Cloud Log Service

Practical Tutorial

Product Documentation

Contents

Practical Tutorial
Log Collection
Collecting Windows Logs to CLS

Last updated：2024-01-20 17:28:40

Overview

This document describes how to use Winlogbeat or Filebeat to collect and upload Windows logs to CLS.

Prerequisites

You have activated CLS and created relevant resources such as logset and log topic.
You have obtained the `SecretId` and `SecretKey` in the Tencent Cloud console.

Directions

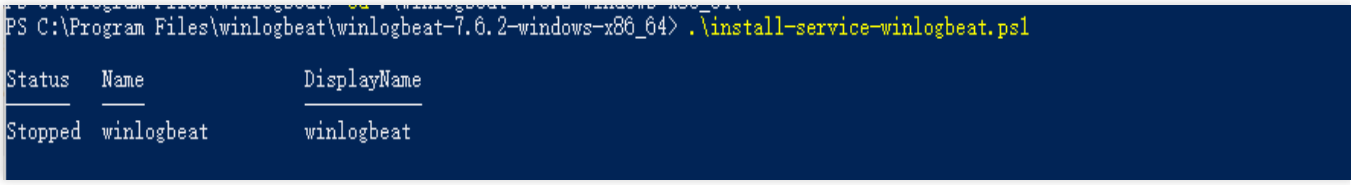
Using Winlogbeat to collect and upload Windows event logs to CLS

Installing Winlogbeat

1. Download the target Winlogbeat version at the official website.
This document takes Winlogbeat 7.6.2 as an example, which can be downloaded here.
2. Decompress the downloaded package to the C drive.
We recommend you create a `winlogbeat` folder under the `Program Files` directory for decompression.
3. Open PowerShell as the admin and run the following command:

```
cd C:\\Program Files
cd .\\winlogbeat-7.6.2-windows-x86_64
.\\install-service-winlogbeat.ps1
```

During execution, if an error is reported, enter the `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned` command and select `y`. Then, enter the above command again.



4. Run the following command to test whether the environment is normal.

```
.\\winlogbeat.exe test config -c .\\winlogbeat.yml -e
```

If `config OK` is returned, the environment is normal.

5. Run the following command to start the program.

```
Start-Service winlogbeat
```

Uploading logs to CLS

In the `winlogbeat.yml` file in `C:\\Program Files\\Winlogbeat`, change `output.kafka` to the following to send logs to CLS.

```
output.kafka:
  enabled: true
  hosts: ["${region}-producer.cls.tencentyun.com:9095"] # TODO: Service
address. The public network port is 9096, and the private network port is 9095.
  topic: "${topicID}" #  TODO: Topic ID
  version: "0.11.0.2"
  compression: "${compress}"   # Configure the compression method. Valid
values: `gzip`, `snappy`, `lz4`.
  username: "${logsetID}"
  password: "${SecurityId}#${SecurityKey}"
```

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, such as 76c63473-c496-466b-XXXX-XXXXXXXXXXXX.
username	Logset ID, such as 0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX.
password	Password in the format of `${SecurityId}#${SecurityKey}`, such as XXXXXXXXXXXXX#YYYYYYYY.

Using Filebeat to collect Windows file logs

Installing Filebeat

1. Download the target version here.
2. Upload and decompress the installation package to the root directory of a drive on the Windows server.
3. Edit the `filebeat.yml` file.

Note:

Use "/" rather than "" in paths.

4. Find the target log path and edit the module configuration file (with mssql as an example below).

```
# Module: mssql
# Docs: https://www.elastic.co/guide/en/beats/filebeat/7.3/filebeat-module-
mssql.html
- module: mssql
# Fileset for native deployment
log:
enabled: true
 # Set custom paths for the log files. If left empty,
# Filebeat will choose the paths depending on your OS.
var.paths: ["D:/Program Files/Microsoft SQL
Server/MSSQL10_50.MSSQLSERVER/MSSQL/Log/ERROR*"]
```

5. Open PowerShell as the admin and run the following command:

```
# Enter the specific Filebeat path
cd c:/filebeat

# Run the installation script to install Filebeat
.\\install-service-filebeat.ps1

# Start the mssql module
.\\filebeat.exe modules enable mssql

# Install the template file
.\\filebeat.exe setup -e

# Start Filebeat
start-service filebeat
```

Uploading logs to CLS

In the `filebeat.yml` file, change `output.kafka` to the following to send logs to CLS:

```
output.kafka:
  enabled: true
  hosts: ["${region}-producer.cls.tencentyun.com:9095"] # TODO: Service
address. The public network port is 9096, and the private network port is 9095.
```

```
  topic: "${topicID}" #  TODO: Topic ID
  version: "0.11.0.2"
  compression: "${compress}"   # Configure the compression method. Valid
values: `gzip`, `snappy`, `lz4`.
  username: "${logsetID}"
  password: "${SecurityId}#${SecurityKey}"
```

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, such as 76c63473-c496-466b-XXXX-XXXXXXXXXXXX.
username	Logset ID, such as 0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX.
password	Password in the format of `${SecurityId}#${SecurityKey}` , such as 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.

Collect/Query Host File Logs

Last updated：2024-01-20 17:28:40

In simple Ops scenarios, logs are directly output to local files on the server, and then you can run the `grep` command in Linux to search for logs. In complex service systems, logs are scattered on different servers, CLI operations are not intuitive, and server permission management is restricted. As a result, it is difficult to find logs, which seriously affects Ops efficiency. It's even more difficult if you need to do some statistical analysis or monitor alarms based on logs.

This document describes how to quickly migrate the local logs searched by the `grep` command to CLS to obtain the following advantages:

Data is stored and searched centrally, and you don't need to log in to multiple servers to query it individually, which is critical in load balancing, microservice, and other architectures.

You can quickly search for logs with a few clicks, eliminating command lines and tedious server permission management.

You can perform statistical analysis based on logs to obtain key business metrics such as PV, API response time, and API error rate.

You can detect exceptional logs in real time and receive notifications through various methods such as SMS, email, and WeChat.

Note:

If your logs have been collected to CLS, you can skip the steps of log collection and index configuration and go to Step 3. Searching for logs.

Step 1. Collect Logs

For server local logs, you can use LogListener to collect raw logs to CLS. For how to install LogListener, please see LogListener Installation Guide.

If you are using Tencent Cloud CVMs, you can also use the automatic installation feature provided by the console to install LogListener. For more information, please see Deploying LogListener on CVMs in Batches.

Different from storing logs locally on servers, collecting logs to CLS makes log search and statistical analysis easier.

When collecting logs, you can convert non-formatted raw logs to formatted data. Assume that the raw log is as follows:

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

You can use the separator `:::` to split the log into eight fields and name each field 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]
```

For operation details, please see Separator Format. In addition to using separators to split logs, you can use regular expression, JSON, and full-text log splitting methods. For more information, please see Collecting Text Logs.

Step 2. Configure Indexes

The purpose of configuring indexes is to define searchable fields and their types to facilitate future log search. For most scenarios, you can use the automatic index configuration feature to complete the configuration with a few clicks. For more information, please see Configuring Indexes.

Step 3. Search for Logs

The following uses the commonly used `grep` command as an example to describe how to achieve a similar log search feature in CLS.
Assume the raw log is as follows:

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

The corresponding formatted log in 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]
```

Example 1

Search for logs where `request` is `/online/sample` .
Use the `grep` command

```
# grep "/online/sample" test.log
```

Use the CLS search mode

```
request:"/online/sample"
```

Example 2

Search for logs where `status` (status code) is not `200`.

Use the `grep` command

```
# grep -v "200" test.log
```

In fact, this mode may exclude certain logs where the number 200 exists but `status` is not `200`.

Use the CLS search mode

```
NOT status:200
```

CLS also supports other more flexible search methods. For example, you can search for logs where `status` is greater than or equal to 500 as follows:

```
status:>=500
```

Example 3

Count the number of logs where `status` is not `200`.

Use the `grep` command

```
# grep -c -v "200" test.log
```

Use the CLS search mode

```
NOT status:200 | select count(*) as errorLogCounts
```

Example 4

Search for logs where `status` is `200` and `request` is `/online/sample`.

Use the `grep` command

```
# grep "200" test.log | grep "/online/sample"
```

Use the CLS search mode

```
status:200 AND request:"/online/sample"
```

Example 5

Search for logs where `request` is `/online/sample` or `/offline/sample`.

