chart and scroll the mouse wheel to zoom in/out the displayed area. This allows you to focus on the local content and display the complete information.

Pie Chart

Last updated：2024-01-20 17:31:30

A pie chart describes the proportions of different types. It measures the proportion of each type by the slice size. It is suitable for proportion statistics scenarios, for example, analyzing the proportions of different error codes.

Chart Configuration

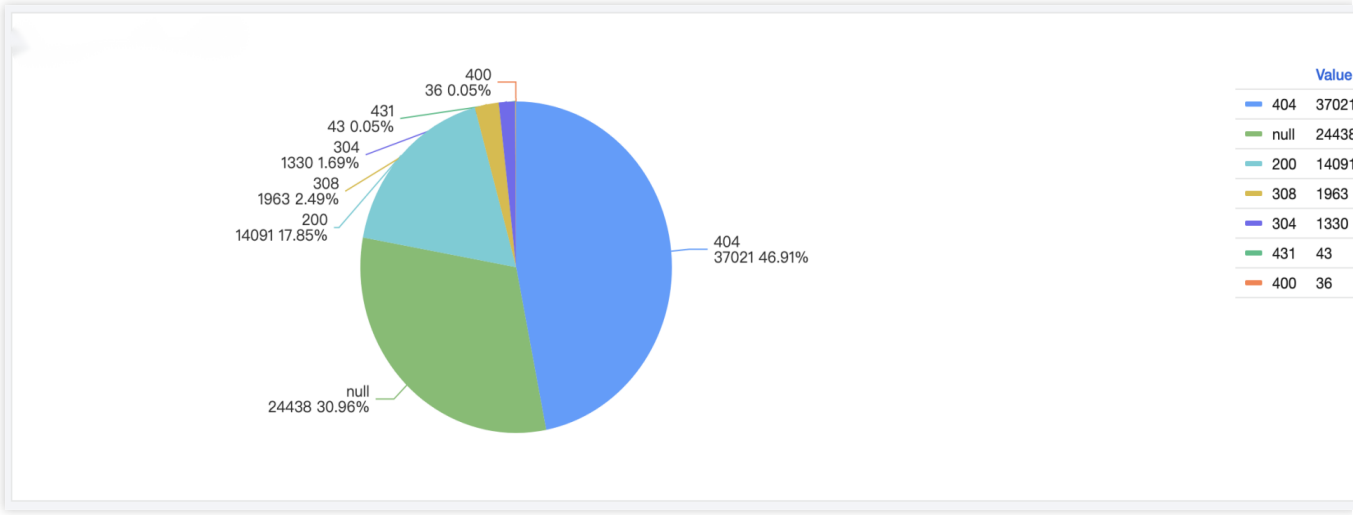
General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Legend	Set the chart legends. You can control the legend styles and positions and add comparison data to legends.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Pie chart configuration

Configuration Item	Description
Pie chart	Display Mode: Control the pie chart style. A solid chart is a pie chart, and a hollow chart is a donut chart. Sort By: Control the slice sorting order, which can be ascending and descending. Sorting is disabled by default. Merge Slices: Merge slices other than top N slices into the "Others" slice. If there are too many slices, you can use this feature to focus on top N slices. Label: Display pie chart labels. You can set name, value, and/or percentage as tags.

Label examples:


400
36 0.05%
431
43 0.05%
304
1330 1.69%
308
1963 2.49%
200
14091 17.85%
404
37021 46.91%
null
24438 30.96%
Value
404 37021
null 24438
200 14091
308 1963
304 1330
431 43
400 36

Individual Value Plot

Last updated：2024-01-20 17:31:30

An individual value plot describes a single metric, typically a key metric of business value. It is suitable for collecting daily, weekly, or monthly metrics such as PV and UV.

Chart Configuration

General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Changes

Configuration Item	Description
Changes	After the changes feature is enabled, you can compare the data in a time period with the data in the same period X hours, days, months, or years ago. You can choose to compare absolute values or percentages.

Individual value plot configuration

Configuration Item	Description
Individual value plot	Display: Control whether to display metric names on the individual value plot. Value: If a metric has multiple statistical results, they need to be aggregated to one value or one of them needs to be selected for display on the individual value plot. By default, the latest non-null value will be used. Metric: Set the target statistical metric, which is **Auto** by default, in which case the first metric field in the returned data will be selected.

Statistical method example:

Three data entries are returned, and the latest non-null value will be selected by default, that is, the last value `383`

will be displayed. If you set **Value** to **Sum**, the sum of the three data entries will be displayed.

Threshold configuration

Configuration Item	Description
Threshold configuration	Threshold point: Set the threshold points. You can add multiple threshold intervals. You can click a threshold color to open the color picker to customize the color.

Gauge Chart

Last updated：2024-01-20 17:31:30

A gauge chart describes a single metric. Unlike an individual value plot, it is generally used with a threshold to measure the metric status. It is suitable for rating scenarios, such as system health monitoring.

Chart Configuration

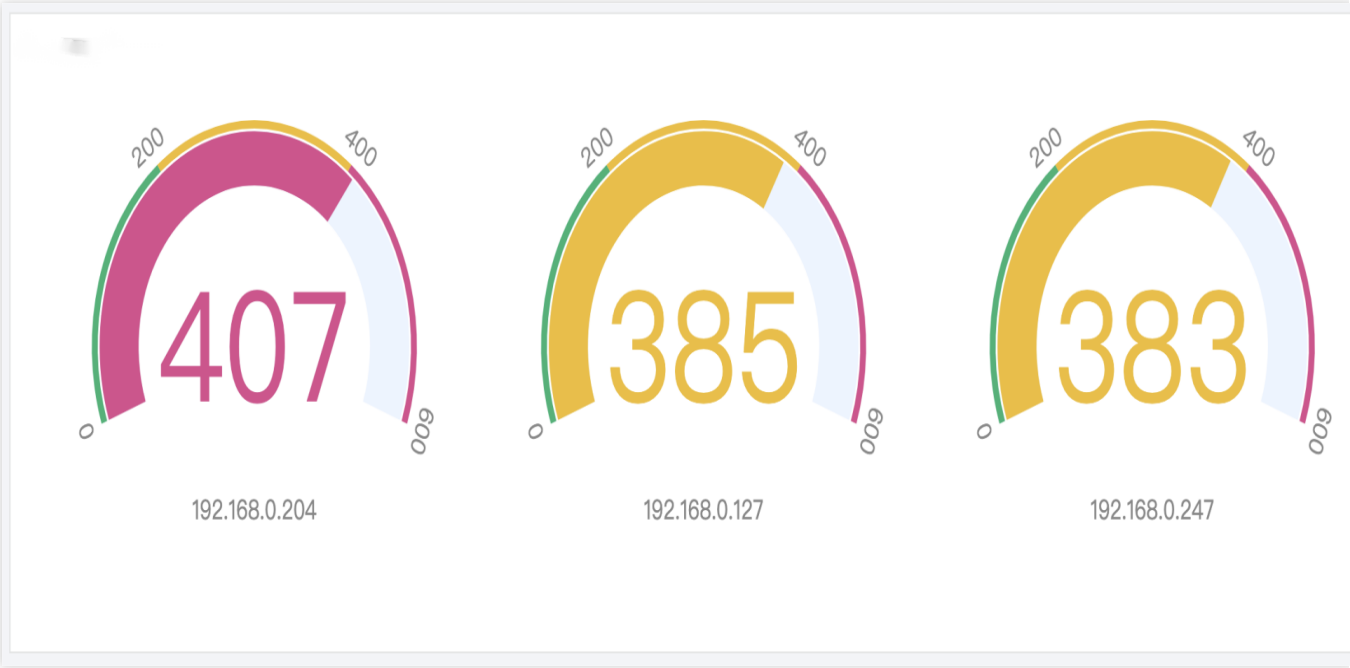
General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Threshold configuration

Configuration Item	Description
Threshold configuration	Threshold point: Set the threshold points. You can add multiple threshold intervals. You can click a threshold color to open the color picker to customize the color. MAX/MIN: Control the maximum and minimum values on the gauge. Data outside the range will not be displayed on the chart.

Threshold configuration example:


0
200
400
600
407
192.168.0.204
0
200
400
600
385
192.168.0.127
0
200
400
600
383
192.168.0.247

Map

Last updated：2024-01-20 17:31:30

A map shows the geographic location of data through the position of graphics. It is generally used to display the distribution of data in different geographic locations. It is suitable for geographic statistics scenarios, such as the geographic distribution of attacker IPs.

Chart Configuration

General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Map	Location: Set the region range displayed on the map. You can select China map or world map. It is set to **Auto** by default, in which case the map is automatically adapted to the region information contained in the data.
Legend	Set the chart legends. You can control the legend styles and positions and add comparison data to legends.

Chart Operations

Filtering regions by value

You can drag the value range in the bottom-left corner of the map to display regions in the specified value range.

Sankey diagram

Last updated：2024-01-20 17:31:30

A Sankey diagram is a special type of flow diagram used to describe the flow of one set of values to another set. It is suitable for directional statistics scenarios, such as firewall source and destination IP traffic.

Chart Configuration

General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Sankey diagram	Direction: Set the display direction of the Sankey diagram.

Sankey diagram direction examples:


192.168.0.72
192.168.0.206
192.168.0.216
192.168.0.55
192.168.0.1
192.168.0.23
192.168.0.249
192.168.0.134
192.168.0.252
192.168.0.7
113.108.77.72
113.108.77.60
113.108.77.71
113.108.77.51
113.108.77.65
111.206.145.52
113.57.28.111
113.108.77.61
192.168.0.72
192.168.0.206
192.168.0.216
192.168.0.55
192.168.0.1
192.168.0.23
192.168.0.249
192.168.0.134
192.168.0.252
192.168.0.7
113.108.77.72
113.108.77.60
113.108.77.71
113.108.77.51
113.108.77.65
111.206.145.52
113.57.28.111
113.108.77.61

Word Cloud

Last updated：2024-01-20 17:31:30

A word cloud is a visual representation of the frequency of words. It is suitable for audit statistics scenarios, such as high-frequency operations.

Chart Configuration

General configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Word cloud	Max Words: Control the maximum number of words to be displayed, i.e., top N words. Up to 100 words can be displayed. Font Size: Control the font size range of words in the word cloud.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Funnel Chart

Last updated：2024-01-20 17:31:31

A funnel chart is suitable for business processes with one single flow direction and path. It collects the statistics of each stage and uses a trapezoidal area to represent the business volume difference between two stages.

Chart Configuration

General configuration

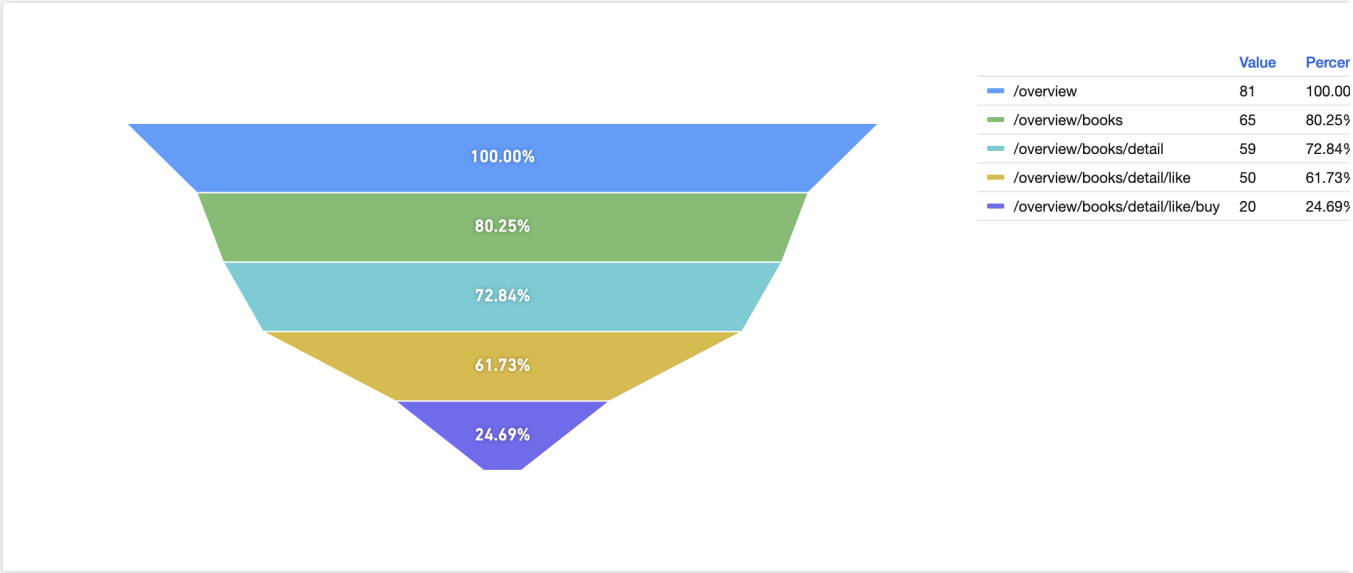
Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Legend	Set the chart legends. You can control the legend styles and positions and configure the data to be displayed as legends.
Standard configuration	Set the unit of all metric-type fields in the chart. For more information, see Unit Configuration.

Funnel chart configuration

Configuration Item	Description
Funnel chart	Display Value: Set the label display form of each stage in the funnel chart, which can be **Value** or **Conversion rate**. Max Rendered Stages: Set the number of rendered stages in the funnel chart, which can be up to 20. Conversion Rate: Set the calculation method of the conversion rate, which can be **Percentage of the first stage** or **Percentage of the previous stage**.

Sample funnel chart:

```
* | select url, count(*) as pv group by url limit 5
```


100.00%
80.25%
72.84%
61.73%
24.69%
Value
Percer
/overview
81
100.00
/overview/books
65
80.25%
/overview/books/detail
59
72.84%
/overview/books/detail/like
50
61.73%
/overview/books/detail/like/buy
20
24.69%

Log

Last updated：2024-01-20 17:31:30

Log charts allow you to save raw logs to the dashboard. You can quickly view the analysis result and the associated log content on the dashboard page, with no need to redirect to the search and analysis page.

Chart Configuration

Configuration Item	Description
Basic information	Chart Name: Set the display name of the table, which can be left empty.
Log	Layout: Select the original layout to display logs as text, or select the table layout to display logs as structured fields and field values in a table. Showed Field: Select the fields to be displayed. If this parameter is left empty, all fields will be displayed. Line Break: After it is enabled, log text in the original layout will be divided into lines by field, where each field occupies a line. Line No.: After it is enabled, the log number will be displayed. Log Time: After it is enabled, the log time will be displayed.

Chart Operations

Adding a log chart to the dashboard

Option 1: Select **Search and Analysis** > **Raw Data** and click **Add to Dashboard** in the top-right corner to add the current log to the dashboard.
Option 2: Click **Dashboard** > **Add Chart**, select the **Log** chart type on the dashboard chart edit page, and enter the search statement.

Loading more logs

A log chart displays the first 20 logs by default, and more logs will be loaded automatically when you scroll down the page.

Text

Last updated：2024-03-11 16:07:32

The text type chart supports MarkDown syntax, you can insert text, image links, hyperlinks, etc. on the dashboard page. This chart is only supported for use in the dashboard.

Directions

1. Log in to the CLS console, click **Dashboard > Dashboard List**, select the dashboard you want to view, click Add chart, create a new chart, and select the chart type as **Text**.
2. Edit the text according to the MarkDown syntax, the example is as follows:

Syntax	Note
`# Heading`	Heading 1
`## Heading`	Heading 2
`**Bold**`	Bold font
`*Italic*`	Italic font
`[Link](http://a.com)`	Hyperlink
`![Image](http://url/a.png)`	Image link
`> Blockquote`	Quote
`* List` `* List` `* List`	List

`Horizontal rule:`
 	
`---`
	Dividing line

`` `Inline code` ``	Inline code block
```` ``` # code block print '3 backticks or' ````	Code block

```print 'indent 4 spaces' ```	
 Microsoft Yahei, green, size=5	Styled text

Heatmap

Last updated：2024-12-25 12:02:21

Heatmaps display statistical charts by coloring the blocks. For statistical indicators, higher values are represented by darker colors and lower values are represented by lighter colors. Heatmaps are suitable for viewing the overall situation, detecting anomalies, displaying differences among multiple variables, and checking whether there is any correlation between them.

Directions

1. Log in to the CLS console, click **Dashboard > Dashboard List**, select the dashboard you want to view, click Add Chart, create a new chart, and choose **Heat Map** as the chart type.
2. Configure the chart according to needs.

Chart Configuration

General configuration

Configuration Item	Description
Basic Information	Chart Name: Set the display name of the table, which can be left empty.
Statistical Analysis Field	Define the chart fields and support manually specifying the fields corresponding to the chart content with default automatic adaptation. X-axis field: Select the field corresponding to the heatmap X-axis. Y-axis field: Select the field corresponding to the heatmap Y-axis. Metric field: Select the field corresponding to the heatmap block.
Axes	Show X-axis: Show/hide X-axis. Y-axis position: Configure the display position of the Y-axis.
Standard configuration	Set the field units for all metric types within the chart. For details, see Unit Configuration.
Visual mapping	Show/hide Visual Map Configuration.
Interaction event	Support clicking the numbers in the chart to trigger interactive events, such as jumping to the set URL address, opening the search analysis page, opening the dashboard page, adding

	filter conditions, etc. For details, see Interaction Event.

Heatmap configuration

Configuration Item	Description
Heat Map	Counts: Setting the display and hide of heatmap block value tags. The value is hidden by default.

Sample Statement

Statistics of average request delays from ISPs on the timestamp:

```
* | select histogram(__TIMESTAMP__,interval 1 hour) as time,isp,round(avg( "request
```

Statistics of URL access the province distribution of PV:

```
* | select  url,prov, count(*) as count group by url,prov order by count desc limit
```

Data conversion

Last updated：2024-01-20 17:31:30

Data conversion allows you to perform further processing of search results, including modifying data types, selecting fields for chart creation, and merging groups. This satisfies your chart creation needs without modifying SQL statements.

Enabling Data Conversion

1. Log in to the CLS console.
2. On the left sidebar, click **Search and Analysis** to go to the search and analysis page.
3. Select the **Chart Analysis** tab and set **Data Conversion** below to

.
4. On the **Dashboard Chart Editing** page, select the **Data Conversion** tab and set **Data Conversion** to

.

Use Cases

After executing a SQL analysis statement, you may want to visualize the results on different types of charts. As different charts require different attributes and numbers of fields, a problem will occur if the attributes and number of fields of the statement results are different from those required by the chart. In this case, you can modify the SQL statement to adapt it to the requirements of the chart. You can also configure data conversion to further process the results in order to meet the requirements for chart creation.

Using Data Conversion

Selecting fields for chart creation

After using a SQL statement to find the results in the above list, you can select fields from the field list to create a chart based on the selected fields. This operation is equivalent to `SELECT` . Below is the result after some fields are selected:

Converting the field type

In the SQL statistics results, each field has a default type. Fields of the `# (numeric)` type will be identified as metrics, fields of the `t (string)` type will be identified as common dimensions, and fields of the `time type` will be identified as time dimensions based on the chart type. Then, they will be matched with the field attributes required for chart creation. For example, in a sequence diagram, the time dimension field needs to be the X-axis, and the metric field needs to be the Y-axis.

The `time` field in the above figure is treated as a common dimension, which doesn't meet the requirements for field attributes of a sequence diagram. If you modify the `time` field attribute to the time type, you can see that the `time` field changes to the time format (this operation is equivalent to the `CAST` function). At this point, you can use a sequence diagram.

The `histogram` function is usually used to process fields, and the result can be in a non-standard time format. Therefore, if the default field attribute is common dimension, the sequence diagram cannot be used. You need to use the `CAST` function to convert the field to the time type or use data conversion to change the field attribute to the time dimension.

Merging groups

After using a SQL statement to find the above results, you can group them by `server_addr` and `server_name` and collect the PV and UV of each group. If you want to merge the results by `server_name` , you can hide the `server_addr` field and merge the results in the selected `server_name` dimension. This operation is equivalent to the `GROUP BY` function.

Unit Configuration

Last updated：2024-01-20 17:31:31

Automatic unit conversion is available in charts. When you select an original unit, the value is automatically converted to the next higher unit if it meets the conversion factor. Units can be configured to display decimal places.

Overview

The results of SQL aggregation are unitless values by default. In the following cases, you need to use unit conversion to make the data more readable.

Selecting a unit and automatically converting it

In CLS chart analysis, SQL aggregation results are unitless raw values by default, and you need to specify a basic unit, such as byte in the example. After a basic unit is selected, if a value is great enough to be converted to a value with a higher-level unit, it will be automatically converted. In this example, the value in bytes is automatically converted to a value in GiB.
In unit configuration, the precision decides the number of decimal places. By default, if `Auto` is selected, two decimal places will be retained. You can modify the precision as needed.

Manually entering a unit

If data has a special unit that cannot be found in the unit configuration options, you can manually add a custom unit with the `custom` prefix. However, note that the custom unit is fixed and cannot be automatically converted. Therefore, if you want to use the raw unit in a unified manner without triggering automatic unit conversion, you can manually add a fixed unit.

Filters and Variables

Last updated：2024-01-20 17:31:31

Type Description

Type	Description	Scope
Filter in the drop-down list	Filter data in all charts on the dashboard by specifying the field value. If statistics is enabled in the index configuration of the log topic for the filter field, the field value can be automatically obtained as a list item.	All charts on the dashboard
Filter by search statement	Filter data in all charts on the dashboard by entering a search statement, that is, add a filter in the query statement of the charts.Filter by search statement includes filter by range, NOT, and full text.	All charts on the dashboard
Data source variable	A data source variable enables batch switching data sources of the charts on the dashboard. It is applicable to scenarios such as applying a dashboard to multiple log topics and comparing data in blue and green on the dashboard.	Charts that use the variable on the dashboard
Custom variable	A custom variable can be set to a static input or a value from a dynamic query and applied to search statements, titles, and text charts for quick batch statement modification.	Charts that use the variable on the dashboard

Filter

Configuring the filter in the drop-down list

1. Log in to the CLS console.
2. On the left sidebar, click **Dashboard** to enter the dashboard management page.
3. Click the ID/name of the target dashboard to enter the dashboard details page.
4. Click **Add Filter and Variable** at the top to enter the settings page.
5. In the pop-up window, configure the filter in the drop-down list and click **OK**.

Form Element	Description
Type	Different types correspond to different configuration items and application scenarios. Here, select **Filter in the drop-down list**.
Filter alias	It is the filter name displayed on the UI, which is optional. If it is left empty, the filter field will be used automatically.

Log topic	It is the log topic to which the filter field belongs.
Filter field	It is the object field to be filtered.
Dynamic option	After it is enabled, the filter field value will be obtained automatically as the filter option.
Static option	A static option is optional, needs to be added manually, and will be always displayed. You can configure its alias.
Default filter	It is the default filter of the dashboard and is optional.
Support for multiple items	After it is enabled, multiple filters can be selected as the filter condition.

6. Go back to the dashboard details page, click the filter icon, and select the target filter. Then, the dashboard data will be refreshed to the filtered content.

Configuring filter by search statement

1. Log in to the CLS console.
2. On the left sidebar, click **Dashboard** to enter the dashboard management page.
3. Click the ID/name of the target dashboard to enter the dashboard details page.
4. Click **Add Filter and Variable** at the top to enter the settings page.
5. In the pop-up window, set the information of the filter by search statement and click **Submit**.

Form Element	Description
Type	Different types correspond to different configuration items and application scenarios. Here, select **Filter by search statement**.
Filter name	It is the unique filter name.
Filter alias	It is the filter name displayed on the UI, which is optional.
Log topic	It is the log topic to which the filter field belongs.
Mode	It is the mode for inputting search statements. Here, interactive and statement modes are supported.
Default filter	It is the default filter of the dashboard and is optional.

6. Go back to the dashboard details page, click the filter icon, and select the target filter. Then, the dashboard data will be refreshed to the filtered content. **Filter by search statement includes filter by range, NOT, and full text**.

Variable

Configuring the data source variable

1. Log in to the CLS console.
2. On the left sidebar, click **Dashboard** to enter the dashboard management page.
3. Click the ID/name of the target dashboard to enter the dashboard details page.
4. Click **Add Filter and Variable** at the top to enter the settings page.
5. In the pop-up window, configure the template variable information and click **Submit**.

Form Element	Description
Variable type	It is the variable type. Different types correspond to different configuration items and application scenarios. Here, select **Data source variable**.
Variable name	It is the name of the variable in the search statement and can contain only letters and digits.
Displayed name	It is the variable name displayed on the dashboard, which is optional. If it is empty, the variable name will be used automatically.
Data source scope	It is the optional scope of the variable value and defaults to **All Log Topics**. You can select **Custom Filter** and set a filter to view only log topics that meet the condition.
Default log topic	It is the default log topic.

6. Go back to the dashboard details page and click **More** > **Edit Chart**.

Note:

If there are no charts on your dashboard, add charts.

7. On the **Search Statement** tab of **Edit Chart**, click **Log Topic**, select **Use Data Source Variable**, and select the created template variable.
8. Click **Save**.
9. On the dashboard details page, click the **Data Source** drop-down list and select another log topic. Then the system changes automatically changes the data source of the chart using the variable accordingly.

Configuring the custom variable

1. Log in to the CLS console.
2. On the left sidebar, click **Dashboard** to enter the dashboard management page.
3. Click the ID/name of the target dashboard to enter the dashboard details page.
4. Click **Add Filter and Variable** at the top to enter the settings page.
5. In the pop-up window, set the information of the custom variable and click **Submit**.

Form Element	Description
Type	It is the variable type. Different types correspond to different configuration items and application scenarios. Here, select **Data source variable**.
Variable name	It is the name of the variable in the search statement and can contain only letters and digits. A variable is referenced in the format of ${Variable name}.
Variable alias	It is the variable name displayed on the dashboard, which is optional. If it is empty, the variable name will be used automatically.
Static variable value	A static variable value needs to be added manually and will be always displayed. You can configure its alias.
Dynamic variable value	After it is enabled, you can select a log topic, enter a search and analysis statement, and use the search and analysis result as the optional variable value.
Default value	The default value is the variable value and is required.

6. Go back to the dashboard details page and click **More** > **Edit Chart**.

Note:

If there are no charts on your dashboard, add charts.

7. On the **Search Statement** tab of **Edit Chart**, insert the created custom variable **${interval}** to replace the original text.

8. Click **Apply to Dashboard**.

9. Go back to the dashboard details page, click the **Time Granularity** drop-down list at the top, and change the time granularity. Then, you can see that the charts inserted with the variable change accordingly.

FAQs

Why does a configured data source variable not take effect or take effect only on some charts?

A data source variable does not apply directly to all charts on the dashboard. It applies only to those charts that use it on the chart editing page.

Examples

Analyzing the performance metrics of different application APIs on the dashboard (filter in the drop-down list)

Requirement

Log topic A stores NGINX access logs of an application, and you are to use the dashboard to view the throughput, number of error requests, and response time of the **entire application** and **a specified API**. The following is the

sample log information:

```
body_bytes_sent:1344
client_ip:127.0.0.1
host:www.example.com
http_method:POST
http_referer:www.example.com
http_user_agent:Mozilla/5.0
proxy_upstream_name:proxy_upstream_name_4
remote_user:example
req_id:5EC4EE87A478DA3436A79550
request_length:13506
request_time:1
http_status:201
time:27/Oct/2021:03:25:24
upstream_addr:219.147.70.216
upstream_response_length:406
upstream_response_time:18
upstream_status:200
interface:proxy/upstream/example/1
```

Solution

1. Create a dashboard.

2. Create three charts (sequence diagrams) based on the application performance metrics. The corresponding query statements are as follows:

Throughput

```
* | select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as analyt
```

Number of error requests

```
 http_status:>=400 | select histogram( cast(TIMESTAMP as timestamp),interval 1
minute) as analytic_time, count(*) as pv_lost group by analytic_time order by
analytic_time limit 1000
```

Average response time

```
* | select histogram( cast(TIMESTAMP as timestamp),interval 1 minute) as analytic_t
```

3. Add a filter in the drop-down list.

Type: Filter in the drop-down list

Display Name: API name

Log Topic: Log topic A

Select field: interface

4. Go back to the dashboard details page, and you can see this variable on the top of the page.
If **API Name** is not specified, data is not filtered, and the charts on the dashboard display all data, that is, the overall performance metrics of the application.
If **API Name** is specified, all charts on the dashboard use the specified API as the filter condition to filter data and display the performance metrics of the API.

Viewing the production and test environment performance metrics on the dashboard separately (data source variable)

Requirement

An application has a production environment and a test environment, and logs are collected to **Log topic A (production environment)** and **Log topic B (test environment)**. Therefore, during application development, testing, and Ops, you need to pay attention to the performance metrics of the two environments.

Solution

1. Create a dashboard.
2. Add variables.
Variable Type: Data Source
Variable Name: env
Display Name: Application Environment
Data Source Scope: All log topics
Default Log Topic: Log topic A (production environment)
3. Add charts.
In the **Log Topic** drop-down list, select **Use Data Source Variable** and select the `${env}` variable created in the previous step. Then, charts will use the value of the variable as the data source, that is, **Log topic A (production environment)**.
4. Repeat **Step 3** to add other charts.
5. Go back to the dashboard details page, click the data source variable **Application Environment** at the top of the page, and switch the log topic in the drop-down list of the variable. Then, the charts that use the variable will switch the log topic accordingly.

Interaction Event

Last updated：2024-03-11 16:04:40

The chart has its interaction event feature, which supports triggering interactive events by clicking a data point in the chart, such as redirecting to a set URL, opening the Search and Analysis page, opening the dashboard, adding filter conditions, etc.

Directions

1. Log in to the CLS console.
2. On the left navigation bar, enter **Search and Analysis** or **Dashboard**, and create statistical analysis charts.
3. In chart configuration, select **Interaction Event** and add relevant interactions. The following variables can be referenced:

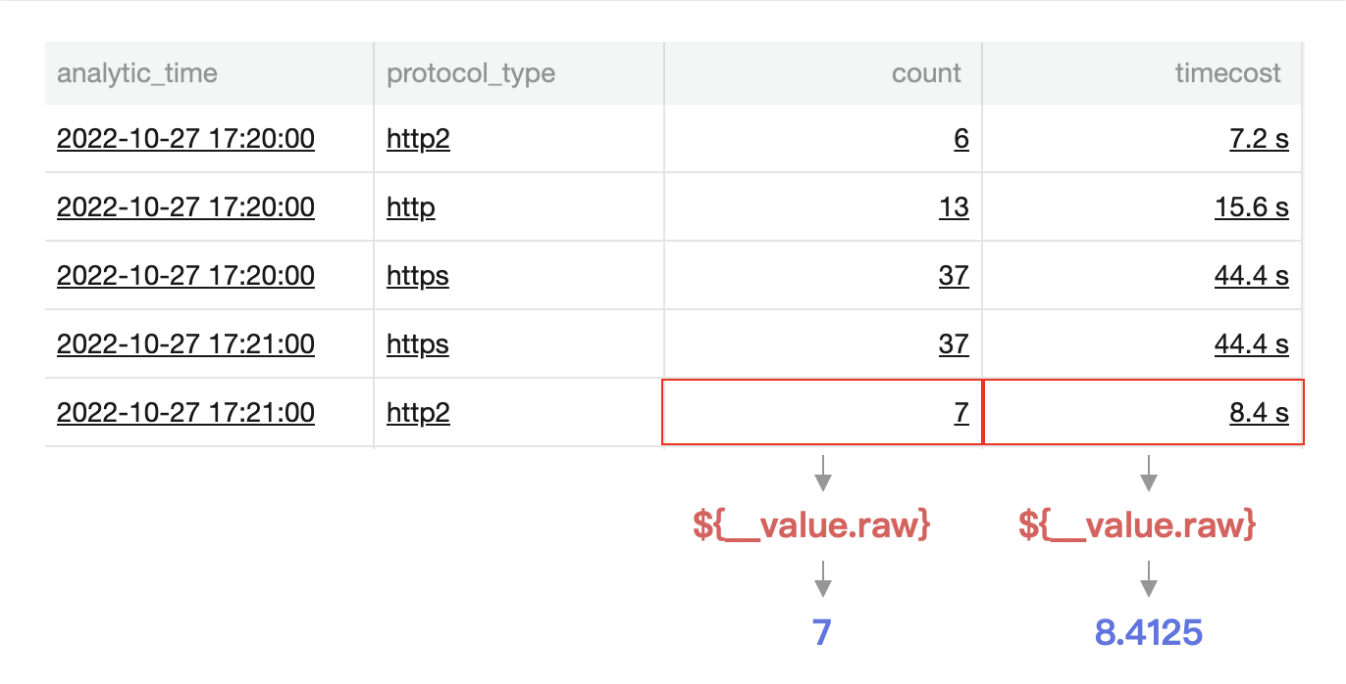
Variable Description

`${__field.Name}` : References the field name of the clicked value.

As shown below, clicking **8.4s** triggers the redirect URL embedded with the **${__field.Name}** variable, which will reference the field name of the value, that is, populate **timecost** in the URL.

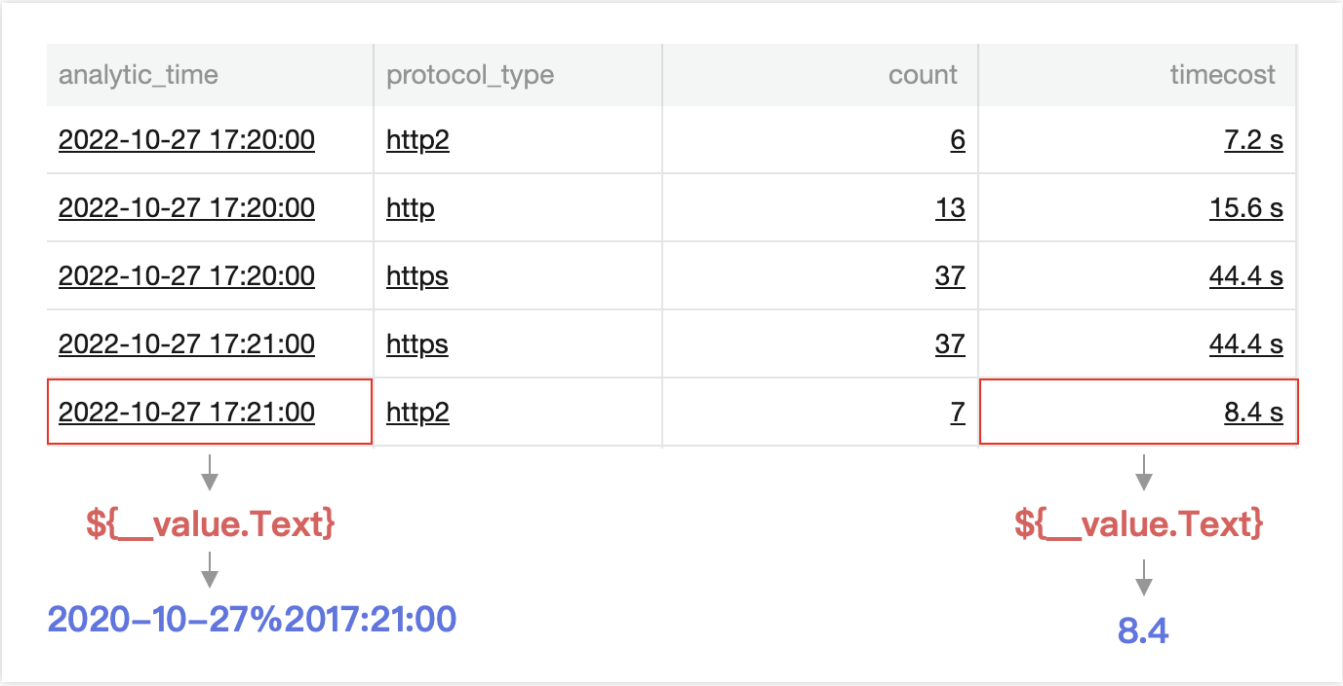
`${__value.raw}` : References the clicked value (populated in the original format).

As shown below, clicking **8.4s** triggers the redirect URL embedded with the **${__value.raw}** variable, which will reference the raw data of the clicked value, that is, the value **8.4125** without unit or decimal place processing.



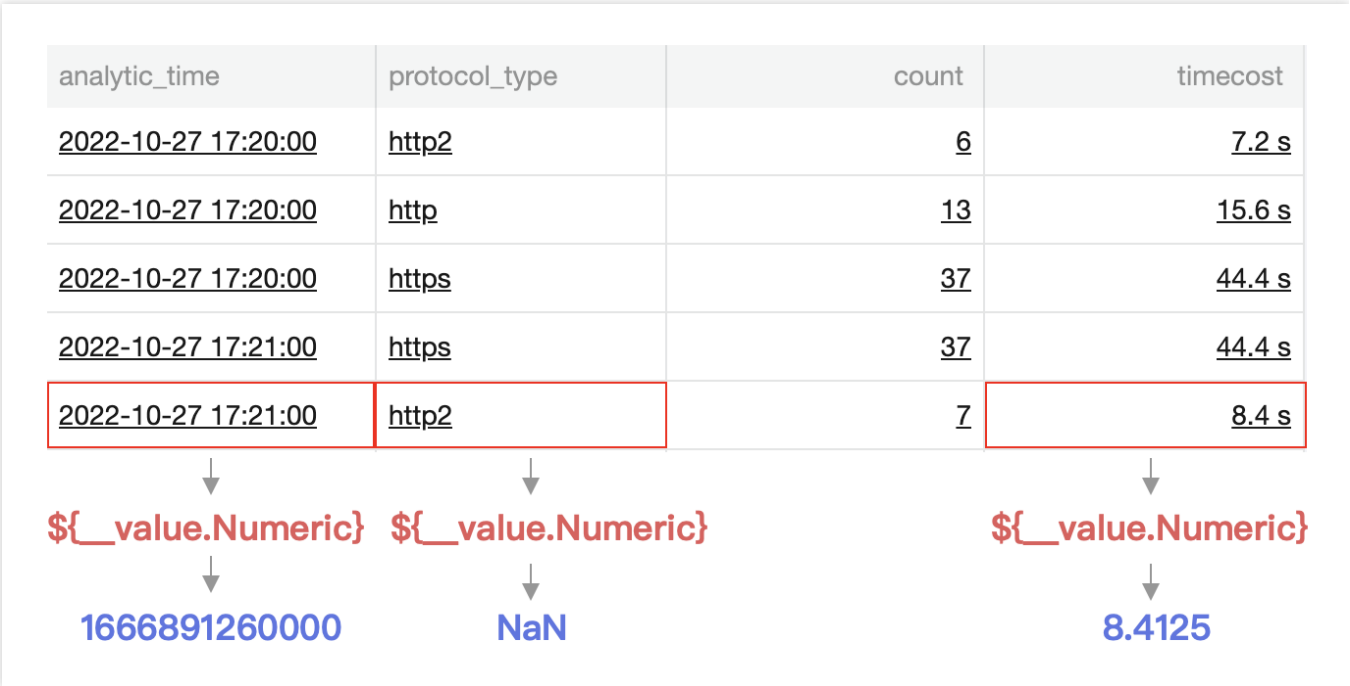
`${__value.Text}` : References the clicked value (populated in the string format).

As shown below, clicking **2020-10-27 17:21:00** triggers the redirect URL embedded with the **${__value.Text}** variable, which will reference the clicked value and convert it into a string, that is, **2020-10-27%2017:21:00** (here, `%20` is a URL-encoded space).



`${__value.Numeric}` : References the clicked value (populated in the numeric format).

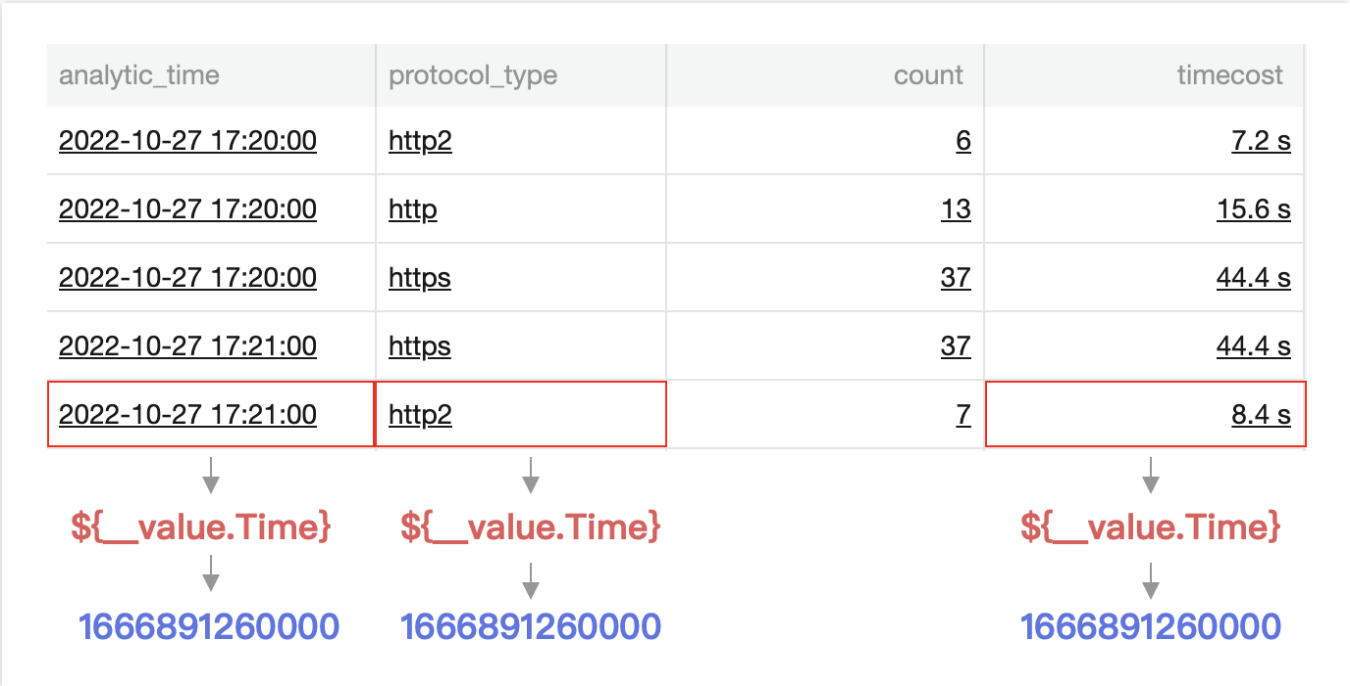
As shown below, clicking **8.4s** triggers the redirect URL embedded with the **${__value.Numeric}** variable, which will reference the clicked value and convert it into a number, that is **8.4125**. Here, a time value will be converted into a Unix timestamp in the numeric format, and a string value will fail to be referenced.



`${__value.Time}` : The timestamp of the clicked value (populated in the Unix time format).

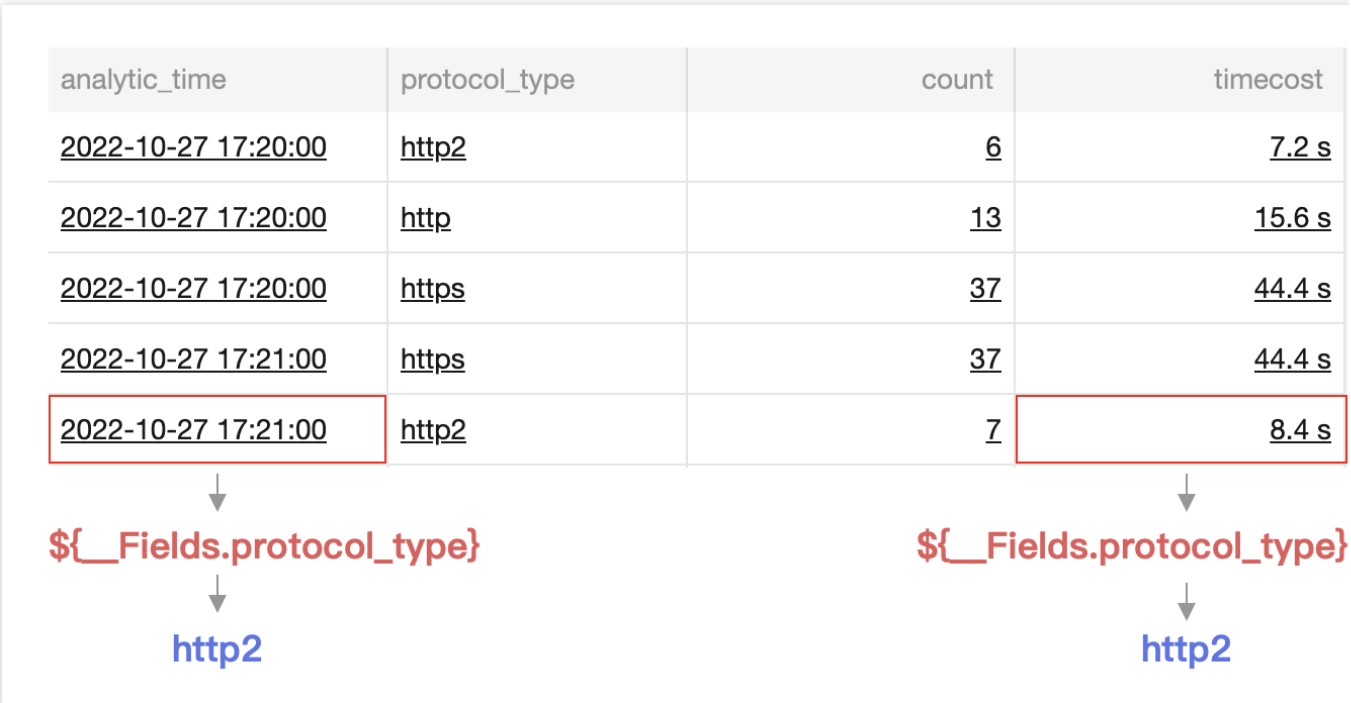
As shown below, clicking **8.4s** triggers the redirect URL embedded with the **${__value.Time}** variable, which will reference the timestamp in the same line as the clicked value, that is, **2022-10-27 17:21:00** of `analytic_time` .