Use the `grep` command

```
# grep -E "/online/sample|/offline/sample" test.log
```

Use the CLS search mode

```
request:"/online/sample" OR request:"/offline/sample"
```

Example 6

Search for logs where `request` is `/online/sample` but the log file is not `test.log`.

Use the `grep` command

```
# grep -rn "/online/sample" --exclude=test.log
```

Use the CLS search mode

```
request:"/online/sample" AND NOT __FILENAME__:"test.log"
```

Example 7

Search for the first 10 lines in logs where `time` is `[Tue Jan 22 14:49:45 CST 2019 +0800]`.

Use the `grep` command

```
# grep "[Tue Jan 22 14:49:45 CST 2019 +0800]" -B 10 test.log
```

Use the CLS search mode

```
time:"[Tue Jan 22 14:49:45 CST 2019 +0800]"
```

If a matching log is found, you can click **Context Search** in the console to view logs near the log found.

Search and Analysis
CLB Access Log Analysis

Last updated：2024-01-20 17:35:08

Cloud Load Balancer (CLB) is a product of hundreds of billions of QPS and relies heavily on refined operation. Access logs provided by Cloud Log Service (CLS) are key to refined operation. You can go to CLB > Access Logs to mine the value of massive amounts of access log data. By analyzing access logs, you can monitor client requests, troubleshoot issues, and analyze user behaviors to provide data support for refined operation. This document introduces how to use CLS to analyze CLB access logs.

Example: OPS Monitoring Scenario

Jack is in charge of the OPS of the advertising platform of an internet business. He received feedback from an advertising partner complaining that, when users click on an advertisement on the platform, the advertisement opens very slowly. The advertising partner has high requirements for timeliness and stability and proposed that if the service experiences an exception, an alarm should be generated within 1 minute and solved within 5 minutes. To meet this requirement, CLB logs can be used to implement the following capabilities:
Monitor client access latency and exceptional requests, and report alarms when the specified threshold is exceeded.
When there are alarms, the following additional information is provided to help you determine fault causes:
Websites, LB instances, and real server (RS) instances that the requests whose latency exceeds the specified threshold access
Access latency statistics of the LB and RS instances

OPS Monitoring Solution

Based on the 1-minute real-time alarms and multidimensional analysis capabilities of CLS, you can quickly monitor CLB access logs to locate and rectify faults.

Step 1. Enable the feature of shipping CLB access logs to CLS

1. Log in to the CLB console.
2. On the left sidebar, choose **Instance Management** to go to the instance management page.
3. Click a CLB ID/name to go to the CLB management page.
4. Locate and enable **Cloud Log Service (CLS)**, as shown in the following figure.

For more information, please see Configuring Access Logs.

Step 2. Configure the client access latency statistics and alarm policy

Client access latency statistics

```
* | select time_series(__TIMESTAMP__, '1m', '%Y-%m-%d %H:%i:%s', '0')  as time, rou
```

Error request statistics

```
status:>200 | select time_series(__TIMESTAMP__, '1m', '%Y-%m-%d %H:%i:%s', '0')
as time, status, count(1) group by time,status order by time limit 1000
```

Configure an alarm policy to detect alarms whose average latency per minute is higher than a specified threshold

Configure multidimensional analysis in the alarm policy to obtain additional information when an alarm occurs

Access latency statistics of the LB and RS instances

Websites, LB instances, and real server (RS) instances that the requests whose latency exceeds the specified threshold access

Configure notification channels. The following channels are supported:

Email

SMS

WeChat

WeCom

Phone

Custom API callback

Step 3. Receive alarms and perform quick fault locating

Once an alarm is triggered, you can receive alarm information and details through channels such as WeChat, WeCom, SMS, and phone.

Alarm details include information such as affected RS and LB instances.

It can be seen that the average latency of LB instances is relatively high and the LB instance most affected is `9.*****.1`. In this way, Jack can quickly locate the exceptional LB instance and restore it. The entire process takes only 1 minute.

Example: Operations Statistics Scenario

Jane, responsible for operating a technology content app, needs to plan an offline salon next month to increase the stickiness of existing users and take the opportunity to promote products and attract new users. Due to the short preparation time and limited funds, in order to achieve the KPI target, Jane lists the following information that needs to be obtained:

Where to hold the offline salon: it is necessary to understand the geographical sources of visiting customers and the geographical locations of key customer groups.

What is the theme of salon: collect statistics on the top ranking of hot websites to understand which content users pay more attention to
Which clients are currently used by users to access CLS: design the landing page according to the current client distribution.
Which channels should the landing page be placed: collect statistics on the request sources of the current website to find out the diversion entries with high traffic and use them as the key areas for advertising.

Operations Statistics Solution

Analyzing CLB access logs helps Jane solve operations statistics problems with ease.

Step 1. Find out the geographical sources of visiting customers

Use the IP function provided by CLS to convert client IPs to the corresponding provinces or countries.
Collect statistics on customer distribution by province in China

```
* | select count(1) as c, ip_to_province(remote_addr) as address group by address l
```

Collect statistics on customer distribution by country

```
* | select count(1) as c, ip_to_country(remote_addr) as address group by address li
```

Step 2. Sort hot websites

The `http_host` field records request domain names. By calculating the PV and UV of request domain names, you can sort out top hosts.

```
* | select http_host, count(*) as pv, approx_distinct(remote_addr) as uv group by h
```

Step 3. Collect statistics on client distribution

```
* | select http_user_agent, count(*) group by http_user_agent
```

Step 4. Collect statistics on the request sources of the current website

The `http_referer` records the request sources of the website.

```
* | select http_referer, count(*) as count group by http_referer order by count des
```

Summary

By analyzing CLB access logs, you can obtain various valuable results, such as PV and UV trends, client packet traffic statistics, status code distribution, and p99 and p95 access latency statistics. To help you quickly analyze CLB access logs, CLS and CLB have jointly created a visual analysis scheme out of the box. You can enjoy it immediately by enabling the feature of shipping CLB access logs to CLS.

NGINX Access Log Analysis

Last updated：2024-01-20 17:28:40

Overview

NGINX is a high-performance HTTP and reverse proxy server. By analyzing NGINX logs, you can obtain various valuable results, such as data support for website diagnosis and tuning, website stability monitoring, and operations statistics. This document introduces how to use CLS to comprehensively mine NGINX log data.

Prerequisites

NGINX logs have been collected to CLS. For more information, please see Collecting and Searching NGINX Access Logs.
This document uses standard NGINX log configuration:

```
log_format  main  '$server_name $remote_addr - $remote_user [$time_local]
"$request" '
                      '$status $uptream_status $body_bytes_sent "$http_referer"
'
                      '"$http_user_agent" "$http_x_forwarded_for" ';
```

Scenarios

Diagnosis and tuning

Requirement description

Tune pages with high access latency to improve user experience.

Solution

Calculate the average latency and highest latency of requests every 5 minutes to understand the overall latency situation.

```
* | select time_series(__TIMESTAMP__, '5m', '%Y-%m-%d %H:%i:%s', '0')  as time, avg
```

Find out the request page corresponding to the highest request latency and further optimize the response speed of the page.

```
* | select time_series(__TIMESTAMP__, '5m', '%Y-%m-%d %H:%i:%s', '0')  as time, max
```

Tune the page with the highest access latency.

For example, the /**4nm8c.html** page has the highest access latency and needs to be tuned. To tune the page, you need to calculate the page access PV, UV, request method statistics, request status statistics, browser statistics, average access latency, and highest access latency.

```
request_uri:"/4nm8c.html*" | select count(1) as pv,
      approx_distinct(remote_addr) as uv,
      histogram(method) as method_pv,
      histogram(status) as status_pv,
      histogram(user_agent) as user_agent_pv,
      avg(request_time) as avg_latency,
      max(request_time) as max_latency
```

Monitoring website stability

Requirement description

In terms of performance problems, website errors, and traffic slump or surge, you can detect problems before users based on threshold-based log monitoring.

Solution

It is more accurate to use percentiles (such as highest latency of the 99% percentile) in mathematical statistics as alarm trigger conditions. If average or individual values are used as alarm trigger conditions, the latency of some individual requests will be averaged, failing to reflect the actual situation. For example, you can use the following query analysis statement to calculate the average latency of each minute, the latency of the 50th percentile, and the latency of the 90th percentile in a one-day window (1,440 minutes).

```
* | select avg(request_time) as l, approx_percentile(request_time, 0.5) as p50, app
```

The system will report alarms when the latency of the 99th percentile is greater than 100 ms and display the affected URLs and users affected in the alarm information. In this way, you can quickly understand the error situation.

```
* | select approx_percentile(request_time, 0.99) as p99
```

When receiving the alarm information, you can find out the top affected URLs and users and make targeted alarm recovery measures accordingly.

Analyzing website access information

With CLS, you can build an operations data dashboard to quickly analyze website access information such as access PV/UV statistics, access geographic information statistics, top 10 access sources, and top 10 access addresses.

Collect statistics on the access IP sources of the recent day

```
* | select count(1) as c, ip_to_province(remote_addr) as address group by address l
```

Display the top 10 access source pages with the highest PV value of the recent day to get the hot pages

```
* | select count(1) as pv , http_referer  group by http_referer order by pv desc li
```

pv ↑	http_referer ↕
13	http://0lomc.com/v_19rrknzf8g.html?vfm=m_141_xmsp
13	http://02adv.com/lc03_p/201509/29/17/19/17/newplayer/LetvPlayer.
13	http://0e3cs.com/q595k1o5z1jn/article/1444002350-106960478.html
13	http://02bal.com/ui/swf/player/v0929/main.swf
13	http://0kipu.com/ads/index.htm?v=3.63
19	http://jwf.cc:8085/wap/wifi/wait
32	http://jwf.cc:8085/wap/wifi/confirm
51	http://www.guilin.cm/

Display the PV and UV statistics of the recent day

```
*| select approx_distinct(remote_addr) as uv ,count(1) as pv , time_series(__TIMEST
```

CDN Access Log Analysis

Last updated：2024-01-20 17:42:52

Overview

Content Delivery Network (CDN) is a very important internet infrastructure. It allows users to access various images, videos and other resources on the network quickly. During the access process, CDN will generate a large amount of log data. Through the analysis of CDN access logs, users can mine a large amount of useful information for CDN quality and performance analysis, error diagnosis, client distribution analysis, and user behavior analysis.

Prerequisites

CDN logs have been collected to the Cloud Log Service (CLS). For more information, please see operation details.

Scenarios

Traditional CDN log analysis

At present, CDN service providers usually provide basic monitoring metrics in real time, such as the number of requests, bandwidth, and other information. However, in many specific analysis scenarios, these default real-time metrics may not be sufficient for custom analysis requirements. Usually, users will download raw CDN logs for offline in-depth analysis and mining.
In that case, users need to set up offline analysis clusters themselves, not only requiring a lot of OPS and development costs and manpower but also making it difficult to guarantee the timeliness of offline logs in some analysis scenarios such as CDN log-based alarm reporting and troubleshooting.

CDN-to-CLS solution

Interconnect Tencent Cloud CDN and CLS so that users can ship CDN data to CLS in real time to further use the search and SQL analysis capabilities of CLS to meet users' personalized real-time log analysis needs in different scenarios:
Push-button log shipping
Analyzing tens of billions of log data entries within seconds
Visualizing real-time logs on dashboards
Real-time alarm reporting in 1 minute

Introduction to CDN logs

CDN log fields are described as follows.

Log Field	Raw Log Type	Log Service Type	Description
app_id	Integer	long	Tencent Cloud account `APPID`
client_ip	String	text	Client IP
file_size	Integer	long	File size
hit	String	text	Cache hit/miss. Both hits on CDN edge servers and parent nodes are marked as hit
host	String	text	Domain name
http_code	Integer	long	HTTP status code
isp	String	text	ISP
method	String	text	HTTP method
param	String	text	Parameter carried in URL
proto	String	text	HTTP protocol identifier
prov	String	text	ISP province
referer	String	text	Referer information, i.e., HTTP source address
request_range	String	text	Range parameter, i.e., request range
request_time	Integer	long	Response time (in milliseconds), which refers to the time it takes for a node to return all packets to the client after receiving a request.
request_port	String	long	A port connecting the client and CDN nodes. This field will be displayed as `-` if the port does not exist.
rsp_size	Integer	long	Number of returned bytes
time	Integer	long	Request timestamp in UNIX format (in seconds)
ua	String	text	`User-Agent` information
url	String	text	Request path
uuid	String	text	Unique request ID
version	Integer	long	CDN real-time log version

CDN quality monitoring

Scenario 1: monitor CDN access latency and report an alarm when a specified threshold is exceeded

It is more accurate to use percentiles (such as 99% highest latency) in mathematical statistics as alarm trigger conditions. If average or individual values are used as alarm trigger conditions, the latency of some individual requests will be averaged, failing to reflect the actual situation. For example, you can use the following query analysis statement to calculate the average latency of each minute, the latency of the 50th percentile, and the latency of the 90th percentile in a one-day window (1,440 minutes).

```
* | select avg(request_time) as l, approx_percentile(request_time, 0.5) as p50, app
```

If the latency of the 99th percentile is greater than 100 ms, an alarm is reported and the affected domain name, URL, and client IP are included in the alarm information to facilitate fault locating. The alarm setting statement is as follows:

```
* | select approx_percentile(request_time, 0.99) as p99
```

By configuring multidimensional analysis, you can include the affected domain name, URL, and client IP in the alarm information to help developers in fault locating.
Once an alarm is triggered, users can obtain key information via WeChat, WeCom, and SMS.

Scenario 2: monitor resource access errors and report an alarm when the period-on-period increase exceeds a specified threshold

When the number of page access errors surges, it is possible that the CDN backend server fails or is overloaded with requests. You can set an alarm to monitor the increase in the number of request errors in a certain period of time (e.g., 1 minute). When the period-on-period increase exceeds a certain threshold, an alarm is reported.

Number of errors in the recent minute

```
* | select * from (select * from (select * from (select date_trunc('minute', __TIME
```

Number of errors in the last minute

```
* | select *  from (select * from (select * from (select date_trunc('minute', __TIM
```

The trigger condition in the alarm policy is configured as follows: Number of errors in the recent minute – Number of errors in the last minute > Specified threshold.

```
$2.errct-$1.errct >100
```

CDN quality and performance analysis

CDN logs contain rich content, enabling users to conduct comprehensive statistics and analysis of the overall quality and performance of CDN from multiple dimensions.

Health

Cache hit rate

Average download speed

ISPs' download count, download traffic, and download speed

Response latency

Health

Calculate the percentage of requests whose `http_code` is less than 500 of all requests.

```
* | select round(sum(case when http_code<500 then 1.00 else 0.00 end) / cast(count(
```

Cache hit rate

Calculate the percentage of requests whose `return_code` is less than 400 of requests whose value of `hit` is `hit` .

```
http_code<400 | select round(sum(case when hit='hit' then 1.00 else 0.00 end) / cas
```

Average download speed

Divide the total downloads over a period of time by the total elapsed time to get the average download speed.

```
* | select sum(rsp_size/1024.0) / sum(request_time/1000.0) as "Average download spe
```

ISPs' download count, download traffic, and download speed

Use the `ip_to_provider` function to convert `client_ip` to the corresponding ISP.

```
* | select ip_to_provider(client_ip) as isp , sum(rsp_size)* 1.0 /(sum(request_time
```

Response latency

Collect access latency statistics by window, and appropriate latency time windows can be divided according to the actual situation of the application.

```
* | select case when request_time < 5000 then  '~5s'  when request_time < 6000 then
```

CDN quality and performance analysis

Access errors have always been an important factor that affects service experience. When an access error occurs, you need to quickly locate the following information of the error: QPS and the proportion, domain name and URI that are affected the most, whether the error is region or ISP related, and whether the error is caused by the newly published version.

Solutions

Check the distribution of 4xx and 5xx errors.

```
* |  select  http_code , count(*) as c where http_code >= 400 group by
http_code order by c  desc
```

As shown in the figure blow, a 404 error occurs, indicating that the accessed file or content does not exist. In this case, it is necessary to check whether the resource has been deleted or terminated.

For requests whose `http_code` is greater than 400, we conduct multidimensional analysis, for example, sorting requests by domain name and URL separately in descending order, checking error counts by province and ISP, and checking client distribution.