The value will be further converted into the Unix format and populated as **1666891260000**. If there is no such timestamp, reference will fail.



`${__Fields.specific field}` : Field value in the same line.

As shown below, clicking **8.4s** triggers the redirect URL embedded with the **${__Fields.protocol_type}** variable, which will reference the field value in the same line as the clicked value, that is, **http2** of `protocol_type` .



Use Case

Scenario: The following counts the top error rates of server IPs. In case of an IP with a high error rate, you need to be redirected to the search and analysis page, where you can view the IP logs with the 4XX status code.

```
https://console.intl.cloud.tencent.com/cls/search?
region=xxxxxxx&topic_id=xxxxxxxx&query=server_addr:${__value.text} AND status:
[400 TO 499]&time=now-1h,now
```

After the target IP is clicked, the search page will be opened automatically for search and display of the search result.

Chart Time Configuration

Last updated：2024-03-11 16:04:05

Overview

Dashboard charts support independent chart time configurations. In contrast to chart time is the dashboard time which is used by default when a dashboard chart is created, its data range is governed by the dashboard time. Changing the dashboard time will alter the time range of all affiliated charts. If the use of dashboard time is ceased, the chart time independently controls itself, it no longer changes with the fluctuations of the global time and the charts in the dashboard can possess different time ranges, thereby supporting a more diverse array of comparison scenarios.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **View Dashboard** to enter the dashboard page.
3. Click **Edit Chart** or **Add Chart**, and select **Custom Chart** to enter the chart editing page.
4. By default, the chart uses the dashboard's global time. Turning off the **Using Global Time** switch will support an independent chart time.
5. View the chart. You can see that the chart uses a different time from the dashboard.

Preset Dashboard

Last updated：2024-01-20 17:31:30

Overview

CLS supports accessing the standard logs of multiple Tencent Cloud products and provides out-of-the-box dashboards for quick log analysis. For more information, see Tencent Cloud Service Log Access.
In addition, CLS provides free demos for you to try out preset dashboards.

List of Preset Dashboards

Tencent Cloud Product	Preset Dashboard
CLB	CLB access log dashboard
NGINX	NGINX access dashboard NGINX monitoring dashboard
CDN	CDN access log - quality monitoring and analysis dashboard CDN access log - user behavior analysis dashboard
COS	COS access log analysis dashboard
FL	ENI flow log - advanced analysis dashboard CCN flow log - advanced analysis dashboard
TKE	TKE audit log - overview dashboard TKE audit log - node operation dashboard TKE audit log - Kubernetes object operation overview dashboard TKE event log - overview dashboard TKE event log - abnormal event aggregation search dashboard

Directions

Viewing a preset dashboard

1. Log in to the CLS console.
2. Select **Dashboard** on the left sidebar to enter the dashboard list page.
3. On the dashboard list page, expand **Preset Dashboards** and select the target dashboard.

4. Select the log topic of the target Tencent Cloud product.

Editing a preset dashboard

If the current preset dashboards cannot meet your needs, you can add statistical charts based on them. Click **Copy** to copy a preset dashboard to the list of custom dashboards for customization.

Subscribing to Dashboard

Last updated：2024-01-20 17:31:30

Overview

CLS allows you to subscribe to a dashboard and export it as an image. Daily, weekly, and monthly reports can be emailed to specified recipients regularly.

Usage Limits

Each account can have up to 100 subscription tasks.

Directions

Creating a subscription task

1. Log in to the CLS console.
2. On the left sidebar, click **View Dashboard** to enter the dashboard page.
3. Click **Subscription** at the top and select **Create Subscription Task** to enter the subscription configuration page.
4. Enter the **Task Name** and select the **Time Range** of the subscribed dashboard.
5. Set the **Variables** of the subscribed dashboard.
If you select **Default Configuration**, the default configuration items will be used for the variables of the subscribed dashboard. When the default variables change, the subscribed dashboard will also change.
If you select **Custom Configuration**, you can customize variables to render the subscribed dashboard.
6. Click **Preview Content** to preview the subscribed dashboard.
7. Select the subscription period, which can be by day, week, or month. Multiple weeks and months can be selected.
8. Configure the subscription method:

Type	Description
Tencent Cloud user	Select a Tencent Cloud user as the email recipient of the subscribed dashboard. Users with no email address configured cannot receive email notifications.
Custom email address	Enter one or multiple custom email addresses.

9. Click **Send Immediately**, and CLS will send a subscription email to all the configured recipients.

Managing a subscription task

1. Log in to the CLS console.
2. On the left sidebar, click **View Dashboard** to enter the dashboard page.
3. Click **Subscription** at the top and select **Manage Subscription Task** to enter the subscription task management page.

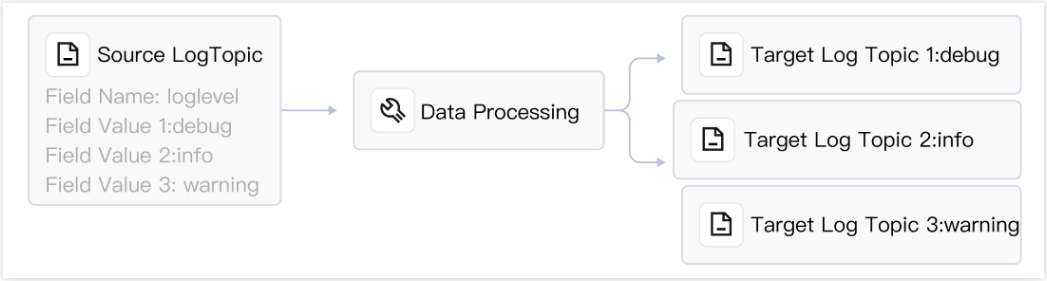
You can create, edit, or delete dashboard subscription tasks and view the last operation.

Data Processing documents
Data Processing
Data Processing Overview

Last updated：2024-12-20 16:13:10

Data processing
provides capabilities for log data filtering, cleansing, data masking, enrichment, and distribution.
According to the position of data processing in the data pipeline, and the different sources and sinks, the following data processing scenarios are currently supported:

Scenario	Description
	**Log Collection Processing Log Topic:** Logs are collected to CLS, processed (filtered, structured), and then written to the log topic. As shown, data processing occurs before the log topic the data pipeline, referred to a preprocessi of data. Performing l filtering in preprocessi can effective reduce log write traffic,

	index traffic, index storag and log storage. Performing l structuring i preprocessi with key-val indexing enabled, allows for S analysis of logs, dashboard configuratio and alarms.
	**Log Topic Processing Fixed Log Topic:** Store the da from the source log topic into a l topic after processing, distribute lo to multiple lo topics.
	**Log Topic Processing Dynamic L Topic:** Based on th field value o the source lo topic, dynamically create log topics and distribute related logs the correspondi

	log topics. F example, if there is a fie named Service in th source log topic with values like "Mysql", "Nginx", "LB etc., CLS ca automaticall create log topics name Mysql, Ngin LB, etc., and write relatec logs into the topics.

Basic Concepts

For terms and concepts related to data processing tasks, see Basic Concepts.

Features

Extract structured data to facilitate subsequent BI analysis, generation of monitoring charts (dashboard), etc. If your raw logs are not structured data, there is no way to perform SQL calculations, which means that OLAP analysis and CLS dashboard (drawing charts based on SQL results) cannot be performed on the logs. Therefore, it is recommended that you use data processing to convert unstructured data into structured data. If your logs are regular, you can also extract structured data at the time of log collection. See Full Regular Expression Format (Single Line) or Delimiter Format. Compared to the collection side, data processing offers more complex structured processing logic.
Log filtering: Reduces subsequent usage costs. Discard unnecessary log data to save storage and traffic costs on the cloud. For example, you may deliver logs to Tencent Cloud COS and CKafka in the future, which can effectively save delivery traffic.
Sensitive Data Masking: For example, masking information such as ID numbers and phone numbers.
Log Distribution: For example, classifying logs according to log levels: ERROR, WARNING, and INFO, and then distributing them to different log topics.

Advantages

Easy to use, especially friendly to data analysts and operations engineers. Provides out-of-the-box functions with no need to purchase, configure, or maintain a Flink cluster. You can achieve stream processing of massive logs, including data cleaning and filtering, data masking, structuring, and distribution, by using our packaged DSL functions. For more information, see Function Overview.
High-throughput real-time log data stream processing. High processing efficiency (millisecond latency) and high throughput, reaching 10-20 MB/s per partition (source log topic partition).

Customer Cases

Data cleaning and filtering: Client A discards invalid logs, retains only specified fields, and fills in some missing fields and values. If the log does not contain the fields product_name and sales_manager, it is considered invalid and discarded. Otherwise, the log is retained, keeping only the fields price, sales_amount, and discount. Other fields are dropped. If the log is missing the discount field, this field is added with a default value, such as "70%".
Data Conversion: Client B has field values in the original logs that are IP addresses. The client needs to add fields and values for country and city based on the IP. For example, for 2X0.18X.51.X5, add the fields country: China and city: Beijing. Convert the UNIX timestamp to Beijing time, for example, 1675826327 to 2023/2/8 11:18:47.
Log classification and delivery: Client C has original logs in multi-level JSON, which includes Array arrays. Client C uses data processing to extract the array from specified nodes in the multi-level JSON as field values. For example, extract the Auth field value from Array[0], and then distribute the log data based on the Auth field value. When the value is "SASL", deliver to target topic A; when the value is "Kerberos", deliver to target topic B; when the value is "SSL", deliver to target topic C.
Log structuring: Client D processes the original log "2021-12-02 14:33:35.022 [1] INFO org.apache.Load - Response:status: 200, resp msg: OK" to complete structuring, resulting in log_time: 2021-12-02 14:33:35.022, loglevel: info, status: 200.
For more detailed information, see Data Processing Cases.

Fee Description

Data processing incurs related costs. For details, see Billing Overview.
If your business only needs processed logs, it is recommended to set the retention period of the source log topic to 3-7 days and disable the index of the source log topic to save costs effectively.

Specifications and Limits

For specifications and limitations of data processing, see Data Processing.

Creating Processing Task

Last updated：2024-01-20 17:44:35

Overview

The data processing feature provides the capabilities to filter, cleanse, mask, enrich, and distribute log data. It can be benchmarked against the open-source component Logstash, but its input and output are CLS log topics.

You can select a source log topic and write Domain Specific Language (DSL) processing function statements to process logs line by line and send logs that meet certain requirements to specified target log topics.If your business uses only processed logs, we recommend you configure the log retention period of the source log topic as one day and not enable the index in order to reduce costs.

Billing

Data processing incurs fees. For example, if you have a log topic in Guangzhou and 100 GB raw data that becomes 50 GB after being filtered, and the target topic is written (with index not enabled), you need to pay:
Data processing fees: 100 GB * 0.026 USD/GB = 2.6 USD
Write traffic fees of the target topic: 50 GB * 0.3 (compression ratio) * 0.032 USD/GB = 0.48 USD
Storage fees of the target topic: 50 GB * 0.3 (compression ratio) * 0.0024 USD/GB = 0.036 USD
The total is 3.116 USD, which is generally the case for regions in the Chinese mainland. For billing details in special regions, see Product Pricing.

Use cases

For most users, data processing can achieve the following:
Extract structured data to facilitate future operations such as search, analysis, and dashboard generation.
Simplify logs to reduce use costs. Unnecessary logs will be discarded to reduce storage and traffic costs.
Mask sensitive data, such as users' ID numbers and phone numbers.
Ship logs by category. For example, logs are categorized by ERROR, WARNING, or INFO and then distributed to different log topics.

Common DSL functions

The following describes some DSL functions that are commonly used for data processing. The corresponding use cases can be viewed in **DSL Statement Generator** on the **Create Data Processing Task** > **Edit Processing Statement** page in the console. You can directly copy the sample code and modify parameters as needed for quick use.
Extracting structured data: `ext_sep()` for extracting by separator, `ext_json` or `ext_json_jmes` for extracting by JSON, and `ext_regex()` for extracting by regular expression
Simplifying logs: `log_drop` for discarding a line of log and `fields_drop()` for discarding log fields
Masking sensitive data: `regex_replace()`
Classifying logs: `t_if_else()` and `t_switch()` for determining conditions, and `log_output()` for distributing logs
Editing fields: `fields_set()` for adding/resetting fields, and `fields_rename()` for renaming fields
Determining the presence of a field or data in logs: `has_field()` for determining whether a field exists and `regex_match()` for determining whether certain data exists
Comparing numerical values in logs: `op_gt()` (greater than), `op_ge()` (greater than or equal to), and `op_eq()` (equal to); functions for addition, subtraction, multiplication, and division, such as `op_mul()` for multiplication and `op_add()` for addition
Processing text in logs: `str_uppercase()` for uppercase or lowercase switching and `str_replace()` for text replacement
Processing time values in logs: `dt_to_timestamp()` for converting a time value to a UTC timestamp
The preceding are common DSL functions used in data processing scenarios. The DSL statement generator also provides other DSL functions for your reference. You can copy a desired function to the DSL function editing box, modify parameters, and use it.
The following describes how to create a data processing task.

Prerequisites

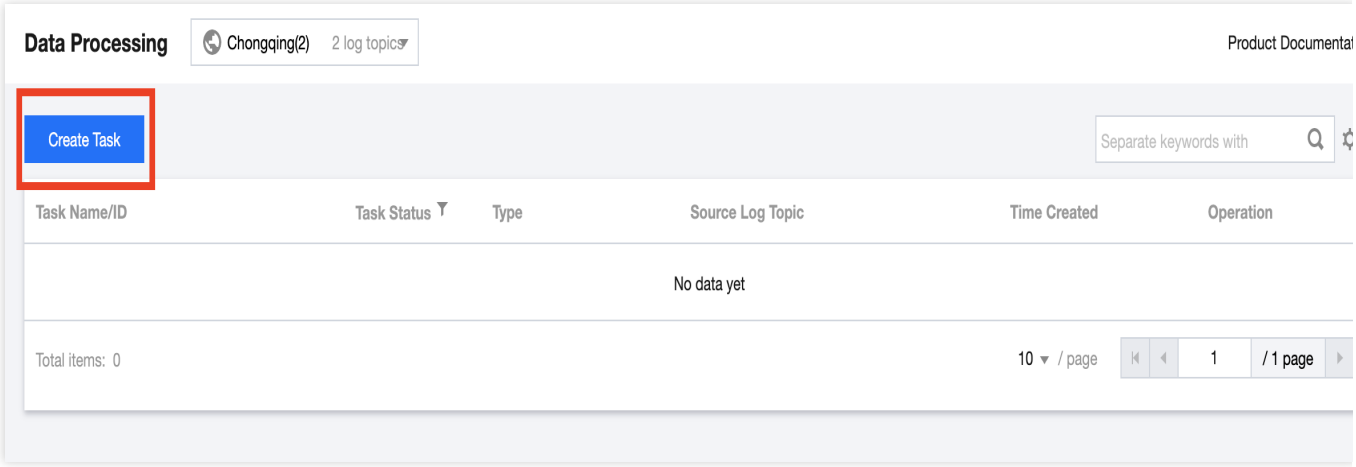
You have activated CLS, created a source log topic, and successfully collected log data.
You have created a target log topic, which is recommended to be empty to allow the writing of processed data.
The current account has the permission to configure data processing tasks.

Note:

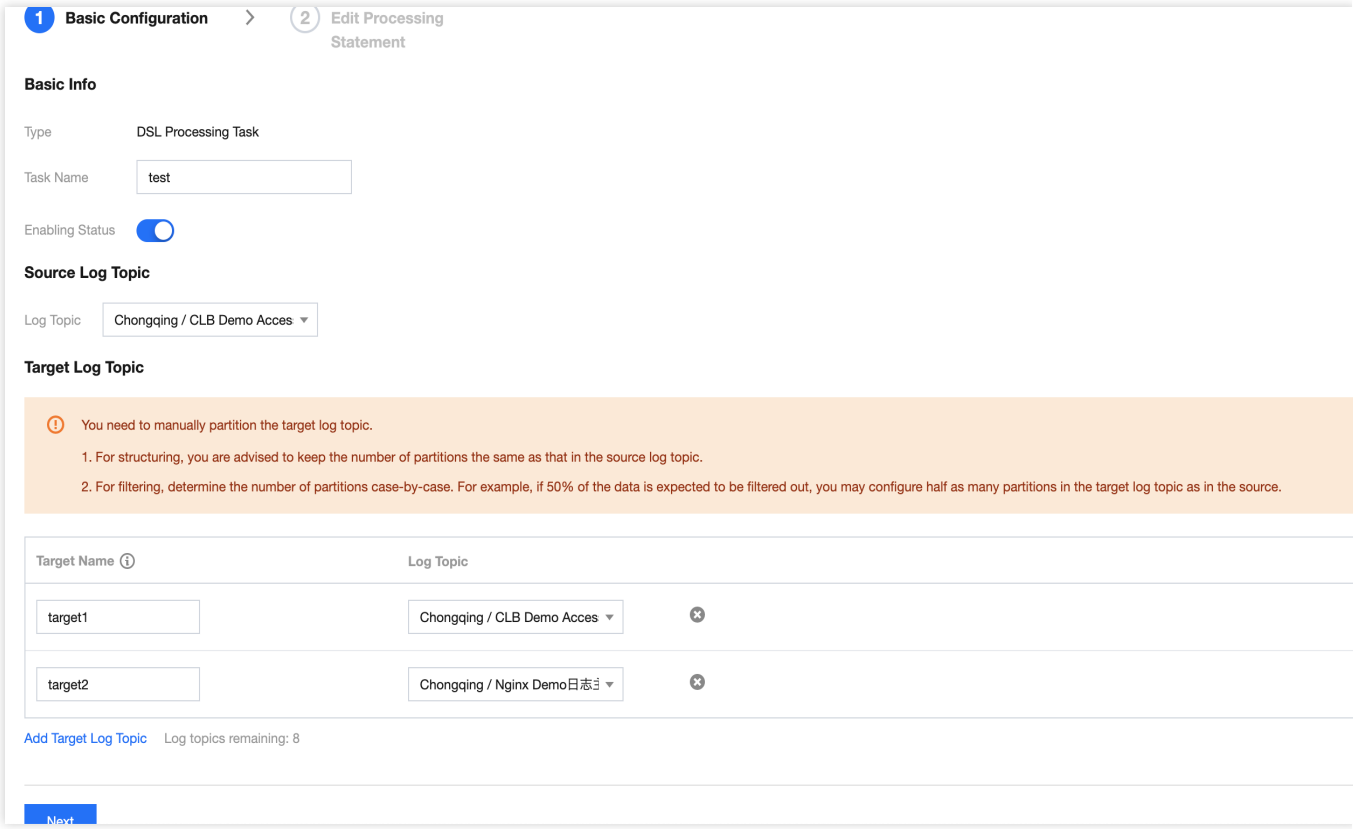
Data processing can process only real-time log streams but not historical logs.
If the data processing console does not automatically load raw log data, the possible cause is that your log topic does not have real-time log streams. In that case, you need to manually add custom logs in JSON format to complete writing the data processing script.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Data Processing**.
3. Click **Create Processing Task**.



4. On the **Basic Info** page, configure the following information:



Task Name: Enter the custom task name.

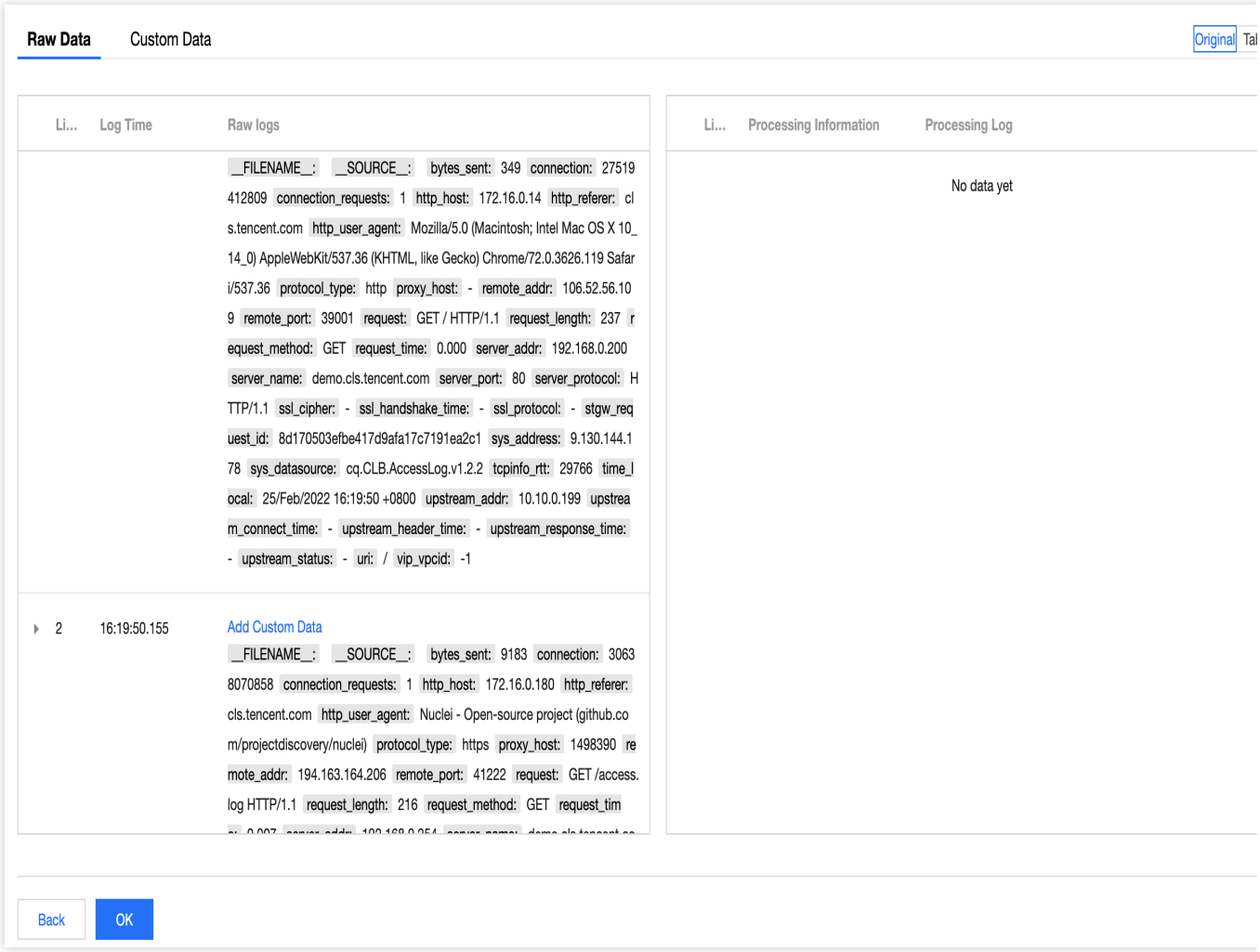
Source Log Topic: Select the source log topic.

Target Log Topic: Enter the target name and select the log topic. Here, the target name can be the alias of the target log topic and used as an input parameter of the written DSL function. You can configure up to ten target log topics.

5. Click **Next**.

6. On the **Edit Processing Statement** page, perform the following operations:

A DSL function can be tested by two types of data: raw log data and custom data. The system automatically loads raw log data, 100 records by default. If you think the raw log data is insufficient for testing the DSL function, you can directly enter custom data on the **Custom Data** tab. Alternatively, you can click **Add Custom Data** on the **Raw Data** tab, modify the data, and use it as your custom data.



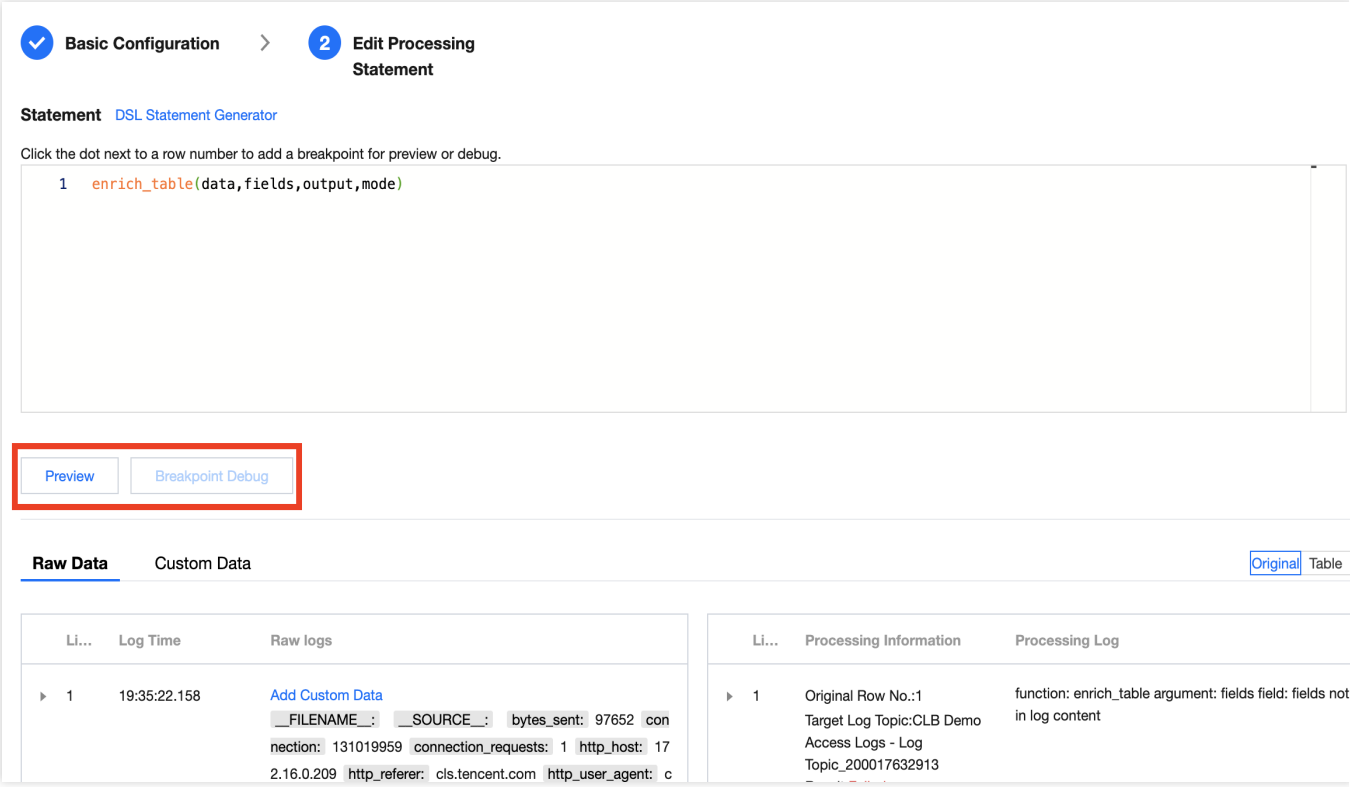
Note:

Custom data must be in JSON format.

If there are multiple entries of custom data, add them in the following format:

```
[
{
    "content": "2021-10-07 13: 32: 21|100|Customer not checked
in|Jack|Beijing|101123199001230123|"
},
{
    "content": "2021-10-07 13: 32: 21|500|Checked in
successfully|Jane|Sanya|501123199001230123|"
},
{
    "content": "2021-10-07 13: 32: 21|1000|Checked
out|Lily|Sanya|101123196001230123|"
}
]
```

7. Click **DSL Statement Generator**.
8. In the pop-up window, select a function, and click **Insert Function**.
The DSL statement generator provides the descriptions and examples of multiple types of functions. You can copy and paste the examples to the processing statement editing box and modify parameters as needed to write your own DSL functions. You can also refer to Processing Example to quickly understand how to write a DSL function.
9. After writing the DSL processing statement, click **Preview** or **Checkpoint Debugging** to run and debug the DSL function.
The running result will be displayed at the bottom right of the page. You can adjust the DSL statement according to the running result until it meets your requirements.



10. Click **OK** to submit the data processing task.

Viewing Data Processing Details

Last updated：2024-01-20 17:44:35

Overview

The data processing details page displays the number of lines of logs are processed, processing results, and failure causes. The following describes how to enter the data processing details page to view the processing content of a processing task.

Directions

Viewing task execution details

1. Log in to the CLS console.
2. On the left sidebar, click **Data Processing** to enter the data processing management page.
3. Click the ID/name of the target processing task to enter the task details page.
4. Click the **Execution Details** tab. On the tab page, you can view the task execution details by time segment.
Task execution details are collected in a statistical period of 10 minutes and include the numbers of input lines, output lines, and failed lines of log data processing. If the execution details include a failure, you can click **View Failure Details** to view the failure details.

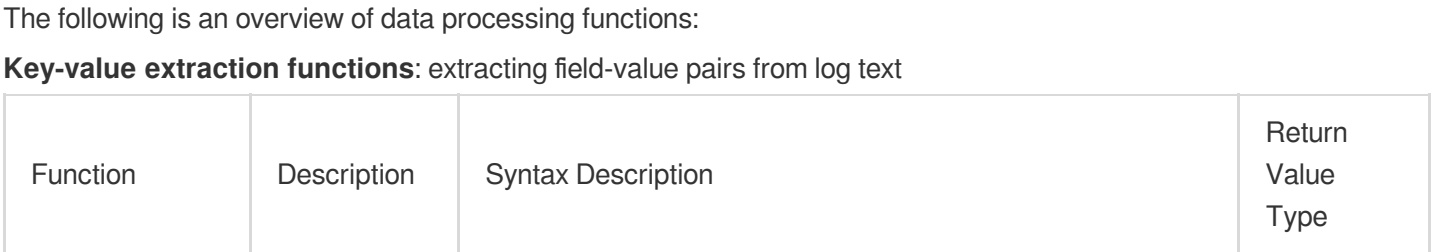
Viewing basic task information

1. Log in to the CLS console.
2. On the left sidebar, click **Data Processing** to enter the data processing management page.
3. Click the ID/name of the target processing task to enter the task details page.
4. Click the **Basic Info** tab. On the tab page, you can view the task's basic configuration, source log topic, target log topic, and processing statement.
In addition, you can click the target log topic to enter the index analysis page. (You can view the data of the target log topic only when index configuration is enabled.)

Data Processing Functions
Function Overview

Last updated：2025-06-11 17:26:46

CLS's data processing functions can be flexibly combined and used to cleanse, structure, filter, distribute, mask log data and more.
The figure below shows how a log containing JSON data is processed: the log data is processed and transformed into structured data, and then a field in the log is masked before the processed log is distributed.



The following is an overview of data processing functions:
Key-value extraction functions: extracting field-value pairs from log text

Function	Description	Syntax Description	Return Value Type
ext_sep	Extracts field value content based on a separator.	ext_sep("Source field name", "Target field 1,Target field 2,Target field...", sep="Separator", quote="Non-segmentation part", restrict=False, mode="overwrite")	Log after extraction (LOG)
ext_sepstr	Extracts field value content based on specified characters (string).	ext_sepstr("Source field name","Target field 1,Target field 2,Target field...", sep="abc", restrict=False, mode="overwrite")	Log after extraction (LOG)
ext_json	Extracts field values from JSON data.	Log after extraction (LOG) ext_json("Source field name",prefix="",suffix="",format="full",exclude_node="JSON nodes not to expand")	Log after extraction (LOG)
ext_json_jmes	Extracts a field value based on a JMES expression.	ext_json_jmes("Source field name", jmes= "JSON extraction expression", output="Target field", ignore_null=True, mode="overwrite")	Log after extraction (LOG)

ext_kv	Extracts field values by using two levels of separators.	ext_kv("Source field name", pair_sep=r"\\s", kv_sep="=", prefix="", suffix="", mode="fill-auto")	Log after extraction (LOG)
ext_regex	Extracts field values by using a regular expression.	ext_regex("Source field name", regex="Regular expression", output="Target field 1,Target field 2,Target field...", mode="overwrite")	Log after extraction (LOG)
ext_first_notnull	Returns the first non-null and non-empty result value.	ext_first_notnull(Value 1,Value 2,...)	The first non-null result value

Enrichment functions: Adding fields to existing fields according to rules

Function	Description	Syntax Description	Return Value Type
enrich_table	Uses CSV structure data to match fields in logs and, when matched fields are found, the function adds other fields and values in the CSV data to the source logs.	enrich_table("CSV source data", "CSV enrichment field", output="Target field 1,Target field 2,Target field....", mode="overwrite")	Mapped log (LOG)
enrich_dict	Uses dict structure data to match a field value in a log. If the specified field and value match a key in the dict structure data, the function assigns the value of the key to another field in the log.	enrich_dict("JSON dictionary", "Source field name", output=Target field name, mode="overwrite")	Mapped log (LOG)

Flow control functions: condition determination

Function	Description	Syntax Description	Return Value Type
compose	Combines multiple operation functions. Providing combination capabilities similar to those of branch code blocks, this function can combine multiple operation functions and execute them in sequence. It can be used in	compose("Function 1","Function 2", ...)	Log (LOG)

	combination with branches and output functions.		
t_if	Executes a corresponding function if a condition is met and does not perform any processing if the condition is not met.	t_if("Condition", Function)	Log (LOG)
t_if_not	Executes a corresponding function if a condition is not met and does not perform any processing if the condition is met.	t_if_not("Condition",Function)	Log (LOG)
t_if_else	Executes a function based on the evaluation result of a condition.	t_if_else("Condition", Function 1, Function 2)	Log (LOG)
t_switch	Executes different functions depending on whether branch conditions are met. If all conditions are not met, the data is deleted.	t_switch("Condition 1", Function 1, "Condition 2", Function 2, ...)	Log (LOG)

Row processing functions: log distribution, deletion, and splitting.

Function	Description	Syntax Description	Return Value Type
log_output	Outputs a row of log to a specified log topic. This function can be used independently or together with branch conditions.	log_output(Log topic alias) (The alias here is the target log topic alias specified when the data processing task is created.)	No return value
log_split	Splits a row of log into multiple rows of logs based on the value of a specified field by using a separator and JMES expression.	log_split(Field name, sep=",", quote="\\", jmes="", output="")	Log (LOG)
log_drop	Deletes logs that meet a specified condition.	log_drop(Condition 1)	Log (LOG)
log_keep	Retains logs that meet a specified condition.	log_keep(Condition 1)	Log (LOG)
log_split_jsonarray_jmes	Splits and expands the JSON array in the log according to JMES syntax.	log_split_jsonarray_jmes("field", jmes="items", prefix="")	Log (LOG)

Field processing functions: field Create/Read/Update/Delete/Rename

Function	Description	Syntax Description	Return Value Type
Extract Tag	Extract tag values from log fields and use these tag values as tags for dynamically generated topics.	extract_tag(tag name 1, tag value 1, tag name 2, tag value 2....)	Value string type (STRING)
fields_drop	Deletes the fields that meet a specified condition.	fields_drop(Field name 1, Field name 2, ..., regex=False,nest=False)	Log (LOG)
fields_keep	Retains the fields that meet a specified condition.	fields_keep(Field name 1, Field name 2, ..., regex=False)	Log (LOG)
fields_pack	Matches field names based on a regular expression and encapsulates the matched fields into a new field whose value is in JSON format.	fields_pack(Target field name, include=".*", exclude="", drop_packed=False)	Log (LOG)
fields_set	Sets field values or adds fields.	fields_set(Field name 1, Field value 1, Field name 2, Field value 2, mode="overwrite")	Log (LOG)
fields_rename	Renames fields.	fields_rename(Field name 1, New field name 1, Field name 2, New field name 2, regex=False)	Log (LOG)
has_field	If the specified field exists, returns `True`. Otherwise, returns `False`.	has_field(Field name)	Condition value (BOOL)
not_has_field	If the specified field does not exist, returns `True`. Otherwise, returns `False`.	not_has_field(Field name)	Condition value (BOOL)
v	Gets the value of a specified field and returns the corresponding string.	v(Field name)	Value string type (STRING)

Value structuring functions: JSON data processing

Function	Description	Syntax Description	Return Value Type

array_get	Retrieve the value of the array, return String.	array_get(array,index_position)	Value string type (STRING)
json_select	Extracts a JSON field value with a JMES expression and returns the JSON string of the extraction result.	json_select(v(Field name), jmes="")	Value string type (STRING)
xml_to_json	Parses and converts an XML-formatted value to a JSON string. The input value must be an XML string. Otherwise, a conversion exception will occur.	xml_to_json(Field value)	Value string type (STRING)
json_to_xml	Parses and converts a JSON string value to an XML string.	json_to_xml(Field value)	Value string type (STRING)
if_json	Checks whether a value is a JSON string.	if_json(Field value)	Condition value (BOOL)

Regular expression functions: matching and replacing characters in text by using regular expressions

Function	Description	Syntax Description	Return Value Type
sensitive_detection	Sensitive information detection, for example, ID card, bank card.	sensitive_detection(scope="", ratio=1, discover_items="", replace_items="")	Value string type (STRING)
regex_match	Matches data in full or partial match mode based on a regular expression and returns whether the match is successful.	regex_match(Field value, regex="", full=True)	Condition value (BOOL)
regex_select	Matches data based on a regular expression and returns the corresponding partial match result. You can specify the sequence number of the matched expression and the sequence number of the group to return (partial match + sequence number of the specified matched group). If no data is matched, an empty string is returned.	regex_select(Field value, regex="", index=1, group=1)	Value string type (STRING)
regex_split	Splits a string and returns a	regex_split(Field value,	Value string

	JSON array of the split strings (partial match).	regex=\\"\\", limit=100)	type (STRING)
regex_replace	Matches data based on a regular expression and replaces the matched data (partial match).	regex_replace(Field value, regex="", replace="", count=0)	Value string type (STRING)
regex_findall	Matches data based on a regular expression and returns a JSON array of the matched data (partial match).	regex_findall(Field value, regex="")	Value string type (STRING)

Date value processing functions

Function	Description	Syntax Description	Return Value Type
custom_cls_log_time	Customize log time. A new log time will be generated based on your processing rules. Seconds, milliseconds, microseconds, and nanoseconds are supported.	custom_cls_log_time(time)	STRING
dt_str	Converts a time field value (a date string in a specific format or timestamp) to a target date string of a specified time zone and format.	dt_str(Value, format="Formatted string", zone="")	STRING
dt_to_timestamp	Converts a time field value (a date string in a specified format; time zone specified) to a UTC timestamp.	dt_to_timestamp(Value, zone="")	STRING
dt_from_timestamp	Converts a timestamp field value to a time string in the specified time zone.	dt_from_timestamp(Value, zone="")	STRING
dt_now	Obtains the current datetime of the processing calculation.	dt_now(format="Formatted string", zone="")	STRING

String processing functions

Function	Description	Syntax Description	Return Value Type
str_exist	Find a substring within the specified range of values and return True or False.	str_exist(data1, data2, ignore_upper=False)	BOOL
str_count	Searches for a substring in a specified	str_count(Value, sub="",	INT

	range of a value and returns the number of occurrences of the substring.	start=0, end=-1)	
str_len	Returns the length of a string.	str_len(Value)	INT
str_uppercase	Converts a string to uppercase.	str_uppercase(Value)	STRING
str_lowercase	Converts a string to lowercase.	str_lowercase(Value)	STRING
str_join	Concatenates input values by using a concatenation string.	str_join(Concatenation string 1, Value 1, Value 2, ...)	STRING
str_replace	Replaces an old string with a new string.	str_replace(Value, old="", new="", count=0)	STRING
str_format	Formats strings.	str_format(Formatting string, Value 1, Value 2, ...)	STRING
str_strip	Deletes specified characters from a string concurrently from the start and end of the string, and returns the remaining part.	str_strip(Value, chars="\\t\\r\\n")	STRING
str_lstrip	Deletes specified characters from a string from the start of the string, and returns the remaining part.	str_strip(Value, chars="\\t\\r\\n")	STRING
str_rstrip	Deletes specified characters from a string from the end of the string, and returns the remaining part.	str_strip(Value, chars="\\t\\r\\n")	STRING
str_find	Checks whether a string contains a specified substring and returns the position of the first occurrence of the substring in the string.	str_find(Value, sub="", start=0, end=-1)	INT
str_start_with	Checks whether a string starts with a specified string.	str_start_with(Value, sub="", start=0, end=-1)	BOOL
str_end_with	Checks whether a string ends with a specified string.	str_end_with(Value, sub="", start=0, end=-1)	BOOL

Logical expression functions

Function	Description	Syntax Description	Return Value Type
op_if	Returns a value based on a specified condition.	op_if(Condition 1, Value 1, Value 2)	If the condition is `true`, `Value 1` is returned;

			otherwise, `Value 2` is returned.
op_and	Performs the AND operation on values. If all the specified parameter values are evaluated to true, `True` is returned; otherwise, `False` is returned.	op_and(Value 1, Value 2, ...)	BOOL
op_or	Performs the OR operation on values. If one or more of the specified parameter values are evaluated to false, `False` is returned; otherwise, `True` is returned.	op_or(Value 1, Value 2, ...)	BOOL
op_not	Performs the NOT operation on values.	op_not(Value)	BOOL
op_eq	Compares two values. If the values are equal, `True` is returned.	op_eq(Value 1, Value 2)	BOOL
op_ge	Compares two values. If `Value 1` is greater than or equal to `Value 2`, `True` is returned.	op_ge(Value 1, Value 2)	BOOL
op_gt	Compares two values. If `Value 1` is greater than `Value 2`, `True` is returned.	op_gt(Value 1, Value 2)	BOOL
op_le	Compares two values. If `Value 1` is less than or equal to `Value 2`, `True` is returned.	op_le(Value 1, Value 2)	BOOL
op_lt	Compares two values. If `Value 1` is less than `Value 2`, `True` is returned.	op_lt(Value 1, Value 2)	BOOL
op_add	Returns the sum of two specified values.	op_add(Value 1, Value 2)	Calculation result
op_sub	Returns the difference between two specified values.	op_sub(Value 1, Value 2)	Calculation result
op_mul	Returns the product of two specified values.	op_mul(Value 1, Value 2)	Calculation result
op_div	Returns the quotient of two specified values.	op_div(Value 1, Value 2)	Calculation result
op_sum	Returns the sum of multiple specified values.	op_sum(Value 1, Value 2, ...)	Calculation result
op_mod	Returns the remainder of a specified value divided by the other specified value.	op_mod(Value 1, Value 2)	Calculation result

op_null	Checks whether a value is `null`. If so, `true` is returned; otherwise, `false` is returned.	op_null(Value)	BOOL
op_notnull	Checks whether a value is not `null`. If so, `true` is returned; otherwise, `false` is returned.	op_notnull(Value)	BOOL
op_str_eq	Compares string values. If they are equal to each other, `true` is returned.	op_str_eq(Value 1, Value 2, ignore_upper=False)	BOOL

Type conversion functions

Function	Description	Syntax Description	Return Value Type
ct_int	Converts a value (whose base can be specified) to a decimal integer.	ct_int(Value 1, base=10)	Calculation result
ct_float	Converts a value to a floating-point number.	ct_float(Value)	Calculation result
ct_str	Converts a value to a string.	ct_str(Value)	Calculation result
ct_bool	Converts a value to a Boolean value.	ct_bool(Value)	Calculation result

Encoding/Decoding functions

Function	Description	Syntax Description	Return Value Type
md5_encoding	Calculate and return the MD5 checksum	md5_encoding(value)	Calculation result
uuid	Generate a universally unique identifier (UUID).	uuid()	STRING
str_encode	Encode a string in the specified format.	str_encode(data, encoding="utf8", errors="ignore")	STRING(UTF8 Format)
str_decode	Decode a string in the specified format.	str_decode(data, encoding="utf8", errors="ignore")	STRING
base64_encode	Encode the string in base64.	base64_encode(value, format="RFC3548")	STRING(base64 Format)

base64_decode	Decode the string in base64.	base64_decode(value, format="RFC3548")	STRING
decode_url	Decodes an encoded URL.	decode_url(Value)	STRING

IP parsing functions

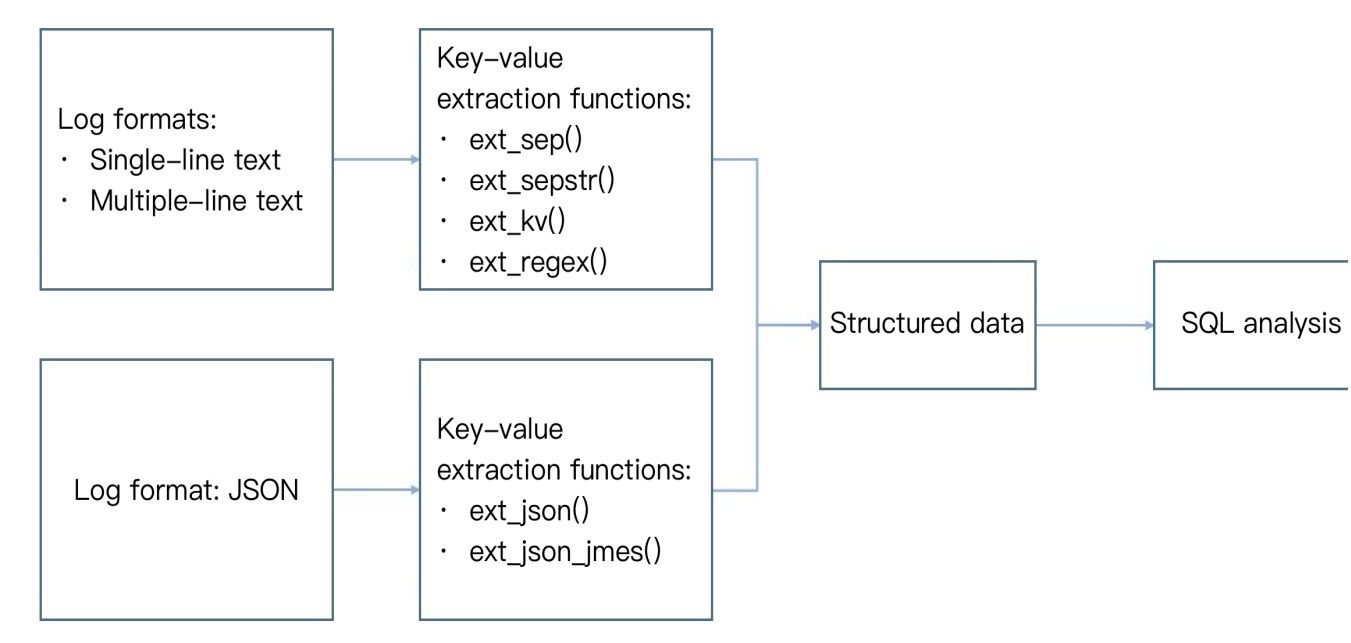
Function	Description	Syntax Description	Return Value Type
geo_parse	Parses the geographical location.	geo_parse(Field value, keep=("country","province","city"), ip_sep=",")	JSON string
is_subnet_of	Checks whether an IP is in the target IP range. Multiple IP ranges are supported.	is subnet of (network segments, iP)	BOOL

Key-Value Extraction Functions

Last updated：2024-01-20 17:44:35

Overview

The figure below shows the common use cases of key-value extraction functions. After key-value extraction, logs are processed into structured data, which can be used for SQL analysis.