Sort requests by domain name

```
* |  select  host , count(*) as count where http_code > 400   group by host  order
```

Sort requests by URL

```
* |  select  url , count(*) as count where http_code > 400   group by url  order by
```

Check error counts by province and ISP

```
* | select client_ip, ip_to_province(client_ip) as "province", ip_to_provider(clien
```

Check client distribution

```
* | select ua as "Client version", count(*) as "Error count" where http_code > 400
```

As shown in the figure, all errors occurred on the Safari client. Fault locating finds that the error is caused by a bug in the new version, which causes frequent failure to access resources via the Safari browser window.

User behavior analysis

Requirements

Where do most users come from, internally or externally?

What are the most popular resources for users?

Are there users who download massive resources frequently, and does the behavior meet expectations?

Solutions

Analyzing access sources

```
* | select ip_to_province(client_ip) as province ,  count(*) as c group by province
```

Analyzing top access URLs

```
http_code < 400 | select url ,count(*) as  "Access count", round(sum(rsp_size)/1024
```

Analyzing top domain names in terms of traffic (volume of data downloaded)

```
* | select host, sum(rsp_size/1024) as  "Total Downloads" group by host order by  "
```

Analyzing top users in terms of download amount

```
* | SELECT CASE WHEN ip_to_country(client_ip)='Hong Kong' THEN concat(client_ip, '
```

Analyzing top users with valid access

```
* | SELECT CASE WHEN ip_to_country(client_ip)='Hong Kong' THEN concat(client_ip, '
```

Analyzing access PVs and UVs (access count within a certain period of time and independent client IP change trend)

```
* | select time_series(__TIMESTAMP__, '1m', '%Y-%m-%dT%H:%i:%s+08:00', '0') as time
```

COS Access Log Analysis

Last updated：2024-12-16 16:48:53

Overview

Cloud Object Storage (COS) access logs record information about users' access to COS resources, including object upload (`PUT`), object deletion (`DELETE`), and object getting (`GET`). By analyzing access logs, you can perform audit backtracking, such as deleting resource records and collecting statistics on popular resources. This document introduces how to analyze COS access logs.

Prerequisite

COS logs have been collected to Cloud Log Service (CLS). For more information, please see Enabling Real-Time Log Feature on COS.

Introduction to Access Logs

COS access logs record information such as the source bucket, user ID, and request method.

No.	Field	Description	Example
1	eventVersion	Log version	1.0
2	bucketName	Bucket name	examplebucket-1250000000
3	qcsRegion	Request region	ap-beijing
4	eventTime	Event time (request end time, which is a timestamp in UTC+0 time zone)	2018-12-01T11:02:33Z
5	eventSource	Access domain name	examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
6	eventName	Event name	UploadPart
7	remoteIp	Source IP	192.168.0.1

8	userSecretKeyId	User access KeyId	AKIDNYVCdoJQyGJ5brTf
9	reservedField	Reserved field	Displayed as `-`
10	reqBytesSent	Request bytes	83886080
11	deltaDataSize	Change in storage made by the request (in bytes)	808
12	reqPath	Requested file path	/folder/text.txt
13	reqMethod	Request method	put
14	userAgent	User agent (UA)	cos-go-sdk-v5.2.9
15	resHttpCode	HTTP return code	404
16	resErrorCode	Error code	NoSuchKey
17	resErrorMsg	Error message	The specified key does not exist.
18	resBytesSent	Bytes returned	197
19	resTotalTime	Total time used by the request (in milliseconds, i.e., the time between the last byte of the response and the first byte of the request)	4295
20	logSourceType	Source type of the log	`USER` (user access requests), `CDN` (CDN origin-pull requests)
21	storageClass	Storage class	STANDARD, STANDARD_IA, ARCHIVE
22	accountId	Bucket owner ID	100000000001
23	resTurnAroundTime	Time used by the request server (in milliseconds, i.e., the time between	4295

		the first byte of the response and the last byte of the request)	
24	requester	Requester	Root account ID, sub-account ID, or `-` (anonymous access)
25	requestId	Request ID	NWQ1ZjY4MTBfMjZiMjU4NjRfOWI1N180NDBiYTY=
26	objectSize	Object size, in bytes	808. If you use multipart upload, `objectSize` will only be displayed when the upload is completed, and will be `-` during the multipart upload process
27	versionId	Object version ID	Random string
28	targetStorageClass	Destination storage class, recorded for replication requests	STANDARD, STANDARD_IA, ARCHIVE
29	referer	HTTP referer of the request	`*.example.com` or 111.111.111.1
30	requestUri	Request URI	"GET /fdgfdgsf%20/%E6%B5%AE%E7%82%B9%E6%95%B0 HTTP/1.1"

Examples

Example 1: audit backtracking

Requirement

An object file cannot be accessed, and the cause needs to be located.

Solution

Go to the COS access log search page, and enter the object name as the keyword to search for logs.

```
json-log2019-05-09_00645d9a-1118-4d69-8411-cfd57ede9ea1_000
```

According to the time column chart, 14 logs are recorded on the last day. For the drill-down analysis of the 14 log records, click the quick analysis bar on the left to view the **resHttpCode** information.

According to the quick analysis, there are 6 request log records whose **resHttpCode** is not 200: **resHttpCode** is 403 for 5 log records and 204 for 1 log record. Click to search for these logs quickly.
According to the logs, the 5 log records whose error code is **Access Deny** are object access failure logs. According to the logs whose **resHttpCode** is 204, object access failed because user `1000******` performed object deletion at around 20:16 on August 24 in the COS console.

Example 2: operations statistics

Requirement

Collect statistics on the top 10 most visited buckets of the day
Collect statistics on the access trend of a certain bucket
Collect statistics on the top 10 visitors of the error requests
Collect statistics on the bucket distribution of the failed operations
User request efficiency trend

Solution

Collect statistics on the top 10 most visited buckets of the day

```
(reqMethod:"GET") | select bucketName, count(*) group by bucketName
```

Collect statistics on the access trend of a certain bucket

```
* | select time_series(TIMESTAMP, '1m', '%Y-%m-%dT%H:%i:%s+08:00', '0') AS time, co
```

Collect statistics on the top 10 visitors of the error requests

```
resHttpCode:>200 | select remoteIp, count(*) group by remoteIp
```

Collect statistics on the bucket distribution of the failed operations

```
resHttpCode:>200 | select bucketName, count(*) group by bucketName
```

User request efficiency trend

```
* | select time_series(__TIMESTAMP__, '5m', '%Y-%m-%d %H:%i:%s', '0')  as time,roun
```

User request source distribution

```
* | select ip_to_province(remoteIp) as province ,  count(*) as c group by province
```

CCN Flow Log Analysis

Last updated：2024-08-12 16:05:21

Overview

Tencent Cloud Flow Logs (FL) provides a full-time, full-flow, and non-intrusive traffic collection service. It enables you to store and analyze the collected network traffic in real time for troubleshooting, compliance auditing, architecture optimization, and security detection.
You can create a flow log within the specified collection range (such as ENI, NAT Gateway, and cross-region CCN traffic) to collect inbound/outbound traffic within the range.

Prerequisites

You have collected Cloud Connect Network (CCN) flow logs to Cloud Log Service (CLS). For more information, see Creating Flow Logs.
If you have not yet collected FL to Cloud Log Service (CLS), you can use the Demo log topic provided for free by CLS to experience this feature. For directions, see Use Demo Log to Quickly Experience CLS.

Example

Using CLS to analyze a CCN flow log

FL is interconnected with CLS, so you can ship CCN flow log data to CLS in real time to further use the search and SQL analysis capabilities of CLS to meet your personalized real-time log analysis needs in different scenarios:
Push-button log shipping
Analyzing tens of billions of log data entries within seconds
Visualizing real-time logs on dashboards
Real-time alarm reporting in 1 minute

Log Field Description

FL of Cloud Connect Network Cross-Region Traffic
Other Types of FL

FL will record network flow filtered by the **Quintuple + Traffic Source Region + Traffic Target Region** rule in a specific capture window. This means that only FL that meets the rule in the capture window will be recorded as FL of Cloud Connect Network Cross-Region Traffic.

Quintuple + Traffic Source Region + Traffic Target Region

The quintuple is a collection containing five parts: the source IP address, source port, target IP address, target port, and transport layer protocol.

Traffic Source Region refers to the region where Cloud Connect Network cross-region traffic is sent.

Traffic Target Region refers to the region where Cloud Connect Network cross-region traffic arrives.

Capture Window

This is a period of continuous time during which CLS aggregates data and then publishes flow log records. The capture window is about 1 minute, and the push time is about 5 minutes.

Field	Data Type	Description
version	text	Flow log version.
region-id	text	The region where logs are recorded.
ccn-id	text	Unique CCN instance ID. To get the information of your CCN instance, contact us.
srcaddr	text	Source IP.
dstaddr	text	Destination IP.
srcport	text	Traffic source port. This field will take effect only for UDP/TCP protocols and will be displayed as "-" for other protocols.
dstport	long	Traffic destination port. This field will take effect only for UDP/TCP protocols and will be displayed as "-" for other protocols.
protocol	long	IANA protocol number of the traffic. For more information, see Assigned Internet Protocol Numbers.
srcregionid	text	Traffic source region.
dstregionid	text	Traffic destination region.
packets	long	Number of packets transferred in the capture window. This field will be displayed as "-" when `log-status` is `NODATA` .
bytes	long	Number of bytes transferred in the capture window. This field will be displayed as "-" when `log-status` is `NODATA` .
start	long	The timestamp when the first packet is received in the current capture window. If there are no packets in the capture window, it will be displayed as the start time of

		the capture window in Unix seconds.
end	long	The timestamp when the last packet is received in the current capture window. If there are no packets in the capture window, it will be displayed as the end time of the capture window in Unix seconds.
action	text	Operation associated with the traffic: ACCEPT: Cross-region traffic normally forwarded over CCN. REJECT: Cross-region traffic prevented from being forwarded due to traffic throttling.
log-status	text	Logging status of the flow log. Valid values: OK: Data is normally logged to the specified destination. NODATA: There was no inbound or outbound network flow in the capture window, in which case both the `packets` and `bytes` fields will be displayed as `-1`.

Flow logs record the network flow filtered by quintuple rules in the specified capture window.

Quintuple

That is a collection composed of the source IP address, source port, target IP address, target port, and transport layer protocol.

Capture Window

This is a period of continuous time during which CLS aggregates data and then publishes flow log records. The capture window is about 5 minute, and the push time is about 5 minutes.

Field	Description
version	Flow log version.
account-id	Account AppID of the flow logs.
interface-id	ENI ID.
srcaddr	Source IP address.
dstaddr	Target IP address.
srcport	The source port of traffic. When the traffic is the ICMP protocol, this field represents the ICMP ID.
dstport	The target port of traffic. When the traffic is ICMP protocol, this field represents a combination of ICMP type (high 8 bits) and code (low 8 bits).
protocol	The IANA protocol number of traffic. For more information, go to the assigned Internet Protocol number.
packets	This shows the number of data packets transmitted in the capture window.
bytes	This shows the bytes transmitted in the capture window.

start	This shows the start time of the capture window, in the Unix second format.
end	This shows the end time of the capture window, in the Unix second format.
action	Operations associated with the traffic: ACCEPT: It shows the traffic allowed to be recorded by the security group or network ACL. REJECT: It shows the traffic not allowed to be recorded by the security group or network ACL.
log-status	Log record status of the flow log: OK: It indicates that data is successfully recorded to the specified target. NODATA: It indicates that there is no incoming or outgoing network traffic in the capture window. In this case, packets and bytes fields will be displayed as -1. SKIPDATA: It indicates that some flow log records were skipped in the capture window. This may be caused by internal capacity limits or internal errors.