Function ext_sep()

Function definition

This function is used to extract field value content based on a separator (single character).

Syntax description

```
ext_sep("Source field name", "Target field 1,Target field 2,Target field...", sep="
```

Field description

Parameter	Description	Parameter	Required	Default	Value Range

		Type		Value	
field	Field to extract	string	Yes	-	Name of an existing field in the user log
output	A single field name or multiple new field names concatenated with commas	string	Yes	-	-
sep	Separator	string	No	,	Any single character
quote	Characters that enclose the value	string	No	-	-
restrict	Handling mode when the number of extracted values is inconsistent with the number of target fields entered by the user: True: ignore the extraction function and do not perform any extraction processing. False: try to match the first few fields	bool	No	False	-
mode	Write mode of the new field	string	No	overwrite	-

Sample

Example 1. Extract values from logs by using a comma as the separatorRaw log:

```
{"content": "hello Go,hello Java,hello python"}
```

Processing rule:

```
// Use a comma as the separator to divide the `content` field into three parts,
corresponding to the `f1`, `f2`, and `f3` fields separately.
ext_sep("content", "f1, f2, f3", sep=",", quote="", restrict=False,
mode="overwrite")
// Delete the `content` field.
fields_drop("content")
```

Processing result:

```
{"f1":"hello Go","f2":"hello Java","f3":"hello python"}
```

Example 2. Process the `content` string as a whole by using `quote` Raw log:

```
{"content": " Go,%hello ,Java%,python"}
```

Processing rule:

```
ext_sep("content", "f1, f2",  quote="%", restrict=False)
```

Processing result:

```
// Though `%hello ,Java%` does contain a comma, it does not participate in separato
{"content":" Go,%hello ,Java%,python","f1":" Go","f2":"hello ,Java"}
```

Example 3: `restrict=True` indicates the number of divided values is different from the target fields, the function is not executed.Raw log:

```
{"content": "1,2,3"}
```

Processing rule:

```
ext_sep("content", "f1, f2", restrict=True)
```

Processing result:

```
{"content":"1,2,3"}
```

Function ext_sepstr()

Function definition

This function is used to extract field value content based on multiple characters (string).

Syntax description

```
ext_sepstr("Source field name","Target field 1,Target field 2,Target field...", sep
```

Field description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field to extract	string	Yes	-	Name of an existing field in the user log
output	A single field name or multiple new field names concatenated with commas	string	Yes	-	-

sep	Separator (string)	string	No	,	-
restrict	Handling mode when the number of extracted values is inconsistent with the number of target fields entered by the user: True: ignore the extraction function and do not perform any extraction processing. False: try to match the first few fields	bool	No	False	-
mode	Write mode of the new field	string	No	overwrite	-

Sample

Raw log:

```
{"message":"1##2##3"}
```

Processing rule:

```
// Use "##" as the separator to extract key-values.
ext_sepstr("message", "f1,f2,f3,f4", sep="##")
```

Processing result:

```
// If the number of target fields is greater than the number of divided values, `""
{"f1":"1","f2":"2","message":"1##2##3","f3":"3","f4":""}
```

Function ext_json()

Function definition

This function is used to extract field values from JSON data.

Syntax description

```
ext_json("Source field name",prefix="",suffix="",format="full",exclude_node="JSON n
```

Field description

Parameter	Description	Parameter Type	Required	Default Value	Value Range

field	Field to extract	string	Yes	-	-
prefix	Prefix of the new field	string	No	-	-
suffix	Suffix of the new field	string	No	-	-
format	`full` : The field name format is in full path format (parent + sep + prefix + key + suffix). `simple` : non-full path format (prefix + key + suffix)	string	No	simple	-
sep	Concatenation character, used to concatenate node names	string	No	#	-
depth	Depth to which the function expands the source field, beyond which nodes will not be expanded any more	number	No	100	1-500
expand_array	Whether to expand an array node	bool	No	False	-
include_node	Allowlist of node names that match the specified regular expression	string	No	-	-
exclude_node	Blocklist of node names that match the specified regular expression	string	No	-	-
include_path	Allowlist of node paths that match the specified regular expression	string	No	-	-
exclude_path	Allowlist of node paths that match the specified regular expression	string	No	-	-
retain	Retains some special symbols without escaping them, such as \\n and \\t.	string	No	-	-
escape	Whether to escape data. Default value: `True` . If special symbols are contained, escaping cannot be performed.	bool	No	True	-

Sample

Example 1. Extract the key-values of all nodes and construct new fields based on the extracted values. The example log is multi-level nesting, but the extraction does not distinguish hierarchy.Raw log:

```
{
```

```
"data": "{  \\"k1\\": 100, \\"k2\\": {  \\"k3\\": 200,  \\"k4\\": { \\"k5\\":
300}}}"
}
```

Processing rule:

```
ext_json("data")
```

Processing result:

```
{"data":"{  \\"k1\\": 100, \\"k2\\": {  \\"k3\\": 200,  \\"k4\\": { \\"k5\\":
300}}}","k1":"100","k3":"200","k5":"300"}
```

Example 2. Perform extraction excluding `sub_field1`

Raw log:

```
{"content": "{\\"sub_field1\\":1,\\"sub_field2\\":\\"2\\"}"}
```

Processing rule:

```
// `exclude_node=subfield1` indicates not to extract the node.
ext_json("content", format="full", exclude_node="sub_field1")
```

Processing result:

```
{"sub_field2":"2","content":"{\\"sub_field1\\":1,\\"sub_field2\\":\\"2\\"}"}
```

Example 3. Add `prefix` to subnodesRaw log:

```
{"content": "{\\"sub_field1\\":
{\\"sub_sub_field3\\":1},\\"sub_field2\\":\\"2\\"}"}
```

Processing rule 1:

```
// When `sub_field2` is extracted, the prefix `udf\\_` is automatically added
to it, making it `udf\\_\\_sub\\_field2`.
ext_json("content", prefix="udf_", format="simple")
```

Processing result 1:

```
{"content":"{\\"sub_field1\\":
{\\"sub_sub_field3\\":1},\\"sub_field2\\":\\"2\\"}","udf_sub_field2":"2","udf_s
ub_sub_field3":"1"}
```

Processing rule 2:

```
// `format=full` indicates to retain the hierarchy of the extracted field name.
When `sub_field2` is extracted, the name of its parent node is automatically to
```

```
it, making it `#content#__sub_field2`.
ext_json("content", prefix="__", format="full")
```

Processing result 2:

```
{"#content#__sub_field2":"2","#content#sub_field1#__sub_sub_field3":"1","conten
t":"{\\"sub_field1\\":{\\"sub_sub_field3\\":1},\\"sub_field2\\":\\"2\\"}"}
```

Example 4. Support special symbols

Raw log 1:

```
{"content": "{\\"sub_field1\\":1,\\"sub_field2\\":\\"\\\\n2\\"}"}
```

Processing rule 1:

```
ext_json("content",retain="\\n")
```

Processing result 1:

```
{"sub_field2":"\\\\n2","content":"
{\\"sub_field1\\":1,\\"sub_field2\\":\\"\\\\n2\\"}","sub_field1":"1"}
```

Raw log 2:

```
{"content": "{\\"sub_field1\\":1,\\"sub_field2\\":\\"\\\\n2\\\\t\\"}"}
```

Processing rule 2:

```
ext_json("content",retain="\\n,\\t")
```

Processing result 2:

```
{"sub_field2":"\\\\n2\\\\t","content":"
{\\"sub_field1\\":1,\\"sub_field2\\":\\"\\\\n2\\\\t\\"}","sub_field1":"1"}
```

Example 5. Specify whether to escape

Raw log:

```
{"message":"{\\"ip\\":\\"183.6.104.157\\",\\"params\\":\\"
[{\\\\\\"tokenType\\\\\\":\\\\\\"RESERVED30\\\\\\",\\\\\\"otherTokenInfo\\\\\\"
:
{\\\\\\"unionId\\\\\\":\\\\\\"123\\\\\\"},\\\\\\"unionId\\\\\\":\\\\\\"adv\\\\\
\"}]\\"}"}
```

Processing rule:

```
ext_json("message", escape=False)
fields_drop("message")
```

Processing result:

```
{"ip":"183.6.104.157", "params":"
[{\\"tokenType\\":\\"RESERVED30\\",\\"otherTokenInfo\\":
{\\"unionId\\":\\"123\\"},\\"unionId\\":\\"adv\\"}]"}
```

Function ext_json_jmes()

Function definition

This function is used to extract field values from JSON data.

Syntax description

```
ext_json_jmes("Source field name", jmes= "JSON extraction expression", output="Targ
```

Field description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field to extract	string	Yes	-	-
jmes	JMES expression. For more information, see JMESPath.	string	Yes	-	-
output	Output field name. Only a single field is supported.	string	Yes	-	-
ignore_null	Whether to ignore a node whose value is null. The default value is `True`, ignoring fields whose value is null. Otherwise, an empty string is returned.	bool	No	True	-
mode	Write mode of the new field. Default value: `overwrite`	string	No	overwrite	-

Sample

Example 1. Extract only one node from multi-layer JSON data

Raw log:

```
{"content": "{\\"a\\":{\\"b\\":{\\"c\\":{\\"d\\":\\"value\\"}}}}"}
```

Processing rule:

```
// `jmes="a.b.c.d"` means to extract the value of `a.b.c.d`.
ext_json_jmes("content", jmes="a.b.c.d", output="target")
```

Processing result:

```
{"content":"{\\"a\\":{\\"b\\":{\\"c\\":
{\\"d\\":\\"value\\"}}}}","target":"value"}
```

Example 2Raw log:

```
{"content": "{\\"a\\":{\\"b\\":{\\"c\\":{\\"d\\":\\"value\\"}}}}"}
```

Processing rule:

```
// `jmes="a.b.c.d"` means to extract the value of `a.b.c`.
ext_json_jmes("content", jmes="a.b.c", output="target")
```

Processing result:

```
{"content":"{\\"a\\":{\\"b\\":{\\"c\\":{\\"d\\":\\"value\\"}}}}","target":"
{\\"d\\":\\"value\\"}"}
```

Function ext_regex()

Function definition

This function is used to extract the value of a field by using a regular expression.

Syntax description

```
ext_regex("Source field name", regex="Regular expression", output="Target field 1,T
```

Field description looking for b

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field to extract	string	Yes	-	-
regex	Regular expression. If the expression contains a special character, escaping is required. Otherwise, syntax error is reported.	string	Yes	-	-

output	A single field name or multiple new field names concatenated with commas	string	No	-	-
mode	Write mode of the new field. Default value: `overwrite`	string	No	overwrite	-

Sample

Example 1. Match digitsRaw log:

```
{"content": "1234abcd5678"}
```

Processing rule:

```
ext_regex("content", regex="\\d+", output="target1,target2")
```

Processing result:

```
{"target2":"5678","content":"1234abcd5678","target1":"1234"}
```

Example 2. The regular expression contains named capturing group, and some field values are automatically filled

Raw log:

```
{"content": "1234abcd"}
```

Processing rule:

```
ext_regex("content", regex="(?<target1>\\d+)(.*)", output="target2")
```

Processing result:

```
{"target2":"abcd","content":"1234abcd","target1":"1234"}
```

Function ext_kv()

Function definition

This function is used to extract key-value pairs by using two levels of separators.

Syntax description

```
ext_kv("Source field name", pair_sep=r"\\s", kv_sep="=", prefix="", suffix="", mode
```

Field description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field to extract	string	Yes	-	-
pair_sep	Level-1 separator, separating multiple key-value pairs	string	Yes	-	-
kv_sep	Level-2 separator, separating keys and values	string	Yes	-	-
prefix	Prefix of the new field	string	No	-	-
suffix	Suffix of the new field	string	No	-	-
mode	Write mode of the new field. Default value: `overwrite`	string	No	-	-

Sample

The raw log contains two levels of separators: "|" and "=".

Raw log:

```
{"content": "a=1|b=2|c=3"}
```

Processing rule:

```
ext_kv("content", pair_sep="|", kv_sep="=")
```

Processing result:

```
{"a":"1","b":"2","c":"3","content":"a=1|b=2|c=3"}
```

Function ext_first_notnull()

Function definition

This function is used to return the first non-null and non-empty result value.

Syntax description

```
ext_first_notnull(value 1, value 2, ...)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
Variable parameter list	Parameters or expressions that participate in the calculation	string	Yes	-	-

Sample

Raw log:

```
{"data1": null, "data2": "", "data3": "first not null"}
```

Processing rule:

```
fields_set("result", ext_first_notnull(v("data1"), v("data2"), v("data3")))
```

Processing result:

```
{"result":"first not null","data3":"first not null","data2":"","data1":"null"}
```

Function ext_grok

Function definition

This function is used to extract the matched result value according to the Grok syntax.

Syntax description

```
ext_grok(Field value, grok="", extend="")
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
field	Field value	string	Yes	-	-
grok	Expression	string	Yes	-	-
extend	Custom Grok expression	string	Yes	-	-

Sample

Example 1

Raw log:

```
{"content":"2019 June 24 \\"I am iron man\\""}
```

Processing rule:

```
ext_grok("content", grok="%{YEAR:year} %{MONTH:month} %{MONTHDAY:day} %
{QUOTEDSTRING:motto}")
fields_drop("content")
```

Processing result:

```
{"day":"24", "month":"June", "motto":"I am iron man", "year":"2019"}
```

Example 2

Raw log:

```
{"content":"Beijing-1104,Beijing-Beijing"}
```

Processing rule:

```
ext_grok("content", grok="%{ID1:user_id1},%{ID2:user_id2}",extend="ID1=%{WORD}-
%{INT},ID2=%{WORD}-%{WORD}")
fields_drop("content")
```

Processing result:

```
{"user_id1":"Beijing-1104", "user_id2":"Beijing-Beijing"}
```

Enrichment Functions

Last updated：2024-01-20 17:44:35

Function enrich_table

Function definition

This function uses CSV structure data to match fields in logs and, when matched fields are found, the function adds other fields and values in the CSV data to the source logs.

Syntax description

```
enrich_table(CSV data string, CSV column name string, output=Target field name or l
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Input CSV data, where the first row is column names and the rest rows are corresponding values. Example: region,count\\nbj, 200\\ngz, 300	string	Yes	-	-
fields	Column name to match. If the field name in the CSV data is the same as the field with the same name in the log, the matching is successful. The value can be a single field name or multiple new field names concatenated with commas.	string	Yes	-	-
output	Output field list. The value can be a single field name or multiple new field names concatenated with commas.	string	Yes	-	-
mode	Write mode of the new field. Default value: `overwrite`	string	No	overwrite	-

Example

Raw log:

```
{"region": "gz"}
```

Processing rule:

```
enrich_table("region,count\\nbj,200\\ngz,300", "region", output="count")
```

Processing result:

```
{"count":"300","region":"gz"}
```

Function enrich_dict

Function definition

This function uses dict structure data to match a field value in a log. If the specified field and value match a key in the dict structure data, the function assigns the value of the key to another field in the log.

Syntax description

```
enrich_dict(Dict data string, Source field name, output=Target field, mode="overwri
```

Parameter description

Parameter	Description	Parameter Type
data	Input dict data, which must be the escaped string of a JSON object. Example: enrich_dict(" {"200":"SUCCESS","500":"FAILED"}", "status", output="message")	string
fields	Field name to match. If the value of the key in the dict data is the same as the value of the specified field, the matching is successful. The value can be a single field name or multiple new field names concatenated with commas.	string
output	Target field list. After successful matching, the function writes the corresponding values in the dict data to the target field list. The value can be a single field name or multiple new field names concatenated with commas.	string
mode	Write mode of the new field.	string

Example

Raw log:

```
{"status": "500"}
```

Processing rule:

```
enrich_dict("{\\"200\\":\\"SUCCESS\\",\\"500\\":\\"FAILED\\"}", "status",
output="message")
```

Processing result:

```
{"message":"FAILED","status":"500"}
```

Flow Control

Last updated：2024-01-20 17:44:35

Overview

The writing method of the flow control logic in commonly used programming languages is different from that in DSL functions. See the figure below.

Function compose

Function definition

This function is used to combine multiple operation functions. Providing combination capabilities similar to those of branch code blocks, this function can combine multiple operation functions and execute them in sequence. It can be used in combination with branches and output functions.

Syntax description

```
compose(Function 1,Function 2, ...)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range

Variable parameter, function	The parameter must be a function whose return value type is LOG.	string	At least one function parameter	-	-

Example

Example 1. Call functions in sequence, executing the `enrich` function first and then the `fields_set` function

Raw log:

```
{"status": "500"}
```

Processing rule:

```
// 1. `enrich` function: use the data in `dict` to enrich the raw log, where `statu
//2. `fields_Set` function: add a field `new` and assign value `1` to it.
compose(enrich_dict("{\\"200\\":\\"SUCCESS\\",\\"500\\":\\"FAILED\\"}", "status", o
```

Processing result:

```
// The final log contains 3 fields:
{"new":"1","message":"FAILED","status":"500"}
```

Example 2

Raw log:

```
{"status": "500"}
```

Processing rule:

```
compose(fields_set("new", 1))
```

Processing result:

```
{"new":"1","status":"500"}
```

Example 3

Raw log:

```
{"condition1": 0,"condition2": 1, "status": "500"}
```

Processing rule:

```
t_if_else(v("condition2"), compose(fields_set("new", 1),log_output("target")),
log_output("target2"))
```

Processing result, `target` output:

```
{"new":"1","condition1":"0","condition2":"1","status":"500"}
```

Function t_if

Function definition

This function is used to execute a corresponding function if a condition is met and does not perform any processing if the condition is not met.

Syntax description

```
t_if(Condition 1, Function 1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
condition	Function expression whose return value is of bool type	bool	Yes	-	-
function	Function expression whose return value is of LOG type	string	Yes	-	-

Example

Example 1Raw log:

```
{"condition": 1, "status": "500"}
```

Processing rule:

```
t_if(True, fields_set("new", 1))
```

Processing result:

```
{"new":"1","condition":"1","status":"500"}
```

Example 2

Raw log:

```
// If the value of `condition` is `1` (true), add a field `new` and assign value `1
{"condition": 1, "status": "500"}
```

Processing rule:

```
t_if(v("condition"), fields_set("new", 1))
```

Processing result:

```
{"new":"1","condition":"1","status":"500"}
```

Function t_if_not

Function definition

This function is used to execute a corresponding function if a condition is not met and does not perform any processing if the condition is met.

Syntax description

```
t_if_not(Condition 1, Function 1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
condition	Function expression whose return value is of bool type	bool	Yes	-	-
function	Function expression whose return value is of LOG type	string	Yes	-	-

Example

Raw log:

```
{"condition": 0, "status": "500"}
```

Processing rule:

```
t_if_not(v("condition"), fields_set("new", 1))
```

Processing result:

```
{"new":"1","condition":"0","status":"500"}
```

Function t_if_else

Function definition

This function is used to execute a function based on the evaluation result of a condition.

Syntax description

```
t_if_else("Condition 1", Function 1, Function 2)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
condition	Function expression whose return value is of bool type	bool	Yes	-	-
function	Function expression whose return value is of LOG type	string	Yes	-	-
function	Function expression whose return value is of LOG type	string	Yes	-	-

Example

Raw log:

```
{"condition": 1, "status": "500"}
```

Processing rule:

```
t_if_else(v("condition"), fields_set("new", 1), fields_set("new", 2))
```

Processing result:

```
{"new":"1","condition":"1","status":"500"}
```

unction t_switch

Function definition

This function is used to execute different functions depending on whether branch conditions are met. If all conditions are not met, the data is deleted.

Syntax description

```
t_switch("Condition 1", Function 1, "Condition 2", Function 2, ...)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
Variable parameter, which is a list of condition-function expression pairs	Similar to a combination of multiple t_if functions. For more information, see Function t_if.	-	-	-	-

Example

Raw log:

```
{"condition1": 0,"condition2": 1, "status": "500"}
```

Processing rule:

```
// If `condition1` is `1` (true), add a field `new` and assign value `1` to it. Her
t_switch(v("condition1"), fields_set("new", 1), v("condition2"), fields_set("new",
```

Processing result:

```
{"new":"2","condition1":"0","condition2":"1","status":"500"}
```

Row Processing Functions

Last updated：2025-04-29 17:22:21

Overview

Row processing functions process log rows, such as filtering, distributing, and splitting log rows.

Function log_output

Function definition

This function is used to output a row of log to a specified log topic. It can be used independently or together with branch conditions.

Syntax description

log_output(Alias). The alias is defined when the processing task is configured.

Field description

Parameter	Description	Type	Required	Default Value	Value Range
alias	Log topic alias	string	Yes	-	-

Sample

Distribute the log to 3 different log topics according to the values (`waring` , `info` , and `error`) of the `loglevel` field.

Raw log:

```
[
  {
    "loglevel": "warning"
  },
  {
    "loglevel": "info"
  },
  {
    "loglevel": "error"
```

```
    }
]
```

Processing rule:

```
// The `loglevel` field has 3 values (`waring`, `info`, and `error`) and
therefore the log is distributed to 3 different log topics accordingly.
t_switch(regex_match(v("loglevel"),regex="info"),log_output("info_log"),regex_m
atch(v("loglevel"),regex="warning"),log_output("warning_log"),regex_match(v("lo
glevel"),regex="error"),log_output("error_log"))
```

Function log_auto_output

Function definition

Output logs to dynamic target topics. For example, if you need to dynamically create multiple target log topics based on the value of the log field "pd" and distribute the corresponding logs to the target log topics. Assuming the value of pd is "CLB", "Ckafka", "COS", "CDN", this function will dynamically create target log topics named "CLB", "Ckafka", "COS", "CDN" and write the related logs into the corresponding topics. Meanwhile, you can configure the index type and storage period for these newly created topics.

Syntax description

```
log_auto_output(topic_name="", logset_name="", index_options="", period=3, storage_t
```

Field description

Parameter	Description	Type	Required	Default Value
topic_name	Log Topic Name	string	y	-

logset_name	Logset Name	string	y	-
index_options	all_index: Enable key-value and full-text indexing no_index: Disable indexing content_index: Enable full-text indexing key_index: Enable key-value indexing	string	n	all_index
period	Storage period, generally the range is 1 to 3600 days 3640 means permanent storage	number	n	3
storage_type	Storage type of log topic, optional values. hot: Standard Storage cold: Infrequent Storage	string	n	hot
hot_period	0: Disable log settlement Non-0: Number of days of standard storage after enabling log settlement HotPeriod must be greater than or equal to 7 and less than Period, effective only when StorageType is hot	number	n	0
tag_dynamic	Add dynamic tags to the log topic. Use with the extract_tag() function to extract tag KV from log fields. For example: tag_dynamic=extract_tag(v("pd"),v("env"),v("team"), v("person"))	string	n	-
tag_static	Add static tags to the log topic. For example: tag_static="Ckafka:test_env,developer_team:MikeWang"	string	n	-

Sample

Raw Log

```
[
```

```
    {
        "pd": "CLB",
        "dateTime": "2023-05-25T00:00:26.579"
    },
    {
        "pd": "Ckafka",
        "time": "2023-05-25T18:00:55.350+08:00"
    },
    {
        "pd": "COS",
        "time": "2023-05-25T00:06:20.314+08:00"
    },
    {
        "pd": "CDN",
        "time": "2023-05-25T00:03:52.051+08:00"
    }
]
```

Processing rules

```
log_auto_output(v("pd"),"My Log Set",index_options="content_index", period=3)
```

Processing results: Four log topics were automatically generated, namely "CLB", "Ckafka", "COS", and "CDN". The log set name is "my log set".

Function log_split

Function definition

This function is used to split a row of log into multiple rows of logs based on the value of a specified field by using a separator and JMES expression.

Syntax description

```
log_split(Field name, sep=",", quote="\\"", jmes="", output="")
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
field	Field to extract	string	Yes	-	-

sep	Separator	string	No	,	Any single character
quote	Characters that enclose the value	string	No	-	-
jmes	JMES expression. For more information, see JMESPath.	string	No	-	-
output	Name of a single field	string	Yes	-	-

Sample

Example 1. Split a log whose `field` has multiple values

```
{"field": "hello Go,hello Java,hello python","status":"500"}
```

Processing rule:

```
// Use the separator "," to split the log into 3 logs.
log_split("field", sep=",",  output="new_field")
```

Processing result:

```
{"new_field":"hello Go","status":"500"}
{"new_field":"hello Java","status":"500"}
{"new_field":"hello python","status":"500"}
```

Example 2. Use a JMES expression to split a log

```
{"field": "{\\"a\\":{\\"b\\":{\\"c\\":{\\"d\\":\\"a,b,c\\"}}}}", "status":
"500"}
```

Processing rule:

```
// The value of `a.b.c.d` is `a,b,c`.
log_split("field", jmes="a.b.c.d",  output="new_field")
```

Processing result:

```
{"new_field":"a","status":"500"}
{"new_field":"b","status":"500"}
{"new_field":"c","status":"500"}
```

Example 3. Split a log that contains a JSON array

```
{"field": "{\\"a\\":{\\"b\\":{\\"c\\":{\\"d\\":[\\"a\\",\\"b\\",\\"c\\"]}}}}",
"status": "500"}
```

Processing rule:

```
log_split("field", jmes="a.b.c.d",  output="new_field")
```

Processing result:

```
{"new_field":"a","status":"500"}
{"new_field":"b","status":"500"}
{"new_field":"c","status":"500"}
```

Function log_drop

Function definition

This function is used to delete logs that meet a specified condition.

Syntax description

```
log_drop(Condition 1)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
condition	Function expression whose return value is of bool type	bool	Yes	-	-

Sample

Delete logs where `status` is `200` and retain other logs.

Raw log:

```
{"field": "a,b,c", "status": "500"}
{"field": "a,b,c", "status": "200"}
```

Processing rule:

```
log_drop(op_eq(v("status"), 200))
```

Processing result:

```
{"field":"a,b,c","status":"500"}
```

Function log_keep

Function definition

This function is used to retain logs that meet a specified condition.

Syntax description

```
log_keep(Condition 1)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
condition	Function expression whose return value is of bool type	bool	Yes	-	-

Sample

Retain logs where `status` is `500` and delete other logs.

Raw log:

```
{"field": "a,b,c", "status": "500"}
{"field": "a,b,c", "status": "200"}
```

Processing rule:

```
log_keep(op_eq(v("status"), 500))
```

Processing result:

```
{"field":"a,b,c","status":"500"}
```

Function log_split_jsonarray_jmes

Function definition

This function is used to split and expand the JSON array in the log according to JMES syntax.

Syntax description

```
log_split_jsonarray_jmes("field", jmes="items", prefix="")
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
field	Field to extract	string	Yes	-	-

Sample

Example 1

Raw log:

```
{"common":"common","result":"{\\"target\\":[{\\"a\\":\\"a\\"},
{\\"b\\":\\"b\\"}]}"}
```

Processing rule:

```
log_split_jsonarray_jmes("result",jmes="target")
fields_drop("result")
```

Processing result:

```
{"common":"common", "a":"a"}
{"common":"common", "b":"b"}
```

Example 2

Raw log:

```
{"common":"common","target":"[{\\"a\\":\\"a\\"},{\\"b\\":\\"b\\"}]"}
```

Processing rule:

```
log_split_jsonarray_jmes("target",prefix="prefix_")
fields_drop("target")
```

Processing result:

```
{"prefix_a":"a", "common":"common"}
{"prefix_b":"b", "common":"common"}
```

Field Processing Functions

Last updated：2025-04-29 17:22:21

Overview

Field processing functions are used to process fields in logs. See the figure below.

Function v

Function definition

This function is used to get the value of a specified field and return the corresponding string.

Syntax description

```
v(Field name)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field name	string	Yes	-	-

Example

Get the value of the "message" field and assign the value to a new field "new_message".
Raw log:

```
{"message": "failed", "status": "500"}
```

Processing rule:

```
fields_set("new_message", v("message"))
```

Processing result:

```
{"message": "failed", "new_message": "failed","status": "500"}
```

Function fields_drop

Function definition

This function is used to delete the fields that meet a specified condition.

Syntax description

```
fields_drop(Field name 1, Field name 2, ..., regex=False,nest=False)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
Variable parameter, which can be a field name or regular expression of the field name	Variable parameter, which can be a field name or regular expression of the field name	string	Yes	-	-
regex	Whether to enable regular expression and use the full match mode	bool	No	False	-
nest	Whether the field is a nested field	bool	No	False	-

Example

Example 1. Delete the field whose name is "field"

Raw log:

```
{"field": "a,b,c", "status": "500"}
```

Processing rule:

```
fields_drop("field")
```

Processing result:

```
{"status":"500"}
```

Example 2. Nested field processing

Raw log:

```
{"condition":"{\\"a\\":\\"aaa\\", \\"c\\":\\"ccc\\",
\\"e\\":\\"eee\\"}","status":"500"}
```

Processing rule:

```
// `nest=True` indicates that the field is a nested field. After `condition.a`
and `condition.c` are deleted, only the `condition.e` field is left.
t_if(if_json(v("condition")), fields_drop("condition.a", "condition.c",
nest=True))
```

Processing result:

```
{"condition":"{\\"e\\":\\"eee\\"}","status":"500"}
```

Example 3. Nested field processing

Raw log:

```
{"App": "thcomm","Message": "{\\"f_httpstatus\\": \\"200\\",\\"f_requestId\\":
\\"2021-11-09 08:40:17.832\\tINFO\\tservices/http_service.go:361\\tbb20ac02-
fcbc-4a56-b1f1-4064853b79da\\",\\"f_url\\":
\\"wechat.wecity.qq.com/trpcapi/MbpsPaymentServer/scanCode\\"}"}
```

Processing rule:

```
// `nest=True` indicates that the filed is a nested field. After
`Message.f_requestId` and `Message.f_url` are deleted, only the `f_httpstatus`
field is left.
t_if(if_json(v("Message")), fields_drop("Message.f_requestId", "Message.f_url",
nest=True))
```

Processing result:

```
{"App":"thcomm","Message":"{\\"f_httpstatus\\":\\"200\\"}"}
```

Function fields_keep

Function definition

This function is used to retain the fields that meet a specified condition.

Syntax description

```
fields_keep(Field name 1, Field name 2, ..., regex=False)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
Variable parameter, which can be a field name or regular expression of the field name	Variable parameter, which can be a field name or regular expression of the field name	string	Yes	-	-
regex	Whether to enable regular expression and use the full match mode	bool	No	False	-

Example

Retain the field whose name is "field" and delete the other fields.

Raw log:

```
{"field": "a,b,c", "status": "500"}
```

Processing rule:

```
fields_keep("field")
```

Processing result:

```
{"field":"a,b,c"}
```

Function fields_pack

Function definition

This function is used to match field names based on a regular expression and encapsulate the matched fields into a new field whose value is in JSON format.

Syntax description

```
fields_pack(Target field name, include=".*", exclude="", drop_packed=False)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range

output	Name of the new field after encapsulation	string	Yes	-	-
include	Regular expression to include the field name	string	No	-	-
exclude	Regular expression to exclude the field name	string	No	-	-
drop_packed	Whether to delete the original fields that are encapsulated	bool	No	False	-

Example

Raw log:

```
{"field_a": "a,b,c","field_b": "abc", "status": "500"}
```

Processing rule:

```
fields_pack("new_field","field.*", drop_packed=False)
```

Processing result:

```
{"new_field":"
{\\"field_a\\":\\"a,b,c\\",\\"field_b\\":\\"abc\\"}","field_a":"a,b,c","field_b
":"abc","status":"500"}
```

Function fields_set

Function definition

This function is used to set field values or add fields.

Syntax description

```
fields_set(Field name 1, Field value 1, Field name 2, Field value 2, mode="overwrit
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
Variable parameter	List of key-value pairs	string	-	-	-

mode	Field overwrite mode	string	No	overwrite	-

Example

Example 1. Change the log level from Info to Waring

Raw log:

```
{"Level": "Info"}
```

Processing rule:

```
fields_set("Level", "Warning")
```

Processing result:

```
{"Level", "Warning"}
```

Example 2. Add two fields: `new` and `new2`

Raw log:

```
{"a": "1", "b": "2", "c": "3"}
```

Processing rule:

```
fields_set("new", v("b"), "new2", v("c"))
```

Processing result:

```
{"a":"1","b":"2","c":"3","new":"2","new2":"3"}
```

Function fields_rename

Function definition

This function is used to rename fields.

Syntax description

```
fields_rename(Field name 1, New field name 1, Field name 2, New field name 2, regex
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range

Variable parameter	List of original-new field name pairs	string	-	-	-
regex	Whether to enable regular expression match for field names. If yes, use a regular expression to match the original field name. If no, use equal match.	bool	No	False	-

Example

Raw log:

```
{"regieeen": "bj", "status": "500"}
```

Processing rule:

```
fields_rename("reg.*", "region", regex=True)
```

Processing result:

```
{"region":"bj","status":"500"}
```

Function has_field

Function definition

If the specified field exists, the function returns `True`. Otherwise, the function returns `False`.

Syntax description

```
has_field(Field name)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field name	string	Yes	-	-

Example

Raw log:

```
{"regiooon": "bj", "status": "500"}
```

Processing rule:

```
t_if(has_field("regiooon"), fields_rename("regiooon", "region"))
```

Processing result:

```
{"region":"bj","status":"500"}
```

Function not_has_field

Function definition

If the field does not exist, the function returns `True` . Otherwise, the function returns `False` .

Syntax description

```
not_has_field(Field name)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
field	Field name	string	Yes	-	-

Example

Raw log:

```
{"status": "500"}
```

Processing rule:

```
t_if(not_has_field("message"), fields_set("no_message", True))
```

Processing result:

```
{"no_message":"TRUE","status":"500"}
```

Function Extract Tag

Function definition

Tag dynamically generated topics. Extract tag values from log fields and use these tag values as tags for dynamically generated topics.

Notes:

Applicable only after the log_auto_output() function.

Among them, the total number of static tags and dynamic tags does not exceed 10 pairs.

Note this statement: `log_auto_output(v("pd"),"My Log Set",index_options="content_index", period=3, tag_static ="Ckafka:test_env,developer:MikeWang", tag_dynamic =extract_tag("pd",v("pd"),"team", v("team")))`

Used tag_Static and tag_dynamic.

Syntax description

```
extract_tag(tag name 1, tag value 1, tag name 2, tag value 2....)
```

Description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
Tag Name	Tag Name	string	Yes	-	-
Tag Value	Tag Value	string	Yes	-	-

Example

Raw log:

```
[
{"pd": "CDN", "team": "test"},
{"pd": "CLS", "team": "product"},
{"pd": "COS", "team": "sales"},
{"pd": "CLS", "team": "test"},
{"pd": "CKafka", "team": "product"}
]
```

Note:

For the same log topic, with the same tag name, only the earliest tag value takes effect. In this example, for the log topic named CLS, "team:produc" is the earliest tag value and it takes effect. The subsequent "team:test" tag does not take effect because their tag names are identical.

Processing rule:

```
log_auto_output(v("pd"),"My Log Set",index_options="content_index",
period=3,tag_static="Ckafka:test_env",tag_dynamic=extract_tag("pd",v("pd"),"tea
```

```
m", v("team")))
```

Processing result:

Generated log topics named COS, CKafka, CLS, and CDN, and tagged these log topics. There is 1 static tag (tag_static) and 2 dynamic tags (tag_dynamic) in the statement, so a total of 3 pairs of tags were generated.

Dictionary and List Functions

Last updated：2025-04-29 17:24:07

Function Dictionary

Function Edit Json

Function Definition

Add/modify/delete key-value pairs in the dictionary, and return dict.

Syntax Description

```
json_edit("result", path="", key="",value="333",index=1,mode="edit")
```

Parameter Description

Parameter	Description	Type	Required	Default Value	Value Range
field	Nested json corresponding key	string	Yes	-	-
path	Need to delete or modify the directory corresponding to the target field. When the field to be deleted or modified is at the first level of nested json, leave it blank; when operating on array elements, fill in up to the key corresponding to the array. Support jmes grammar.	string	No	-	-
key	Target field that needs to be deleted or modified. No need to specify when performing operations on array elements.	string	No	-	-
value	New value to be set, required when modifying a value	string	No	-	-
index	Fill in this field when operating on an array. Array elements start from 1.	number	No	0	-

mode	mode, edit: modify, move: delete, default move	string	No	move	-

Example 1: Modify the Value of the Designated Node

Raw log:

```
{"content":"{\\"p9\\":
[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"CN\\"}","time":"1650440364"}
```

Processing rule:

```
json_edit("content", path="", key="p18", value="hello", mode="edit")
```

Processing result:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"hello\\"}",
"time":"1650440364"}
```

Example 2: Modify the Value of the Designated Node (Multi-Level)

Raw log:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"CN\\",\\"info\\":
{\\"province\\":\\"hubei\\",\\"geo\\":
{\\"long\\":\\"111\\",\\"lati\\":\\"222\\"}}}","time":"1650440364"}
```

Processing rule:

```
json_edit("content", path="info.geo", key="long", value="333", mode="edit")
```

Processing result:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"CN\\",\\"info\\":
{\\"province\\":\\"hubei\\",\\"geo\\":
{\\"long\\":\\"333\\",\\"lati\\":\\"222\\"}}}","time":"1650440364"}
```

Example 3: Delete Node

Raw log:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"CN\\",\\"info\\":
{\\"province\\":\\"hubei\\",\\"geo\\":
{\\"long\\":\\"111\\",\\"lati\\":\\"222\\"}}}","time":"1650440364"}
```

Processing rule:

```
json_edit("content", path="", key="p18", mode="move")
```

Processing result:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"info\\":
{\\"province\\":\\"hubei\\",\\"geo\\":
{\\"long\\":\\"111\\",\\"lati\\":\\"222\\"}}}","time":"1650440364"}
```

Example 4: Delete Array Element

Raw log:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"CN\\",\\"info\\":
{\\"province\\":[\\"hello\\",\\"world\\"],\\"geo\\":{\\"long\\":
[\\"1.0\\",\\"2.0\\"],\\"lati\\":\\"222\\"}}}","time":"1650440364"}
```

Processing rule:

```
json_edit("content", path="info.province", index=1, mode="move")
```

Processing result:

```
{"content":"{\\"p9\\":[\\"0.0\\",\\"0.0\\"],\\"p18\\":\\"CN\\",\\"info\\":
{\\"province\\":[\\"world\\"],\\"geo\\":{\\"long\\":
[\\"1.0\\",\\"2.0\\"],\\"lati\\":\\"222\\"}}}","time":"1650440364"}
```

Function json_select

Function Definition

With a Jmes expression, extract the JSON field value and return the Jmes extracted node value (String).

Syntax Description

```
json_select(data, jmes="")
```

Parameter Description

Parameter	Description	Type	Required	Default Value	Value Range
data	Field value, which can be extracted by other functions.	string	Yes	-	-
jmes	Jmes expression	string	Yes	-	-

Example

Raw log:

```
{"field": "{\\"a\\":{\\"b\\":{\\"c\\":{\\"d\\":\\"success\\"}}}}", "status":
"500"}
```

Processing rule:

```
fields_set("message", json_select(v("field"), jmes="a.b.c.d"))
```

Processing result:

```
{"field":"{\\"a\\":{\\"b\\":{\\"c\\":
{\\"d\\":\\"success\\"}}}}","message":"success","status":"500"}
```

Function xml_to_json

Function Definition

Parse the XML value and convert it to a JSON string. The input value must be in the XML string structure; otherwise, the conversion will fail and return a dict.

Syntax Description

```
xml_to_json(data)
```

Parameter Description

Parameter	Description	Type	Required	Default Value	Value Range
data	Field value.	string	Yes	-	-

Example

Raw log:

```
{"xml_field": "<note><to>B</to><from>A</from><heading>Reminder</heading>
<body>Don't forget me this weekend!</body></note>", "status": "500"}
```

Processing rule:

```
fields_set("json_field", xml_to_json(v("xml_field")))
```

Processing result:

```
{"xml_field":"<note><to>B</to><from>A</from><heading>Reminder</heading>
<body>Don't forget me this weekend!</body></note>","json_field":"
{\\"to\\":\\"B\\",\\"from\\":\\"A\\",\\"heading\\":\\"Reminder\\",\\"body\\":\\
"Don't forget me this weekend!\\"}","status":"500"}
```

Function json_to_xml

Function Definition

Parse JSON string values and convert them to XML strings, and return a String.

Syntax Description

```
json_to_xml(data)
```

Parameter Description

Parameter	Description	Type	Required	Default Value	Value Range
data	Field value.	string	Yes	-	-

Example

Raw log:

```
{"json_field":"
{\\"to\\":\\"B\\",\\"from\\":\\"A\\",\\"heading\\":\\"Reminder\\",\\"body\\":\\
"Don't forget me this weekend!\\"}", "status": "200"}
```

Processing rule:

```
fields_set("xml_field", json_to_xml(v("json_field")))
```

Processing result:

```
{"json_field":"
{\\"to\\":\\"B\\",\\"from\\":\\"A\\",\\"heading\\":\\"Reminder\\",\\"body\\":\\
"Don't forget me this weekend!\\"}","xml_field":"<ObjectNode><to>B</to>
<from>A</from><heading>Reminder</heading><body>Don't forget me this weekend!
</body></ObjectNode>","status":"200"}
```

Function if_json

Function Definition

Check whether it is a JSON string, and return TRUE/FALSE.

Syntax Description

```
if_json(data)
```

Parameter Description

Parameter	Description	Type	Required	Default Value	Value Range
data	Field value.	string	Yes	-	-

Example

Sample 1

Raw log:

```
{"condition":"{\\"a\\":\\"b\\"}","status":"500"}
```

Processing Statement:

```
t_if(if_json(v("condition")), fields_set("new", 1))
```

Processing result:

```
{"new":"1","condition":"{\\"a\\":\\"b\\"}","status":"500"}
```

Sample 2

Raw log:

```
{"condition":"haha","status":"500"}
```

Processing Statement:

```
t_if(if_json(v("condition")), fields_set("new", 1))
```

Processing result:

```
{"condition":"haha","status":"500"}
```

List Functions

Function Get Array

Function Definition

Retrieve the value of the array, return String.

Syntax Description

```
array_get(array, index_position)
```

Parameter Description

Parameter	Description	Type	Required	Default Value	Value Range
Array	Array	string	Yes	-	-
index_position	Get the value at a specific index in an array	int	Yes	-	-

Sample 1

Raw log:

```
{
    "field1": "[1,2,3]"
}
```

Processing rule:

```
fields_set("field2", array_get(v("field1"), 0))
```

Processing result:

```
{"field1":"[1,2,3]","field2":"1"}
```

Sample 2

Raw log:

```
{
```

```
    "field1": "['tom','jerry','bobo']"
}
```

Processing rule:

```
fields_set("field2", array_get(v("field1"), 0))
```

Processing result:

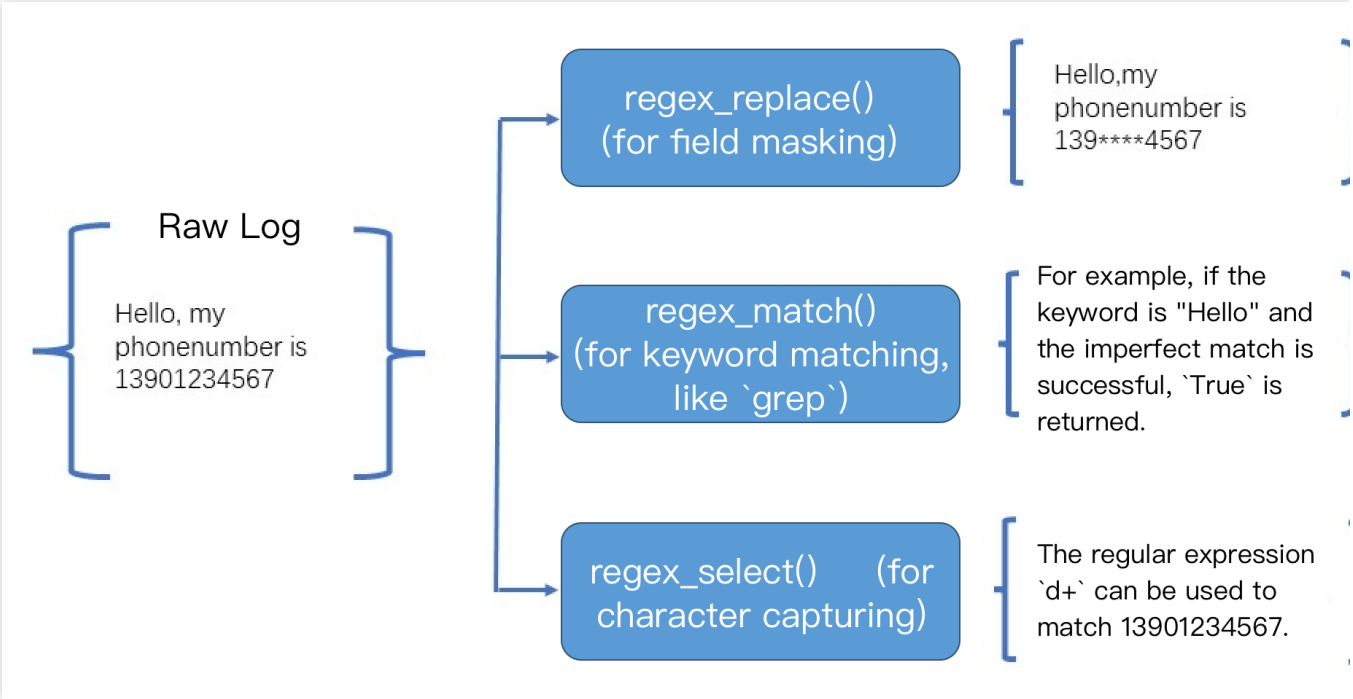
```
{"field1":"['tom','jerry','bobo']","field2":"tom"}
```

Regular Expression Processing Functions

Last updated：2024-01-20 17:44:35

Overview

Logs contain a large volume of text. When processing text, you can use regular expression functions to flexibly extract keywords, mask fields, or determine whether the text contains specified characters. See the figure below.