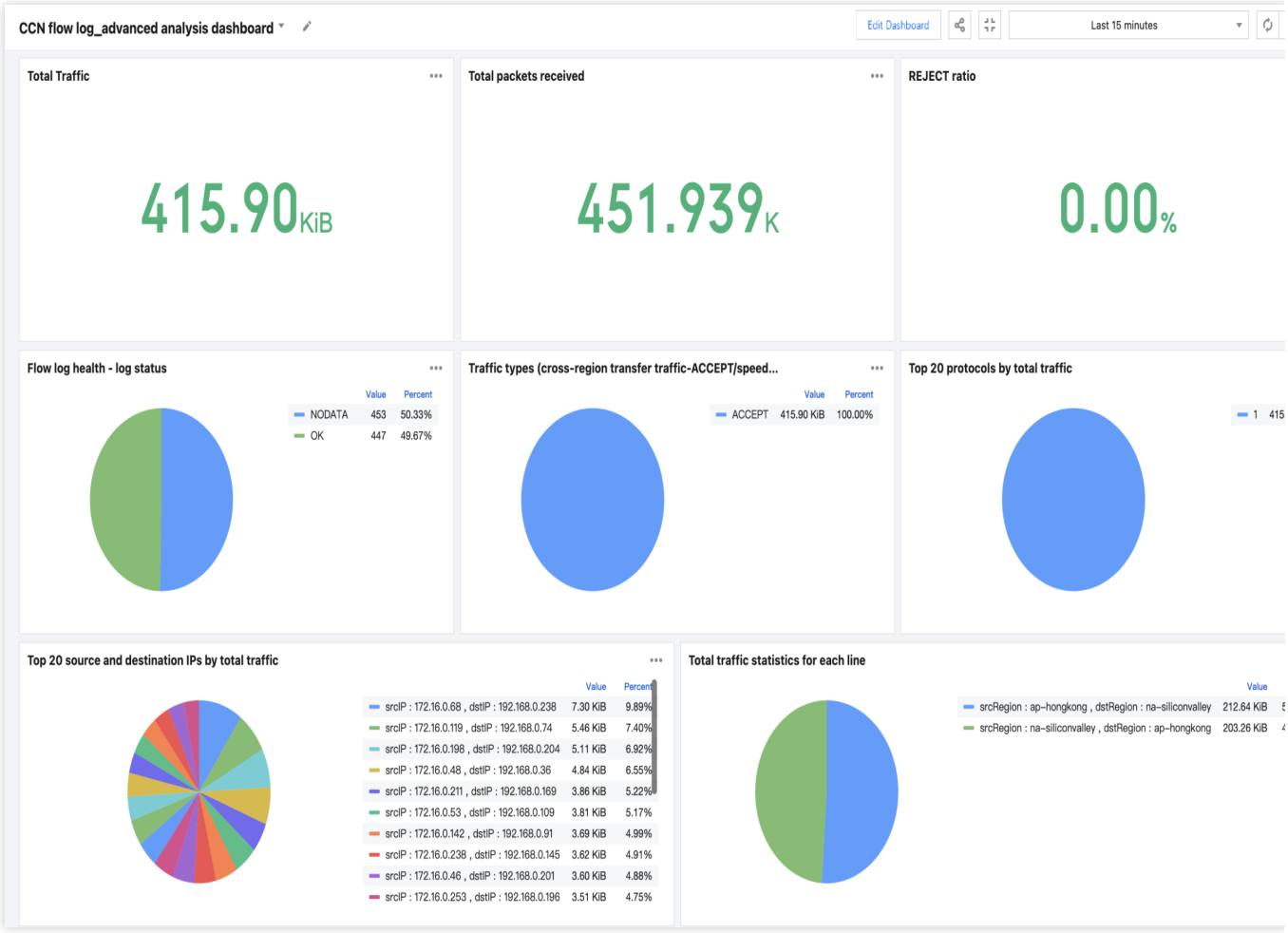
Preset Dashboard

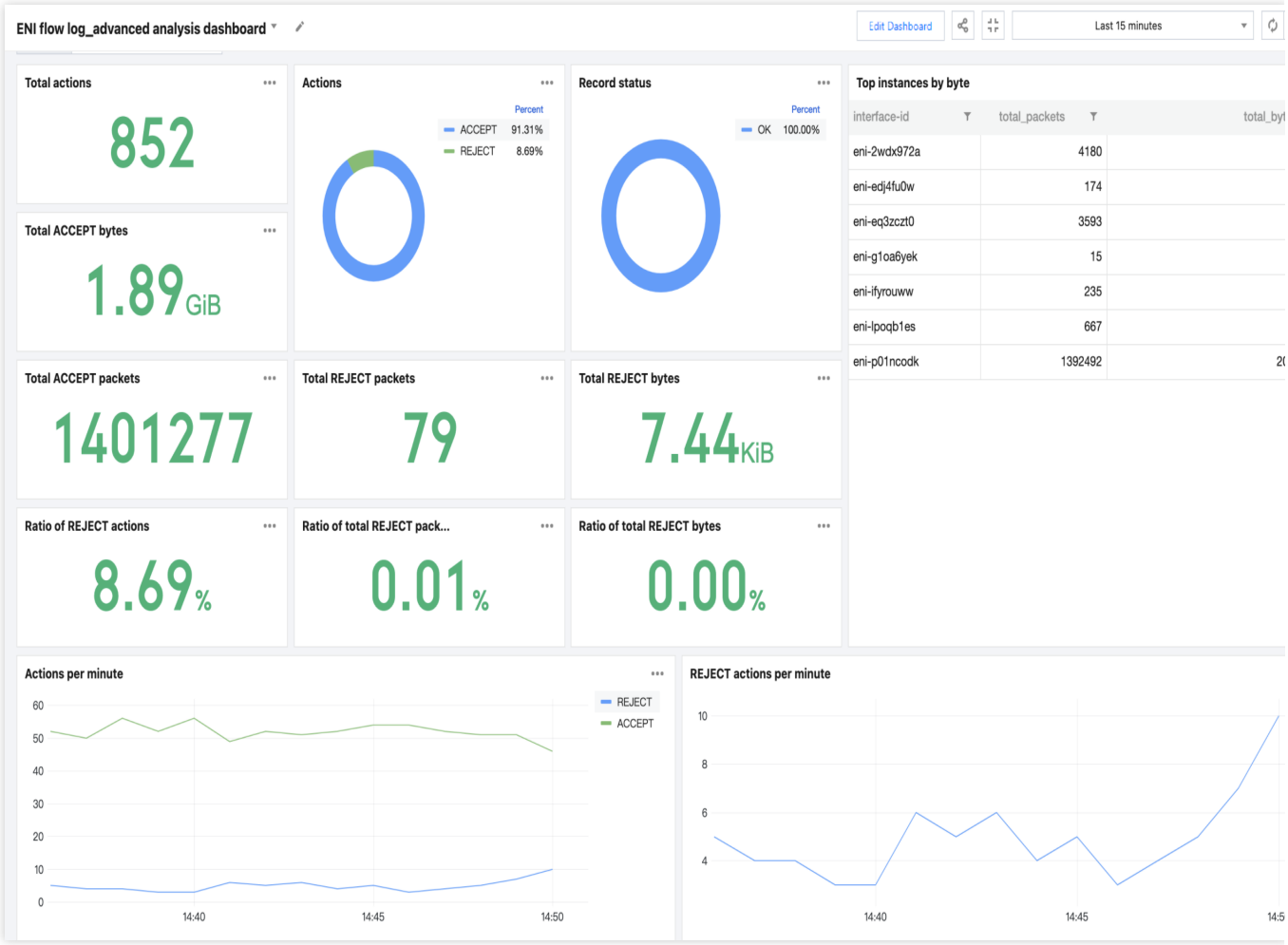
CLS has preset common Cloud Connect Network and Elastic Network Interface flow log statistics as dashboards. You can quickly know the current network status through these dashboards.

CCN :https://console.intl.cloud.tencent.com/cls/dashboard/d?templateId=flow-log-ccn-analysis-dashboard

ENI :https://console.intl.cloud.tencent.com/cls/dashboard/d?templateId=flow-log-eni-analysis-dashboard

Click **Edit Dashboard** in the upper right corner of the dashboard to edit based on the preset dashboard.


CCN flow log_advanced analysis dashboard
Edit Dashboard
Last 15 minutes
Total Traffic
415.90KiB
Total packets received
451.939K
REJECT ratio
0.00%
Flow log health - log status
Value Percent
NODATA 453 50.33%
OK 447 49.67%
Traffic types (cross-region transfer traffic-ACCEPT/speed...
Value Percent
ACCEPT 415.90 KiB 100.00%
Top 20 protocols by total traffic
Top 20 source and destination IPs by total traffic
Value Percent
srcIP : 172.16.0.68 , dstIP : 192.168.0.238 7.30 KiB 9.89%
srcIP : 172.16.0.119 , dstIP : 192.168.0.74 5.46 KiB 7.40%
srcIP : 172.16.0.198 , dstIP : 192.168.0.204 5.11 KiB 6.92%
srcIP : 172.16.0.48 , dstIP : 192.168.0.36 4.84 KiB 6.55%
srcIP : 172.16.0.211 , dstIP : 192.168.0.169 3.86 KiB 5.22%
srcIP : 172.16.0.53 , dstIP : 192.168.0.109 3.81 KiB 5.17%
srcIP : 172.16.0.142 , dstIP : 192.168.0.91 3.69 KiB 4.99%
srcIP : 172.16.0.238 , dstIP : 192.168.0.145 3.62 KiB 4.91%
srcIP : 172.16.0.46 , dstIP : 192.168.0.201 3.60 KiB 4.88%
srcIP : 172.16.0.253 , dstIP : 192.168.0.196 3.51 KiB 4.75%
Total traffic statistics for each line
Value
srcRegion : ap-hongkong , dstRegion : na-siliconvalley 212.64 KiB
srcRegion : na-siliconvalley , dstRegion : ap-hongkong 203.26 KiB



Configure Alarm

For example, if the bandwidth cap of 100 Mbps is set for the Cloud Connect Network Hong Kong (China) - Silicon Valley line, you need to monitor the current bandwidth usage. If the bandwidth is greater than or equal to 95 Mbps for ten consecutive minutes, an alarm will be triggered to adjust the bandwidth cap when necessary.

1. Go to the Create Alarm Policy page. For directions, see Configure Alarm Policy.

2. Enter the following statement in the execution statement, select a time range of 1 minute, and count the bandwidth usage of the Hong Kong (China) - Silicon Valley line in the past one minute. The bandwidth in the result of this execution statement is the one-minute bandwidth in Mbps.

```
log-status:OK AND srcregionid:ap-hongkong AND dstregionid:na-siliconvalley | select
```

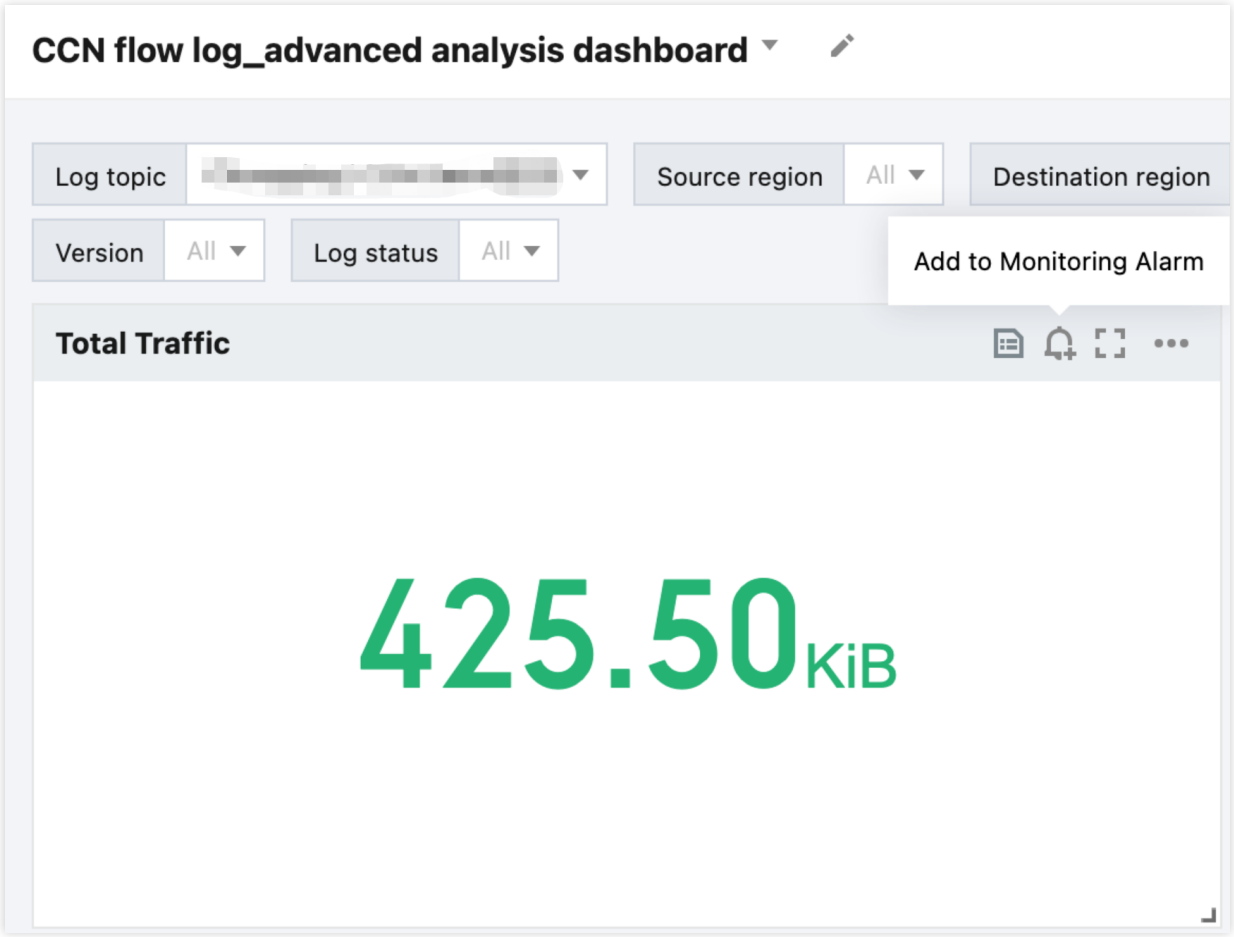
3. The trigger condition is as follows: If the bandwidth is greater than or equal to 95 Mbps, the alarm condition is met.

```
$1.bandwidth > 95
```

4. Execution cycle: The system is executed every minute at a fixed frequency.

5. Alarm notification - alarm frequency: An alarm is always triggered if the trigger condition is met for 10 consecutive cycles. That is, if the bandwidth is greater than or equal to 95Mbps for 10 consecutive minutes, an alarm will be triggered.

For charts in the preset dashboard, you can click **Add to Monitoring and Alarming** in the upper right corner to add the metrics in the chart to the alarm policy.



TKE Event Log Analysis

Last updated：2024-01-20 17:28:40

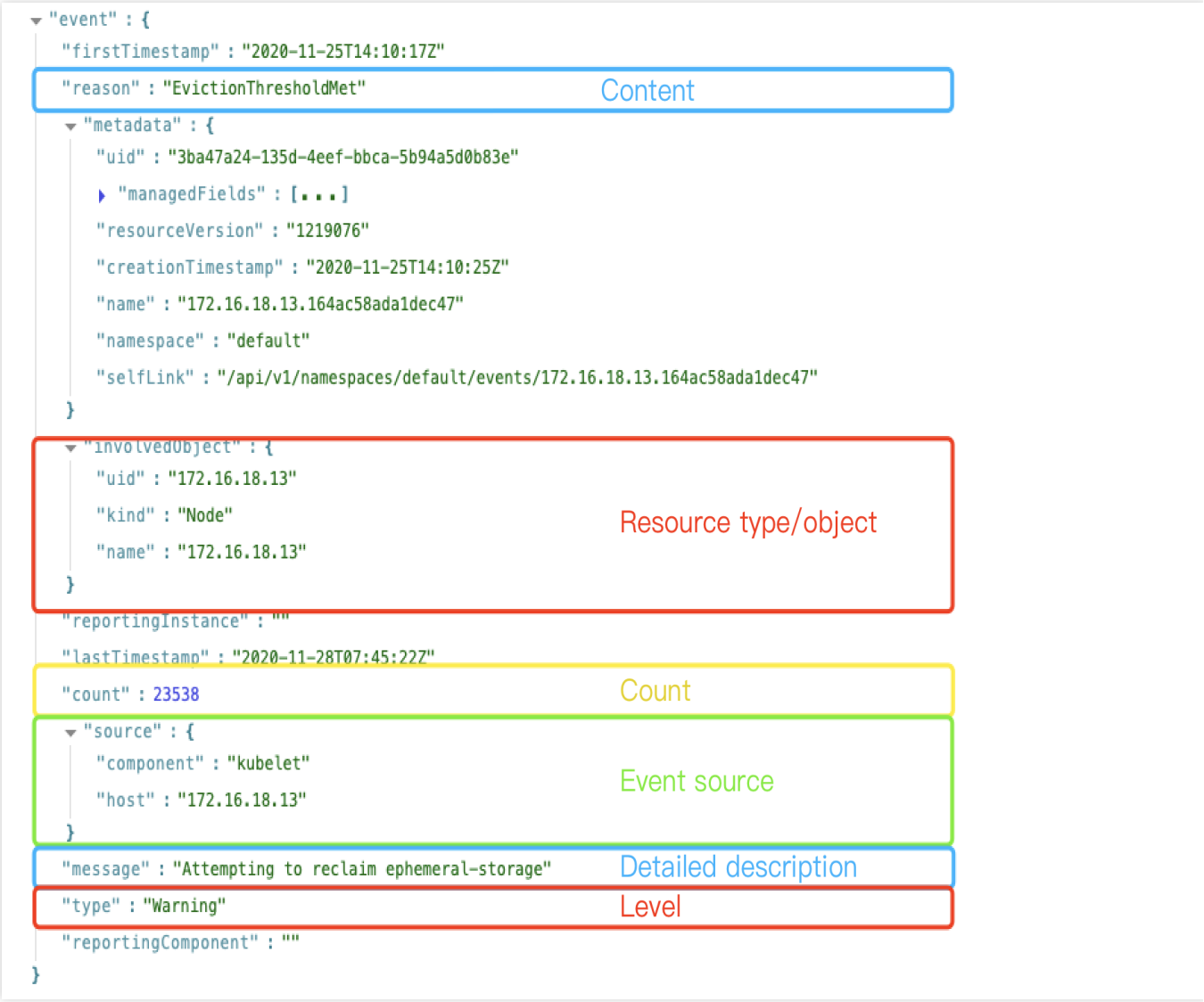
Overview

Situations in the cluster emerge one after another and are unpredictable, such as abnormal node status and pod restarts. If these situations cannot be perceived in the first place, users will miss the best time to deal with them. It's often too late to find out when the problem worsens and affects businesses.
Event logs record comprehensive information about cluster status changes, helping users find and troubleshoot problems in the first place.

Event Log Definition

An event log is one of many resource objects in Kubernetes and is usually used to record status changes within a cluster, ranging from cluster node exceptions to pod startup and scheduling success. You can use the 'kubectl describe' command to view the event log information of resources.

Event Log Fields



Level (`type`): Currently only the Normal and Warning levels are supported. If necessary, you can customize a level.
Resource type/object (`involvedobject`): Objects involved in the event, such as Pod, Deployment, and Node.
Event source (`source`): Component that reports the event, such as Scheduler and Kubelet.
Content (`reason`): Brief description of the current event. Generally, an enumerated value is used. This field is used within the program.
Detailed description (`message`): Detailed description of the current event.
Number of occurrences (`count`): Number of times the event occurs

Using Event Logs for Troubleshooting

CLS provides a one-stop service for Kubernetes event logs, including collection, storage, search, and analysis capabilities. You only need to enable the cluster event log feature with a few clicks to obtain a visual event log analysis dashboard out of the box. With visual charts, you can easily solve most common Ops problems via the console.

Prerequisites

You have purchased the Tencent Kubernetes Engine (TKE) service and enabled the cluster event log feature. For more information, see Event Storage.

Scenario 1: An exception occurred on a node, and you need to locate the cause

1. Log in to the TKE console.
2. On the left sidebar, click **Log Management** > **Event Log**.
3. On the **Event Search** page, click **Event Overview** tab and enter the exception node name as the filter item.

The query result is displayed.

Check the exception event trend and top exception events.

Starting from `2020-11-25` , the node `172.16.18.13` was exceptional due to insufficient disk space. Then Kubelet began to drain pods on the node to repossess the node's disk space.

Scenario 2: A node triggered expansion, and you need to backtrack the expansion process to determine the cause

For clusters with node pool**auto scaling** enabled, the Cluster Autoscaler (CA) component will automatically increase or decrease the number of nodes in the cluster based on the actual load. If the nodes in the cluster are automatically scaled out, users can trace the entire scaling process through event search.