For examples of regular expressions commonly used in log scenarios, visit Online Test of Regular Expressions.

Purpose	Raw Log	Regular Expression	Extraction Result
Extract content in braces.	`[2021-11-24 11:11:08,232] [328495eb-b562-478f-9d5d-3bf7e] [INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}`	\\{[^\\}]+\\}	{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 10], "orderField": "createTime"}

Extract content in brackets.	`[2021-11-24 11:11:08,232] [328495eb-b562-478f-9d5d-3bf7e] [INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}`	\\[\\S+\\]	[328495eb-b562-478f-9d5d-3bf7e] [INFO]
Extract time.	`[2021-11-24 11:11:08,232] [328495eb-b562-478f-9d5d-3bf7e] [INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}`	\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2},\\d{3}	`2021-11-08 11:11:08,232`
Extract uppercase characters of a specific length.	`[2021-11-24 11:11:08,232] [328495eb-b562-478f-9d5d-3bf7e] [INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}`	[A-Z]{4}	INFO
Extract lowercase characters of a specific length.	`[2021-11-24 11:11:08,232] [328495eb-b562-478f-9d5d-3bf7e] [INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468,`	[a-z]{6}	versio passwo timest interf create

	`"interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 15], "orderField": "createTime"}}}`		
Extract letters and digits.	`[2021-11-24 11:11:08,232] [328495eb-b562-478f-9d5d-3bf7e] [INFO] curl -H 'Host: ' http://abc.com:8080/pc/api -d '{"version": "1.0", "user": "CGW", "password": "123", "timestamp": 1637723468, "interface": {"Name": "ListDetail", "para": {"owner": "1253", "limit": [10, 14], "orderField": "createTime"}}}`	([a-z]{3}):([0-9]{4})	com:8080

Function regex_match

Function definition

This function is used to match data in full or partial match mode based on a regular expression and return whether the match is successful.

Syntax description

```
regex_match(Field value, regex="", full=True)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value	string	Yes	-	-
regex	Regular expression	string	Yes	-	-
full	Whether to enable full match. For full match, the entire value must fully match the regular expression. For partial match, only part of the value needs to match the regular expression.	bool	No	True	-

Sample

Example 1. Check whether the regular expression "192.168.*" fully matches the value `192.168.0.1` of the field `IP` (full=True). The `regex_match` function returns `True` for the case of full match.

Raw log:

```
{"IP":"192.168.0.1", "status": "500"}
```

Processing rule:

```
// Check whether the regular expression "192\\.168.*" fully matches the value
`192.168.0.1` of the field `IP` and save the result to the new field `matched`.
t_if(regex_match(v("IP"), regex="192\\.168.*", full=True),
fields_set("matched", True))
```

Processing result:

```
{"IP":"192.168.0.1","matched":"TRUE","status":"500"}
```

Example 2. Check whether the regular expression "192*" partially matches the value `192.168.0.1` of the field `IP` (full=False). The `regex_match` function returns `True` for the case of partial match.

Raw log:

```
{"IP":"192.168.0.1", "status": "500"}
```

Processing rule:

```
t_if(regex_match(v("ip"), regex="192", full=False), fields_set("matched",
True))
```

Processing result:

```
{"IP":"192.168.0.1","matched":"TRUE","status":"500"}
```

Function regex_select

Function definition

This function is used to match data based on a regular expression and returns the corresponding partial match result. You can specify the sequence number of the matched expression and the sequence number of the group to return (partial match + sequence number of the specified matched group). If no data is matched, an empty string is returned.

Syntax description

```
regex_select(Field value, regex="", index=1, group=1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value	string	Yes	-	-
regex	Regular expression	string	Yes	-	-
index	Sequence number of the matched expression in the match result	number	No	First	-
group	Sequence number of the matched group in the match result	number	No	First	-

Sample

Capture different content from a field value based on a regular expression.

Raw log:

```
{"data":"hello123,world456", "status": "500"}
```

Processing rule:

```
fields_set("match_result", regex_select(v("data"), regex="[a-z]+
(\\d+)",index=0, group=0))
fields_set("match_result1", regex_select(v("data"), regex="[a-z]+(\\d+)",
index=1, group=0))
fields_set("match_result2", regex_select(v("data"), regex="([a-z]+)
(\\d+)",index=0, group=0))
fields_set("match_result3", regex_select(v("data"), regex="([a-z]+)
(\\d+)",index=0, group=1))
```

Processing result:

```
{"match_result2":"hello123","match_result1":"world456","data":"hello123,world45
6","match_result3":"hello","match_result":"hello123","status":"500"}
```

Function regex_split

Function definition

This function is used to split a string and return a JSON array of the split strings (partial match).

Syntax description

```
regex_split(Field value, regex=\\"\\", limit=100)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value	string	Yes	-	-
regex	Regular expression	string	Yes	-	-
limit	Maximum array length for splitting. When this length is exceeded, the excessive part will be split, constructed as an element, and added to the array.	number	No	100	-

Sample

Raw log:

```
{"data":"hello123world456", "status": "500"}
```

Processing rule:

```
fields_set("split_result", regex_split(v("data"), regex="\\d+"))
```

Processing result:

```
{"data":"hello123world456","split_result":"
[\\"hello\\",\\"world\\"]","status":"500"}
```

Function regex_replace

Function definition

This function is used to match data based on a regular expression and replace the matched data (partial match).

Syntax description

```
regex_replace(Field value, regex="", replace="", count=0)
```

Parameter description

Parameter	Description	Parameter	Required	Default	Value

		Type		Value	Range
data	Field value	string	Yes	-	-
regex	Regular expression	string	Yes	-	-
replace	Target string, which is used to replace the matched result	string	Yes	-	-
count	Replacement count. The default value is `0`, indicating complete replacement.	number	No	0	-

Sample

Example 1. Replaces a field value based on a regular expression

Raw log:

```
{"data":"hello123world456", "status": "500"}
```

Processing rule:

```
fields_set("replace_result", regex_replace(v("data"), regex="\\d+", replace="",
count=0))
```

Processing result:

```
{"replace_result":"helloworld","data":"hello123world456","status":"500"}
```

Example 2. Mask the user ID, phone number, and IP address

Raw log:

```
{"Id": "dev@12345","Ip": "11.111.137.225","phonenumber": "13912345678"}
```

Processing rule:

```
// Mask the `Id` field. The result is `dev@***45`.
fields_set("Id",regex_replace(v("Id"),regex="\\d{3}", replace="***",count=0))
fields_set("Id",regex_replace(v("Id"),regex="\\S{2}", replace="**",count=1))
// Mask the `phonenumber` field by replacing the middle 4 digits with ****. The
result is `139****5678`.
fields_set("phonenumber",regex_replace(v("phonenumber"),regex="(\\d{0,3})\\d{4}
(\\d{4})", replace="$1****$2"))
// Mask the `Ip` field by replacing the octet with ***. The result is
`11.***137.225`.
fields_set("Ip",regex_replace(v("Ip"),regex="(\\d+\\.)\\d+(\\.\\d+\\.\\d+)",
replace="$1***$2",count=0))
```

Processing result:

```
{"Id":"**v@***45","Ip":"11.***.137.225","phonenumber":"139****5678"}
```

Function regex_findall

Function definition

This function is used to match data based on a regular expression and return a JSON array of the matched data (partial match).

Syntax description

```
regex_findall(Field value, regex="")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value	string	Yes	-	-
regex	Regular expression	string	Yes	-	-

Sample

Raw log:

```
{"data":"hello123world456", "status": "500"}
```

Processing rule:

```
fields_set("result", regex_findall(v("data"), regex="\\d+"))
```

Processing result:

```
{"result":"[\\"123\\",\\"456\\"]","data":"hello123world456","status":"500"}
```

Time Value Processing Functions

Last updated：2025-04-29 17:22:21

Overview

CLS's time processing functions include functions for converting date values to string values, converting time field values to UTC time values and vice versa, and getting the current time.

Function dt_str

Function definition

This function is used to convert a time field value (a date string in a specific format or timestamp) to a target date string of a specified time zone and format.

Syntax description

```
dt_str(Value, format="Formatted string", zone="")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value. For the parsing formats supported, see dateparser.	string	Yes	-	-
format	Formatted date. For more information, see DateTimeFormatter.	string	No	-	-
zone	Default UTC time, without a specified time zone. For time zone definitions, see ZoneId.	string	No	UTC+00:00	-

Example

Raw log:

```
{"date":"2014-04-26 13:13:44 +09:00"}
```

Processing rule:

```
fields_set("result", dt_str(v("date"), format="yyyy-MM-dd HH:mm:ss",
zone="UTC+8"))
```

Processing result:

```
{"date":"2014-04-26 13:13:44 +09:00","result":"2014-04-26 12:13:44"}
```

Function dt_to_timestamp

Function definition

This function is used to convert a time field value (a date string in a specified format; time zone specified) to a UTC timestamp.

Syntax description

```
dt_to_timestamp(Value, zone="")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value. For the parsing formats supported, see dateparser.	string	Yes	-	-
zone	UTC time is used by default, without a time zone specified. If you specify a time zone, make sure that it corresponds to the time field value. Otherwise, a time zone error occurs. For time zone definitions, see ZoneId.	string	No	UTC+00:00	-

Example

Raw log:

```
{"date":"2021-10-26 15:48:15"}
```

Processing rule:

```
fields_set("result", dt_to_timestamp(v("date"), zone="UTC+8"))
```

Processing result:

```
{"date":"2021-10-26 15:48:15","result":"1635234495000"}
```

Function dt_from_timestamp

Function definition

This function is used to convert a timestamp field value to a time string in the specified time zone.

Syntax description

```
dt_from_timestamp(Value, zone="")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Field value. For the parsing formats supported, see dateparser.	string	Yes	-	-
zone	Default UTC time, without a specified time zone. For time zone definitions, see ZoneId.	string	No	UTC+00:00	-

Example

Raw log:

```
{"date":"1635234495000"}
```

Processing rule:

```
fields_set("result", dt_from_timestamp(v("date"), zone="UTC+8"))
```

Processing result:

```
{"date":"1635234495000","result":"2021-10-26 15:48:15"}
```

Function dt_now

Function definition

This function is used to obtain the current datetime of the processing calculation.

Syntax description

```
dt_now(format="Formatted string", zone="")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
format	Formatted date. For more information, see DateTimeFormatter.	string	No	-	-
zone	Default UTC time, without a specified time zone. For time zone definitions, see ZoneId.	string	No	UTC+00:00	-

Example

Raw log:

```
{"date":"1635234495000"}
```

Processing rule:

```
fields_set("now", dt_now(format="yyyy-MM-dd HH:mm:ss", zone="UTC+8"))
```

Processing result: (The actual processing result depends on the system time, and the following is for reference only.)

```
{"date":"1635234495000","now":"2021-MM-dd HH:mm:ss"}
```

Function Custom Cls Log Time

Function definition

Customize log time. A new log time will be generated based on your processing rules. Seconds, milliseconds, microseconds, and nanoseconds are supported. After you click Execute Preview, you can view the field value of __TIMESTAMP__ in the processing result to check whether it has been changed to your expected result.

Syntax description

```
custom_cls_log_time(time)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
time	UTC Timestamp Type. For the definition of time zone, refer to ZoneId. It supports seconds, milliseconds, microseconds, and nanoseconds. For example, 1565064739000.	string	No	UTC+00:00	-

Return value

UTC Timestamp Type. For example, 1565064739000.

Example

Raw log:

```
{"field1": "1","time":"06/Aug/2019 12:12:19"}
```

Processing rules:

```
custom_cls_log_time(dt_to_timestamp(v("time"), zone="UTC+8"))
```

Processing results are for reference only. Specific results are related to the system time:

```
{"__TIMESTAMP__":"1565064739000", "field1":"1", "time":"06/Aug/2019 12:12:19"}
```

String Processing Functions

Last updated：2025-04-29 17:22:22

Overview

String functions support string length calculation, case conversion, string concatenation, substring replacement, substring deletion, character locating, prefix/suffix matching, and more.

Function str_exist

Function definition

Find a substring within the specified range of values and return True or False.

Syntax description

```
str_exist(data1, data2, ignore_upper=False)
```

Parameter Description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data1	Value of string type	String	Yes		-
data2	Value of string type	String	Yes		-
ignore_upper	Is It Case sensitivity	Bool	No	False	-

Example

Raw log:

```
{"data": "cls nihao"}
```

Processing rule:

```
fields_set("result", str_exist(v(data), "nihao"))
```

Processing result:

```
{"result":"true","data":"cls nihao"}
```

Function str_count

Function definition

This function is used to search for a substring in a specified range of a value and return the number of occurrences of the substring.

Syntax description

```
str_count(Value, sub="", start=0, end=-1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
sub	Substring whose number of occurrences you want to count	string	Yes	-	-
start	Start position to search	number	No	0	-
end	End position to search	number	No	-1	-

Example

Raw log:

```
{"data": "warn,error,error"}
```

Processing rule:

```
fields_set("result", str_count(v("data"), sub="err"))
```

Processing result:

```
{"result":"2","data":"warn,error,error"}
```

Function str_len

Function definition

This function is used to return the length of a string.

Syntax description

```
str_len(Value)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-

Example

Raw log:

```
{"data": "warn,error,error"}
```

Processing rule:

```
fields_set("result", str_len(v("data")))
```

Processing result:

```
{"result":"16","data":"warn,error,error"}
```

Function str_uppercase

Function definition

This function is used to convert a string to uppercase.

Syntax description

```
str_uppercase(Value)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range

data	Value of string type	string	Yes	-	-

Example

Raw log:

```
{"data": "warn,error,error"}
```

Processing rule:

```
fields_set("result", str_uppercase(v("data")))
```

Processing result:

```
{"result":"WARN,ERROR,ERROR","data":"warn,error,error"}
```

Function str_lowercase

Function definition

This function is used to convert a string to lowercase.

Syntax description

```
str_lowercase(Value)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-

Example

Raw log:

```
fields_set("result", str_lowercase(v("data")))
```

Processing rule:

```
{"data": "WARN,ERROR,ERROR"}
```

Processing result:

```
{"result":"warn,error,error","data":"WARN,ERROR,ERROR"}
```

Function str_join

Function definition

This function is used to concatenate input values by using a concatenation string.

Syntax description

```
str_join(Concatenation string 1, Value 1, Value 2, ...)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
join	Value of string type	string	Yes	-	-
Value parameter, list of variable parameters	Value of string type	string	Yes	-	-

Example

Raw log:

```
{"data": "WARN,ERROR,ERROR"}
```

Processing rule:

```
fields_set("result", str_join(",", v("data"), "INFO"))
```

Processing result:

```
{"result":"WARN,ERROR,ERROR,INFO","data":"WARN,ERROR,ERROR"}
```

Function str_replace

Function definition

This function is used to replace an old string with a new string.

Syntax description

```
str_replace(Value, old="", new="", count=0)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
old	String to the replaced	string	Yes	-	-
new	Target string after replacement	string	Yes	-	-
count	Maximum replacement count. The default value is `0`, replacing all matched content.	number	No	0	-

Example

Replace "WARN" in the value of the `data` field with "ERROR".

Raw log:

```
{"data": "WARN,ERROR,ERROR"}
```

Processing rule:

```
fields_set("result", str_replace( v("data"), old="WARN", new="ERROR"))
```

Save the replacement result to the new field `result`.

Processing result:

```
{"result":"ERROR,ERROR,ERROR","data":"WARN,ERROR,ERROR"}
```

Function str_format

Function definition

This function is used to format strings.

Syntax description

```
str_format(Formatted string, Value 1, Value 2, ...)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
format	Target format, using "{}" as placeholders, such as "The disk "{1}" contains {0} file(s).". The numbers in "{}" correspond to the sequence numbers of the parameter values, and the numbers start from 0. For usage details, see MessageFormat.format.	string	Yes	-	-
Value parameter, list of variable parameters	Value of string type	string	Yes	-	-

Example

Raw log:

```
{"status": 200, "message":"OK"}
```

Processing rule:

```
fields_set("result", str_format("status:{0}, message:{1}", v("status"),
v("message")))
```

Processing result:

```
{"result":"status:200, message:OK","message":"OK","status":"200"}
```

Function str_strip

Function definition

This function is used to delete specified characters from a string concurrently from the start and end of the string and return the remaining part.

Syntax description

```
str_strip(Value, chars="\\t\\r\\n")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
chars	String to delete	string	No	\\t\\r\\n	-

Example

Example 1

Raw log:

```
{"data": " abc  "}
```

Processing rule:

```
fields_set("result", str_strip(v("data"), chars=" "))
```

Processing result:

```
{"result":"abc","data":" abc  "}
```

Example 2

Raw log:

```
{"data": " **abc**  "}
```

Processing rule:

```
fields_set("result", str_strip(v("data"), chars=" *"))
```

Processing result:

```
{"result":"abc","data":" **abc**  "}
```

Function str_lstrip

Function definition

This function is used to delete specified characters from a string from the start of the string and return the remaining part.

Syntax description

```
str_strip(Value, chars="\\t\\r\\n")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
chars	String to delete	string	No	\\t\\r\\n	-

Example

Raw log:

```
{"data": " abc  "}
```

Processing rule:

```
fields_set("result", str_lstrip(v("data"), chars=" "))
```

Processing result:

```
{"result":"abc  ","data":" abc  "}
```

Function str_rstrip

Function definition

This function is used to delete specified characters from a string from the end of the string and return the remaining part.

Syntax description

```
str_strip(Value, chars="\\t\\r\\n")
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
chars	String to delete	string	No	\\t\\r\\n	-

Example

Raw log:

```
{"data": " abc  "}
```

Processing rule:

```
fields_set("result", str_rstrip(v("data"), chars=" "))
```

Processing result:

```
{"result":" abc","data":" abc  "}
```

Function str_find

Function definition

This function is used to check whether a string contains a specified substring and return the position of the substring in the string.

Syntax description

```
str_find(Value, sub="", start=0, end=-1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
sub	Substring whose number of occurrences you want to count	string	Yes	-	-
start	Start position to search	number	No	0	-
end	End position to search	number	No	-1	-

Example

Raw log:

```
{"data": "warn,error,error"}
```

Processing rule:

```
fields_set("result", str_find(v("data"), sub="err"))
```

Processing result:

```
{"result":"5","data":"warn,error,error"}
```

Function str_start_with

Function definition

This function is used to check whether a string starts with a specified prefix.

Syntax description

```
str_start_with(Value, sub="", start=0, end=-1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
sub	Prefix string or character	string	Yes	-	-
start	Start position to search	number	No	0	-
end	End position to search	number	No	-1	-

Example

Example 1

Raw log:

```
{"data": "something"}
```

Processing rule:

```
fields_set("result", str_start_with(v("data"), sub="some"))
```

Processing result:

```
{"result":"true","data":"something"}
```

Example 2

Raw log:

```
{"data": "something"}
```

Processing rule:

```
fields_set("result", str_start_with(v("data"), sub="*"))
```

Processing result:

```
{"result":"false","data":"something"}
```

Function str_end_with

Function definition

This function is used to check whether a string starts with a specified prefix.

Syntax description

```
str_end_with(Value, sub="", start=0, end=-1)
```

Parameter description

Parameter	Description	Parameter Type	Required	Default Value	Value Range
data	Value of string type	string	Yes	-	-
sub	Prefix string or character	string	Yes	-	-
start	Start position to search	number	No	0	-
end	End position to search	number	No	-1	-

Example

Raw log:

```
{"data": "endwith something"}
```

Processing rule:

```
fields_set("result", str_end_with(v("data"), sub="ing"))
```

Processing result:

```
{"result":"true","data":"endwith something"}
```

Type Conversion Functions

Last updated：2024-01-20 17:44:35

Overview

Type conversion functions provide commonly type conversion features. They can be used to convert field values to the int, float, bool, and Str types.

Function ct_int

Function definition

This function is used to convert a value (whose base can be specified) to a decimal integer.

Syntax description

```
ct_int(Value 1, base=10)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
base	Base	number	No	10	[2-36]

Examples

Example 1

Raw log:

```
{"field1": "10"}
```

Processing rule:

```
fields_set("result", ct_int(v("field1")))
```

Processing result:

```
{"result":"10","field1":"10"}
```

Example 2

Raw log:

```
{"field1": "AB"}
```

Processing rule:

```
fields_set("result", ct_int(v("field1"), 16))
```

Processing result:

```
{"result":"171","field1":"AB"}
```

Function ct_float

Function definition

This function is used to convert a value to a floating-point number.

Syntax description

```
ct_float(Value)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Examples

Raw log:

```
{"field1": "123"}
```

Processing rule:

```
fields_set("result", ct_float(v("field1")))
```

Processing result:

```
{"result":"123.0","field1":"123"}
```

Function ct_str

Function definition

This function is used to convert a value to a string.

Syntax description

```
ct_str(Value)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Examples

Raw log:

```
{"field1": 123}
```

Processing rule:

```
fields_set("result", ct_str(v("field1")))
```

Processing result:

```
{"result":"123","field1":"123"}
```

Function ct_bool

Function definition

This function is used to convert a value to a Boolean value.

Syntax description

```
ct_bool(Value)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Examples

Example 1

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", ct_bool(0))
```

Processing result:

```
{"result":"false"}
```

Example 2

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", ct_bool(1))
```

Processing result:

```
{"result":"true"}
```

Example 3

Raw log:

```
{"field1": 1}
```

Processing rule:

```
fields_set("result", ct_bool(v("field1")))
```

Processing result:

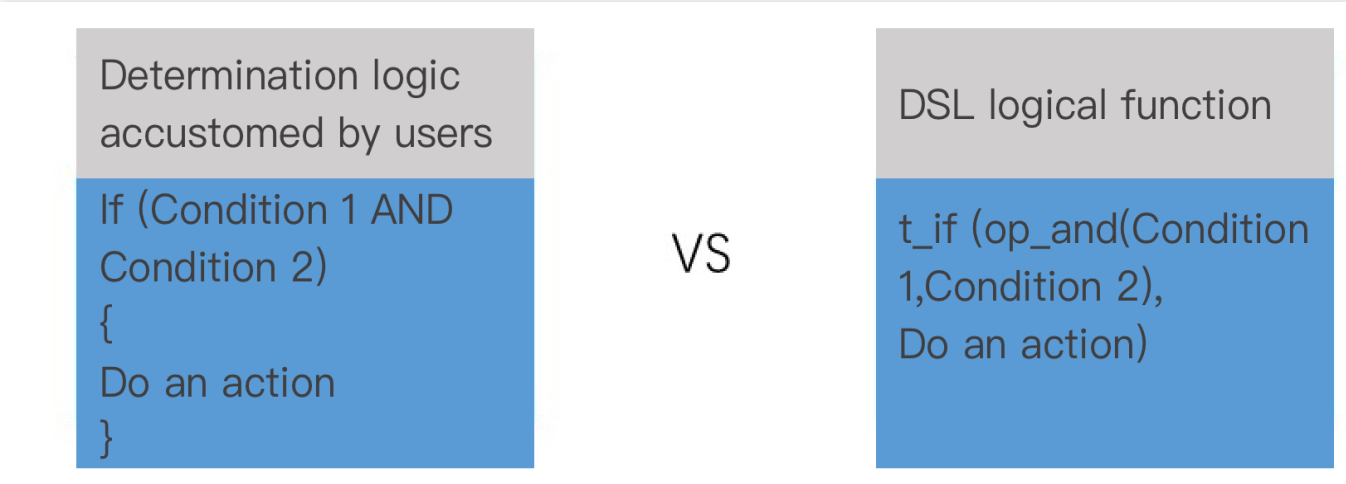
```
{"result":"true","field1":"1"}
```

Logical and Mathematical Functions

Last updated：2024-01-20 17:44:35

Overview

Logic and arithmetic functions include AND, OR, greater than, less than, equal to, addition, subtraction, multiplication, division, and modulus operation functions. Their writing method is slightly different from that of commonly used programming languages, as shown in the figure below.



Function op_if

Function definition

This function is used to return a value based on a specified condition.

Syntax description

```
op_if(Condition 1, Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
condition	Condition expression	bool	Yes	-	-
data1	If the condition is `True` , the value of this	string	Yes	-	-

	parameter is returned.				
data2	If the condition is `False`, the value of this parameter is returned.	string	Yes	-	-

Sample

Example 1

Raw log:

```
{"data": "abc"}
```

Processing rule:

```
fields_set("result", op_if(True, v("data"), "false"))
```

Processing result:

```
{"result":"abc","data":"abc"}
```

Example 2

Raw log:

```
{"data": "abc"}
```

Processing rule:

```
fields_set("result", op_if(False, v("data"), "123"))
```

Processing result:

```
{"result":"123","data":"abc"}
```

Function op_and

Function definition

This function is used to perform the AND operation on values. If all the specified parameter values are evaluated to true, `True` is returned. Otherwise, `False` is returned.

Syntax description

```
op_and(Value 1, Value 2, ...)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
Variable parameter list	Parameters or expressions that participate in the calculation	string	Yes	-	-

Sample

Example 1

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_and(True, False))
```

Processing result:

```
{"result":"false"}
```

Example 2

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_and(1, 1))
```

Processing result:

```
{"result":"true"}
```

Example 3

Raw log:

```
{"data":"false"}
```

Processing rule:

```
fields_set("result", op_and(1, v("data")))
```

Processing result:

```
{"result":"false","data":"false"}
```

Function op_or

Function definition

This function is used to perform the OR operation on values. If one or more of the specified parameter values are evaluated to false, `False` is returned. Otherwise, `True` is returned.

Syntax description

```
op_or(Value 1, Value 2, ...)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
Variable parameter list	Parameters or expressions that participate in the calculation	string	Yes	-	-

Sample

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_or(True, False))
```

Processing result:

```
{"result":"true"}
```

Function op_not

Function definition

This function is used to perform the NOT operation on values.

Syntax description

```
op_not(Value)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data	Value of any type	any	Yes	-	-

Sample

Example 1

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_not(True))
```

Processing result:

```
{"result":"false"}
```

Example 2

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_not("True"))
```

Processing result:

```
{"result":"false"}
```

Function op_eq

Function definition

This function is used to compare two values. If the values are equal, `True` is returned.

Syntax description

```
op_eq(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Example 1. Determine whether the values of the `Post` and `Get` fields are equal

Raw log:

```
{"Post": "10", "Get": "11"}
```

Processing rule:

```
fields_set("result", op_eq(v("Post"), v("Get")))
```

Save the result to `result` .

Processing result:

```
{"result":"false","Post":"10","Get":"11"}
```

Example 2. Determine whether the values of the `field1` and `field2` fields are equal

Raw log:

```
{"field1": "1", "field2": "1"}
```

Processing rule:

```
fields_set("result", op_eq(v("field1"), v("field2")))
```

Processing result:

```
{"result":"true","field1":"1","field2":"1"}
```

Function op_ge

Function definition

This function is used to compare two values. If `Value 1` is greater than or equal to `Value 2`, `True` is returned.

Syntax description

```
op_ge(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Example 1

Raw log:

```
{"field1": "20", "field2": "9"}
```

Processing rule:

```
fields_set("result", op_ge(v("field1"), v("field2")))
```

Processing result:

```
{"result":"true","field1":"20","field2":"9"}
```

Example 2

Raw log:

```
{"field1": "2", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_ge(v("field1"), v("field2")))
```

Processing result:

```
{"result":"true","field1":"2","field2":"2"}
```

Function op_gt

Function definition

This function is used to compare two values. If `Value 1` is greater than `Value 2`, `True` is returned.

Syntax description

```
op_gt(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Raw log:

```
{"field1": "20", "field2": "9"}
```

Processing rule:

```
fields_set("result", op_ge(v("field1"), v("field2")))
```

Processing result:

```
{"result":"true","field1":"20","field2":"9"}
```

Function op_le

Function definition

This function is used to compare two values. If `Value 1` is less than or equal to `Value 2`, `True` is returned.

Syntax description

```
op_le(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Raw log:

```
{"field1": "2", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_le(v("field1"), v("field2")))
```

Processing result:

```
{"result":"true","field1":"2","field2":"2"}
```

Function op_lt

Function definition

This function is used to compare two values. If `Value 1` is less than `Value 2`, `True` is returned.

Syntax description

```
op_lt(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to	number	Yes	-	-

	a numeric value				
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Raw log:

```
{"field1": "2", "field2": "3"}
```

Processing rule:

```
fields_set("result", op_lt(v("field1"), v("field2")))
```

Processing result:

```
{"result":"true","field1":"2","field2":"3"}
```

Function op_add

Function definition

This function is used to return the sum of two specified values.

Syntax description

```
op_add(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Raw log:

```
{"field1": "1", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_add(v("field1"), v("field2")))
```

Processing result:

```
{"result":"3","field1":"1","field2":"2"}
```

Function op_sub

Function definition

This function is used to return the difference between two specified values.

Syntax description

op_sub(Value 1, Value 2)

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Raw log:

```
{"field1": "1", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_sub(v("field1"), v("field2")))
```

Processing result:

```
{"result":"-1","field1":"1","field2":"2"}
```

Function op_mul

Function definition

This function is used to return the product of two specified values.

Syntax description

```
op_mul(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Raw log:

```
{"field1": "1", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_mul(v("field1"), v("field2")))
```

Processing result:

```
{"result":"2","field1":"1","field2":"2"}
```

Function op_div

Function definition

This function is used to return the quotient of two specified values.

Syntax description

```
op_div(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Example 1

Raw log:

```
{"field1": "1", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_div(v("field1"), v("field2")))
```

Processing result:

```
{"result":"0","field1":"1","field2":"2"}
```

Example 2

Raw log:

```
{"field1": "1.0", "field2": "2"}
```

Processing rule:

```
fields_set("result", op_div(v("field1"), v("field2")))
```

Processing result:

```
{"result":"0.5","field1":"1.0","field2":"2"}
```

Function op_sum

Function definition

This function is used to return the sum of multiple specified values.

Syntax description

```
op_sum(Value 1, Value 2, ...)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
Variable parameter list	Numeric value or string that can be converted to a numeric value	string	Yes	-	-

Sample

Raw log:

```
{"field1": "1.0", "field2": "10"}
```

Processing rule:

```
fields_set("result", op_sum(v("field1"), v("field2")))
```

Processing result:

```
{"result":"11.0","field1":"1.0","field2":"10"}
```

Function op_mod

Function definition

This function is used to return the remainder of a specified value divided by the other specified value.

Syntax description

```
op_mod(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Example 1

Raw log:

```
{"field1": "1.0", "field2": "0"}
```

Processing rule:

```
fields_set("result", op_mod(v("field1"), v("field2")))
```

Processing result:

```
{"result":"2","field1":"1","field2":"2"}
```

Example 2

Raw log:

```
{"field1": "1.0", "field2": "5"}
```

Processing rule:

```
fields_set("result", op_mod(v("field1"), v("field2")))
```

Processing result:

```
{"result":"1.0","field1":"1.0","field2":"5"}
```

Example 3

Raw log:

```
{"field1": "6", "field2": "4"}
```

Processing rule:

```
fields_set("result", op_mod(v("field1"), v("field2")))
```

Processing result:

```
{"result":"2","field1":"6","field2":"4"}
```

Function op_null

Function definition

This function is used to check whether a value is `null`. If so, `true` is returned; otherwise, `false` is returned.

Syntax description

```
op_null(Value)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data	Value of any type	any	Yes	-	-

Sample

Example 1

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_null("null"))
```

Processing result:

```
{"result":"true"}
```

Example 2

Raw log:

```
{"data": null}
```

Processing rule:

```
fields_set("result", op_null(v("data")))
```

Processing result:

```
{"data": "null", "result":"true"}
```

Function op_notnull

Function definition

This function is used to check whether a value is not `null`. If so, `true` is returned; otherwise, `false` is returned.

Syntax description

```
op_notnull(Value)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data	Value of any type	any	Yes	-	-

Sample

Example 1

Raw log:

```
{}
```

Processing rule:

```
fields_set("result", op_notnull("null"))
```

Processing result:

```
{"result":"false"}
```

Example 2

Raw log:

```
{"data": null}
```

Processing rule:

```
fields_set("result", op_notnull(v("data")))
```

Processing result:

```
{"data": "null", "result":"false"}
```

Function op_str_eq

Function definition

This function is used to compare string values. If the values are equal, `true` is returned.

Syntax description

```
op_str_eq(Value 1, Value 2, ignore_upper=False)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	String value	string	Yes	-	-
data2	String value	string	Yes	-	-
ignore_upper	Case sensitivity	bool	No	False	-

Sample

Example 1

Raw log:

```
{"field": "cls"}
```

Processing rule:

```
fields_set("result", op_str_eq(v("field"), "cls"))
```

Processing result:

```
{"result":"true","field":"cls"}
```

Example 2

Raw log:

```
{"field": "cls"}
```

Processing rule:

```
fields_set("result", op_str_eq(v("field"), "etl|cls|data"))
```

Processing result:

```
{"result":"true","field":"cls"}
```

Example 3

Raw log:

```
{"field": "CLS"}
```

Processing rule:

```
fields_set("result", op_str_eq(v("field"), "cls", ignore_upper=True))
```

Processing result:

```
{"result":"true","field":"CLS"}
```

Example 4

Raw log:

```
{"field": "CLS"}
```

Processing rule:

```
fields_set("result", op_str_eq(v("field"), "etl|cls|data", ignore_upper=True))
```

Processing result:

```
{"result":"true","field":"CLS"}
```

Function random

Function definition

This function is used to generate a random number between two values. The value range is left-closed and right-closed.

Syntax description

```
random(Value 1, Value 2)
```

Field description

Parameter	Description	Type	Required	Default Value	Value Range
data1	Numeric value or string that can be converted to a numeric value	number	Yes	-	-
data2	Numeric value or string that can be converted to a numeric value	number	Yes	-	-

Sample

Example 1

Raw log:

```
{"field1": "1"}
```

Processing rule:

```
log_keep(op_eq(random(1, 5), 3))
```

Processing result:

```
{"field1": "1"}
```

Example 2

Raw log:

```
{"field1": "1"}
```

Processing rule:

```
fields_set("field2", random(1, 5))
```

Processing result:

```
{"field1":"1", "field2":"4"}
```

Encoding and Decoding Functions

Last updated：2025-04-30 09:45:11

Function decode_url

Function definition

This function is used to decode an encoded URL.

Syntax description

```
decode_url(value)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
url	URL value	string	Yes	-	-

Sample

Raw log:

```
{"url":"https%3A%2F%2Fcloud.tencent.com%2F"}
```

Processing rule:

```
fields_set("result",decode_url(v("url")))
```

Processing result:

```
{"result":"https://cloud.tencent.com/","url":"https%3A%2F%2Fcloud.tencent.com%2
F"}
```

Function md5_encoding

Function definition

This function is used to calculate and return the MD5 checksum.

Syntax description

```
md5_encoding(value)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
Value	The data for which to calculate the MD5 checksum	String	Yes	-	-

Sample

Raw log:

```
{"field": "haha"}
```

Processing rule:

```
fields_set("field", md5_encoding(v("field")))
```

Processing result:

```
{"field":"4e4d6c332b6fe62a63afe56171fd3725"}
```

Function uuid

Function definition

This function is used to generate a universally unique identifier (UUID).

Syntax description

```
uuid()
```

Parameter description

No input parameters

Sample

Raw log:

```
{"key":"value"}
```

Processing rule:

```
fields_set("field",uuid())
```

Processing result:

```
{"field":"8c2db704-45c0-4ea1-9e2c-cf9c966e35cd","key":"value"}
```

Function str_encode

Function definition

Encode a string in the specified format.

Syntax description

```
str_encode(data, encoding="utf8", errors="ignore")
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	Data to be encoded	String	Yes	-	-
encoding	Encoding format, utf8 by default, supporting ASCII, latin1, and unicode-escape.	String	No	utf8	-
errors	Handling method when encoding format cannot recognize characters: ignore (default): Ignore and do not encode. Strict control: Report an error directly and discard this log data. replace: Use a half-angle question mark (?), and replace the unrecognized part. xmlcharrefreplace: Use corresponding XML character references to replace unrecognized parts	String	No	ignore	-

Sample

Raw log:

```
{"field1": "asd encode encode \\\\u1234"}
```

Processing rule:

```
fields_set("field2", str_decode(str_encode(v("field1"), "unicode-escape"),
"unicode-escape"))
```

Processing result:

```
{"field1":"asd encode encode \\\\u1234","field2":"asd code code ሴ"}
```

Function Str_decode

Function definition

Decode a string in the specified format.

Syntax description

```
str_decode(data, encoding="utf8", errors="ignore")
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	Data to be decoded	String	Yes	-	-
encoding	Encoding format, utf8 by default, supporting ASCII, latin1, and unicode-escape.	String	No	utf8	-
errors	Handling method when encoding format cannot recognize characters: ignore (default): Ignore and do not decode. Strict control: Report an error directly and discard this log data. replace: Replace the undecodable part with a half-width question mark (?).	String	No	ignore	-

	xmlcharrefreplace: Use corresponding XML characters to replace undecodable parts				

Sample

Raw log:

```
{"field1": "Test in English and Chinese: qwertyuiopasdfghjklzxcvbnm QWERTYUIOPASDFGHJKLZXCVBNM special symbols:]] [! @#$%^&*()_++~"}
```

Processing rule:

```
fields_set("field2", str_decode(str_encode(v("field1"))))
```

Processing result:

```
{"field1":"Test in English and Chinese: qwertyuiopasdfghjklzxcvbnm QWERTYUIOPASDFGHJKLZXCVBNM Special Symbols:]] [! @#$%^&*()_++~", "field2":"Test in English and Chinese: qwertyuiopasdfghjklzxcvbnm QWERTYUIOPASDFGHJKLZXCVBNM special symbols:]] [! @#$%^&*()_++~"}
```

Function Base64 Encode

Function definition

Encode the string in base64.

Syntax description

```
base64_encode(value, format="RFC3548")
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
value	String to be encoded	string	Yes	-	-
format	Encoding format, supports RFC4648 (default), RFC3548	string	No	RFC4648	-

Sample

Raw log:

```
{"field": "hello world"}
```

Processing rule:

```
fields_set("encode", base64_encode(v("field")))
```

Processing result:

```
{"encode":"aGVsbG8gd29ybGQ=", "field":"hello world"}
```

Function Base64 Decode

Function definition

Decode the string in base64.

Syntax description

```
base64_decode(value, format="RFC3548")
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
value	The string to be decoded	string	Yes	-	-
format	Decoding format, supports RFC4648 (default), RFC3548	string	No	RFC4648	-

Sample

Raw log:

```
{"field": "aGVsbG8gd29ybGQ="}
```

Processing rule:

```
fields_set("decode", base64_decode(v("field")))
```

Processing result:

```
{"decode":"hello world", "field":"aGVsbG8gd29ybGQ="}
```

IP Parsing Functions

Last updated：2024-01-20 17:44:35

Function geo_parse

Function definition

This function is used to parse the geographical location.

Syntax description

```
geo_parse(field value, keep=("country","province","city"), ip_sep=",")
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
data	IP value. Separate multiple IPs by separator.	string	Yes	-	-
keep	The field to be reserved	string	No	("country","province","city")	-
ip_sep	The number of the expression in the match result	string	No	-	-

Sample

Example 1

Raw log:

```
{"ip":"101.132.57.150"}
```

Processing rule:

```
fields_set("result", geo_parse(v("ip")))
```

Processing result:

```
{"ip":"101.132.57.150","result":"
{\\"country\\":\\"China\\",\\"province\\":\\"Shanghai\\",\\"city\\":\\"Shanghai
\\"}"}
```

Example 2

Raw log:

```
{"ip":"101.132.57.150,101.14.57.157"}
```

Processing rule:

```
fields_set("result", geo_parse(v("ip"),keep="province,city",ip_sep=","))
```

Processing result:

```
{"ip":"101.132.57.150,101.14.57.157", "result":"{\\"101.14.57.157\\":
{\\"province\\":\\"Taiwan\\",\\"city\\":\\"NULL\\"},\\"101.132.57.150\\":
{\\"province\\":\\"Shanghai\\",\\"city\\":\\"Shanghai\\"}}"}
```

Function is_subnet_of

Function definition

This function is used to check whether an IP is in the target IP range. Multiple IP ranges are supported.

Syntax description

```
is_subnet_of(IP range list, IP)
```

Parameter description

Parameter	Description	Type	Required	Default Value	Value Range
IP range list	IP range. Separate multiple IP ranges by comma.	String	Yes	-	-
IP	The IP to be checked	String	Yes	-	-

Sample

Example 1

Raw log:

```
{"ip": "192.168.1.127"}
```

Processing rule:

```
log_keep(is_subnet_of("192.168.1.64/26",v("ip")))
```

Processing result:

```
{"ip": "192.168.1.127"}
```

Example 2

Raw log:

```
{"ip": "192.168.1.127"}
```

Processing rule:

```
fields_set("is_subnet",is_subnet_of("192.168.1.64/26",v("ip")))
```

Processing result:

```
{"ip": "192.168.1.127", "is_subnet":"true"}
```

Example 3

Raw log:

```
{"ip": "192.168.1.127"}
```

Processing rule:

```
fields_set("is_subnet",is_subnet_of("172.16.0.0/16",v("ip")))
```

Processing result:

```
{"ip": "192.168.1.127", "is_subnet":"false"}
```

Example 4

Raw log:

```
{"ip": "192.168.1.127"}
```

Processing rule:

```
fields_set("is_subnet",is_subnet_of("172.16.0.0/16,192.168.1.64/26",v("ip")))
```

Processing result:

```
{"ip": "192.168.1.127", "is_subnet":"true"}
```

Processing Cases
Case Overview

Last updated：2024-12-18 16:38:39

Overview

The cases in this document will help you get a general and perceptual understanding of data processing. You can write your own data processing scripts according to the examples mentioned below .

Common Mistakes

v **function**: v("Field A"), which means "value of 'Field A'", that is, the value of Field A. Among the parameters of data processing functions, some are fields, such as fields_drop("Field A"), and some are **field values**. In such cases, the v function needs to be used to obtain the values. For example, str_len(v("Field A")). A common mistake made by beginners is that when the parameter is a field value, they fail to use the **v function** to obtain the field value, resulting in the failure of the function execution.
Log distribution: the **target log topics** and their **target names** need to be configured in advance. The target names will be used in the **distribution functions**.
fields_set function: the `fields_set` function is used to set field values and store the content processed by data processing function. For example, `fields_set("A+B",op_add(v("Field A"),v("Field B")))` is to add the values of fields A and B, and the op_add function needs to leverage the `fields_set` function to complete result writing and storage.

Overview

You can refer to the following cases to complete your data processing:

Log Structuring - Single-line Text
Log Structuring - JSON
Log Structuring - Multiple Formats
Log Structuring - Separator
Log Structuring - Grok
Log Distribution
Log Masking
Log Filtering

Single-Line Text Log Structuration

Last updated：2024-01-20 17:44:35

Use Case

Tom has collected a log to CLS. The log does not use a fixed separator and is in single-line text format. Now Tom wants to structuralize the log and **extract the log time, log level, operation, and URL information** from the text for subsequent search and analysis.

Scenario Analysis

According to Tom's requirements, the processing ideas are as follows:1.The content in {...} is the detailed information of operations and can be extracted with a regular expression.2. Use a regular expression to extract the **log time**, **log level**, and **URL**.

Raw Log

```
{
    "content": "[2021-11-24 11:11:08,232][328495eb-b562-478f-9d5d-3bf7e][INFO]
curl -H 'Host: ' http://abc.com:8080/pc/api -d {\\"version\\":
\\"1.0\\",\\"user\\": \\"CGW\\",\\"password\\": \\"123\\",\\"interface\\":
{\\"Name\\": \\"ListDetail\\",\\"para\\": {\\"owner\\":
\\"1253\\",\\"orderField\\": \\"createTime\\"}}}"
}
```

DSL Processing Function

```
fields_set("Action",regex_select(v("content"),regex="\\
{[^\\}]+\\}",index=0,group=0))
fields_set("loglevel",regex_select(v("content"),regex="\\[[A-Z]
{4}\\]",index=0,group=0)).
fields_set("logtime",regex_select(v("content"),regex="\\d{4}-\\d{2}-\\d{2}
\\d{2}:\\d{2}:\\d{2},\\d{3}",index=0,group=0))
fields_set("Url",regex_select(v("content"),regex="([a-z]{3}).([a-z]{3}):([0-9]
{4})",index=0,group=0))
```

```
fields_drop("content")
```

DSL Processing Function Details

1. Create a field named **Action** and use the regular expression {[^}]+} to match **{...}**.

```
fields_set("Action",regex_select(v("content"),regex="\\
{[^\\}]+\\}",index=0,group=0))
```

2. Create a field named **loglevel** and use the regular expression [A-Z]{4} to match **INFO**.

```
fields_set("loglevel",regex_select(v("content"),regex="\\[[A-Z]
{4}\\]",index=0,group=0)).
```

3. Create a field named **logtime** and use the regular expression d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2},\\d{3} to match **2021-11-24 11:11:08**.

```
fields_set("logtime",regex_select(v("content"),regex="\\d{4}-\\d{2}-\\d{2}
\\d{2}:\\d{2}:\\d{2},\\d{3}",index=0,group=0))
```

4. Create a field named **Url**, use the regular expression [a-z]{3}.[a-z]{3} to match **abc.com**, and use [0-9]{4} to match **8080**.

```
fields_set("Url",regex_select(v("content"),regex="([a-z]{3}).([a-z]{3}):([0-9]
{4})",index=0,group=0))
```

5. Discard the **content** field.

```
fields_drop("content")
```

Processing Result

```
{"Action":"{\\"version\\": \\"1.0\\",\\"user\\": \\"CGW\\",\\"password\\":
\\"123\\",\\"interface\\": {\\"Name\\": \\"ListDetail\\",\\"para\\":
{\\"owner\\": \\"1253\\",\\"orderField\\":
\\"createTime\\"}","Url":"abc.com:8080","loglevel":"[INFO]","logtime":"2021-11-
24 11:11:08,232"}
```

Log Structuration-JSON

Last updated：2024-12-18 16:37:58

Scenario description

Xiaowang collects logs in JSON format to CLS (Cloud Log Service, CLS) under the following two conditions:
1. The JSON is multi-layer nested. Xiaowang wants to extract the user and App fields, where user is a secondary nested field.
2. Xiaowang's JSON log is an array and needs to split multiple logs from the array.

Raw Log

Log 1: Nested JSON
Log 2: JSON Array

```
[
    {
        "content": {
            "App": "App-1",
            "start_time": "2021-10-14T02:15:08.221",
            "resonsebody": {
                "method": "GET",
                "user": "Tom"
            },
            "response_code_details": "3000",
            "bytes_sent": 69
        }
    },
    {
        "content": {
            "App": "App-2",
            "start_time": "2222-10-14T02:15:08.221",
            "resonsebody": {
                "method": "POST",
                "user": "Jerry"
            },
            "response_code_details": "2222",
            "bytes_sent": 1
        }
    }
]
```

```
{
    "timestamp": 1732099684144000,
    "topic": "log-containers",
    "records": [
        {
            "category": "kube-request",
            "log": "{\\"requestID\\":\\"12345\\",\\"stage\\":\\"Complete\\"}"
        },
        {
            "category": "db-request",
            "log": "{\\"requestID\\":\\"67890\\",\\"stage\\":\\"Response\\"}"
        }
    ]
}
```

Processing result

Log 1

Log 2

```
[{"App":"App-1","user":"Tom"},
{"App":"App-2","user":"Jerry"}]
```

```
[
{
"category":"kube-request",
"requestID":"12345",
"stage":"Complete",
"timestamp":"1732099684144000",
"topic":"log-containers"
},
{
"category":"db-request",
"requestID":"67890",
"stage":"Response",
"timestamp":"1732099684144000",
"topic":"log-containers"
}
]
```

DSL Processing Function

Log 1:Processing statement

Log 2:Processing statement

```
//Use the ext_json function to extract structured data from JSON data, by
default, it will flatten all fields
ext_json("content")
//Discard the content field
fields_drop("content")
//Discard unnecessary fields bytes_sent,method,response_code_details,start_time
fields_drop("bytes_sent","method","response_code_details","start_time")
```

```
//Split logs from the array, splitting into 2 logs
log_split_jsonarray_jmes("records")
//Discard the original field records
fields_drop("records")
//Expand the KV pairs of the log
ext_json("log")
//Discard the original field log
fields_drop("log")
```

Multi-Format Log Structuration

Last updated：2024-01-20 17:44:35

Use Case

Tom has collected **user operation and result** logs to CLS in single-line text format. The formats of contents of the logs **are not identical**. Tom wants to write a set of statements to structure the logs in different formats.
Analysis found that the logs are basically in three formats: the first contains four fields (**uin**, **requestid**, **action**, and **Reqbody**), the second contains three fields (**uin**, **requestid**, and **action**), and the third contains three fields (**requestid**, **action**, and **TaskId**).

Scenario Analysis

According to Tom's requirements, the processing ideas are as follows:
1. Since all three formats of logs contain the **requestid** and **action** fields, use a regular expression to extract these two fields.
2. Perform special processing on the **uin**, **reqbody**, and **TaskId** fields: determine whether the fields exist and then extract them if they exist.

Raw Log

```
[
    {
        "__CONTENT__": "2021-11-29 15:51:33.201 INFO request 7143a51d-caa4-
4a6d-bbf3-771b4ac9e135 action: Describe uin: 15432829 reqbody {\\"Key\\":
\\"config\\",\\"Values\\": \\"appisrunnning\\",\\"Action\\":
\\"Describe\\",\\"RequestId\\": \\"7143a51d-caa4-4a6d-bbf3-
771b4ac9e135\\",\\"AppId\\": 1302953499,\\"Uin\\": \\"100015432829\\"}"
    },
    {
        "__CONTENT__": "2021-11-2915: 51: 33.272 ERROR request 2ade9fc4-2db2-
49d8-b3e0-a6ea78ce8d96 has error action DataETL uin 15432829"
    },
    {
        "__CONTENT__": "2021-11-2915: 51: 33.200 INFO request 6059b946-25b3-
4164-ae93-9178c9e73d75 action: UploadData hUWZSs69yGc5HxgQ TaskId 51d-caa-a6d-
bf3-7ac9e"
```

```
    }
]
```

DSL Processing Function

```
fields_set("requestid",regex_select(v("__CONTENT__"),regex="request [A-Za-z0-
9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+",index=0,group=0))
fields_set("action",regex_select(v("__CONTENT__"),regex="action: \\S+|action
\\S+",index=0,group=0))
t_if(regex_match(v("__CONTENT__"),regex="uin",
full=False),fields_set("uin",regex_select(v("__CONTENT__"),regex="uin: \\d+|uin
\\d+",index=0,group=0)))
t_if(regex_match(v("__CONTENT__"),regex="TaskId",
full=False),fields_set("TaskId",regex_select(v("__CONTENT__"),regex="TaskId [A-
Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-
9]+",index=0,group=0)))
t_if(regex_match(v("__CONTENT__"),regex="reqbody",
full=False),fields_set("requestbody",regex_select(v("__CONTENT__"),regex="reqbo
dy \\{[^\\}]+\\}")))
t_if(has_field("requestbody"),fields_set("requestbody",str_replace(v("requestbo
dy"),old="reqbody",new="")))
fields_drop("__CONTENT__")
fields_set("requestid",str_replace(v("requestid"),old="request",new=""))
t_if(has_field("action"),fields_set("action",str_replace(v("action"),old="actio
n:|action",new="")))
t_if(has_field("uin"),fields_set("uin",str_replace(v("uin"),old="uin:|uin",new=
"")))
t_if(has_field("TaskId"),fields_set("TaskId",str_replace(v("TaskId"),old="TaskI
d",new="")))
```

DSL Processing Function Details

1. Create a field named **requestid** and use a regular expression to match "**request 7143a51d-caa4-4a6d-bbf3-771b4ac9e135**".

```
fields_set("requestid",regex_select(v("__CONTENT__"),regex="request [A-Za-z0-
9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+",index=0,group=0))
```

2. Create a field named **action** and use a regular expression to match "**action: UploadData**" and "**action DataETL**" (they exist in two formats of the raw logs).

```
fields_set("action",regex_select(v("__CONTENT__"),regex="action: \\S+|action
\\S+",index=0,group=0))
```

If the **__CONTENT__** field contains the **uin** keyword, create the **uin** field and use the regular expression "**uin: \\d+|uin \\d+**" to match **uin: 15432829** and **uin 15432829**.

```
t_if(regex_match(v("__CONTENT__"),regex="uin",
full=False),fields_set("uin",regex_select(v("__CONTENT__"),regex="uin: \\d+|uin
\\d+",index=0,group=0)))
```

If the **TaskId** keyword exists, create the **TaskId** field and use a regular expression to match "**TaskId 51d-caa-a6d-bf3-7ac9e**".

```
t_if(regex_match(v("__CONTENT__"),regex="TaskId",
full=False),fields_set("TaskId",regex_select(v("__CONTENT__"),regex="TaskId [A-
Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-9]+-[A-Za-z0-
9]+",index=0,group=0)))
```

If the **reqbody** keyword exists, create the **requestbody** field and use a regular expression to match "**reqbody{...}**".

```
t_if(regex_match(v("__CONTENT__"),regex="reqbody",
full=False),fields_set("requestbody",regex_select(v("__CONTENT__"),regex="reqbo
dy \\{[^\\}]+\\}")))
```

3. Discard the **__CONTENT__** field.

```
fields_drop("__CONTENT__")
```

Now we have extracted the fields we need. However, unnecessary characters (**action**, **uin**, **requestbody**, **requestid**, and **TaskId**) are generated during regular expression matching. Therefore, we need to use the **str_replace()** function to remove the unnecessary characters and use the **fields_set()** function to reset field values.

1. If the **requestbody** field exists, remove the unnecessary characters **reqbody** from the field value.

```
t_if(has_field("requestbody"),fields_set("requestbody",str_replace(v("requestbo
dy"),old="reqbody",new="")))
```

2. Remove the unnecessary characters **requestid** from the v("**requestid**") field value. Because every log contains **requestid**, we do not determine whether the field exists.

```
fields_set("requestid",str_replace(v("requestid"),old="request",new=""))
```

3. If the **action** field exists, remove the unnecessary characters **action:** or **action** from the field value.

```
t_if(has_field("action"),fields_set("action",str_replace(v("action"),old="actio
n:|action",new="")))
```

4. If the **uin** field exists, remove the unnecessary characters **uin:** or **uin** from the field value.

```
t_if(has_field("uin"),fields_set("uin",str_replace(v("uin"),old="uin:|uin",new=
""")))
```

5. If the **TaskId** field exists, remove the unnecessary characters **TaskId** from the field value.

```
t_if(has_field("tTaskId"),fields_set("TaskId",str_replace(v("TaskId"),old="Task
Id",new="")))
```

Processing Result

```
[
{"action":" Describe","requestid":" 7143a51d-caa4-4a6d-bbf3-
771b4ac9e135","requestbody":" {\\"Key\\": \\"config\\",\\"Values\\":
\\"appisrunnning\\",\\"Action\\": \\"Describe\\",\\"RequestId\\": \\"7143a51d-
caa4-4a6d-bbf3-771b4ac9e135\\",\\"AppId\\": 1302953499,\\"Uin\\":
\\"100015432829\\"}","uin":" 15432829"},
{"action":" DataETL","requestid":" 2ade9fc4-2db2-49d8-b3e0-
a6ea78ce8d96","uin":" 15432829"},
{"action":" UploadData","requestid":" 6059b946-25b3-4164-ae93-
9178c9e73d75","TaskId":" 51d-caa-a6d-bf3-7ac9e"}
]
```

Using Separators to Extract Specified Content from Logs

Last updated：2024-01-20 17:44:35

Use Case

Tom has collected **Flink task running logs** to CLS in single-line text format. The log content is divided into segments with the **comma (,)** and **colon (:)** separators. Among these segments, there is a segment in escaped JSON format containing Flink task execution details. Tom wants to extract the task details and structure them.

Scenario Analysis

According to Tom's requirements, the processing ideas are as follows:
1. Extract the content in escaped JSON format.
2. Extract structured data from the JSON content.

Raw Log

```
{
    "regex": "2021-12-02 14:33:35.022 [1] INFO  org.apache.Load -
Response:status: 200, resp msg: OK, resp content: {    \\"TxnId\\": 58322,
\\"Label\\": \\"flink_connector_20211202_1de749d8c80015a8\\",    \\"Status\\":
\\"Success\\",    \\"Message\\": \\"OK\\",    \\"TotalRows\\": 1,
\\"LoadedRows\\": 1,    \\"FilteredRows\\": 0,  \\"CommitAndPublishTimeMs\\":
16}"
}
```

DSL Processing Function

```
ext_sepstr("regex", "f1, f2, f3", sep=",")
fields_drop("regex")
fields_drop("f1")
fields_drop("f2")
ext_sepstr("f3", "f1,resp_content", sep=":")
```

```
fields_drop("f1")
fields_drop("f3")
ext_json("resp_content", prefix="")
fields_drop("resp_content")
```

DSL Processing Function Details

1. Use commas (,) to divide the log into 3 segments, where the third segment **f3** is **resp content:{JSON}**.

```
ext_sepstr("regex", "f1, f2, f3", sep=",")
```

2. Discard unwanted fields.

```
fields_drop("regex")
fields_drop("f1")
fields_drop("f2")
```

3. Use colons (:) to divide the **f3** field into two segments.

```
ext_sepstr("f3", "f1,resp_content", sep=":")
```

4. Discard useless fields.

```
fields_drop("f1")
fields_drop("f3")
```

5. Use the **ext_json** function to extract structured data from the **resp_content** field.

```
ext_json("resp_content", prefix="")
```

6. Discard the **resp_content** field.

```
fields_drop("resp_content")
```

Processing Result

```
{"CommitAndPublishTimeMs":"16","FilteredRows":"0","Label":"flink_connector_2021
1202_1de749d8c80015a8","LoadedRows":"1","Message":"OK","Status":"Success","Tota
lRows":"1","TxnId":"58322"}
```

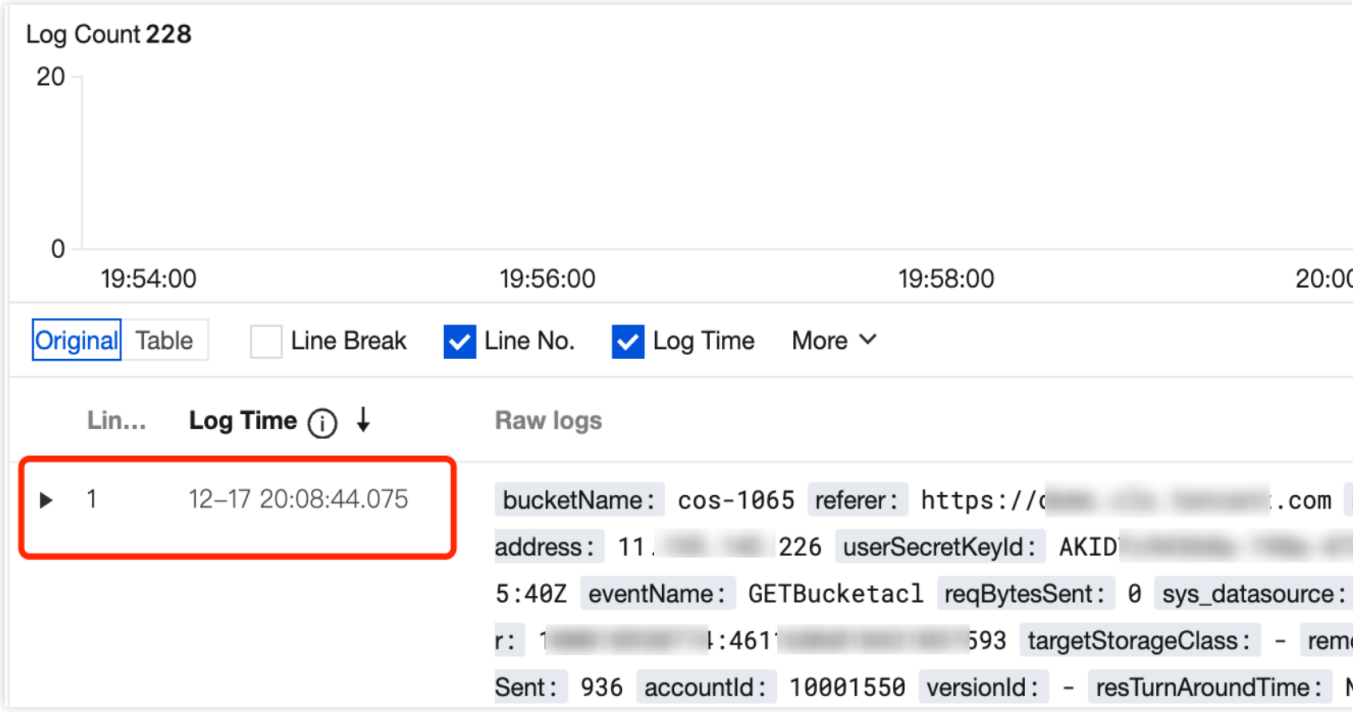
Log Structuring - Grok

Last updated：2024-12-18 16:36:30

Scenario description

XiaoWang reports the logs collected by Beats to CLS through Kafka protocol to upload logs. The approach is as follows:

1. Use the grok function to structure the logs.
2. Use the time field in the logs to **replace the log time of CLS**(__TIMESTAMP__).



Raw Log

```
{
  "__FILENAME__": "",
  "__SOURCE__": "192.168.100.123",
  "message": "2024-10-11 15:32:10.003 DEBUG [gateway,746db87efd1bbcf5434cb9835c59
}
```

Processing result

```
{
"__FILENAME__":"",
"__SOURCE__":"192.168.100.123",
"__TIMESTAMP__":"1728631930003",
"level":"DEBUG",
"service":"gateway",
"spanid":"47c3036810e0c33b",
"time":"2024-10-11 15:32:10.003",
"traceid":"746db87efd1bbcf5434cb9835c59e522"
}
```

Processing statement

```
// Use the grok function to extract time, log level, service, traceid, and spanid f
ext_grok("message",grok="%{TIMESTAMP_ISO8601:time} %{DATA:level} \\[%{DATA:service}
// Delete message field
fields_drop("message")
// custom_cls_log_time function, use the new field time to replace the log time of
custom_cls_log_time(dt_to_timestamp(v("time"), zone="UTC+8"))
```

Log Distribution

Last updated：2024-01-20 17:44:35

Use Case

Tom has collected logs to CLS. The logs contain information such as the log time, log level, log content, task ID, process name, and host IP, and the information is separated by two vertical bars (||). Now Tom wants to structure the log to facilitate subsequent indexing and dashboard display. He also wants to **distribute** the logs to three different target log topics according to three log levels (**ERROR**, **WARNING**, and **INFO**) for subsequent analysis. Tom also wants the logs whose content contains the **team B is working** keywords to be filtered out (discarded)**.

Scenario Analysis

According to Tom's requirements, the processing ideas are as follows:
1. Filter out (discard) logs that contain the **team B is working** keywords and place the discarded logs up front to reduce subsequent computation.
2. Structure logs based on the separator of **two vertical bars** (||).
3. Log distribution: Distribute the logs to three different target log topics according to three log levels (**ERROR**, **WARNING**, and **INFO**).

Note:
To distribute logs to multiple target log topics, you need to define the target names of the log topics when creating the data processing task. The target names will be used in the **log_output("Target name")** functions.

Raw Log

```
[
    {
        "message": "2021-12-09 11:34:28.279||team A is
working||INFO||605c643e29e4||BIN--COMPILE||192.168.1.1"
    },
    {
        "message": "2021-12-09 11:35:28.279||team A is working
||WARNING||615c643e22e4||BIN--Java||192.168.1.1"
    },
    {
```

```
        "message": "2021-12-09 11:36:28.279||team A is working
||ERROR||635c643e22e4||BIN--Go||192.168.1.1"
    },
    {
        "message": "2021-12-09 11:37:28.279||team B is
working||WARNING||665c643e22e4||BIN--Python||192.168.1.1"
    }
]
```

DSL Processing Function

```
log_drop(regex_match(v("message"),regex="team B is working",full=False))
ext_sepstr("message","time,log,loglevel,taskId,ProcessName,ip",sep="\\|\\|")
fields_drop("message")
t_switch(regex_match(v("loglevel"),regex="INFO",full=True),log_output("info_log
"),regex_match(v("loglevel"),regex="WARNING",full=True),log_output("warning_log
"),regex_match(v("loglevel"),regex="ERROR",full=True),log_output("error_log"))
```

DSL Processing Function Details

1. Discard logs that contain the **team B is working** keywords. The fourth log contains the **team B is working** keywords and needs to be discarded.

```
log_drop(regex_match(v("message"),regex="team B is working",full=False))
```

2. Extract structured data based on the separator of **two vertical bars** (||).

```
ext_sepstr("message","time,log,loglevel,taskId,ProcessName,ip",sep="\\|\\|")
```

3. Discard the **message** field.

```
fields_drop("message")
```

4. According to the value of the **loglevel** field, **INFO**, **WARNING**, and **ERROR** logs will be distributed to different target log topics.

```
t_switch(regex_match(v("loglevel"),regex="INFO",full=True),log_output("info_log
"),regex_match(v("loglevel"),regex="WARNING",full=True),log_output("warning_log
"),regex_match(v("loglevel"),regex="ERROR",full=True),log_output("error_log"))
```

Processing Result

Note:

Target log topics and target names must be configured in advance.

The following log is distributed to **info_log** (**Data processing-target 3**). See the mappings between target names and log topics in the figure above.

```
{"ProcessName":"BIN--COMPILE","ip":"192.168.1.1","log":"team A is
working","loglevel":"INFO","taskId":"605c643e29e4","time":"2021-12-09
11:34:28.279"}
```

The following log is distributed to **warning_log** (**Data processing-target 2**).

```
{"ProcessName":"BIN--COMPILE","ip":"192.168.1.1","log":"team A is
working","loglevel":"INFO","taskId":"605c643e29e4","time":"2021-12-09
11:34:28.279"}
```

The following log is distributed to **error_log** (**Data processing-target 1**).

```
{"ProcessName":"BIN--Go","ip":"192.168.1.1","log":"team A is working
","loglevel":"ERROR","taskId":"635c643e22e4","time":"2021-12-09 11:36:28.279"}
```

Log Masking

Last updated：2024-12-03 19:08:26

Use Case

Tom has collected a log to CLS. The log contains sensitive information such as the user ID (**dev@12345**), login IP (**11.111.137.225**), and mobile number (**13912345678**). Tom wants to mask these sensitive information.

Scenario Analysis

The log itself is a structured log, and therefore its fields can be masked directly.

Raw Log

```
{
    "Id": "dev@12345",
    "Ip": "11.111.137.225",
    "phonenumber": "13912345678"
}
```

DSL Processing Function

```
fields_set("Id",regex_replace(v("Id"),regex="\\d{3}", replace="***",count=0))
fields_set("Id",regex_replace(v("Id"),regex="\\S{2}", replace="**",count=1))
fields_set("phonenumber",regex_replace(v("phonenumber"),regex="(\\d{0,3})\\d{4}
(\\d{4})", replace="$1****$2"))
fields_set("Ip",regex_replace(v("Ip"),regex="(\\d+\\.)\\d+(\\.\\d+\\.\\d+)",
replace="$1***$2",count=0))
```

DSL Processing Function Details

1. Mask the **Id** field. The result is **dev@***45**.

```
fields_set("Id",regex_replace(v("Id"),regex="\\d{3}", replace="***",count=0))
```

2. Mask the **Id** field again. The result is ****v@***45**.

```
fields_set("Id",regex_replace(v("Id"),regex="\\S{2}", replace="**",count=1))
```

3. Mask the **phonenumber** field by replacing the middle 4 digits with ********. The result is **139****5678**.

```
fields_set("phonenumber",regex_replace(v("phonenumber"),regex="(\\d{0,3})\\d{4}
(\\d{4})", replace="$1****$2"))
```

4. Mask the **IP** field by replacing the octet with *******. The result is **11.***137.225**.

```
fields_set("Ip",regex_replace(v("Ip"),regex="(\\d+\\.)\\d+(\\.\\d+\\.\\d+)",
replace="$1***$2",count=0))
```

Processing Result

```
{"Id":"**v@***45","Ip":"11.***.137.225","phonenumber":"139****5678"}
```

Log Filtering

Last updated：2024-12-18 16:37:22

Scenario description

Xiao Wang collected the tool's logs into CLS. Now, it is necessary to filter out logs with the operation method of PUT and POST from the cmdb logs.

Raw Log

```
[
    {
        "path": "/eks/pod/running",
        "clientIP": "1.139.21.123",
        "method": "POST"
    },
    {
        "path": "/cmdb/login",
        "clientIP": "1.139.21.123",
        "method": "PUT"
    },
    {
        "path": "/cmdb/start",
        "clientIP": "1.139.21.123",
        "method": "GET"
    }
]
```

Processing statement

```
//path contains 'cmdb' characters, keep the log, filter out the rest
log_keep(regex_match(v("path"),regex="cmdb",full=False))
//method contains POST or PUT characters, keep the log
log_keep(regex_match(v("method"),regex="POST|PUT",full=False))
```

Processing result

```
{
"clientIP":"1.139.21.123",
"method":"PUT",
"path":"/cmdb/login"
}
```

Scheduled SQL Analysis Overview

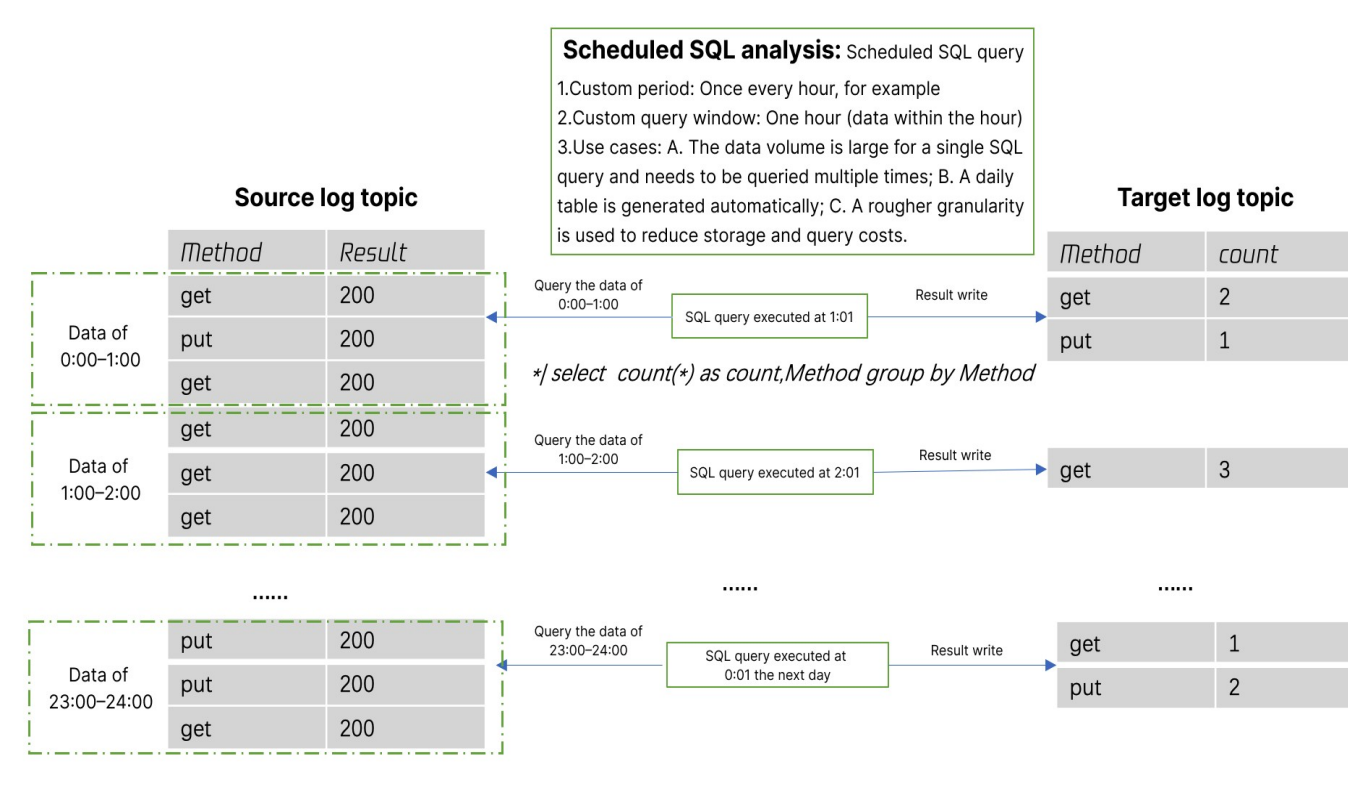
Last updated：2024-01-20 17:44:35

Note:

This feature has been in beta test in Beijing, Shanghai, and Guangzhou, Nanjing, and Chongqing regions free of charge since August 15, 2022.

Overview

Scheduled SQL analysis can be simply understood as crontab SQL. You can configure a scheduling policy, and the system will execute SQL queries on the source log topic regularly based on the policy and save query results to the specified target log topic.



Prerequisites

The CLS service has been activated, and key-value index has been enabled.
Make sure that the current account has the permission to configure scheduled SQL analysis. For more information, see Examples of Custom Access Policies.

Use cases

For a query with a high data volume, you can use the scheduled SQL analysis feature to break it down into multiple queries with a low data volume each to avoid query failure and timeout.
For example, if the original SQL query involves data of one day, you can split it into 24 queries for execution once every hour.
Generate daily and weekly reports.
Aggregate logs, which can greatly reduce index and log storage fees.
For example, aggregating 1-minute historical logs into 1-hour logs can effectively save the storage costs.
Filter and save data to a new log topic. Scheduled SQL analysis can meet the needs in some simple filtering scenarios through WHERE and WHEN statements. However, as a SQL query can only return up to 10,000 results, data integrity cannot be guaranteed, and the query is not conducted by real-time stream computing. Therefore, this feature is only applicable to use cases with a small amount of data that don't require real-time computing. For similar use cases, we recommend you use this feature.

Use limits

Up to 10,000 results can be returned per query, and the excess will be truncated.
Cross-region query is not supported. The source and target log topics must be in the same region.

Creating Task

Last updated：2024-01-20 17:44:35

Overview

This document describes how to create a scheduled SQL analysis task.

Prerequisites

The CLS service has been activated, and key-value index has been enabled.
Make sure that the current account has the permission to configure scheduled SQL analysis. For more information, see Examples of Custom Access Policies.

Directions

1. Log in to the CLS console.
2. Click **Data Processing** > **Scheduled SQL Task** on the left sidebar and click

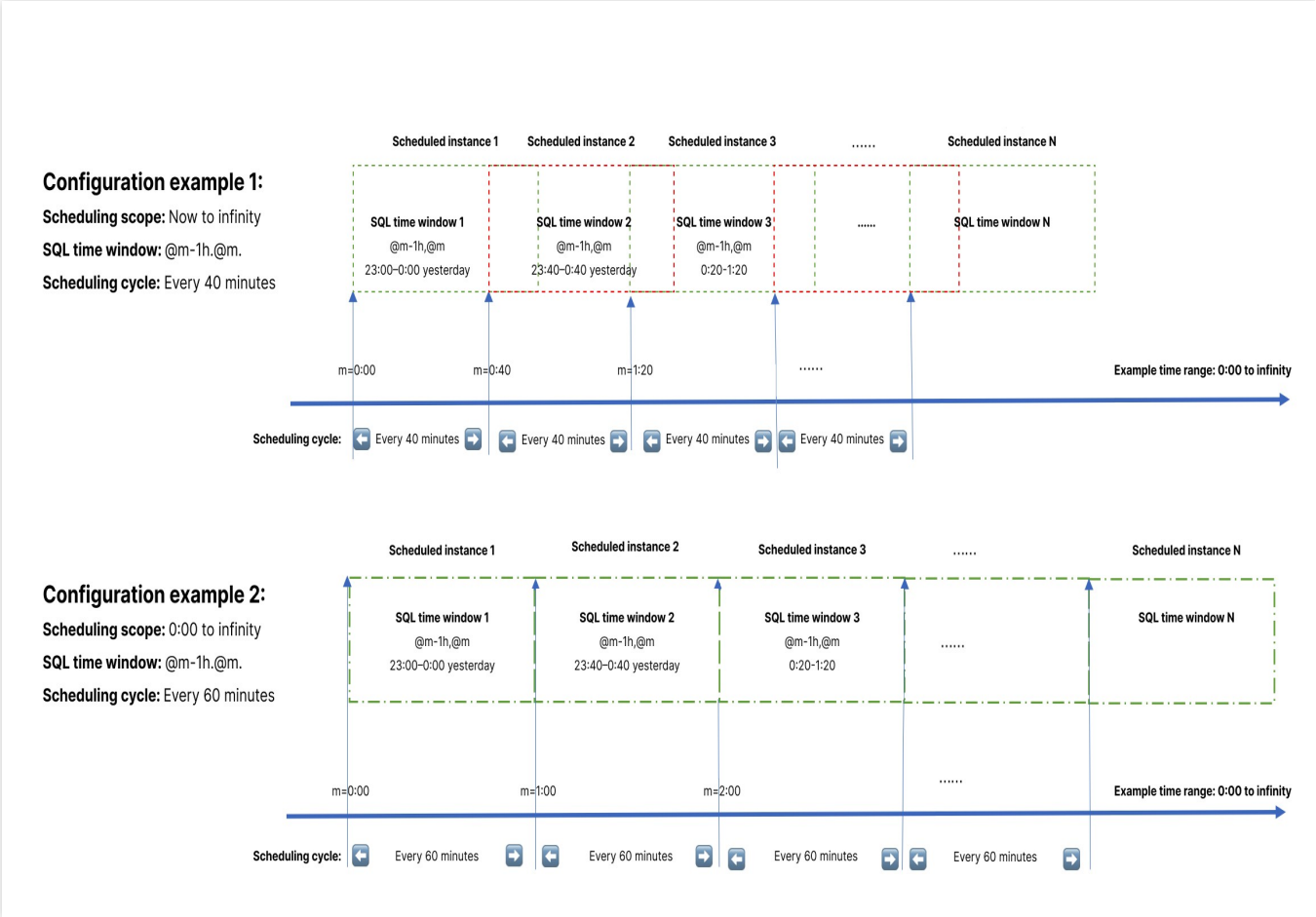

to create a task.
3. On the basic configuration page, configure the following information and click **Next**:
Task Name: Enter a custom task name.
Source Log Topic: Select the log topic where to run the SQL analysis task.
SQL Statement: Enter the SQL statement in **Query Statement**, and the system will return the preview results (up to 100 items).
Target Log Topic: Select the target log topic where to save SQL analysis results.
4. On the scheduling configuration page, configure the following information and click **OK**.
Scheduling Range: Set the time range for running the scheduled task. The default value is to start at the current time and last forever, i.e., running continuously.
Scheduling Cycle: Set the cycle of the scheduled task, i.e., execution every X minutes. The maximum value is 1,440.
SQL Time Window: Set the start and end time of the SQL query log data.

Common SQL Time Window Expression (Suppose It's 12:06 Now)	SQL Time Window	Description

@m-1h, @m	11:06 - 12:06	`@m` and `-1h` indicate to take the value down to the minute and subtract 1 hour, respectively.
@h-1h,@h	11:00 - 12:00	`@h` and `-1h` indicate to take the value down to the hour and subtract 1 hour, respectively.
@m-1h+20m,@h+25m	11:26 - 12:25	`@m`, `-1h`, `+20m`, `@h`, and `+25m` indicate to take the value down to the minute, subtract 1 hour, add 20 minutes, take the value down to the hour, and add 25 minutes, respectively.

Example

The following example illustrates the configurations of the scheduling range, scheduling cycle, and SQL time window:



Viewing Task

Last updated：2024-01-20 17:44:35

Overview

This document describes how to view the information of a scheduled SQL analysis task.

Directions

1. Log in to the CLS console.
2. On the left sidebar, select **Data Processing** > **Scheduled SQL Tasks** to view the following information:
Basic task information: View the task's name, ID, source log topic, target log topic, creation time, and last modified time.
Scheduling details: View each SQL query's instance ID, execution time, SQL time window, processed data volume, and scheduling result.
Preview data: Click **Result Data** to view the result data.
You can also go to the target log topic to view the results of scheduled SQL analysis.

SCF

Last updated：2024-01-20 17:44:35

Overview

You can use SCF to process CLS logs. SCF and CLS are independent of each other and are connected via triggers.

Prerequisites

You have logged in to the CLS console.

Directions

Creating a log topic

Create two log topics as instructed in Managing Log Topic.

Note:

As both the source and destination of data ETL are CLS, you need to create at least two topics.

Creating an SCF function

Create a function as instructed in Creating and Testing Function.

The main parameters are as follows:

Region: Select **Beijing** region.

Function name: Enter "CLSdemo".

Creation method: Click **Template** and select the **CLSLogETL** template.

Note:

You should select the VPC and subnet of CLS for the created function on the **Function Configuration** page.

Configuring a CLS trigger

1. Log in to the CLS console.
2. On the left sidebar, click **Log Topic** to go to the log topic management page.
3. Find the log topic you just created and click its ID/name to enter the log topic details page.
4. On the log topic details page, select the **Function Processing** tab and click **Create**.
5. In the **Function Processing** pop-up window, add the created function and click **OK**.

The main parameter information is as follows. Use the default values for the remaining configuration items.

Namespace: Select the function namespace.

Function Name: Select the function created in the Creating SCF function step.

Alias: Select a function alias.

Maximum waiting time: Configure the longest waiting time for a single event pull. Default value: 60s.

Testing the function

1. Download the log file in the test sample, extract `demo-scf1.txt` , and import it to the source CLS service.

2. Switch to the SCF console to view the execution result.

On the function details page, select the **Log Query** tab to view the printed log information.

3. Switch to the target CLS service to view the data processing result.

Note:

You can write specific data processing methods as needed.

Shipping and Consumption
Shipping Overview

Last updated：2024-01-20 17:44:35

Shipping to COS

CLS can ship data in a log topic to COS to meet the needs in the following scenarios:
Logs are shipped to and stored in COS in STANDARD storage class. If you need other storage classes, perform relevant operations in COS. For more information, see Overview.
Log data is processed through offline computing or other computing programs. Such data is shipped to COS first and then loaded by Data Lake Compute (data lake) or EMR (big data platform) for further analysis. For more information, see Using Data Lake Compute (Hive) to Analyze CLS Log. We recommend you choose CSV or Parquet as the shipping format.

Billing Description

Log shipping generates private network read traffic fees (cross-region shipping is not supported for now), and CLS will charge fees based on the compression format (Snappy/GZIP/lzop). If your raw log is 100 GB and you choose Snappy for compression, around 50 GB will be billable. As the read traffic price is 0.032 USD/GB, the fees will be 50 GB * 0.032 USD/GB = 1.6 USD.

Feature Limits

Historical data cannot be shipped.
Cross-region shipping is not supported. The log topic and COS bucket must be in the same region.
Cross-account shipping is not supported.

Shipping Format

Data Shipping Format	Description	Recommended Scenario
CSV shipping	Log data is shipped to COS based on the specified separator, such as space,	It can be used for computing in Data Lake Compute.

	tab, comma, semicolon, and vertical bar.	It can be used to ship raw logs (logs collected in a single line, in multiple lines, and with separators).
JSON shipping	Log data is shipped to COS in JSON format.	It is a common data format and can be selected as needed.
Parquet shipping	Log data is shipped to COS in Parquet format.	Log data needs to be structured data. The data type can be converted (data not collected in a single line or multiple lines). This format is mainly used for Hive batch processing.

Note:

After log data is shipped to COS, COS storage fees will be incurred. For billing details, see Billing Overview.

To cleanse the log data before shipping to COS, see Log Filtering and Distribution.

CLS Service Role Authorization

Last updated：2024-01-20 17:44:35

Overview

When creating shipping to COS/CKafka tasks, you need to grant the CLS service role the permissions to access COS/CKafka. If you perform operations in the console, the system will guide you through the authorization process. If you directly call APIs, manual authorization will be required. Before manual authorization, check whether the CLS service role has been authorized in the following steps.

Checking CLS Authorization

1. Log in to the CAM console, and select **Role** on the left sidebar.
2. On the **Role** page, check whether you have the `CLS_QcsRole` role. You can use the search box in the top-right corner of the role list to search for the role.
3. Click the role name to go to the role details page.
Select the **Permission** tab to see if the role has the `QcloudCOSAccessForCLSRole` and `QcloudCKAFKAAccessForCLSRole` permissions.
Select the **Role Entity** tab to see whether the role entity is `cls.cloud.tencent.com` .
If there is no such role or permission, create one as instructed below.

Directions

Granting CLS access permissions

You can use either of the following methods to grant CLS the permissions to ship logs to COS/CKafka:
Automatic Creation via CLS Console
Manual Creation via CAM Console
If this is the first time you create a task to ship logs to COS/CKafka in the CLS console, follow the instructions in the console to create the required role and policies:
1. In the pop-up window that reads **This feature requires creating a service role**, click **Go to Cloud Access Management**.
2. On the **Role Management** page, click **Grant**.
1. Log in to the CAM console, and select **Role** on the left sidebar.
2. On the **Role** page, click **Create Role**.

3. In the **Select role entity** dialog box, click **Tencent Cloud Product Service**.
4. On the **Create Custom Role** page, perform the following operations:
4.1 In the **Enter role entity info** step, select **Cloud Log Service (cls)** and click **Next**.
4.2 In the **Configure Role Policy** step, use `clsrole` to search for data, select the `QcloudCKAFKAAccessForCLSRole` and `QcloudCOSAccessForCLSRole` policies in the search result, and click **Next**.
4.3 In the **Review** step, enter the role name `CLS_QcsRole` and click **Complete**.
At this point, you have authorized the CLS service role to access COS/CKafka. If you are using a root account, you can directly ship logs. If you are using a sub-account or collaborator account, you need to be authorized by the root account. For more information on granting permissions, see CAM Access Management. For more information on copying authorization policies, see Examples of Custom Access Policies.

Shipping to COS
JSON Shipping

Last updated：2024-01-20 17:44:35

Overview

You can log in to the CLS console and ship data in JSON format to COS. This document describes how to create a JSON shipping task.

Prerequisite

1. You have activated CLS, created a logset and a log topic, and successfully collected the log data.
2. You have activated COS and created a bucket in the target region for log topic shipping. For more information, see Creating Bucket.
3. Sub-accounts and collaborators need to be authorized by the root account. For more information on granting permissions, see CAM Access Management. For more information on copying authorization policies, see Examples of Custom Access Policies.
4. You have authorized the CLS service role to access COS. If you perform operations in the console, the system will guide you through the authorization process. If you directly call APIs, manual authorization will be required. For more information, see Viewing and Configuring Shipping Permissions.

Directions

1. Log in to the CLS console.
2. Click **Log Topic** on the left sidebar.
3. Click the desired log topic ID/name to go to the log topic management page.
4. Select the **Ship to COS** tab, click **Add Shipping Configuration**, and finish the configuration as detailed below.

The parameters are described as follows:

Configuration Item	Description	Limit	Required
Shipping Task Name	Configures the name of a shipping task	The name can contain letters, numbers,	Yes

		underscores (_), and hyphens (-)	
COS Bucket	Target bucket for shipping. The target bucket must be in the same region as the current log topic.	A value selected from the drop-down list	Yes
COS Path	COS bucket path, which is in the format of `/year/month/day/hour/` by default, such as `/2022/7/31/14/`. This format is used in COS for storing shipped log files. Here, the strftime syntax is supported. For example, if a log was shipped at 14:00 on July 31, 2022, the generated `/%Y/%m/%d/` path is `/2022/7/31/`, and the `/%Y%M%d/%H/` path is `/2022/07/31/14/`.	Cannot start with /	No
Filename	Option 1 (recommended): Use the shipping time as the name. For example, `202208251645_000_132612782.gz` indicates the shipping time_log topic partition_offset. This type of files can be loaded in Hive.Option 2: Use a random number as the name. This is the legacy practice of naming files and cannot be recognized by Hive as it cannot recognize filenames starting with "_". You can add a custom prefix to the COS path, such as `/%Y%M%d/%H/Yourname`.	/	Yes
Compression Format	To reduce read traffic fees, log files are compressed before being shipped to COS. Snappy, lzop, and GZIP are supported.	GZIP, Snappy, and lzop	Yes
File Size	It indicates the size of the raw log file to be shipped and is used together with the shipping interval parameter. When either condition is met, compression will be performed accordingly. For example, if the size is set to 256 MB and the interval is set to 15 minutes, when the file reaches 256 MB within five minutes, the size condition will be met first to trigger shipping.	The value must be a number ranging from 5 to 256 in MB.	Yes
Shipping Interval	It indicates the interval for shipping and is used together with the file size parameter. When either condition is met, compression will be performed accordingly. For example, if the size is set to 256 MB and the interval is set to 15 minutes, when the file reaches only 200 MB in 15 minutes, the shipping interval condition will be met first to trigger shipping.	Value range: 300-900s	Yes

5. Click**Next** to enter the **Advanced Configuration** page. Set **Shipping Format** to **JSON** and select the field to be shipped. Here, `_CONTENT_` is the user log data, and `_SOURCE_` , `_FILENAME_` , `_HOSTNAME_` , `_TIMESTAMP_` , and `_TAG_` are CLS metadata fields that can be selected as needed.

Raw Log Shipping

Last updated：2024-01-20 17:44:35

Overview

This document describes how to ship the collected raw log to COS. Currently, only logs collected in a single line, in multiple lines, or with certain separators are supported in this scenario.

Notes

You can configure CSV shipping to COS to ship certain raw logs. The following table describes the options of **CSV** shipping to COS but doesn't apply to JSON or Parquet shipping.

Log Collection Format	Support for Raw Log Shipping	User Raw Log	CLS Storage Format	Format for Shipping to COS
Full text in a single line	Yes. For more information, see Logs collected in a single line.	yourlog	__CONTENT__:yourlog	yourlog
Full text in multiple lines	Yes. For more information, see Logs collected in multiple lines.	yourlog	__CONTENT__:yourlog	yourlog
Separator (CSV) format	It depends. For more information, see CSV format. Only space, tab, comma, semicolon, and vertical bar are supported as raw log separators.	V1 separator V2 separator V3 separator...Vn For example, V1,V2,V3...Vn	K1:V1 K2:V2 K3:V3...Kn:Vn	V1 separator V2 separator V3 separator...Vn For example, V1,V2,V3...Vn
JSON format	No	K1:V1 K2:V2 K3:V3...Kn:Vn	K1:V1 K2:V2 K3:V3...Kn:Vn	V1,V2,V3...Vn differs from the original JSON.
Full regex	No	V11V22V33...Vnn	K1:V1 K2:V2	V1,V2,V3...Vn

			K3:V3...Kn:Vn	differs from the raw log.

Directions

Log collected in a single line or multiple lines

For logs collected in a single line or multiple lines, you can configure parameters based on CSV shipping to implement raw log shipping.

1. Complete the first step of **Basic Configuration** as instructed in CSV Shipping.
2. Set **Shipping Format** to **CSV**, retain the `__CONTENT__` field, delete other fields, set **Separator** to **Space**, set **Escape Character** to **Space**, set **Invalid Field Filling** to **None**, and disable **Key in First Line**.

The parameters are described as follows:

Configuration Item	Description	Remarks
Key	__CONTENT__	For full text in a single line or multiple lines, __CONTENT__ is used as the default key, and the raw log is used as the value. When the raw log is shipped, only the __CONTENT__ field is retained.
Separator	Space	Set **Separator** to **Space** for the full text in a single line or multiple lines.
Escape Character	None	To prevent the raw log from being modified due to escape characters, set **Escape Character** to **None**.
Invalid Field Filling	None	Set **Invalid Field Filling** to **None**.
Key in First Line	Disabled	You don't need to add a description of the field name in the first line of the CSV file for raw log shipping.

3. Click **OK** to enable shipping.

Logs collected in CSV format

Note:

CSV shipping supports limited types of separators, including space, tab, comma, semicolon, and vertical bar. Therefore, you can ship log data in the raw format only when the separator in the raw log content is supported by CSV shipping.

1. Complete the first step of **Basic Configuration** as instructed in CSV Shipping.
2. Set **Shipping Format** to **CSV**. During field configuration, you need to delete CLS metadata fields.

The parameters are described as follows:

Configuration Item	Value	Description
Key	Keys	Only the user field is retained.
Separator	A value selected from the drop-down list	Select the separator of the raw log content. If separators are different, raw log shipping is not supported. Currently, only space, tab, comma, semicolon, and vertical bar are supported.
Escape Character	None	To prevent the raw log from being modified due to escape characters, set **Escape Character** to **None**.
Invalid Field Filling	None	Set **Invalid Field Filling** to **None**.
Key in First Line	Disabled	You don't need to add a description of the field name in the first line of the CSV file for raw log shipping.

3. Click **OK** to enable shipping.

Shipping Task Management

Last updated：2024-01-20 17:44:35

Overview

This document describes how to manage shipping tasks, including viewing task details as well as modifying and deleting tasks.

Directions

Viewing shipping task details

1. Log in to the CLS console.
2. Select **Shipping Task Management** on the left sidebar.
3. On the **Shipping Task Management** page, select the region, logset, and log topic to view the list of shipping to COS tasks.
4. Click a task name to view its details and monitoring information.

Shipping Traffic: The traffic generated by successful shipping to COS. When the configured file size or shipping interval is reached to trigger shipping, there will be an actual value for this parameter. At other times, the value will be 0.

Shipped Line Count: The number of data lines successfully shipped to COS. When the configured file size or shipping interval is reached to trigger shipping, there will be an actual value for this parameter. At other times, the value will be 0.

Shipping Latency: The time after log data arrives at CLS and before it is processed in a shipping task. It is normal if this value is within 60 seconds. This metric is mainly used to monitor whether data heaps up and logs are shipped to COS timely.

Shipping Task Status:

`1` indicates a success.

`10001` indicates that the COS bucket doesn't exist. You should check the validity of the bucket.

`10002` indicates that you have no permission to access the COS bucket. You should make sure that you have the permission.

`10003` indicates an internal error. You should try again. If the problem persists, submit a ticket.

Modifying a task

1. Log in to the CLS console.
2. Select **Shipping Task Management** on the left sidebar.

3. On the **Shipping Task Management** page, select the region, logset, log topic, and name of the target shipping task, and click **Modify Configuration** on the right.

Deleting a task

1. Log in to the CLS console.
2. Select **Shipping Task Management** on the left sidebar.
3. On the **Shipping Task Management** page, select the region, logset, log topic, and name of the target shipping task, and click **Delete** on the right.
4. In the pop-up window, click **Delete**.

Shipping to CKafka
Creating Shipping Task

Last updated：2024-01-20 17:44:35

Overview

You can ship log topic data to CKafka for real-time stream computing and storage. If you haven't purchased a CKafka instance, you can consider using the Consumption over Kafka feature of CLS.

Prerequisites

You have activated the CKafka service.
Make sure that the current account has the permission to enable shipping to CKafka. If your account is a sub-account, it needs to be authorized by the root account first. For more information, see Examples of Custom Access Policies.

Billing Description

Log shipping incurs private network read traffic fees (cross-region shipping is not supported for now), which CLS charges based on the size after Snappy or lz4 compression. If your raw log is 100 GB and you choose Snappy for compression, around 50 GB will be billable. As the read traffic price is 0.032 USD/GB, the fees will be 50 GB * 0.032 USD/GB = 1.6 USD.

Directions

1. Create a CKafka instance in the same region as the log topic. For more information, see Creating Instance.
2. Configure the following parameters to create a topic in the same region as the log topic. For more information, see Creating Topic.
Preset ACL Policy: Disable this option.
Show advanced configuration:
CleanUp.policy: Select **delete**; otherwise, shipping will fail.
max.message.bytes: Set this value to 8 MB or above. Otherwise, when the size of a single message in CLS exceeds the specified limit, the message cannot be written to the CKafka topic, and shipping will fail.
3. Go to the CLS console and enter the shipping task management page or log topic management page as needed.

On the left sidebar, click **Shipping Task Management** and select a region, logset, and log topic.
On the left sidebar, click **Log Topic** and select a log topic to be shipped to CKafka to enter the log topic management page.
4. Click the **Ship to CKafka** tab to enter the configuration page.
5. Click **Edit* on the right to enable shipping to CKafka. Then, select the target CKafka instance and topic as well as the log field to be shipped.
6. Click **OK** to start shipping to CKafka. If the task status is **Enabled**, the feature is enabled successfully.

Note:

To cleanse the log data before shipping to CKafka, see Log Filtering and Distribution.

FAQs

What should I do if the log data cannot be shipped to CKafka?

If ACL authentication is enabled in CKafka, the log data cannot be shipped. In this case, you need to disable the ACL of the topic.

What should I do if the system prompts that I have no permissions to read/write the CKafka topic?

If you directly use an API to ship data to CKafka, you may not have the read/write permissions of the CKafka topic. If you ship data in the console, the system will guide you through the authorization process, but if you directly call an API for shipping, you need to authorize manually. For more information, see CLS Service Role Authorization.

Shipping to ES
Dumping to ES Through SCF

Last updated：2024-01-20 17:44:35

Overview

This document describes how to use SCF to dump CLS logs to ES. CLS is mainly used for log collection, and SCF mainly provides node computing capabilities. For the data processing flowchart, please see Function Processing Overview.

Directions

Creating an SCF function

1. Log in to the SCF console and select Function Service on the left sidebar.
2. At the top of the **Function Service** page, select the **Beijing** region and click **Create** to go to the function creating page and configure the following parameters:
Function name: enter **CLSdemo**.
Runtime environment: select **Python 2.7**.
Creation Method: select **Function Template**.
Fuzzy search: enter **CLSToElasticsearch** and search.
3. Click **Learn More** in the template to view relevant information in the **Template Details** pop-up window, which can be downloaded.
4. After configuring the basic information, click **Next** to go to the function configuration page.
5. Keep the default configuration and click **Complete** to complete the function creation.

Note:

You should select the same VPC and subnet of CLS for the created function on the **Function Configuration** page as shown below:

Configuring CLS triggers

On the log topic details page, select **Function Processing** and click **Create**. Add the created function in the **Function Processing** pop-up window. See the figure below:
The main parameter information is as follows. Use the default values for the remaining configuration items.
Namespace: select the function namespace.
Function Name: select the function created in the Creating SCF function step.

Alias: select a function alias.

Maximum waiting time: configure the longest waiting time for a single event pull. Default value: 60s.

Testing the function

1. Download the log file in the test sample, extract `demo-scf1.txt` , and import it to the source CLS service.

2. Switch to the SCF console to view the execution result.

Select the **Log Query** tab on the function details page to view the printed log information.

3. Log in to the ES console to view the data dumping and processing result.

Note:

You can write specific data processing methods as needed.

Log Shipping
Consumption over Kafka
Consumption over Kafka

Last updated：2024-01-20 17:44:35

Overview

You can consume a log topic as a Kafka topic over the Kafka protocol. In practice, collected log data can be consumed through the Kafka Consumer or Kafka connectors provided by open-source communities, such as flink-connector-kafka and Kafka-connector-jdbc, to downstream big data components or data warehouses, including Spark, HDFS, Hive, Flink, and Tencent Cloud products like Oceanus and EMR.
This document provides demos for how to consume log topics with Flink and Flume.

Prerequisites

You have activated CLS, created a logset and a log topic, and collected log data.
Make sure that the current account has the permission to enable **Consumption over Kafka**. For more information, see Access Policy Templates.

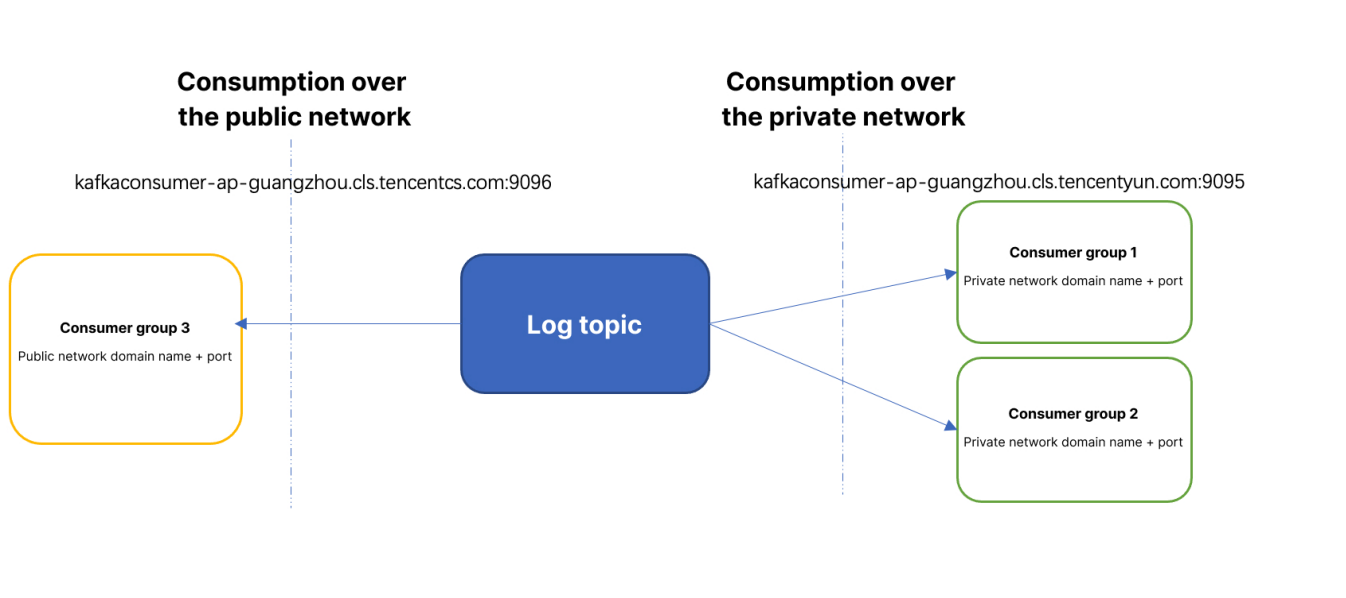
Use Limits

Supported Kafka protocol versions: 1.1.1 and earlier.
Supported compression modes: Snappy and LZ4.
User authentication mode: SASL_PLAINTEXT.
You can consume only current but not historical data.
Data in topics are retained for two hours.

Consumption over private or public network

Consumption over the private network: A private network domain name is used to consume logs, and the traffic price is 0.032 USD/GB. If your raw log is 100 GB and you choose Snappy for compression, around 50 GB will be billable, and the private network read traffic fees will be 50 GB * 0.032 USD/GB = 1.6 USD. In general, you can consume logs over the private network if your consumer and log topic are in the same VPC or region.

Consumption over the public network: A public network domain name is used to consume logs, and the traffic price is 0.141 USD/GB. If your raw log is 100 GB and you choose Snappy for compression, around 50 GB will be billable, and the public network read traffic fees will be 50 GB * 0.141 USD/GB = 7.05 USD. In general, you need to consume logs over the public network if your consumer and log topic are in different VPCs or regions.



Directions

1. Log in to the CLS console and select **Log Topic** on the left sidebar.
2. On the **Log Topic** page, click the target **Log Topic ID/Name** to enter the log topic management page.
3. On the log topic management page, click the **Consumption over Kafka** tab.
4. Click **Edit** on the right, set **Current Status** to **Enable**, and click **OK**.
5. The console displays the topic and host+port information, which can be copied for constructing the consumer SDK.

Consumer Parameter Description

The parameters of the consumer over Kafka are described as follows:

Parameter	Description
User authentication mode	Currently, only SASL_PLAINTEXT is supported.

hosts	Consumption over the private network: `kafkaconsumer-${region}.cls.tencentyun.com:9095`.Consumption over the public network: `kafkaconsumer-${region}.cls.tencentcs.com:9096`. For more information, see Available Regions.
topic	Topic name provided in the CLS console for consumption over Kafka, which can be copied by clicking the button next to it, such as `XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX`.
username	Configure it as `${logsetID}`, i.e., logset ID, such as `0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX`. You can copy the logset ID in the log topic list.
password	Configure it as ${SecretId}#${SecretKey}, such as `XXXXXXXXXXXXXX#YYYYYYYY`. Log in to the CAM console and click Access Key on the left sidebar to get the key information. You can use either the API key or project key. We recommend that you use a sub-account key and follow the principle of least privilege when authorizing a sub-account, that is, configure the minimum permission for `action` and `resource` in the access policy of the sub-account.

Note:

Be sure not to omit the `;` after the `${SecretId}#${SecretKey}` of the `jaas.config` in the following sample code; otherwise, an error will be reported.

SDK for Python

```
import uuid
from kafka import KafkaConsumer,TopicPartition,OffsetAndMetadata
consumer = KafkaConsumer(
#Topic name provided in the CLS console for consumption over Kafka, which can
be copied in the console, such as `XXXXXX-633a268c-XXXX-4a4c-XXXX-
7a9a1a7baXXXX`.
'Your consumption topic',
group_id = uuid.uuid4().hex,
auto_offset_reset='earliest',
# Service address + port (9096 for the public network and 9095 for the private
network). In this example, consumption is performed over the private network.
Enter this field accordingly.
bootstrap_servers = ['kafkaconsumer-${region}.cls.tencentyun.com:9095'],
security_protocol = "SASL_PLAINTEXT",
sasl_mechanism = 'PLAIN',
# The username is the logset ID, such as `ca5cXXXXdd2e-4ac0af12-92d4b677d2c6`.
sasl_plain_username = "${logsetID}",
```

```
#The password is a string of your `SecretId#SecretKey`, such as
`AKIDWrwkHYYHjvqhz1mHVS8YhXXXX#XXXXuXtymIXT0Lac`. Note that `#` is required. We
recommend that you use a sub-account key and follow the principle of least
privilege when authorizing a sub-account, that is, configure the minimum
permission for `action` and `resource` in the access policy of the sub-account.
sasl_plain_password = "${SecretId}#${SecretKey}",
api_version = (1,1,1)
)
print('begin')
for message in consumer:
    print('begins')
    print ("Topic:[%s] Partition:[%d] Offset:[%d] Value:[%s]" % (message.topic,
message.partition, message.offset, message.value))
    print('end')
```

Consumption of CLS Logs by Oceanus

Create a job in the Oceanus console.

```
CREATE TABLE `nginx_source`
(   # Fields in the log
    `@metadata` STRING,
    `@timestamp` TIMESTAMP,
    `agent` STRING,
    `ecs` STRING,
    `host` STRING,
    `input` STRING,
    `log` STRING,
    `message` STRING,
    `partition_id` BIGINT METADATA FROM 'partition' VIRTUAL,    -- Kafka
partition
    `ts` TIMESTAMP(3) METADATA FROM 'timestamp'
)  WITH (
  'connector' = 'kafka',
  #Topic name provided in the CLS console for consumption over Kafka, which can
be copied in the console, such as `XXXXXX-633a268c-XXXX-4a4c-XXXX-
7a9a1a7baXXXX`.
  'topic' = 'Your consumption topic',
  # Service address + port (9096 for the public network and 9095 for the
private network). In this example, consumption is performed over the private
network. Enter this field accordingly.
  'properties.bootstrap.servers' =
'kafkaconsumer-${region}.cls.tencentyun.com:9095',
    # Replace it with the name of your consumer group
```

```
  'properties.group.id' = 'The name of your consumer group',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'json.fail-on-missing-field' = 'false',
  'json.ignore-parse-errors' = 'true' ,
  # The username is the logset ID, such as `ca5cXXXXdd2e-4ac0af12-
92d4b677d2c6`.
  #The password is a string of your `SecretId#SecretKey`, such as
`AKIDWrwkHYYHjvqhz1mHVS8YhXXXX#XXXXuXtymIXT0Lac`. Note that `#` is required. We
recommend that you use a sub-account key and follow the principle of least
privilege when authorizing a sub-account, that is, configure the minimum
permission for `action` and `resource` in the access policy of the sub-account.
Be sure not to omit the `;` at the end of the `jaas.config`; otherwise, an
error will be reported.
  'properties.sasl.jaas.config' =
'org.apache.kafka.common.security.plain.PlainLoginModule required
username="${logsetID}" password="${SecretId}#${SecretKey}";',
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.sasl.mechanism' = 'PLAIN'
);
```

Consumption of CLS Log by Flink

Enabling log consumption over Kafka

Enable log consumption over Kafka and get the consumer service domain name and topic as instructed in Directions.

Confirming flink-connector-kafka dependency

After confirming that `flink-connector-kafka` exists in `flink lib`, directly register a Kafka table in `sql`. The dependency is as follows:

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka</artifactId>
  <version>1.14.4</version>
</dependency>
```

Registering a Flink table

```
CREATE TABLE `nginx_source`
(
    # Fields in the log
```

```
    `@metadata` STRING,
    `@timestamp` TIMESTAMP,
    `agent` STRING,
    `ecs` STRING,
    `host` STRING,
    `input` STRING,
    `log` STRING,
    `message` STRING,
    # Kafka partition
    `partition_id` BIGINT METADATA FROM 'partition' VIRTUAL,
    `ts` TIMESTAMP(3) METADATA FROM 'timestamp'
)  WITH (
  'connector' = 'kafka',
  #Topic name provided in the CLS console for consumption over Kafka, which can be
  'topic' = 'Your consumption topic',
  # Service address + port (9096 for the public network and 9095 for the private ne
  'properties.bootstrap.servers' = 'kafkaconsumer-${region}.cls.tencentyun.com:9095
   # Replace it with the name of your consumer group
  'properties.group.id' = 'The name of your consumer group',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'json.fail-on-missing-field' = 'false',
  'json.ignore-parse-errors' = 'true' ,
  # The username is the logset ID, such as `ca5cXXXXdd2e-4ac0af12-92d4b677d2c6`.
  #The password is a string of your `SecretId#SecretKey`, such as `AKIDWrwkHYYHjvqh
  'properties.sasl.jaas.config' = 'org.apache.kafka.common.security.plain.PlainLogi
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.sasl.mechanism' = 'PLAIN'
);
```

Querying

After successful execution, you can use the statement for query.

```
select count(*) , host from nginx_source group by host;
```

Consumption of CLS Log by Flume

If you need to have log data consumed by a self-built HDFS or Kafka cluster, you can use the Flume component for forwarding as instructed below.

Enabling log consumption over Kafka

Enable log consumption over Kafka and get the consumer service domain name and topic as instructed in Directions.

Flume configuration

```
a1.sources = source_kafka
a1.sinks = sink_local
a1.channels = channel1

#Configure the source
a1.sources.source_kafka.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.source_kafka.batchSize = 10
a1.sources.source_kafka.batchDurationMillis = 200000
# Service address + port (9096 for the public network and 9095 for the private network). In this example, consumption is performed over the private network. Enter this field accordingly.
a1.sources.source_kafka.kafka.bootstrap.servers = $kafkaconsumer-${region}.cls.tencentyun.com:9095
#Topic name provided in the CLS console for consumption over Kafka, which can be copied in the console, such as `XXXXXX-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX`.
a1.sources.source_kafka.kafka.topics = Your consumption topic
#Replace it with the name of your consumer group
a1.sources.source_kafka.kafka.consumer.group.id = The name of your consumer group
a1.sources.source_kafka.kafka.consumer.auto.offset.reset = earliest
a1.sources.source_kafka.kafka.consumer.security.protocol = SASL_PLAINTEXT
a1.sources.source_kafka.kafka.consumer.sasl.mechanism = PLAIN
# The username is the logset ID, such as `ca5cXXXXdd2e-4ac0af12-92d4b677d2c6`.
#The password is a string of your `SecretId#SecretKey`, such as `AKIDWrwkHYYHjvqhz1mHVS8YhXXXX#XXXXuXtymIXT0Lac`. Note that `#` is required. We recommend that you use a sub-account key and follow the principle of least privilege when authorizing a sub-account, that is, configure the minimum permission for `action` and `resource` in the access policy of the sub-account. Be sure not to omit the `;` at the end of the `jaas.config`; otherwise, an error will be reported.
a1.sources.source_kafka.kafka.consumer.sasl.jaas.config = org.apache.kafka.common.security.plain.PlainLoginModule required username="${logsetID}"
password="${SecretId}#${SecretKey}";



// Configure the sink
a1.sinks.sink_local.type = logger

a1.channels.channel1.type = memory
a1.channels.channel1.capacity = 1000
a1.channels.channel1.transactionCapacity = 100
```

```
// Bind the source and sink to the channel
a1.sources.source_kafka.channels = channel1
a1.sinks.sink_local.channel = channel1
```

Customized Consumption

Last updated：2024-10-17 15:55:34

Prerequisites

1. Cloud Log Service is activated. Create a log set and log topic, and you have successfully collected log data.
2. Sub-accounts/Collaborators need root account authorization. For authorization steps, see CAM-Based Permission Management. For copying authorization policy, see CLS Access Policy Templates.

Consumption Process within a Consumer Group

When consuming data within a consumer group, the server manages the consumption tasks for all consumers within the group. It automatically balances these tasks based on the correlation between the number of topic partitions and the number of consumers. Moreover, it records the consumption progress for each partition in the topic to guarantee that different consumers can consume data without any duplication. The detailed process of consumption within a consumer group proceeds as follows:

1. Create a consumer group.
2. Every consumer periodically sends heartbeats to the server.
3. The consumer group automatically assigns topic partitions to consumers according to the load balancing situation of the topic partitions.
4. Consumers retrieve the partition offsets and consume the data according to the list of allocated partitions.
5. Consumers periodically update their consumption progress for each partition to the consumer group, facilitating the next round of task allocation by the group.
6. Repeat steps 2 through 6 until consumption is completed.

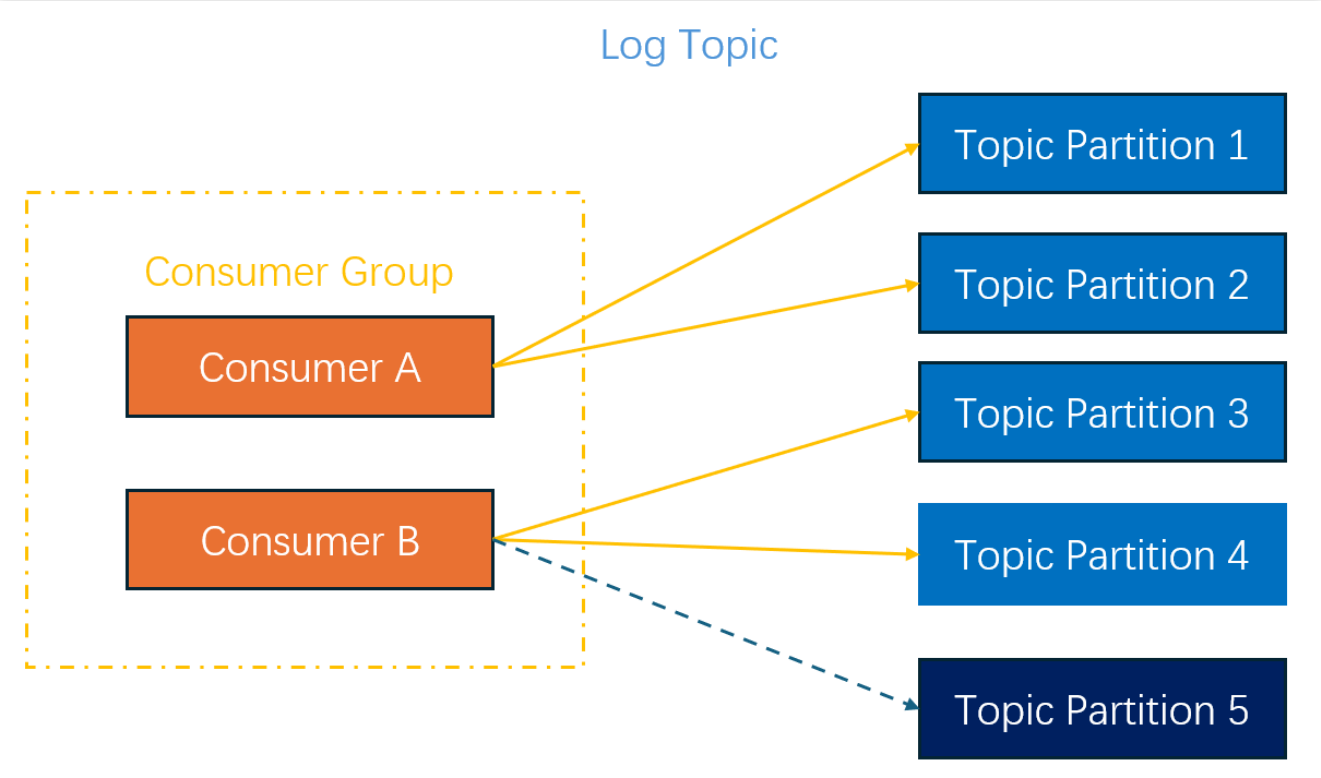
Consumption Balancing

The consumer group will dynamically adjust the consumption tasks of each consumer according to the number of active consumers and topic partitions to ensure balanced consumption. At the same time, consumers can save the consumption progress in each topic partition to ensure that they can continue to consume data after fault recovery and avoid repeated consumption.

Example 1: Topic Partition Change

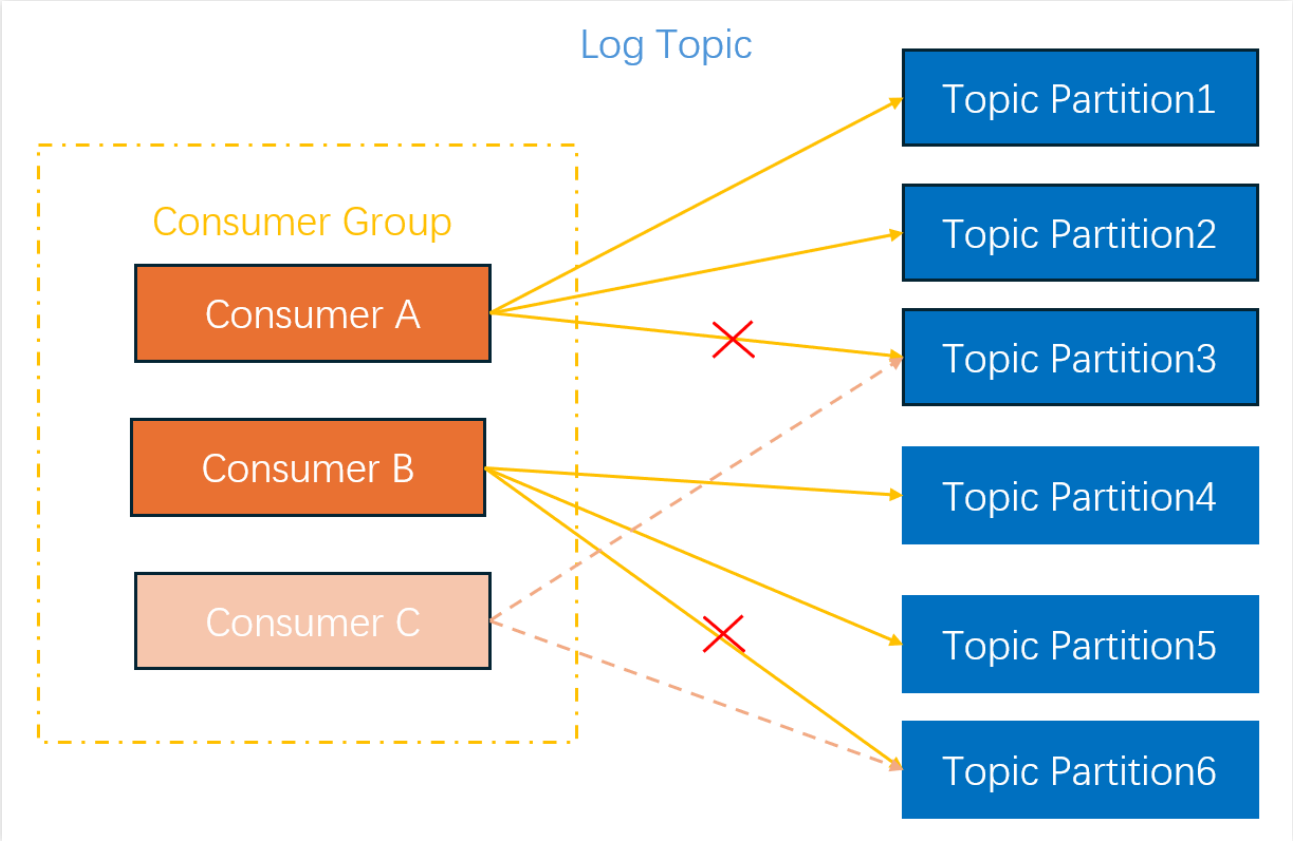
For example, a log topic has two consumers. Consumer A consumes data in partitions 1 and 2, and consumer B in partitions 3 and 4. After partition 5 is added through partition splitting, the consumer group will automatically allocate

partition 5 to consumer B for consumption, as shown in the figure below:



Example 2: Consumer Change

For example, a log topic has two consumers. Consumer A consumes data in partitions 1, 2, and 3, and consumer B in partitions 4, 5, and 6. To ensure that the consumption speed is equal to the generation speed, consumer C is added. The consumer group will reallocate partitions. Then, partitions 3 and 6 will be allocated to consumer C for consumption, as shown in the figure below:



Consumption Demo (Python)

Note:

For the complete Demo, see tencentcloud-cls-sdk-python, It is recommended to use Python version 3.5 or above for data consumption.

The usage and instructions of the Demo are as follows:

1. Install SDK. For details, see tencentcloud-cls-sdk-python.

```
pip install git+https://github.com/TencentCloud/tencentcloud-cls-sdk-python.git
```

2. Process the data to be consumed by the consumer and save the user log data in the log_group struct. The log_group struct is as follows:

```
log_group {
    source   //Log source, which is usually the machine's IP address.
    filename //Log file name
    logs {
        time //Log time, which is a Unix timestamp in microseconds.
        user_defined_log_kvs  //User log fields
    }
}
```

The implementation by using the SampleConsumer method is as follows:

```
class SampleConsumer(ConsumerProcessorBase):
    last_check_time = 0
    log_results = []
    lock = RLock()

    def initialize(self, topic_id):
        self.topic_id = topic_id

    # Process the data to be consumed.
    def process(self, log_groups, offset_tracker):
        for log_group in log_groups:
            for log in log_group.logs:
                # Process a single row of data.
                item = dict()
                item['filename'] = log_group.filename
                item['source'] = log_group.source
                item['time'] = log.time
                for content in log.contents:
                    item[content.key] = content.value

                with SampleConsumer.lock:
                    # Aggregate data to SampleConsumer.log_results.
                    SampleConsumer.log_results.append(item)

        # Submit offset every 3 seconds.
        current_time = time.time()
        if current_time - self.last_check_time > 3:
            try:
                self.last_check_time = current_time
                offset_tracker.save_offset(True)
            except Exception:
                import traceback
                traceback.print_exc()
        else:
            try:
                offset_tracker.save_offset(False)
            except Exception:
                import traceback
                traceback.print_exc()

        return None

    # Call this function when Worker exits for cleanup.
    def shutdown(self, offset_tracker):
```

```
        try:
            offset_tracker.save_offset(True)
        except Exception:
            import traceback
            traceback.print_exc()
```

3. Create a consumer and start the consumer thread. The consumer then consumes data from the specified topic.

Parameter	Description	Default Value	Value Range
endpoint	Request Domain, domain name of the **API for Log Upload Tag** page.	-	Supported regions: Beijing, Shanghai, Guangzhou, Nanjing, Hong Kong (China), Tokyo, Eastern United States, Singapore, and Frankfurt.
access_key_id	For your Secret_id, go to CAM.	-	-
access_key	For your Secret_key, go to CAM.	-	-
region	Topic's region. For example, ap-beijing, ap-guangzhou, ap-shanghai. For more details, see Regions and Access Domains.	-	Supported regions: Beijing, Shanghai, Guangzhou, Nanjing, Hong Kong (China), Tokyo, Eastern United States, Singapore, and Frankfurt.
logset_id	Logset ID. Only one logset is supported.	-	-
topic_ids	Log topic ID. For multiple topics, use , to separate.	-	-
consumer_group_name	Consumer Group Name	-	-
internal	Private network: TRUE Public network: FALSE **Note:** For private network/public network read traffic cost, see Product Pricing.	FALSE	TRUE/FALSE
consumer_name	Consumer name. Within the same consumer group, consumer	-	A string consisting of 0-9, aA-zZ, '-', '_', '.'.

	names must be unique.		
heartbeat_interval	The interval of heartbeats. If consumers fail to report a heartbeat for two intervals, they will be considered offline.	20	0-30 minutes
data_fetch_interval	The interval of consumer data pulling. Cannot be less than 1 second.	2	-
offset_start_time	The start time for data pulling. The string type of **unix Timestamp** , with second-level precision. For example, 1711607794. It can also be directly configured as "begin" and "end". begin: The earliest data within the log topic lifetime. end: The latest data within the log topic lifetime.	"end"	"begin"/"end"/unix Timestamp
max_fetch_log_group_size	The data size for a consumer in a single pulling. Defaults to 2 M and up to 10 M.	2097152	2M - 10M
offset_end_time	The end time for data pulling. Supports a string-type **unix Timestamp** , with second-level precision. For example, 1711607794. **Not filling this field represents continuous pulling**.	-	-

```
def sample_consumer_group():
    # CLS Access Point. Fill in according to the actual situation.
    endpoint = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_ENDPOINT', '')
    # Region to be accessed
    region = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_REGION', '')
    # User Secret_id
    access_key_id = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_ACCESSID', '')
    # User Secret_key
    access_key = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_ACCESSKEY', '')
    # ID of the logset to be consumed
```

```
    logset_id = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_LOGSET_ID', '')
    # ID list for consumed log topics, separated by English commas.
    topic_ids = os.environ.get('TENCENTCLOUD_LOG_SAMPLE_TOPICS', '').split(',')
    # Consumer group name. The consumer group name under the same logset is unique.
    consumer_group = 'consumer-group-1'
    # Consumer name
    consumer_name1 = "consumer-group-1-A"
    consumer_name2 = "consumer-group-1-B"

    assert endpoint and access_key_id and access_key and logset_id, ValueError("end
                                                                               "log
    # Create Client to access the TencentCloud API.
    client = YunApiLogClient(access_key_id, access_key, region=region)
    SampleConsumer.log_results = []

    try:
        # Create two consumer configurations.
        option1 = LogHubConfig(endpoint, access_key_id, access_key, region, logset_
                               consumer_name1, heartbeat_interval=3, data_fetch_int
                               offset_start_time='end', max_fetch_log_group_size=10
        option2 = LogHubConfig(endpoint, access_key_id, access_key, region, logset_
                               consumer_name2, heartbeat_interval=3, data_fetch_int
                               offset_start_time='end', max_fetch_log_group_size=10

        # Create a consumer.
        print("*** start to consume data...")
        client_worker1 = ConsumerWorker(SampleConsumer, consumer_option=option1)
        client_worker2 = ConsumerWorker(SampleConsumer, consumer_option=option2)

        # Start the consumer
        client_worker1.start()
        client_worker2.start()

        # Wait for 2 minutes, or continue execution after the data is obtained.
        sleep_until(120, lambda: len(SampleConsumer.log_results) > 0)

        # Stop the consumer.
        print("*** stopping workers")
        client_worker1.shutdown()
        client_worker2.shutdown()

        # Print the summarized log data
        print("*** get content:")
        for log in SampleConsumer.log_results:
            print(json.dumps(log))

        # Print consumer group information, including the consumer group name, cons
```

```
        print("* consumer group status *")
        ret = client.list_consumer_group(logset_id, topic_ids)
        ret.log_print()

        # Delete consumer group
        print("*** delete consumer group")
        time.sleep(30)
        client.delete_consumer_group(logset_id, consumer_group)
    except Exception as e:
        import traceback
        traceback.print_exc()
        raise e


if __name__ == '__main__':
    sample_consumer_group()
```

Shipping to DLC
Creating a DLC Shipping Task

Last updated：2025-06-27 19:12:54

After logs are collected to CLS, you can ship log data to Tencent Cloud DLC for analysis.

Prerequisites

Enable Cloud Log Service, create a logset and log topic, and ensure log data has been successfully collected.
Activate Tencent Cloud DLC service and create a database and data table in the region of the log topic to be delivered. See SQL editor.
Sub-account/Collaborator requires root account authorization. For authorization steps, see CAM management permissions. For authorization policy, refer to CLS Access Policy Template.
The Tencent Cloud CLS service role has been authorized to access DLC. When most users operate through the console, the system will guide the user to complete authorization; a small number of users cross the console and directly call the API, and these customers need to manually authorize. For details, please refer to Delivery Task Role Authorization.

Directions

1. Log in to the Cloud Log Service console.
2. Select **Log Topic** in the left sidebar.
3. Click the log topic ID/name to deliver, and enter the log topic management page.
4. Click the **Shipping to DLC** tab, enter the Send to DLC configuration page, and click **Create**.

Basic Configuration Item	Description	Rule	Required
Shipping Task Name	Name of the delivery task.	-	Required
Service log	Write the monitoring metrics of the delivery task running to the free log topic cls_service_log.	-	No
Shipping Mode	Currently only support **Batch Shipping**.	-	No
File Size	The size of the raw log file to be delivered works in conjunction with the delivery interval time. Whichever	5 - 256, unit: MB.	No

	condition is met first will trigger the rule to compress the file, and then deliver it to DLC. For example, if you configure 256MB and 15 minutes, and the file size reaches 256MB in 5 minutes, then the file size condition will trigger the delivery task first.		
Shipping Interval	Specify the interval to trigger a delivery. This works with the delivery file size. Whichever condition is met first will trigger the rule to compress the file and deliver it to DLC. For example, if you configure 256MB and 15 minutes, and the file size is only 200MB after 15 minutes, then the interval time condition will trigger the delivery task first.	300 - 900, unit: s.	No

5. Click **Next** to enter **Data Table Configuration**.

Data Table Configuration Item	Description	Rule	Required
Data Catalogs	Currently, only DataLakeCatalog is supported.	-	No
Database	Select your DLC database.	-	Required
Data Table	Select your DLC data table.	-	Required

data Field	Log Field Name: Map fields in the CLS log to corresponding DLC fields, as shown in the log in figure: Only support filling in the key of the first-layer node in JSON, such as app_name. Nested nodes like details.request_id are not supported.
	<pre>{
	"__TIMESTAMP__":1742543131,
	"app_name": "cls",
	"category": "Filter",
	"latency": "11",
	"details": {
	"request_id": "123456dfg",
	"trace_id": "0610df2a4c5d9cdf"
	}
	}</pre>Preview Logs: Click this button to view log samples (JSON format) on the right side of the page, helping you select fields and fill in log field names.
	Data Table Field Name: The data table field is read from DLC and cannot be modified here. Please go to DLC to edit.
		-	Required

	Field Type: Type of DLC field, not supported here. Please go to DLC to modify. Assignment information returned upon resolution failure: NULL/empty/custom value. If the user's raw field cannot be parsed as the specified type, the system will attempt to parse the unresolved field. If the unresolved field also cannot be parsed, it will fill the default zero value. int/bigint: corresponds to 0 float/double/decimal: corresponds to 0 date: today timestamp: corresponding current timestamp Enable Mapping: whether to map this field to the DLC table. If not required, toggle off.		
Partition Field	Log Field Name: the field name of the log, used for Mapping the partition field in DLC. If your DLC table is partitioned by time, we recommend using the log time field here, such as __TIMESTAMP__. Data Table Field Name: partition field, read from DLC, cannot be modified here. Please go to DLC to edit. Field Type: type of partition field, read from DLC, not supported here. Please go to DLC to modify.	-	Required

6. Click **Submit** to complete the configuration of the new task.

Metric Topic Shipping

Last updated：2024-08-27 17:58:40

Overview

After metrics are collected to CLS, they can be shipped to a time series database (such as Prometheus) with RemoteWrite enabled.

Prerequisites

1. CLS has been enabled, metric topics have been created, and metric data has been successfully collected.
2. **RemoteWrite has been enabled** for the time series database.
3. Sub-accounts/collaborators have been authorized by using the root account. For authorization steps, see Sub-account Authorization. To copy the authorization policy, see CLS Access Policy Template.

Directions

1. Log in to the Cloud Log Service console.
2. In the left sidebar, click **Metric Topic**.
3. Click the ID/name of the metric topic to be shipped to enter the metric topic management page.
4. Click **Deliver metrics** to go to the delivery configuration page, then click **Create**.

The configuration items are described as follows:

Configuration Item	Description	Limit	Required
Task Name	Name of the shipping task	No more than 128 characters.	Required
Shipping object	Prometheus Thanos	Configure RemoteWrite of the shipping destination in advance.	-
Access method	**Private network**: Choose a private network if the shipping destination is accessed through the private	The shipping destination and	Required

	network. For example, Prometheus is deployed on a CVM instance in the same region as the CLS instance. **Public network**: **Not supported currently**.	metric topic are in the same region.	
Network (private network)	VPC network of the shipping destination	-	Required
Network service type (private network)	**CVM**: The shipping destination is directly deployed on a CVM instance. **Note:** Choose CVM if you want to ship metric topics to Prometheus on the Tencent Cloud Observability Platform. **CLB:** The service address and port of the shipping destination are forwarded through Tencent Cloud CLB.	-	-
Remote Write address	**Example:** http://192.168.2.17:9090/api/v1/prom/write	-	Required
Authentication method	Authentication method for accessing the time series database via RemoteWrite: **BASIC AUTH:** Username and password are required. **Note:** For Prometheus on the Tencent Cloud Observability Platform, enter the Grafana page, and choose DataSource > Settings > Basic Auth Details on the left configuration pane to view the user and password. **No authentication**	-	Required
Test connectivity	Test the network connectivity between CLS and the shipping destination. **Note:** The metric shipping configuration can be submitted only after the connectivity test passes.	-	Required

Monitoring Alarm
Monitoring Alarm Overview

Last updated：2024-01-20 17:59:35

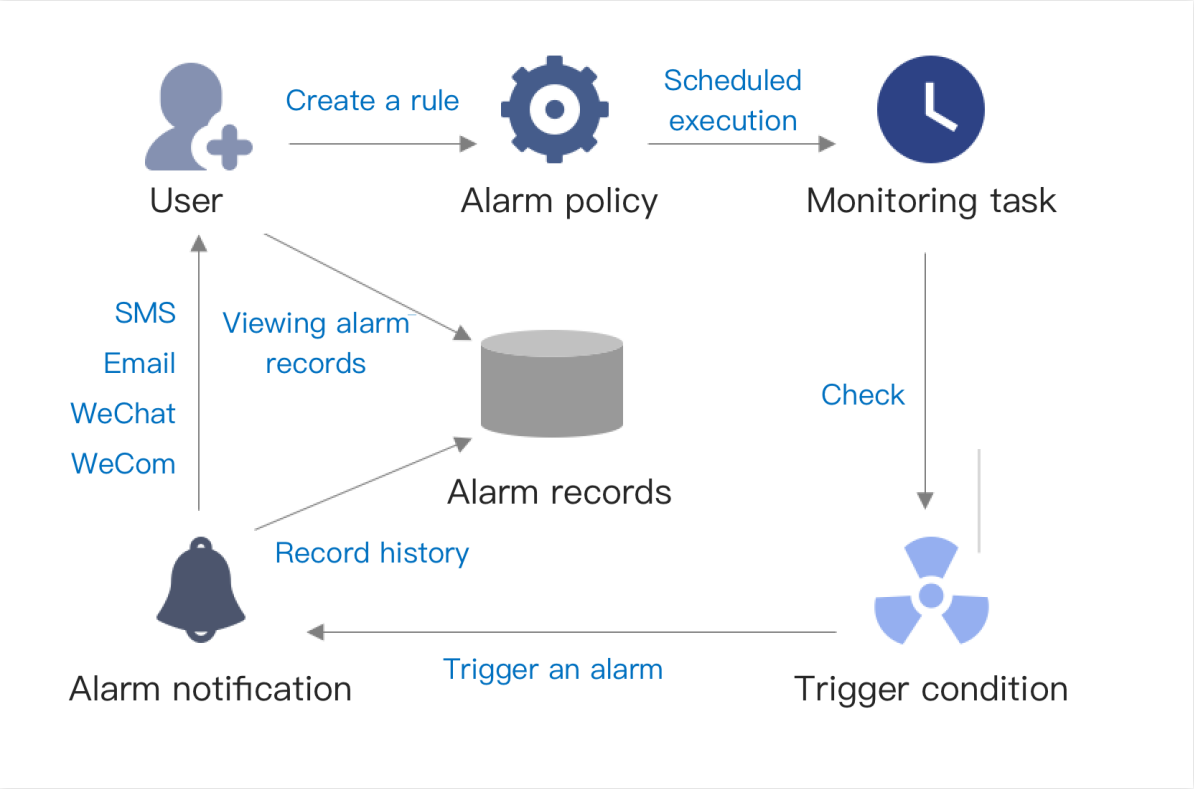
Overview

CLS supports setting alarm policies for one or more log topics. An alarm policy periodically performs monitoring tasks and sends alarm notifications when the query and analysis results meet the trigger condition, so that you can find exceptions in time.

Relevant concepts

Name	Description
Alarm policy	It is the management unit for monitoring alarms. An alarm policy contains various information such as monitoring object, monitoring period, trigger condition, alarm frequency, and notification template.
Monitoring object	A log topic can be used as the monitoring object, a query or analysis statement can be executed on the log topic, and then the query or analysis result can be checked.
Trigger condition	The query and analysis result is checked, and if the trigger condition expression is true, an alarm will be triggered.
Monitoring period	It is the policy execution period. A fixed period (such as every 5 minutes) and a fixed time (such as 12:00 every day) are supported.
Alarm frequency	It is the alarm frequency after the trigger condition is met, which helps avoid frequent alarm notifications.
Notification group	Supported notification channels include SMS, WeChat, phone calls, email, and webhook.

Flowchart


Create a rule
Scheduled execution
User
Alarm policy
Monitoring task
SMS
Email
WeChat
WeCom
Viewing alarm records
Check
Alarm records
Record history
Alarm notification
Trigger an alarm
Trigger condition

Managing Alarm Policies
Configuring Alarm Policies

Last updated：2024-01-20 17:59:36

Overview

This document describes how to configure an alarm policy based on logs so that alarms can be sent when certain conditions are met, such as when there are too many error logs or the API response time is too long.

Prerequisites

You have uploaded the log to a log topic and configured the index.
The log topic is not in STANDARD_IA storage, which doesn't support alarm policy configuration. An alarm policy requires SQL statements. We recommend that you structure logs as instructed in Collection Overview.
You have logged in to the CLS console and entered the Alarm Policy page.

Directions

On the **Alarm Policy** page, click **Create** and configure the following items.

Configuring the monitoring object and monitoring task

Monitoring Object: Select the target log topic(s). It can be determined whether the trigger conditions are met separately for each log topic. You can select up to 20 log topics in the same region. If multiple log topics meet the trigger conditions at the same time, multiple alarms will be generated at a time.
Monitoring Task
Query Statement: It is used for log topics and needs to contain the analysis statement (i.e., SQL statement as described in Overview and Syntax Rules).
Example 1: To count logs with errors, use `status:error | select count(*) as ErrCount`.
Example 2: To calculate the average response time of the domain name "domain:aaa.com", enter `domain:"aaa.com" | select avg(request_time) as Latency`.
Query Time Range: It indicates the time range of data for query by the query statement, which can be up to the last 24 hours.
Trigger Condition: An alarm is triggered when the trigger condition is met. In the condition expression, `$N.keyname` is used to reference the query statement result. Here, `$N` indicates the Nth query statement in

the current alarm policy, and `keyname` indicates the corresponding field name. For more information on the expression syntax, see Trigger Condition Expression.

Example 1: To trigger an alarm when the number of logs with errors exceeds 10, enter `$1.ErrCount > 10`. Here, `$1` indicates the first query statement, and `ErrCount` indicates the `ErrCount` field in the result.

Example 2: To trigger an alarm when the domain name "domain:aaa.com" takes more than 5 seconds on average to respond, enter `$2.Latency > 5`. Here, `$2` indicates the first query statement, and `Latency` indicates the `Latency` field in the result.

Trigger by Group: It specifies whether the trigger condition expression should trigger alarms by group. When it is enabled, if multiple results of the query statement meet the trigger condition, the results will be grouped based on the group field, and an alarm will be triggered for each group.

For example, if the query statement 2 is `* | select avg(request_time) as Latency,domain group by domain order by Latency desc limit 5`, and multiple results are returned:

Latency	Domain
12.56	aaa.com
9.45	bbb.com
7.23	ccc.com
5.21	ddd.com
4.78	eee.com

If the trigger condition is `$2.Latency > 5`, then it is met by four results.

If triggering by group is not enabled, only one alarm will be triggered when the trigger condition is met by one of the above execution results.

If it is enabled and the results are grouped by the `domain` field, four alarms will be triggered separately for the above execution results.

Note:

When triggering by group is enabled, the trigger condition may be met by multiple results, and a large number of alarms will be triggered, leading to an alarm storm. Therefore, configure the group field and trigger condition appropriately.

When specifying the group field, you can divide execution results into up to 1,000 groups. No alarms will be triggered for excessive groups.>

Execution Cycle: It indicates the execution frequency of the monitoring task, which can be configured in the following two ways:

Period Configuration Method	Description	Example
Fixed frequency	Monitoring tasks are performed at fixed intervalsInterval: 1–1,440 minutes. Granularity: Minute	Monitoring tasks are performed once every 5 minutes
Fixed time	Monitoring tasks are performed once at fixed points in timeTime point range: 00:00–23:59. Granularity: Minute	Monitoring tasks are performed once at 02:00 every day

Configuring multi-dimensional analysis

When an alarm is triggered, raw logs can be further analyzed through multi-dimensional analysis, and the analysis result can be added to the alarm notification to facilitate root cause discovery. The multi-dimensional analysis doesn't affect the alarm trigger condition.

Multi-dimensional Analysis Type	Description
Related raw logs	Get the raw logs that meet the search condition of the query statement. The log field, quantity, and display form can be configured. For example, when an alarm is triggered by too many error logs, you can view the detailed logs in the alarm.
Top 5 field values by occurrence and their percentages	For all the logs within the time range when the alarm is triggered, group them based on the specified field and get the top 5 field values and their percentages. For example, when an alarm is triggered by too many error logs, you can get the top 5 URLs and top 5 response status codes.
Custom search and analysis	Execute the custom search and analysis statement for all the logs within the time range when the alarm is triggered. Example 1: `*

Note:

The "related raw logs" and "top 5 field values by occurrence and their percentages" options support the automatic association with the search condition of the specified query statement (excluding the analysis statement, i.e., SQL filter condition), so as to indicate to perform multi-dimensional analysis on raw logs that meet what conditions.

Configuring an alarm notification

Alarm Frequency:

Duration: A notification will be sent only after the trigger condition is met constantly a certain number of times (which can be 1–10 and is 1 by default).

Interval: No notifications will be sent within the specified interval after the last notification. For example, the **an alarm will be triggered every 15 minutes** option indicates that only one alarm will be sent within 15 minutes.

Notification Group:

The notification channels and objects can be set by associating a notification channel group. Notifications can be sent by SMS, email, phone call, Weixin, WeCom, and custom callback API (webhook). For more information, please see Managing Notification Groups.

Notification Content:

By adding preset variables to the notification content, you can add specified information to the alarm notification. For more information on variables, see Alarm Notification Variable.

Custom Webhook Configuration:

If the selected notification group contains a custom webhook, the custom webhook input box will be displayed. You can customize the request header and request body there, which will be used by CLS to call the specified API when an alarm is triggered. In the request header and body, you can use notification content variables to send relevant data to the specified API.

Best Practices

Setting Alarm Trigger Conditions by Time PeriodSetting Interval-Valued Comparison and Periodically-Valued Comparison as Alarm Trigger Conditions

Trigger Condition Expression

Last updated：2024-01-20 17:59:35

A trigger condition expression is used to determine whether to trigger an alarm. The query analysis result of the monitoring object is input as a variable for the trigger expression. If the expression is true, an alarm will be triggered.

Syntax Description

Operator	Description	Example
$N.keyname	Imports the query analysis result. `N` is the monitoring object number, `keyname` is the field name in the **query analysis result** (which must start with a letter and can contain letters, digits, and underscores. We recommend you use the AS syntax to set an alias for the result.)	$1.ErrCount
+	Addition operator	$1.ErrCount+$1.FatCount>10
-	Subtraction operator	$1.Count-$1.InfoCount>100
*	Multiplication operator	$1.RequestMilSec*1000>10
/	Division operator	$1.RequestSec/1000>0.01
%	Modulo operator	$1.keyA%10==0
==	Comparison operator: equal to	$1.ErrCount==100 $1.level=="Error"
>	Comparison operator: greater than	$1.ErrCount>100
<	Comparison operator: less than	$1.pv<100
>=	Comparison operator: greater than or equal to	$1.ErrCount>=100
<=	Comparison operator: less than or equal to	$1.pv<=100
!=	Comparison operator: not equal to	$1.level!="Info"
()	Parentheses for controlling the operation priority	($1.a+$1.b)/$1.c>100
&&	Logical operator: AND	$1.ErrCount>100 && $1.level=="Error"

||	Logical operator: OR	$1.ErrCount>100 || $1.level=="Error"

An alarm will be triggered only if the expression is true. For example, if the calculation result of `$1.a+$1.b` is 100, no alarms will be triggered; if the result is greater than or equal to 100, an alarm will be triggered.

`keyname` in `$N.keyname` is the field name of the query analysis result. It must start with a letter and can contain letters, digits, and underscores, such as `level:error | select count(*) AS errCount`. `errCount` can be directly used as `keyname` in the trigger condition expression. If the field name contains special symbols, you need to enclose the imported variable with `[]`, such as `[$1.count(*)]`. **We recommend you use an** AS analysis statement **to set an alias for the result field name.**

Up to three monitoring objects can be set in an alarm policy. Each one is identified by a number starting from 1. For example, `$1.key1` imports the `key1` field name in the query whose number is `1`, and `$2.key2` imports the `key2` field name in the query whose number is `2`.

If multiple values are returned in the query analysis result, the expression will be calculated according to the values for up to 1,000 times or until the calculation result is `true`. For example, if the expression is `$1.a+$2.b>100`, analysis 1 returns m results, and analysis 2 returns n results, then the expression will be calculated for m * n times, and calculation will stop when `$1.a+$2.b>100` is `true` or after 1,000 times of calculation.

Channels to Receive Alarm Notifications Receiving Alarm Notifications via SMS and Phone

Last updated：2024-01-20 17:59:36

Overview

This document describes how to receive alarm notifications via SMS and phone.

Use Limits

We recommend that you set to receive only urgent alarms via phone, as the number of alarms that can be received in this manner is limited as follows:
A mobile number can receive up to **one** alarm per 30 seconds, up to **three** alarms per 10 minutes, and up to **50** alarms per day.
A mobile number can receive up to **three** alarms between 10:00 PM and 8:00 AM.
The above limits apply to not only phone alarm notifications generated by CLS and are also shared by those generated by CM and Message Center.

Directions

Verifying the mobile number

1. Log in to the CAM console.
2. Click **User** > **User List** on the left sidebar to enter the user list page.
3. Click the **Username** of the target user to enter the **User Details** page.
4. In the **Contact Number** column, click

.
5. In the pop-up window, enter the target mobile number, click **OK**, and verify the message channel as prompted.

Creating an SMS/phone alarm channel

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Notification Group** to enter the notification group list page.
3. Click **Create**.
4. On the **Create** page, set the following information and click **OK**.

Name: Enter a custom notification group name.

Recipient: Select the target **User**/**User Group**.

Type: Select **Email**, **SMS**, or **Phone**.

Notify By: Select **SMS** or **Phone**.

Email

Last updated：2024-01-20 17:59:36

Overview

This document describes how to receive alarm notifications via email.

Directions

Verifying email address

1. Log in to the CAM console.
2. Click **User** > **User List** on the left sidebar to enter the user list page.
3. Find the user for whom to configure the email notification channel and click the username to enter the user details page.
4. In the **Contact Email** column, click

.
5. In the pop-up window, enter the target email address and click **Confirm**.
6. In the **Email** column, click **Send Verification Link**.
7. Check the inbox and click **Confirm to Receive** in the "Tencent Cloud Email Receipt Verification" message.

Selecting email alarm channel

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Alarm Policy** to enter the alarm policy list page.
3. Find the policy for which to enable notification receipt and click the policy name to enter the policy editing page.
4. Select the recipient group and click **Edit**.
5. In the pop-up window, select **Email** and click **OK**.

Custom Callback APIs

Last updated：2024-01-20 17:59:35

Overview

This document describes how to receive alarm notifications via custom callback APIs (webhooks).

Use Limits

Each custom callback API can send up to 20 messages per minute. If many alarm policies are configured, we recommend that you create multiple custom callback APIs and associate them with different alarm policies; otherwise, multiple alarm policies may trigger alarms simultaneously, and you may fail to receive some alarm notifications as a result.

Note:

After you have successfully created custom callback APIs and set callback addresses, CLS will automatically send requests to these APIs as configured.

Custom callback APIs require access over the public network, as CLS cannot call back private network addresses.

Directions

Generating custom callback links for target services

Generate custom callback links (webhooks) for custom services requiring callbacks (such as DingTalk and Slack) to receive alarm notifications.

Configuring custom API callbacks (custom webhooks)

On the **Notification Channel** page in the CLS console, enter the custom callback link address, and set the custom request content as required. For more information on the configuration, please see Adding Notification Channel Groups.

After custom API callbacks are configured, when alarm policies are triggered, CLS will call the custom APIs according to the configured request formats.

Managing Notification Groups

Last updated：2024-01-20 17:59:36

Overview

This document introduces how to manage notification groups.

Directions

Adding notification groups

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Notification Group** to enter the notification group management page.
3. Click **Create** and set the following parameters as required on the notification group creation page:
Basic information
Name: custom notification group name.
Notification Type:
Alarm triggered: when the monitoring result meets the alarm trigger expression, an alarm triggered notification will be sent.
Alarm cleared: when the trigger condition is met in the previous monitoring period but not met in the current period and the alarm triggered notification has been sent, an alarm cleared notification will be sent.
Method: Configure user notification and enter the receipt information.
Type: Select a type as needed.
Recipient: specified users or user groups (a user group contains multiple users).
Notification Period: Define the time period for receiving alarms.
Notify By: email, SMS, Weixin, or phone calls.
Webhook URL: Configure the API callback and enter the callback address.

Note:

Up to 10 custom and WeCom webhooks are supported.
Webhook - WeCom robot
After creating a WeCom bot, just enter the URL of the webhook.
Webhook - custom
Enter a custom webhook address to further process and forward alarm notifications received. You can customize the request content. CLS alarm-related information can be referenced by variables, and when the alarm callback is

triggered, it will be replaced with the corresponding content in the current alarm policy. For more information, please see Custom Callback APIs.

Copying a notification group

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Notification Group** to enter the notification group management page.
3. Select the target notification group and click **Copy**.

Modifying a notification group

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Notification Group** to enter the notification group management page.
3. Select the target notification group and click **Edit**.

Deleting a notification group

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Notification Group** to enter the notification group management page.
3. Select the target notification group and click **Delete**.

Alarm Notification Variable

Last updated：2024-01-20 17:59:36

When setting the **notification content** and **custom webhook configuration** in an **alarm policy**, you can use alarm notification variables to customize the notification content and include a clearer cause description.

Getting Started

When configuring an alarm policy, enter the following configuration information for the notification content:

```
Detailed log:
{{.QueryLog[0][0]}}
```

The configured information should be as follows:
When the alarm notification is received, the notification content will be automatically replaced with the following value, indicating the last detailed log when the alarm is triggered:

```
Detailed log:
{"content":{"body_bytes_sent":"33352","http_referer":"-
","http_user_agent":"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/30.0.1599.17
Safari/537.36","remote_addr":"201.80.83.199","remote_user":"-
","request_method":"GET","request_uri":"/content/themes/test-
com/images/header_about.jpg","status":"404","time_local":"01/Nov/2018:01:16:31"
},"fileName":"/root/testLog/nginx.log","pkg_id":"285A243662909DE3-
70A","source":"172.17.0.2","time":1653831150008,"topicId":"a54de372-ffe0-49ae-
a12e-c340bb2b03f2"}
```

Notification Variables

Variable	Configuration	Sample Variable Value	Description
{{.UIN}}	Account ID	100007xxx827	-
{{.Nickname}}	Account nickname	xx company	-
{{.Region}}	Region	Guangzhou	-
{{.Alarm}}	Alarm policy	Too many NGINX error logs	-

	name		
{{.AlarmID}}	Alarm policy ID	notice-3abd7ad6-15b7-4168-xxxx-52e5b961a561	-
{{.ExecuteQuery}}	Executed Statement	["status:>=400 | select count(*) as errorLogCount","status:>=400 | select count(*) as errorLogCount,request_uri group by request_uri order by count(*) desc"]	It is an array. {{.ExecuteQuery[0]}} indicates the detailed log of the first query statement, {{.ExecuteQuery[1]}} the second, and so on.
{{.Condition}}	Trigger Condition	$1.errorLogCount > 1	-
{{.HappenThreshold}}	Number of times the trigger condition needs to be constantly met before an alarm is triggered	1	-
{{.AlertThreshold}}	Alarm interval	15	Unit: Minute
{{.Topic}}	Log topic name	nginxLog	-
{{.TopicId}}	Log topic ID	a54de372-ffe0-49ae-xxxx-c340bb2b03f2	-
{{.StartTime}}	Time when the alarm is triggered for the first time	2022-05-28 18:56:37	Time zone: Asia/Shanghai
{{.StartTimeUnix}}	Timestamp when the alarm is triggered for the first time	1653735397099	UNIX timestamp in milliseconds

{{.NotifyTime}}	Time of this alarm notification	2022-05-28 19:41:37	Time zone: Asia/Shanghai
{{.NotifyTimeUnix}}	Timestamp of this alarm notification	1653738097099	UNIX timestamp in milliseconds
{{.NotifyType}}	Alarm notification type	1	Valid values: `1` (alarmed), `2` (resolved)
{{.ConsecutiveAlertNums}}	Number of consecutive alarms	2	-
{{.Duration}}	Alarm duration	0	Unit: Minute
{{.TriggerParams}}	Alarm trigger parameter	$1.errorLogCount=5;	-
{{.ConditionGroup}}	Group information when the alarm is triggered	{"$1.AppName":"userManageService"}	This is valid only when triggering by group is enabled in the alarm policy.
{{.DetailUrl}}	URL of the alarm details page	https://alarm.cls.tencentcs.com/MDv2xxJh	No login is required
{{.QueryUrl}}	URL of the search and analysis statement in the first query statement	https://alarm.cls.tencentcs.com/T0pkxxMA	-
{{.Message}}	Notification content	-	It indicates the **notification content** entered ir the alarm policy.
{{.QueryResult}}	Execution result of the	For more information, see the {{.QueryResult}} remarks.	-

	query statement		
{{.QueryLog}}	Detailed log matching the search condition of the query statement	For more information, see the {{.QueryLog}} remarks.	-
{{.AnalysisResult}}	Multi-dimensional analysis result	For more information, see the {{.AnalysisResult}} remarks.	This variable is vali only when an alarm is triggered and becomes invalid when the alarm is cleared.

Below are relevant descriptions:

{{.QueryResult}}

Description: Execution result of the query statement

Remarks: It is an array. `{{.QueryResult[0]}}` indicates the execution result of the first query statement, `{{.QueryResult[1]}}` the second, and so on.

Sample variable value:

If there are two query statements in the alarm policy:

```
The first query statement: status:>=400 | select count(*) as errorLogCount
The second query statement: status:>=400 | select count(*) as errorLogCount,request
```

Then the variable values are:

```
[
  [{
      "errorLogCount": 7
  }],
  [{
      "errorLogCount": 3,
      "request_uri": "/apple-touch-icon-144x144.png"
  }, {
      "errorLogCount": 3,
      "request_uri": "/feed"
  }, {
      "errorLogCount": 1,
```

```
      "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html"
  }]
]
```

{{.QueryLog}}

Description: Detailed log matching the search condition of the query statement (excluding SQL filter condition)

Remarks: It is an array. `{{.QueryLog[0]}}` indicates the detailed log of the first query statement, `{{.QueryLog[1]}}` the second, and so on. Up to last ten detailed logs can be contained in each query statement.

Sample variable value:

```
[
  [{
      "content": {
          "__TAG__": {
              "pod": "nginxPod",
              "cluster": "testCluster"
          },
          "body_bytes_sent": "32847",
          "http_referer": "-",
          "http_user_agent": "Opera/9.80 (Windows NT 6.1; U; en-US)
Presto/2.7.62 Version/11.01",
          "remote_addr": "105.86.148.186",
          "remote_user": "-",
          "request_method": "GET",
          "request_uri": "/apple-touch-icon-144x144.png",
          "status": "404",
          "time_local": "01/Nov/2018:00:55:14"
      },
      "fileName": "/root/testLog/nginx.log",
      "pkg_id": "285A243662909DE3-5CD",
      "source": "172.17.0.2",
      "time": 1653739000013,
      "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
  }, {
      "content": {
          "__TAG__": {
              "pod": "nginxPod",
              "cluster": "testCluster"
          },
          "body_bytes_sent": "33496",
          "http_referer": "-",
          "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36",
          "remote_addr": "222.18.168.242",
```

```
          "remote_user": "-",
          "request_method": "GET",
          "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html",
          "status": "404",
          "time_local": "01/Nov/2018:00:54:37"
      },
      "fileName": "/root/testLog/nginx.log",
      "pkg_id": "285A243662909DE3-5C8",
      "source": "172.17.0.2",
      "time": 1653738975008,
      "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
  }]
]
```

{{.AnalysisResult}}

Description: Multi-dimensional analysis result

Remarks: It is an object. The level-1 object corresponds to each multi-dimensional analysis result, with the `key` being the multi-dimensional analysis name and the `value` being the multi-dimensional analysis result. This variable is valid only when an alarm is triggered (that is, {{.NotifyType}}=1) and becomes invalid when the alarm is cleared (that is, {{.NotifyType}}=2).

Sample variable value:

If there are three multi-dimensional analysis operations in the alarm policy:

```
Name: Top URL
Type: Top 5 field values by occurrence and their percentages
Field: request_uri

Name: Error log URL distribution
Type: Custom search and analysis
Analysis statement: status:>=400 | select count(*) as errorLogCount,request_uri gro

Name: Detailed error log
Type: Custom search and analysis
Analysis statement: status:>=400
```

Then the variable values are:

```
{
  "Top URL": [{
      "count": 77,
      "ratio": 0.45294117647058824,
      "value": "/"
  }, {
      "count": 20,
```

```
        "ratio": 0.11764705882352941,
        "value": "/favicon.ico"
    }, {
        "count": 7,
        "ratio": 0.041176470588235294,
        "value": "/blog/feed"
    }, {
        "count": 5,
        "ratio": 0.029411764705882353,
        "value": "/test-tile-service"
    }, {
        "count": 3,
        "ratio": 0.01764705882352941,
        "value": "/android-chrome-192x192.png"
    }],
    "Detailed error log": [{
        "content": {
            "__TAG__": {
                "pod": "nginxPod",
                "cluster": "testCluster"
            },
            "body_bytes_sent": "32847",
            "http_referer": "-",
            "http_user_agent": "Opera/9.80 (Windows NT 6.1; U; en-US)
Presto/2.7.62 Version/11.01",
            "remote_addr": "105.86.148.186",
            "remote_user": "-",
            "request_method": "GET",
            "request_uri": "/apple-touch-icon-144x144.png",
            "status": "404",
            "time_local": "01/Nov/2018:00:55:14"
        },
        "fileName": "/root/testLog/nginx.log",
        "pkg_id": "285A243662909DE3-5CD",
        "source": "172.17.0.2",
        "time": 1653739000013,
        "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
    }, {
        "content": {
            "__TAG__": {
                "pod": "nginxPod",
                "cluster": "testCluster"
            },
            "body_bytes_sent": "33496",
            "http_referer": "-",
            "http_user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36",
```

```
            "remote_addr": "222.18.168.242",
            "remote_user": "-",
            "request_method": "GET",
            "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html",
            "status": "404",
            "time_local": "01/Nov/2018:00:54:37"
        },
        "fileName": "/root/testLog/nginx.log",
        "pkg_id": "285A243662909DE3-5C8",
        "source": "172.17.0.2",
        "time": 1653738975008,
        "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
    }],
    "Error log URL distribution": [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html"
    }]
}
```


Variable Syntax

The variable syntax is similar to Go template syntax. It extracts and formats alarm notification variables so that they can be displayed more clearly in the alarm notification content. All variables and their syntaxes are within `{{ }}`, and text outside `{{ }}` won't be processed.

Variable extraction

Syntax format:

```
{{.variable[x]}} or {{index .variable x}}
{{.variable.childNodeName}} or {{index .variable "childNodeName"}}
```

Syntax description:

When the variable is an array, `{{.variable[x]}}` (equivalent to `{{index .variable x}}`) is used to extract array elements by subscript. Here, `x` is an integer greater than or equal to 0.

When the variable is an object, `{{.variable.childNodeKey}}` (equivalent to `{{index .variable "childNodeName"}}`) is used to extract sub-object values (`value`) by sub-object name (`key`).

Note:

When a sub-object name contains spaces, use syntax in the format of `{{index .variable "childNodeName"}}`, such as `{{index .AnalysisResult "Top URL"}}`.

Sample:

The `{{.QueryResult}}` variable values are:

```
[
    [{
        "errorLogCount": 7 // Extract the value
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html"
    }]
]
```

Get the `errorLogCount` value of the first array through the following expression:

```
{{.QueryResult[0][0].errorLogCount}}
```

Returned result:

```
7
```

Loop and traversal

Syntax format:

```
{{range .variable}}
Custom content{{.childNode1}}custom content{{.childNode2}}...
{{end}}
```

Or

```
{{range $key,$value := .variable}}
Custom content{{$key}}custom content{{$value}}...
```

```
{{end}}
```

Syntax description:

When the variable is an array or an object that contains multiple sub-objects, you can use this syntax to display each element/object in the specified format.

Sample:

The `{{.QueryResult}}` variable values are:

```
[
    [{
        "errorLogCount": 7
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html"
    }]
]
```

Display the `errorLogCount` value of each `request_uri` in the second array through the following expression:

```
{{range .QueryResult[1]}}
* {{.request_uri}} error log quantity: {{.errorLogCount}}
{{end}}
```

Returned result:

```
* /apple-touch-icon-144x144.png error log quantity: 3

* /feed error log quantity: 3

* /opt/node_apps/test-v5/app/themes/basic/public/static/404.html error log
quantity: 1
```

Condition judgment

Syntax format:

```
{{if boolen}}
xxx
{{end}}
```

Or

```
{{if boolen}}
xxx
{{else}}
xxx
{{end}}
```

Or

```
{{if boolen}}
xxx
{{else if boolen}}
xxx
{{end}}
```

Syntax description:

Execute the expressions based on the condition judgment result, where AND, OR, and NOT can be used for logical operations and values can be compared.

```
eq arg1 arg2: When arg1 == arg2, the value is `true`.
ne arg1 arg2: When arg1 != arg2, the value is `true`.
lt arg1 arg2: When arg1 < arg2, the value is `true`.
le arg1 arg2: When arg1 <= arg2, the value is `true`.
gt arg1 arg2: When arg1 > arg2, the value is `true`.
ge arg1 arg2: When arg1 >= arg2, the value is `true`.
```

Sample:

The `{{.QueryResult}}` variable values are:

```
[
    [{
        "errorLogCount": 7
    }],
    [{
        "errorLogCount": 3,
        "request_uri": "/apple-touch-icon-144x144.png"
    }, {
        "errorLogCount": 3,
        "request_uri": "/feed"
    }, {
```

```
        "errorLogCount": 1,
        "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html"
    }]
]
```

Display the `request_uri` that is ≥ 2 and ≤ 100 and its `errorLogCount` value in the second array through the following expression:

```
{{range .QueryResult[1]}}
{{if and (ge .errorLogCount 2) (le .errorLogCount 100)}}
* {{.request_uri}} error log quantity: {{.errorLogCount}}
{{end}}
{{end}}
```

Returned result:

```
* /apple-touch-icon-144x144.png error log quantity: 3

* /feed error log quantity: 3
```

You can also use `if` to check whether the field value exists. If the field value is an empty string or does not exist, it is equivalent to `false`. For example:

```
{{if .QueryLog[0][0].apple}}
apple exist, value is : {{.QueryLog[0][0].apple}}
{{else}}
apple is not exist
{{end}}
```

Blank area removal

Syntax format:

```
{{- xxx}} or {{xxx -}}
```

Syntax description:

When the variable syntax is executed, its spaces, indents, line breaks, and other blank areas will be retained. For example, many blank lines are contained in the returned result of the samples for loop and traversal as well as condition judgment, adversely affecting the display effect. You can add `-` at the beginning or end in `{{ }}` to remove blank areas.

Sample:

Change the expression in the condition judgment sample to:

```
{{- range .QueryResult[1]}}
{{- if and (ge .errorLogCount 2) (le .errorLogCount 100)}}
* {{.request_uri}} error log quantity: {{.errorLogCount}}
{{- end}}
{{- end}}
```

Returned result:

```
* /apple-touch-icon-144x144.png error log quantity: 3
* /feed error log quantity: 3
```

Variable Function

Special symbol escaping

Syntax format:

```
{{escape .variable}}
```

Syntax description:

Many alarm variables contain special symbols. If these variables are directly concatenated into JSON strings for custom webhooks, the JSON format may be incorrect, leading to callback failures. In this case, you can escape the original variables before they are concatenated.

Sample:

The `{{.ExecuteQuery[0]}}` variable value is `status:>=400 | select count(*) as "error log quantity"` .

If escaping is not used, the request content in the custom webhook configuration will be:

```
{
  "Query":"{{.ExecuteQuery[0]}}"
}
```

The returned result will be as follows, which is not a valid JSON string:

```
```

```
{
   "Query":"status:>=400 | select count(*) as "error log quantity""
}
```

In this case, you can use escaping to change the request content in the custom webhook configuration to:

```
{
  "Query":"{{escape .ExecuteQuery[0]}}"
}
```

The returned result will be as follows, which is in line with the JSON syntax:

```
{
  "Query":"status:>=400 | select count(*) as \\"error log quantity\\""
}
```

String extraction

Extraction by length

Syntax format:

```
{{substr .variable start}} or {{substr .variable start length}}
```

Syntax description:

Extract a string based on the specified start point and length (optional).

Sample:

The `{{.QueryLog[0][0].fileName}}` variable value is:

```
/root/testLog/nginx.log
```

Get a string starting from the sixth character and containing seven characters through the following expression:

```
{{substr .QueryLog[0][0].fileName 6 7 }}
```

Returned result:

```
testLog
```

Extraction based on the start and end characters

Syntax format:

```
{{extract .variable "startstring" ["endstring"]}}
```

Syntax description:

Extract a string based on the specified start and end characters (optional).

Sample:

The `{{.QueryLog[0][0].fileName}}` variable value is:

```
/root/testLog/nginx.log
```

Get a string between `/root/` and `/nginx` through the following expression:

```
{{extract .QueryLog[0][0].fileName "/root/" "/nginx"}}
```

Returned result:

```
testLog
```

Specified string check

Syntax format:

```
{{containstr .variable "searchstring"}}
```

Syntax description:

Check whether the specified string is included in the variable value. The check result can be used in the condition judgment syntax.

Sample:

The `{{.QueryLog[0][0].fileName}}` variable value is:

```
/root/testLog/nginx.log
```

Get a string between `/root/` and `/nginx` through the following expression:

```
{{if containstr .QueryLog[0][0].fileName "test"}}
Test log
{{else}}
Non-test log
{{end}}
```

Returned result:

```
Test log
```

UNIX timestamp conversion

Syntax format:

```
{{fromUnixTime .variable}} or {{fromUnixTime .variable "timezone"}}
```

Syntax description:

It is used to convert UNIX timestamps (in milliseconds or seconds) into readable dates and times. Here, the time zone is optional and is Asia/Shanghai by default.

Sample:

The `{{.QueryLog[0][0].time}}` variable value is:

```
1653893435008
```

Get the dates and times in different time zones through the following expressions:

```
{{fromUnixTime .QueryLog[0][0].time}}
{{fromUnixTime .QueryLog[0][0].time "Asia/Shanghai"}}
{{fromUnixTime .QueryLog[0][0].time "Asia/Tokyo"}}
```

Returned result:

```
2022-05-30 14:50:35.008 +0800 CST
2022-05-30 14:50:35.008 +0800 CST
2022-05-30 15:50:35.008 +0900 JST
```

String concatenation

Syntax format:

```
{{concat .variable1 .variable2 ...}}
```

Syntax description:

Concatenate specified variables or strings.

Sample:

Concatenate the region and alarm policy name.

```
{{concat .Region .Alarm}}
```

Returned result:

```
Guangzhou alarmTest
```

Base64/Base64URL/URL encoding and decoding

Syntax format:

```
{{base64_encode .variable}}
{{base64_decode .variable}}
{{base64url_encode .variable}}
{{base64url_decode .variable}}
{{url_encode .variable}}
{{url_decode .variable}}
```

Syntax description:

Encode or decode specified variables or strings. Here, the *"="* at the end will not be removed or added during Base64URL encoding and decoding.

Sample:

```
{{base64_encode "test"}}
{{base64_decode "dGVzdOa1i+ivlQ=="}}
{{base64url_encode "test"}}
{{base64url_decode "dGVzdOa1i-ivlQ=="}}
{{url_encode "https://console.intl.cloud.tencent.com:80/cls?region=ap-
chongqing"}}
{{url_decode "https%3A%2F%2Fconsole.cloud.tencent.com%3A80%2Fcls%3Fregion%3Dap-
chongqing"}}
```

Returned result:

```
dGVzdOa1i+ivlQ==
test
dGVzdOa1i-ivlQ==
test
https%3A%2F%2Fconsole.cloud.tencent.com%3A80%2Fcls%3Fregion%3Dap-chongqing
https://console.intl.cloud.tencent.com:80/cls?region=ap-chongqing
```

MD5/SHA1/SHA256/SHA512 encryption

Syntax format:

```
{{md5 .variable}}
{{md5 .variable | base64_encode}}
{{md5 .variable | base64url_encode}}
{{sha1 .variable}}
{{sha1 .variable | base64_encode}}
{{sha1 .variable | base64url_encode}}
{{sha256 .variable}}
{{sha256 .variable | base64_encode}}
{{sha256 .variable | base64url_encode}}
{{sha512 .variable}}
{{sha512 .variable | base64_encode}}
{{sha512 .variable | base64url_encode}}
```

Syntax description:

Encrypt specified variables or strings using a certain encryption algorithm. By default, hexadecimal strings are returned, which can be converted into Base64 or Base64URL as needed.

Sample:

```
{{md5 "test"}}
{{md5 "test" | base64_encode}}
{{md5 "test" | base64url_encode}}
{{sha1 "test"}}
{{sha1 "test" | base64_encode}}
{{sha1 "test" | base64url_encode}}
{{sha256 "test"}}
{{sha256 "test" | base64_encode}}
{{sha256 "test" | base64url_encode}}
{{sha512 "test"}}
{{sha512 "test" | base64_encode}}
{{sha512 "test" | base64url_encode}}
```

Returned result:

```
098F6BCD4621D373CADE4E832627B4F6
CY9rzUYh03PK3k6DJie09g==
CY9rzUYh03PK3k6DJie09g==
A94A8FE5CCB19BA61C4C0873D391E987982FBBD3
qUqP5cyxm6YcTAhz05Hph5gvu9M=
qUqP5cyxm6YcTAhz05Hph5gvu9M=
9F86D081884C7D659A2FEAA0C55AD015A3BF4F1B2B0B822CD15D6C15B0F00A08
n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg=
n4bQgYhMfWWaL-qgxVrQFaO_TxsrC4Is0V1sFbDwCgg=
EE26B0DD4AF7E749AA1A8EE3C10AE9923F618980772E473F8819A5D4940E0DB27AC185F8A0E1D5F84F8
7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1nsUNzLDBMxfqa2Ob1f1ACi
7iaw3Ur350mqGo7jwQrpkj9hiYB3Lkc_iBml1JQODbJ6wYX4oOHV-E-IvIh_1nsUNzLDBMxfqa2Ob1f1ACi
```

HMAC_MD5/HMAC_SHA1/HMAC_SHA256/HMAC_SHA512 encryption

Syntax format:

```
{{hmac_md5 .variable "Secretkey"}}
{{hmac_md5 .variable "Secretkey" | base64_encode}}
{{hmac_md5 .variable "Secretkey" | base64url_encode}}
{{hmac_sha1 .variable "Secretkey"}}
{{hmac_sha1 .variable "Secretkey" | base64_encode}}
{{hmac_sha1 .variable "Secretkey" | base64url_encode}}
{{hmac_sha256 .variable "Secretkey"}}
{{hmac_sha256 .variable "Secretkey" | base64_encode}}
```

```
{{hmac_sha256 .variable "Secretkey" | base64url_encode}}
{{hmac_sha512 .variable "Secretkey"}}
{{hmac_sha512 .variable "Secretkey" | base64_encode}}
{{hmac_sha512 .variable "Secretkey" | base64url_encode}}
```

Syntax description:

Encrypt specified variables or strings using a certain encryption algorithm. By default, hexadecimal strings are returned, which can be converted into Base64 or Base64URL as needed. Here, `Secretkey` is the key in the HMAC encryption algorithm and can be modified as needed.

Sample:

```
{{hmac_md5 "test" "Secretkey"}}
{{hmac_md5 "test" "Secretkey" | base64_encode}}
{{hmac_md5 "test" "Secretkey" | base64url_encode}}
{{hmac_sha1 "test" "Secretkey"}}
{{hmac_sha1 "test" "Secretkey" | base64_encode}}
{{hmac_sha1 "test" "Secretkey" | base64url_encode}}
{{hmac_sha256 "test" "Secretkey"}}
{{hmac_sha256 "test" "Secretkey" | base64_encode}}
{{hmac_sha256 "test" "Secretkey" | base64url_encode}}
{{hmac_sha512 "test" "Secretkey"}}
{{hmac_sha512 "test" "Secretkey" | base64_encode}}
{{hmac_sha512 "test" "Secretkey" | base64url_encode}}
```

Returned result:

```
E7B946D930658699AA668601E33E87CE
57lG2TBlhpmqZoYB4z6Hzg==
57lG2TBlhpmqZoYB4z6Hzg==
2AB64F124D932F5033EAC7AF392AC5CC4D52F503
KrZPEk2TL1Az6sevOSrFzE1S9QM=
KrZPEk2TL1Az6sevOSrFzE1S9QM=
FC49EBC05209B1359773D87C216BA85BCE0163FDE459EA37AB603EC9D8445D23
/EnrwFIJsTWXc9h8IWuoW84BY/3kWeo3q2A+ydhEXSM=
_EnrwFIJsTWXc9h8IWuoW84BY_3kWeo3q2A-ydhEXSM=
D18DF3D943F74769A8B66E43D7EF03639BB6B8B8A2EBC9976170DC58EEE58BE98478F3183E4B5AA3481
0Y3z2UP3R2motm5D1+8DY5u2uLii68mXYXDcWO7li+mEePMYPktao0gd4SAmquOEP4ITs51jnqxu6Tc06mZ
0Y3z2UP3R2motm5D1-8DY5u2uLii68mXYXDcWO7li-mEePMYPktao0gd4SAmquOEP4ITs51jnqxu6Tc06mZ
```

Use Cases

Case 1: Displaying the last detailed log in the alarm notification

Scenario:

The last detailed log that meets the search condition of the query statement is added to the alarm notification in the format of `key:value`. There is a key in each row, and CLS preset fields and metadata fields are not included.

Notification content configuration:

```
{{range $key,$value := .QueryLog[0][0].content}}
{{if not (containstr $key "__TAG__")}}
{{- $key}}:{{$value}}
{{- end}}
{{- end}}
```

Here, `.QueryLog[0][0]` indicates the last detailed log that meets the search condition of the first query statement in the alarm policy. Its value is:

```
{
    "content": {
        "__TAG__": {
            "a": "b12fgfe",
            "c": "fgerhcdhgj"
        },
        "body_bytes_sent": "33704",
        "http_referer": "-",
        "http_user_agent": "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/35.0.3319.102 Safari/537.36",
        "remote_addr": "247.0.249.191",
        "remote_user": "-",
        "request_method": "GET",
        "request_uri": "/products/hadoop)",
        "status": "404",
        "time_local": "01/Nov/2018:07:54:08"
    },
    "fileName": "/root/testLog/nginx.log",
    "pkg_id": "285A243662909DE3-210B",
    "source": "172.17.0.2",
    "time": 1653908859008,
    "topicId": "a54de372-ffe0-49ae-a12e-c340bb2b03f2"
}
```

Alarm notification content:

```
remote_addr:247.0.249.191
time_local:01/Nov/2018:07:54:08
http_user_agent:Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/35.0.3319.102 Safari/537.36
remote_user:-
http_referer:-
```

```
body_bytes_sent:33704
request_method:GET
request_uri:/products/hadoop)
status:404
```

Case 2: Displaying the execution result of the query statement in the alarm notification

Scenario:

What meets the trigger condition in the execution result of the query statement is added to the alarm notification and displayed in a list.

The query statement of the alarm policy is `status:>=400 | select count(*) as errorLogCount,request_uri group by request_uri order by count(*) desc`.

The trigger condition is `$1.errorLogCount > 10`.

Notification content configuration:

```
{{range .QueryResult[0]}}
{{- if gt .errorLogCount 10}}
{{.request_uri}} error log quantity: {{.errorLogCount}}
{{- end}}
{{- end}}
```

Here, `.QueryResult[0]` indicates the execution result of the first query statement in the alarm policy. Its value is:

```
[{
    "errorLogCount": 161,
    "request_uri": "/apple-touch-icon-144x144.png"
}, {
    "errorLogCount": 86,
    "request_uri": "/opt/node_apps/test-
v5/app/themes/basic/public/static/404.html"
}, {
    "errorLogCount": 33,
    "request_uri": "/feed"
}, {
    "errorLogCount": 26,
    "request_uri": "/wp-login.php"
}, {
    "errorLogCount": 10,
    "request_uri": "/safari-pinned-tab.svg"
}, {
    "errorLogCount": 7,
    "request_uri": "/mstile-144x144.png"
}, {
    "errorLogCount": 4,
    "request_uri": "/atom.xml"
```

```
}, {
    "errorLogCount": 3,
    "request_uri": "/content/plugins/prettify-gc-syntax-highlighter/launch.js?
ver=3.5.2?ver=3.5.2"
}]
```

Alarm notification content:

```
/apple-touch-icon-144x144.png error log quantity: 161
/opt/node_apps/elastic-v5/app/themes/basic/public/static/404.html error log
quantity: 86
/feed error log quantity: 33
/wp-login.php error log quantity: 26
```

Viewing Alarm Records

Last updated：2024-01-20 17:59:35

This document shows how to view alarm records in the CLS console.

Directions

1. Log in to the CLS console.
2. On the left sidebar, click **Monitoring Alarm** > **Alarm Records** to enter the alarm records page.

Notes

CLS allows you to view the alarm records in the last 30 days.

Alarm statistics

Alarm Statistics displays important information on alarms in the current region, such as alarm policy statistics and monitoring task execution. The metrics are as detailed below:

Metric	Description
Total Alarm Policy Executions	Number of alarm policies executed over the statistical time range
Alarm Policy Executions	Number of times the query and analysis statement in the alarm policy is executed over the statistical time range
Failed Alarm Policy Executions	Number of alarm policy execution failures over the statistical time range. Execution failures include AlarmConfigNotFound, QuerySyntaxError, QueryError, QueryResultParseError, ConditionSyntaxError, ConditionEvaluateError, and ConditionValueTypeError. For more information, please see Execution Result Status Codes
Times of Trigger Conditions Met	Number of times the query and analysis statement in the alarm policy is executed successfully and the result returned meets the trigger condition over the statistical time range
Notifications Triggered by the	Number of times notifications are triggered by the execution of the alarm policy over the statistical time range

Alarm Policy	
Top 10 Alarm Policies by Number of Notifications	Top 10 alarm policies in terms of the number of times notifications are triggered over the statistical time range

Historical records

Once an alarm policy takes effect, it will periodically perform monitoring tasks, and the details of executions of each monitoring task will be recorded in **Historical Records**, including the result of each execution. By viewing the records of alarm policy execution, you can easily trace back historical alarming tasks.

Note:

CLS allows you to view the records in the last 30 days.

Policy execution results are described as follows.

Execution Result	Description
AlarmConfigNotFound	The alarm policy configuration is missing. Please check whether the alarm policy and monitoring object have been configured correctly.
QuerySyntaxError	The analysis statement of the monitoring object has a syntax error. Please check whether the statement is correct. For more information on the syntax, please see Overview.
QueryError	The analysis statement is not executed properly. Please check the analysis statement and the index configuration of the log topic.
QueryResultParseError	Failed to parse the analysis result format.
ConditionSyntaxError	The trigger condition expression has a syntax error. Please check the syntax format of the expression.
ConditionEvaluateError	An error occurred while computing the trigger condition. Please check whether the imported variable exists in the analysis result
ConditionValueTypeError	The evaluation result of the trigger condition is not a Boolean value. Please check whether the trigger condition expression is correct.
EvalTimesLimited	The trigger condition hasn't been met even after it has been computed more than 1,000 times.

QueryResultUnmatch	The analysis result for the current monitoring period doesn't meet the alarm trigger condition.
UnreachedThreshold	The alarm trigger condition is met, but the alarm convergence threshold has not been reached, so no alarm notification is sent. HappenThreshold Unreached: the period convergence condition is not met; for example, an alarm is triggered only if the trigger condition is met in 5 consecutive monitoring periods. AlertThreshold Unreached: the alarm interval condition is not met; for example, an alarm will be triggered once every 15 minutes.
TemplateUnmatched	The alarm configuration information doesn't match the notification template. Specific causes include: TypeUnmatched: the alarm notification type (alarm triggered or alarm cleared) doesn't match the notification template, so no alarm notification is sent. TimeUnmatched: the alarm notification time period doesn't match the notification template, so no alarm notification is sent. SendFail: the notification failed to be sent.
Matched	The alarm condition is met, and the alarm notification is sent successfully.

Policy alarm states are described as follows.

Alarm State	Description
Uncleared	The system continuously meets the trigger condition and triggers an alarm.
Cleared	The current monitoring period does not meet the trigger condition.
Invalid	The alarm policy is deleted or modified.

Alarm Silence

Last updated：2024-06-06 17:05:09

Overview

When an exception occurs in the system, there will be alarms continuously. For some problems that are under processing, the alarms can be silenced for some time to avoid disturbance. In addition, scheduled system changes may also generate alarms as expected and this type of alarms can be silenced in advance.
Cloud Log Service provides two methods for Alarm Silence:
Temporary Alarm Silence: When an alarm notification is received, the alarm can be directly silenced for a period of time (15 minutes to 12 hours). The operation is convenient and suitable for temporarily silencing alarms that are already under processing.
Scheduled Alarm Silence: Create alarm silencing rules in advance on the console, which can be operated in batches. It is suitable for silencing some expected alarms in advance before system changes.
Alarm Silence rules belong to Management Notification Groups. When multiple alarm policies use the same Notification Channel Group, you can silence multiple alarm policies in batches in the Notification Channel Group.

Directions

Temporary Alarm Silence

Note:
This feature does not support phone calls and WeChat.
You need to enable **Block alarms without login** feature in Management Notification Groups, which allows you to silence subsequent notifications of related alarms directly in the alarm notification without logging in to the console.
1. Receive alarm notifications through Short Message Service, email, WeCom, DingTalk and other channels.
2. Click **Alarm Silence** or **Detailed Report** (and click **Alarm Silence** at the bottom) in the notification to enter the configuration page of Alarm Silence, and view or fill in the following information:
Notification Channel Group: The notification channel group used by the current alarm policy. There may be multiple notification channel groups and they cannot be modified.
Range of Silencing Time: It supports 15 minutes, 30 minutes, 1 hour, 6 hours, 1 day. You can only select one option.
Silencing Rule: Automatically generated, only silencing the alarm policy, alarm severity, monitoring object and alarm group (need to open Trigger by Group feature) corresponding to the current alarm. The rules cannot be modified.
Operator: Name of the operator, required.
Reason for Silencing: The reason for silencing the alarm, required.

Scheduled Alarm Silence

1. Log in to the Cloud Log Service Console.
2. In the left sidebar, click **Monitoring Alarm > Notification Group** to enter the management page of Notification Channel Group.
3. In the list, click **Notification Group Name/ID** for which you want to configure silencing rules to enter the details page of Notification Channel Group.
4. Click **Alarm Blocking Rule** tag at the top tab to enter the management page of alarm silence rules.
5. Click **Add Rule** to fill in the following information in the pop-up dialog box:

Blocking Period: the period for which you want to silence the alarms.

Blocking Rule:

All notifications: Silencing all alarms using this Notification Channel Group.

Notifications satisfying the following rules: Only silence alarms that meet custom rules, such as silencing all alarms of Reminder level. For details on supported rules, see Rule Details .

Blocking Reason: The reason for silencing the alarm, required.

Viewing Created Silencing Rules

Regardless of the method used to create the silencing rules, you can view and manage the rules in the console.

1. Log in to the Cloud Log Service Console.
2. In the left sidebar, click **Monitoring Alarm > Notification Group** to enter the management page of Notification Channel Group.
3. In the list, click **Notification Group Name/ID** for which you want to configure silencing rules to enter the details page of Notification Channel Group.
4. Click **Alarm Blocking Rule** tag at the top tab to enter the management page of alarm silence rules.
5. Rules of different status support the following operations:

Activating: The rule supports copying and expiration. After clicking Expire, the rule will no longer be effective.

Expired: The rule supports copying and viewing, but not manual deletion. The system can save up to 1,000 expired silencing rules, and automatically delete old rules after exceeding this limit.

Not activated yet: The rule supports editing, copying, and deleting.

Rule Details

When creating a silencing rule through the console or API, you can define the silencing rule according to the following fields and comparison methods:

Field	Comparison Method	Comparison Value	Example of API Parameter (Rule)
Alarm Severity	It belongs	Reminder `1`	Example meaning: The alarm severity is not reminder.

`Level`	to `In` . It doesn't belong to `NotIn` .	Warning `0` Emergency `2` (Multiple values supported)	<pre>{
			"Type": "Condition",
			"Value": "Level",
			"Children": [{
			"Value": "NotIn",
			"Type": "Compare"
			}, {
			"Value": "[1]",
			"Type": "Value"
			}]
			}</pre>

Alarm Policy `AlarmID`	It belongs to `In` . It doesn't belong to `NotIn` .	Alarm policy (Multiple policies supported)	Example meaning: Alarm Policy belongs to `alarm-4`
			<pre>{
			"Type": "Condition",
			"Value": "AlarmID",
			"Children": [{
			"Value": "In",
			"Type": "Compare"
			}, {
			"Value": "[\\"alarm-4
			"Type": "Value"
			}]
			}</pre>

Alarm Policy Name `AlarmName`	Regular expression match `=~` Regular expression mismatch `! =~`	Alarm policy name (Regular expression)	Example meaning: The regular expression of alarm pol
			<pre>{
			"Type": "Condition",
			"Value": "AlarmName",
			"Children": [{
			"Value": "!=~",
			"Type": "Compare"
			}, {
			"Value": "test",
			"Type": "Value"
			}]
			}</pre>

Alarm Classification Field `Label` . (The specific classified field name	It belongs to `In` . It doesn't belong to	Classified field value (Multiple values	Example meaning: Alarm Classification Field `key1` b
			<pre>{
			"Type": "Condition",
			"Value": "Label",</pre>

is required to be specified.)	`NotIn` .	supported)	`"Children": [{ "Value": "key1", "Type": "Key" }, { "Value": "In", "Type": "Compare" }, { "Value": "[\\"value1\\" "Type": "Value" }] }`
	Regular expression match `=~` Regular expression mismatch `! =~`	Classified field value (Regular expression)	Example meaning: Alarm Classification Field `key2` r `{ "Type": "Condition", "Value": "Label", "Children": [{ "Value": "key2", "Type": "Key" }, { "Value": "=~", "Type": "Compare" }, { "Value": "value3", "Type": "Value" }] }`
Group Trigger Field `Group` (The specific group field name is required to be specified.)	It belongs to `In` . It doesn't belong to `NotIn` .	Classified field value (Multiple values supported)	Example meaning: Group Trigger Field `$1.key1` be `{ "Type": "Condition", "Value": "Group", "Children": [{ "Value": "$1.key1", "Type": "Key" }, { "Value": "In", "Type": "Compare" }, { "Value": "[\\"value1\\" "Type": "Value" }] }`

	Regular expression match `=~` Regular expression mismatch `! =~`	Classified field value (Regular expression)	Example meaning: Group Trigger Field `$1.key2` re `{ "Type": "Condition", "Value": "Group", "Children": [{ "Value": "$1.key2", "Type": "Key" }, { "Value": "!=~", "Type": "Compare" }, { "Value": "value3", "Type": "Value" }] }`
Monitor Object `MonitorObject` .	It belongs to `In` . It doesn't belong to `NotIn` .	Log topic and metric topic (Multiple topics supported)	Example meaning: The monitor object belongs to log to Note: BizType=0 represents the log topic; BizType=1 r `{ "Type": "Condition", "Value": "MonitorObject", "Children": [{ "Value": "In", "Type": "Compare" }, { "Value": "[{\\"TopicI "Type": "Value" }] }`

For example, to silence the alarms at the Reminder level in the demo alarm policy, the corresponding API parameter rule is:

```
{
        "Value": "AND", //Meet the following rules at the same time, and it must be
        "Type": "Operation",
        "Children": [{
                "Type": "Condition", //The first rule
                "Value": "Level", //Alarm severity
                "Children": [{
                        "Value": "In", //Belongs to
                        "Type": "Compare"
                }, {
```

```
                        "Value": "[1]", //Reminder
                        "Type": "Value"
                }]
        }, {
                "Type": "Condition", //The second rule
                "Value": "AlarmID", //Alarm policy
                "Children": [{
                        "Value": "In", //Belongs to
                        "Type": "Compare"
                }, {
                        "Value": "[\\"alarm-57105ec6-xxxx-xxxx-xxxx-892f3b8d143a\\"
                        "Type": "Value"
                }]
        }]
}
```

Cloud Insight

Overview of Cloud Insight

Last updated：2025-04-07 14:50:44

Cloud Insight is a unified cloud product observability platform built on Cloud Log Service (CLS). Through in-depth correlation analysis of multi-dimensional data such as cloud product logs, metrics, and usage, it provides comprehensive services including cloud product usage analysis, performance monitoring, security audit, data protection, anomaly detection, and access analysis. It helps you quickly build observability of cloud products, improve usage efficiency and security of cloud resources, and assist enterprises in achieving higher business value during digital transformation.

Features

Provide out-of-the-box Ops analysis dashboards for cloud products.
Unify the log retrieval and analysis entry for cloud products.
Enable cloud product log collection in batches.

Supported Cloud Products

The following cloud products support managing cloud product logs through Cloud Insight. For integrating other cloud products, see Collecting Cloud Product Logs.
CLB Cloud Insight
CDN Cloud Insight
COS Cloud Insight
SCF Cloud Insight
TKE Cloud Insight

CLB Cloud Insight

Last updated：2025-04-07 14:50:44

CLB Cloud Insight provides one-stop collection and retrieval analysis of Cloud Load Balancer (CLB) access logs. Meanwhile, based on access logs, it provides out-of-the-box access analysis dashboards. It helps you quickly build observability of CLB access analysis.

Features

Provide out-of-the-box CLB instance access analysis dashboards.
Unify the access log retrieval and analysis entry for CLB instances.
Enable CLB instance access log collection in batches.

Prerequisites

Creating a CLB instance is completed.

Directions

Enabling CLB Access Log Collection

Step 1: Entering CLB Cloud Insight

1. Log in to the Cloud Log Service console.
2. In the left navigation bar, select **Cloud Insight** to enter the Cloud Insight page.

3. In **Cloud Insight**, select and click **Cloud Load Balancer (CLB) Cloud Insight.**

Step 2: Enabling CLB Access Log Collection

1. In the **Cloud Load Balancer(CLB) Cloud Insight**, select **Collection Management** > **Instance Collection**.

2. Select the region where the target CLB instance is located. In the instance list, find and select one or more target CLB instances.

3. Click **Enable log collection**.

4. In the Enable Log Collection pop-up window, select or create a target Log Topic and click **Confirm** to start collecting access logs for the target CLB instance.

Note:

Only custom target storage log topics are supported. The CLB dedicated log set named cloud_cdn_logset is used as the default log set.

Viewing the CLB Access Analysis Dashboard

1. After enabling CLB access log collection, select **Dashboard** in the top tab bar.

2. In the CLB instance drop-down list, select the target CLB instance for which access log collection is enabled.

3. After selecting an instance, you can view the CLB access analysis dashboard related to the selected instance.

Retrieving and Analyzing CLB Access Logs

1. After enabling CLB access log collection, select **Log Search** in the top Tab bar.

2. In the CLB Instance drop-down list, select the target CLB instance for which access log collection is enabled.

3. After selecting an instance, you can retrieve and analyze the CLB access logs related to the selected instance.

Disabling CLB Access Log Collection

1. In the **Cloud Load Balancer (CLB) Cloud Insight**, select **Collection Management** > **Instance Collection**.

2. Select the instance for which you want to disable log collection in the CLB instance list and click **Disable Log Collection**.

Note:

Disabling log collection only represents that no new logs will be collected. The existing collected logs will continue to be stored in the log topic until expiration, and storage fees will continue to be incurred during this period. If you do not want to continue storing, choose **Collection Management** > **Storage Topic** and find the target topic and delete it.

3. In the Disable Collection pop-up window, select **Disable**, and then click **Confirm** to disable log collection for the selected CLB instance.

CDN Cloud Insight

Last updated：2025-04-07 14:50:44

CDN Cloud Insight provides one-stop collection, retrieval and analysis of Content Delivery Network (CDN) access logs. Meanwhile, based on access logs, it provides out-of-the-box access analysis dashboards. It helps quickly build observability of CDN access log analysis.

Features

Provide the out-of-the-box CDN domain name access analysis dashboard.
Unify the entry for CDN domain name access log retrieval and analysis.
Enable CDN domain name access log collection in batches.

Prerequisites

Creating a CDN domain name is completed.

Directions

Enabling Domain Name Access Log Collection

Step 1: Entering CDN Cloud Insight

1. Log in to the Cloud Log Service console.
2. Click **Cloud Insight** in the left navigation bar to go to the Cloud Insight page.

3. In **Cloud Product Logs**, select and click **Content Delivery Network (CDN) Cloud Insight**.

Step 2: Enabling Domain Name Access Log Collection

1. In **Content Delivery Network (CDN) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

2. In the CDN domain name list, find and select one or more target CDN domain names.

3. Click **Enable log collection**, and enable **Domestic Access Log** or **Overseas Access Log** in the dropdown options.

Note:
ECDN domain names and domain names in the Chinese mainland do not support enabling access log collection outside the Chinese mainland.
Domain names outside the Chinese mainland do not support enabling access log collection in the Chinese mainland.

4. In the Enable Log Collection pop-up window, select or create a target Log Topic and click **Confirm** to start collecting logs in or outside the Chinese mainland for the target CDN domain name.
Note:
Only custom target storage log topics are supported. The CDN dedicated log set named cloud_cdn_logset is used as the default log set.

Viewing the Domain Name Analysis Dashboard

1. After enabling the collection of CDN domain name access logs, select **Dashboard** in the top tab bar.

2. In the top secondary tab, select the target dashboard.

3. In the CDN domain name drop-down list, select the target CDN domain name with enabled access log collection and select the target region to be analyzed, namely, **Domestic Access Log** or **Overseas Access Log**.

4. After selecting a domain name, you can view the access analysis dashboard related to the selected domain name.

Retrieving and Analyzing Cloud Load Balancer (CLB) Access Logs

1. After enabling the collection of CDN domain name access logs, in the top tab bar, select **Log Search**.

2. Select **Domestic Access Log** or **Overseas Access Log** to select a target region for analysis.

3. In the CDN domain name drop-down list, select the target CDN domain name for which access log collection is enabled.

4. After selecting a CDN domain name, you can retrieve and analyze access logs in or outside the Chinese mainland, which are related to the selected CDN domain name.

Disable Domain Name Access Log Collection

1. In **Content Delivery Network (CDN) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

2. Select the instance for which you want to disable log collection in the list and click **Disable Log Collection**. In the drop-down list, select Disable Access Logs in the Chinese Mainland or Disable Access Logs Outside the Chinese Mainland.

Note:

Disabling log collection only represents that no new logs will be collected, but the existing collected logs will continue to be stored in the log topic until expiration, during which storage fees will continue to be incurred. If you do not want to continue storing, choose **Collection Management** > **Storage Topic** and find the target topic to delete.

3. In the Disable Collection pop-up window, you need to select **Disable**, and then click **Confirm** to disable log collection for the selected domain name.

COS Cloud Insight

Last updated：2025-04-07 14:50:44

COS Cloud Insight provides one-stop collection, search, and analysis of Cloud Object Storage (COS) access logs. Meanwhile, based on access logs, it offers out-of-the-box access analysis dashboards. It helps quickly build observability of COS access log analysis.

Features

Provide the out-of-the-box bucket access analysis dashboard.
Unify the entry for COS bucket access log retrieval and analysis.
Enable COS access log collection in batches.

Prerequisites

Creating a COS bucket is completed.

Directions

Enabling Bucket Access Log Collection

Step 1: Entering COS Cloud Insight

1. Log in to the Cloud Log Service console.
2. Click **Cloud Insight** in the left navigation bar to go to the Cloud Insight page.

3. In **Cloud Product Logs**, select and click **Cloud Object Storage (COS) Cloud Insight.**

Step 2: Enabling COS Access Log Collection

1. In **Cloud Object Storage (COS) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

2. Select the region where the target bucket is located. In the bucket list, find and select one or more target buckets.

3. Click **Enable log collection**.

4. In the Enable Log Collection pop-up window, click **Confirm** to start collecting access logs for the target bucket.

Note:

The target log topic and log set are by default COS object storage dedicated log topics and log sets. The log topic name is cos-log-store, and the log set named is cos-log-ap-${bucket region}-${UIN}.

Viewing the Bucket Access Analysis Dashboard

1. After enabling bucket access log collection, in the top tab bar, select **Dashboard**.

2. In the bucket drop-down list, select the target bucket for which access log collection is enabled.

3. After selecting a bucket, you can view the bucket access analysis dashboard related to the selected bucket.

Retrieving and Analyzing Bucket Access Logs

1. After enabling bucket access log collection, select **Log Search** in the top tab bar.

2. In the bucket drop-down list, select the target bucket for which access log collection is enabled.

3. After selecting a bucket, you can retrieve and analyze the bucket access logs related to the selected bucket.

Disabling Bucket Access Log Collection

1. In **Cloud Object Storage (COS) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

2. Select the bucket for which you want to disable log collection in the bucket list ,and click **Disable Log Collection**.

Note:

Disabling log collection only represents that no new logs will be collected. The existing collected logs will continue to be stored in the log topic until expiration, and storage fees will continue to be incurred during this period. If you do not want to continue storing, please choose **Collection Management** > **Storage Topic** and find the target topic and delete it.

3. In the Disable Collection pop-up window, select **Disable**, and then click **Confirm** to disable log collection for the selected bucket.

SCF Cloud Insight

Last updated：2025-04-07 14:50:44

SCF Cloud Insight provides one-stop collection and retrieval analysis of Serverless Cloud Function (SCF) call logs, helping you quickly build observability of SCF call analysis.

Features

Unify the retrieval and analysis entry for SCF call logs.
Enable SCF call log collection in batches.

Prerequisites

Creating a SCF function is completed.

Directions

Enabling SCF Call Log Collection

Step 1: Entering SCF Cloud Insight

1. Log in to the Cloud Log Service console.
2. In the left navigation bar, select **Cloud Insight** to enter the Cloud Insight page.

3. In **Cloud Insight**, select and click **Serverless Cloud Function (SCF) Cloud Insight.**

Step 2: Enabling Call Log Collection for SCF

1. In **Serverless Cloud Function (SCF) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

2. Select the region where the target SCF function is located. In the SCF list, find and select one or more target SCF functions.

3. Click **Enable log collection**.

4. In the Enable Log Collection pop-up window, click **Confirm** to start collecting the call logs of the target SCF function.

Note:

The target log topic and log set are dedicated to SCF by default. The log topic name is SCF_logtopic_${automatically generated unique identifier}, and the log set name is SCF_logset_${automatically generated unique identifier}.

Retrieving and Analyzing SCF Call Logs

1. After enabling call log collection for SCF, select **Log Search** in the top tab bar.

2. In the SCF drop-down list, select the target SCF with access log collection enabled.

3. After selecting the SCF, you can retrieve and analyze the function call logs related to the selected SCF instance.

Disabling SCF Call Log Collection

1. In **Serverless Cloud Function (SCF) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

2. Select the SCF for which you want to disable log collection in the SCF list, and click **Disable Log Collection**.

Note:

Disabling log collection only represents that no new logs will be collected. The existing collected logs will continue to be stored in the log topic until expiration, and storage fees will continue to be incurred during this period. If you do not want to continue storing, choose **Collection Management** > **Storage Topic** and find the target topic and delete it.

3. In the pop-up for disabling collection, you need to select **Disable**, and then click **Confirm** to disable log collection for the selected SCF instance.

TKE Cloud Insight

Last updated：2025-04-07 14:50:45

TKE Cloud Insight provides one-stop collection, search, and analysis of Tencent Kubernetes Engine (TKE) business logs, audit logs, and event logs. Meanwhile, based on audit and event logs, it offers out-of-the-box audit and event analysis dashboards to help quickly build observability of TKE access analysis.

Features

Provide out-of-the-box TKE cluster audit and event analysis dashboards.
Unify the entry for TKE cluster audit and event log search and analysis.
Start TKE cluster business logs, audit logs, and event logs collection in batches.

Prerequisites

Creating a TKE Cluster is completed.

Directions

Enabling Cluster Business Log Collection

Step1: Selecting a Cluster

1. Log in to the Cloud Log Service console.
2. Click **Cloud Insight** in the left navigation bar to go to the Cloud Insight page.

3. In **Cloud Insight**, select and click **Tencent Kubernetes Engine (TKE) Cloud Insight.**

4. In **Tencent Kubernetes Engine (TKE) Cloud Insight**, choose **Collection Management** > **Instance Collection**.

5. Select the region where the TKE cluster is located, and find the target collection cluster in the cluster list.

If the status of a collection component is **Not installed**, click **Install** to install the log collection component.

Note:

If a log collection component is installed in a cluster, a pod named tke-log-agent pod and a pod named cls-provisioner will be deployed by using DaemonSet in the kube-system namespace of the cluster. Reserve at least 0.1 cores and 16 MiB of available resources for each node.

To install a log collection component on a TKE cluster, you need the BRAC permissions to access the cluster. If not, contact the cluster administrator to add them.

6. If the status of a collection component is **Latest**, click **Create Collection Configuration** on the right to enter the cluster log collection configuration process.

Note:

To configure log collection for a TKE cluster, you need BRAC permissions to access the cluster. If not, contact the cluster administrator to add them.

Step 2: Configuring a Log Topic

Enter the cluster log collection configuration process. In the **Create Log Topic** step, you can select an existing log topic or create a log topic for storing logs. For more information about log topics, see Log Topic.

Step 3: Configuring Collection Rules

After selecting a log topic, click **Next** to enter **Collection Configuration** to configure collection rules. The configuration information is as follows:

Log Source Configuration:

Collection Rule Name: You can customize the log collection rule name.

Collection Type: Currently, the collection types support container STANDARD output, container file path, and node file path.

Container standard output

Container file path

Node file path

Select one of the following 3 methods to specify the collection log source of container STANDARD output: **All containers**, **Specific workload**, and **Specific pod labels**.

All containers: Represents collecting STANDARD output logs from all containers in the specified namespace, as shown below:

Specific workload: Represents collecting STANDARD output logs from the specified container in the specified workload under the specified namespace, as shown below:

Specific pod labels: Represents collecting STANDARD output logs from all containers with specified Pod Labels in the specified namespace, as shown below:

Note:

A container file path cannot be a symbolic link. Otherwise, the actual path of the symbolic link will not exist in the collector's container, resulting in log collection failure.

Select one of the following 2 methods to specify the log source for collecting container file paths: **Specific workload** and **Specific pod labels**.

Specific workload: Represents collection container file paths in the specified container within the specified workload under the specified namespace, as shown below:

Specific pod labels: Represents collection container file paths in all containers with specified pod labels under the specified namespace, as shown in the figure below:

File Path: Consists of a log directory and a file name. The log directory starts with /, and the file name does not start with /. Both the log directory and the file name support the use of wildcard characters (? and *). Commas are not supported. /**/ indicates that the log collection component will monitor all levels of log files that match the specified directory. Multiple file paths are in an OR relationship. For example, if a container file path is `/opt/logs/*.log`, you can specify the log with a directory of `/opt/logs` and a file name of `*.log`.

Note:

Only container collection components of version 1.1.12 or later support multiple paths for collection.

Only collection configurations created after container collection components are upgraded to version 1.1.12 or later support defining multiple paths for collection.

After container collection components are upgraded to version 1.1.12, the corresponding collection configurations created in versions earlier than 1.1.12 do not support configuring multiple paths for collection. The collection configurations need to be recreated.

Collection Path Blocklist: After this configuration is enabled, specified directory paths or complete file paths can be ignored during collection. Directory paths and file paths can be fully matched and support wildcard pattern matching.

Blocklist Configuration: Collection blocklists are divided into two filter types, which can be used simultaneously:

By File Path: Under the collection path, the specified complete file path needs to be ignored for collection. The file path supports wildcard characters (* and ?), and supports ** path fuzzy matching.

By Directory Path: Under the collection path, the specified directory needs to be ignored for collection. The directory supports wildcard characters (* and ?), and supports ** path fuzzy matching.

Note:

Container log collection components of version 1.1.2 or later are required.

Files and directory paths specified in blocklists are excluded during collection. Therefore, whether By File Path or By Directory Path is selected, the specified path should be a subset of the collection paths.
The **file path** of a node consists of a log directory and a file name. The log directory starts with /, and the file name does not start with /. Both the log directory and the file name support the use of wildcards (? and *), but commas are not supported. /**/ indicates that a log collection component will monitor all levels of log files that match the specified log directory. Multiple file paths are in an OR relationship. For example, if a node file path is `/opt/logs/*.log`, you can specify the with a log directory of `/opt/logs` and a file name of `*.log`.
Note:
Only collection components of version 1.1.12 or later support multiple paths for collection.
Only collection configurations created after container collection components are upgraded to version 1.1.12 or later support defining multiple paths for collection.
After container collection components are upgraded to version 1.1.12, the corresponding collection configurations created in versions earlier than 1.1.12 do not support configuring multiple paths for collection. The collection configurations need to be recreated.
Collection Path Blocklist: After the option is enabled, specified directory paths or complete file paths can be ignored during collection. Directory paths and file paths can be fully matched and support wildcard pattern matching.

Blocklist Configuration: Collection blocklists are divided into two filter types, which can be used simultaneously:
By File Path: Under the collection path, the specified complete file path needs to be ignored for collection. The file path supports wildcard characters (* and ?), and supports ** path fuzzy matching.
By Directory Path: Under the collection path, the specified directory needs to be ignored for collection. The directory supports wildcard characters (* and ?), and supports ** path fuzzy matching.
Note:
Container log collection components of version 1.1.2 or later are required.
Files and directory paths specified in blocklists are excluded during collection. Therefore, whether By File Path or By Directory Path is selected, the specified path should be a subset of the collection paths.
Metadata configuration:

In addition to raw log content, Cloud Log Service (CLS) also reports metadata related to containers or Kubernetes to CLS together. For example, the ID of the container that generates logs can be reported to CLS. This is convenient for users to trace the source when viewing logs or retrieve according to container IDs and features (for example, container name and labels). You can report this metadata and select as needed for uploading.
For container or Kubernetes-related metadata, refer to the table below:

Field Name	Description
container_id	Container ID to which logs belong.
container_name	Container name to which logs belong.

image_name	Image name/IP of the container to which logs belong.
namespace	The namespace of the pod to which logs belong.
pod_uid	UID of the pod to which logs belong.
pod_name	Name of the pod to which logs belong.
pod_ip	IP address of the pod to which logs belong.
pod_lable_{label name}	Label of the pod to which logs belong. For example, if a pod has two labels: app=nginx and env=prod, then the uploaded log will be accompanied by two metadata entries: pod_label_app:nginx and pod_label_env:prod.

Note:

If you want to collect some pod labels, you need to manually input one or more desired label keys, ending each one with Enter, and the hits will be collected.

Parsing Rule Configuration：

Configure **Collection Scope**. You can select **All** or **New**.

All: Full collection represents starting to collect from the beginning of the log file.

New: Incremental collection represents collecting only the newly added content in the file.

Encode format: Supports **UTF-8** and **GBK**.

Extraction Mode: Supports multiple types of extraction modes, details are as follows:

Single Line Full Text Format

Multi-Line Full Text Format

Single Line - Full Regular Format

Multi-Line - Full Regular Format

JSON Format

Delimiter

Composite Resolution

A single-line full-text log represents that the content of a complete log contains only one line. If a single-line full-text log is collected, CLS will use the line break \\n as the end identifier of the log. For unified structured management, each log will have a default key-value __CONTENT__, but the log data itself will not be processed in a structured manner, nor will log fields be extracted. The time attribute of a log is determined by the time when the log is collected.

A sample raw data entry of a log is as follows:

```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-
6.el7.x86_64
```

The data collected from Cloud Log Service is:

```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-
1.2.90-6.el7.x86_64
```

A multi-line full-text log represents that the content of a complete log may span multiple lines, for example, Java stack trace. In this case, it is somewhat unreasonable to use the line break \\n as the end identifier of the log. In order to enable the logging system to clearly distinguish each log, the first-line regular expression method is adopted for matching. If a certain line of a log matches the preset regular expression, it is considered as the beginning of the log, and the beginning of the next line appears as the end identifier of the log.

For multi-line full text, a default key-value __CONTENT__ will also be set, but the log data itself will no longer be processed in a structured manner, nor will log fields be extracted. The time attribute of a log is determined by the time when the log is collected.

A sample raw data entry of a multi-line log is as follows:

```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
    at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
    at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

The first-line regular expression method is as follows:

```
\\d{4}-\\d{2}-\\d{2}\\s\\d{2}:\\d{2}:\\d{2},\\d{3}\\s.+
```

The data collected from Cloud Log Service is:

```
__CONTENT__:2019-12-15 17:13:06,043 [main] ERROR
com.test.logging.FooFactory:\\njava.lang.NullPointerException\\n    at
com.test.logging.FooFactory.createFoo(FooFactory.java:15)\\n    at
com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

The single-line full regular expression format is usually used to process structured logs, which represents a log parsing pattern that extracts multiple key-value pairs from a complete log using regular expressions.

A sample raw data entry of a log is as follows:

```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1"
127.0.0.1 200 782 9703 "http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0"  0.354 0.354
```

The configured regular expression is as follows:

```
(\\S+)[^\\[]+(\\[[^:]+:\\d+:\\d+:\\d+\\s\\S+)\\s"(\\w+)\\s(\\S+)\\s([^"]+)"\\s(\\S+
```

The data collected from Cloud Log Service is:

```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?
filter%5Btype%5D=all&filter%5Bprice%5D=all&filter%5BcurrentLevelId%5D=all&order
By=studentNum
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101
Firefox/64.0
remote_addr: 10.135.46.111
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

Assume that one of your log raw data is:

```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

The regular expression of the line header is:

```
\\[\\d+-\\d+-\\w+:\\d+:\\d+,\\d+]\\s\\[\\w+]\\s.*
```

The custom regular expression configured is:

```
\\[(\\d+-\\d+-\\w+:\\d+:\\d+,\\d+)\\]\\s\\[(\\w+)\\]\\s(.*)
```

After the system extracts the corresponding key-value according to `()` capture group, you can customize the key name of each group as follows:

```
time: 2018-10-01T10:30:01,000`
level: INFO`
msg:java.lang.Exception: exception happened
    at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
    at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
    at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

Assume that one of your JSON log raw data is:

```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34 +0800","body_sent":
```

After being processed and structured by CLS, this log will become as follows:

```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

Assume that one of your log raw data is:

```
10.20.20.10 - ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/
```

When the log parsing separator is specified as `:::` , this log will be divided into eight fields and each of these fields will be assigned a unique key, as shown below:

```
IP: 10.20.20.10 -
bytes: 35
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

Assume that the raw data of a log is:

```
1571394459, http://127.0.0.1/my/course/4|10.135.46.111|200, status:DEAD,
```

The content of the custom plugins is as follows:

```
{
 "processors": [
   {
     "type": "processor_split_delimiter",
     "detail": {
       "Delimiter": ",",
       "ExtractKeys": [ "time", "msg1","msg2"]
     },
     "processors": [
       {
```

```
          "type": "processor_timeformat",
          "detail": {
            "KeepSource": true,
            "TimeFormat": "%s",
            "SourceKey": "time"
           }
        },
        {
          "type": "processor_split_delimiter",
          "detail": {
            "KeepSource": false,
            "Delimiter": "|",
            "SourceKey": "msg1",
            "ExtractKeys": [ "submsg1","submsg2","submsg3"]
           },
           "processors": []
        },
        {
          "type": "processor_split_key_value",
          "detail": {
            "KeepSource": false,
            "Delimiter": ":",
            "SourceKey": "msg2"
          }
        }
      ]
    }
  ]
}
```

After being processed and structured by CLS, this log will become as follows:

```
time: 1571394459
submsg1: http://127.0.0.1/my/course/4
submsg2: 10.135.46.111
submsg3: 200
status: DEAD
```

Filter: LogListener only collects logs that match filter rules. Key values support full matching, and filter rules support regular expression matching, such as collecting logs whose `ErrorCode = 404` only. You can enable the filter feature and configure rules as needed.

Step 4: Configuring Index

After configuring the collection rules, click **Next** to enter **Index Configuration**. For information about indexes, see Index. Configure the following information in the index configurations:

Index Status: Confirm whether the index feature is enabled to use analysis features such as log retrieval.
Full-Text Index: Confirm whether case sensitivity needs to be set. By default, the following full-text segmentation symbols are supported: @&()="',;:<>[]{}/ \\n\\t\\r. Confirm whether the default full-text word delimiters need to be modified.
Allow Chinese Characters: Confirm whether to enable this configuration.
Key-Value Index: You can configure the field type, segmentation symbol, and whether to enable statistical analysis according to the key name as needed. If you need to enable the key-value index feature, enable this configuration.

Note:

Index configuration should be enabled for retrieval; otherwise, retrieval is not available.
If you need to retrieve logs based on log fields, you need to configure key-value indexes.
If you need to perform statistical analysis based on log fields, you need to configure key-value indexes and enable statistics.
Index rules are effective only for newly written logs after being edited; existing data will not be updated.

Step 5: Retrieving Business Logs

At this point, all deployments for collecting TKE cluster business logs are completed. You can choose CLS console > Search and Analysis to view the collected logs.

Managing Business Log Collection Configuration

1. In **Collection Management** > **Instance Collection**, find the target TKE cluster and click the cluster name to enter the cluster details page.

2. On the cluster details page, you can view and manage your cluster business log collection configurations in **Cluster business log**.

Enabling Cluster Audit/Event Log Collection

Note:

Cluster audit logs record visits to the kube-apiserver as events, sequentially logging each user, administrator, or system component's activities affecting the cluster.
Cluster event logs record the operation of the cluster and the scheduling of various resources.

Step1: Selecting a Cluster

1. If the status of a collection component is **Not installed**, click **Install** to install the log collection component.

Note:

If a log collection component is installed in a cluster, a pod named tke-log-agent pod and a pod named cls-provisioner will be deployed by using DaemonSet in the kube-system namespace of the cluster. Reserve at least 0.1 cores and 16

MiB of available resources for each node.
2. If the status of a collection component is **Latest**, click the cluster name to enter the Cluster Details page, and find **Cluster audit log** or **Cluster event log** on the cluster details page.

3. **Enable Cluster audit log** or **Cluster event log** and enter the corresponding configuration process**.**

Step 2: Selecting a Log Topic

Enter the audit or event log configuration process. In the **Create Log Topic** step, you can select an existing log topic or create a log topic for storing logs. For more information about log topics, see Log Topic.

Step 3: Configuring Index

After configuring the log topic, click **Next** to enter **Index Configuration**. For information about indexes, see Index. The configuration information in the index configuration is as follows:

Index Status: Confirm whether the index feature is enabled to use analysis features such as log retrieval.
Full-Text Index: Confirm whether case sensitivity needs to be set. By default, the following full-text segmentation symbols are supported: @&()='",;:<>[]{}/ \\n\\t\\r. Confirm whether the default full-text word delimiters need to be modified.
Allow Chinese Characters: Confirm whether to enable this configuration.
Key-Value Index: You can configure the field type, segmentation symbol, and whether to enable statistical analysis according to the key name as needed. If you need to enable the key-value index feature, enable this configuration.
Note
Index configuration should be enabled for retrieval; otherwise, retrieval is not available.
If you need to retrieve logs based on log fields, you need to configure key-value indexes.
If you need to perform statistical analysis based on log fields, you need to configure key-value indexes and enable statistics.
Index rules are effective only for newly written logs after being edited; existing data will not be updated.

Step 4: Retrieving Audit and Event Logs

At this point, all deployments for collecting audit or event business logs of the TKE Kubernetes cluster are completed. You can click **Log Search**, select a log type (audit or event) you want to retrieve, and select the target cluster to retrieve audit or event logs related to the selected cluster.

Viewing the TKE Cluster Audit and Event Analysis Dashboard

1. After enabling the audit or event log collection for the TKE cluster, select **Dashboard** in the top tab bar.

2. In the top secondary Tab, select the target audit or event dashboard.

3. In the TKE cluster drop-down list, select the target TKE cluster with audit or event log collection enabled.

4. After selecting a cluster, you can view the event or audit analysis dashboard related to the selected cluster.

Upgrading Log Collection Component

On the **Collection Management** page, find the target TKE cluster. If the status of a collection component shows **Upgrade available**, you can click **Update** to upgrade the log collection component to the latest version.

Uninstalling Log Collection Component

On the **Collection Management** page, find the target TKE cluster, click **More** in the operation column, and then click **Uninstall Collection Component** in the drop-down list.

Historical Documentation
Operation guide of earlier LogListener versions

Last updated：2024-01-20 17:55:44

Note:

This document provides an operation guide for LogListener 2.2.4 and earlier. We recommend that you update to the latest version as this document may no longer be maintained. For information on how to install the latest version, see the LogListener Installation Guide.

Starting LogListener

Go to the installation directory `loglistener` and start LogListener by running the following script:

```
cd loglistener/tools; ./start.sh
```

Stopping LogListener

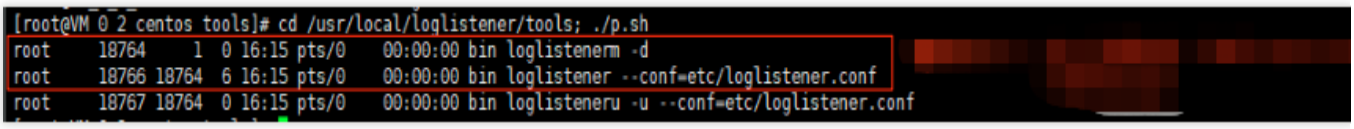
Go to the installation directory `loglistener` and stop LogListener by running the following script:

```
cd loglistener/tools; ./stop.sh
```

Checking LogListener process status

Go to the installation directory `loglistener` and check the status of the LogListener processes by running the following command:

```
cd loglistener/tools; ./p.sh
```



Normally, there are three processes:

```
bin/loglistenerm -d                                 #Daemon process
bin/loglistener --conf=etc/loglistener.conf         #Main process
bin/loglisteneru -u --conf=etc/loglistener.conf     #Update process
```

Uninstalling LogListener

Go to the installation directory `loglistener` and uninstall LogListener by running the following command:

```
cd loglistener/tools; ./uninstall.sh
```

Checking LogListener heartbeat and configuration

Go to the installation directory `loglistener` and check the heartbeat and configuration of LogListener by running the following command:

```
cd loglistener/tools; ./check.sh
```

Troubleshooting earlier LogListener versions

Last updated：2024-01-20 17:55:44

Note:
This document provides troubleshooting information for LogListener 2.2.4 and earlier. For information on troubleshooting the latest version, see Server Group Exceptions.

Error Description

An exception occurred with log collection, and the associated server group is found to be exceptional.

Possible Causes

The heartbeat between the server group and the CLS system is interrupted, resulting in failure to collect and report logs. Possible causes for the server group exception include:
1. The IP address is incorrect.
2. The network is disconnected.
3. LogListener process failure.
4. LogListener is configured incorrectly.

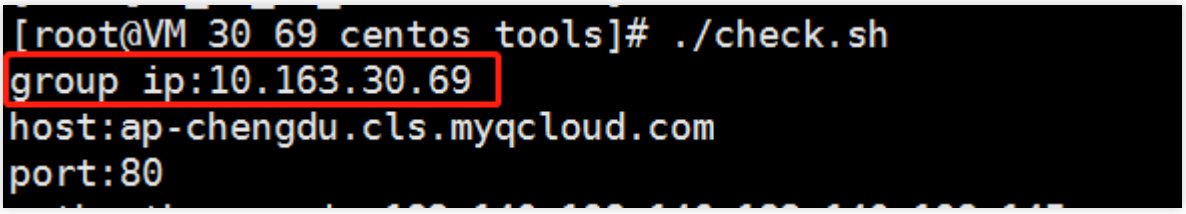
Solution

Troubleshoot problems according to the above causes.

Directions

1. Check whether the IP address added to the server group is correct.
1.1 Check the IP address obtained by LogListener by running the following command:

```
cd loglistener/tools && ./check.sh
```



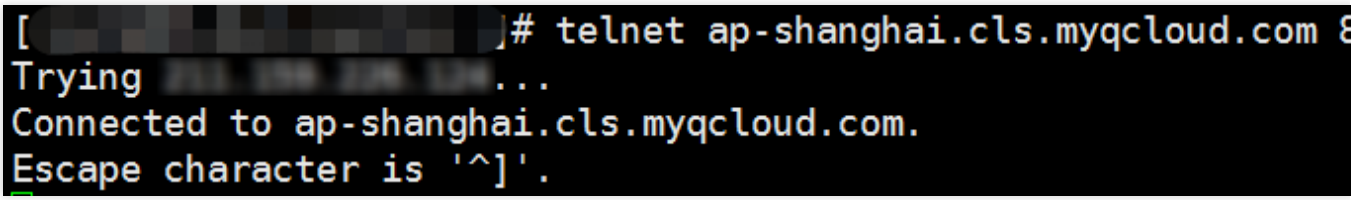
1.2 Log in to the Cloud Log Service Console, and click **Server Group** in the leftside bar. On the Server Group Management page, check the IP address of the server group. The IP address must be the same as that for collection.

2. Check whether the network is connected by running the following command:

```
telnet <region>.cls.myqcloud.com 80
```

`<region>` is the abbreviation for the region where CLS resides. For more information on regions, see Available Regions.

The following code appears upon normal network connection. Otherwise, connection fails. Check the network and ensure normal connection.



3. Check whether LogListener processes are running normally. Go to the installation directory and run the following command:

```
cd loglistener/tools && ./p.sh
```

Normally, there are three processes:

```
bin/loglistenerm -d                                  #Daemon process
bin/loglistener --conf=etc/loglistener.conf          #Main process
bin/loglisteneru -u --conf=etc/loglistener.conf      #Update process
```

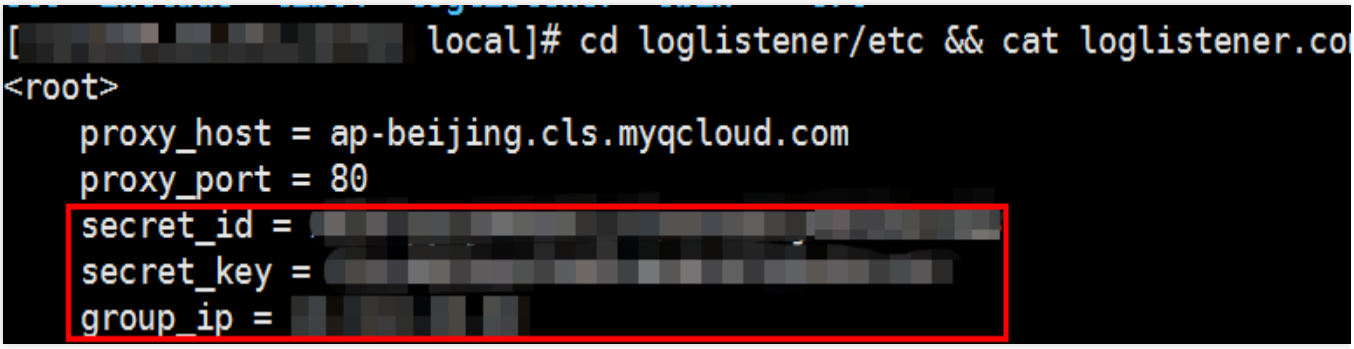
If any process fails, restart it. Go to the installation directory and run the following command:

```
cd loglistener/tools && ./start.sh
```

4. Check whether the key and IP address are correctly configured in LogListener. Go to the installation directory to check the configuration information by running the following command:

```
cd loglistener/etc && cat loglistener.conf
```

See the following figure:



The key is the API key for the Tencent Cloud account or the collaborator. Project keys are not supported.

group_ip in the configuration file must be consistent with the IP address entered in the server group on the console. Since LogListener obtains the server IP address automatically, check the consistency regularly when the server is bound to multiple ENIs.