1. Log in to the TKE console.
2. On the left sidebar, click **Log Management** > **Event Log**.
3. On the **Event Search** page, click the **Global Search** tab, and enter the following search command:

```
event.source.component : "cluster-autoscaler"
```

4. In the **Hidden Field** on the left side, select `event.reason` , `event.message` , `event.involvedObject.name` , and `event.involvedObject.name` to display. Sort the query results in reverse order by `Log Time` .

According to the event flow, you can find that the node scaling occurred around `2020-11-25 20:35:45` and was triggered by three NGINX pods (nginx-5dbf784b68-tq8rd, nginx-5dbf784b68-fpvbx, and nginx-5dbf784b68-v9jv5). After three nodes were scaled out, the subsequent scaling was not triggered because the number of nodes in the node pool reached the upper limit.

TKE Audit Log Analysis

Last updated：2024-01-20 17:28:40

Overview

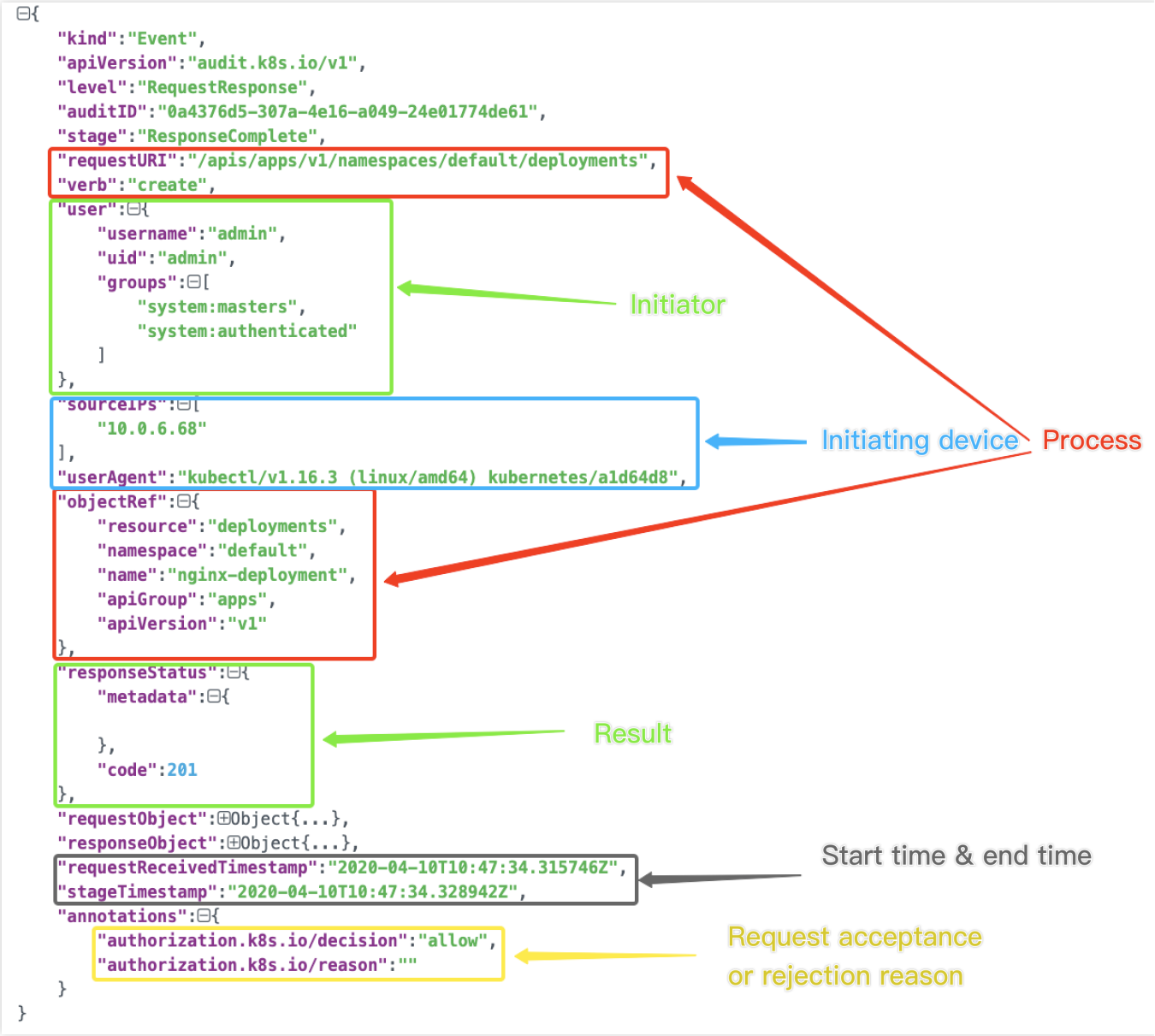
In the past, it was not easy for users to troubleshoot Tencent Kubernetes Engine (TKE) problems. A Kubernetes cluster in a production environment is usually a very complex system. The bottom layer accommodates a variety of heterogeneous hosts, networks, storage devices, and other cloud infrastructure. The upper layer carries a large amount of application load. In the middle, various native components (e.g., Scheduler and Kubelet) and third-party components (e.g., various operators) run to manage and schedule infrastructure and applications. In addition, personnel with different roles frequently deploy applications, add nodes, and perform other operations on the cluster. Therefore, in the cluster OPS scenario, users often encounter the following problems:
An application in the cluster was deleted. Who did it?
The load of apiserver suddenly becomes high and a large number of access failures occur. What happened in the cluster?
Cluster nodes are cordoned off. Who did it and when did it happen?
Cloud Log Service (CLS) is now interconnected with Tencent Kubernetes Engine (TKE). Kubernetes audit logs will be an important tool to help users quickly solve the above mentioned problems.

Audit Log Definition

In Kubernetes, all cluster status queries and changes are implemented by sending requests to the apiserver. Audit logs are structured logs with configurable policies generated by Kube-apiserver and record apiserver access events. You can view and analyze audit logs to trace cluster status changes, understand the health of the cluster, troubleshoot exceptions, and discover potential security and performance risks of the cluster, and so on.

Audit Log Fields

Each audit log is a structured record in JSON format, and includes three parts: metadata, requestObject, and responseObject. The metadata is a required part (it contains the request context information, such as who initiated the request, where it was initiated, and the accessed URI). requestObject and responseObject are optional, depending on the audit level.



Using Audit Logs for Troubleshooting

CLS provides a one-stop service for Kubernetes audit logs, including collection, storage, search, and analysis capabilities. You only need to enable the cluster audit log feature with a few clicks to obtain a visual audit log analysis dashboard out of the box. With visual charts, you can easily solve most common OPS problems via the console.

Prerequisites

You have purchased TKE and enabled the cluster audit log feature. For more information, please see Directions.

Scenario 1: An application in the cluster was deleted. Who did it?

1. Log in to the TKE console.

2. On the left sidebar, choose **Cluster OPS** > **Auditing Search**.
3. On the **Auditing Search** page, click the **K8s Object Operation Overview** tab and specify the operation type as **delete** and specify the resource object nginx.
The following figure shows an example of the query result.
As shown in the above figure, account `10001****7138` deleted the NGINX application. You can use the account ID to query the detailed information about this account in **CAM** > **User List**.

Scenario 2: The load of apiserver suddenly becomes high and a large number of access failures occur. What happened in the cluster?

1. Log in to the TKE console.
2. On the left sidebar, choose **Cluster OPS** > **Auditing Search**.
3. On the **Auditing Search** page, click the **Aggregation Search** tab. The tab page displays the trend of apiserver access in multiple dimensions such as user, operation type, and status code.
As shown in the above figures, you can find that user `tke-kube-state-metrics` has the maximum number of accesses; in the Trend of Operation Type Distribution chart, most operations are list operations; and in the Trend of Status Code Distribution chart, most return codes are 403. Then use the `tke-kube-state-metrics` keyword to search for logs.
Combining with business logs, you can find that `tke-kube-state-metrics` frequently sends requests to the apiserver due to RBAC permission issues, resulting in a sharp increase in apiserver access.

Scenario 3: Cluster nodes are cordoned off. Who did it and when did it happen?

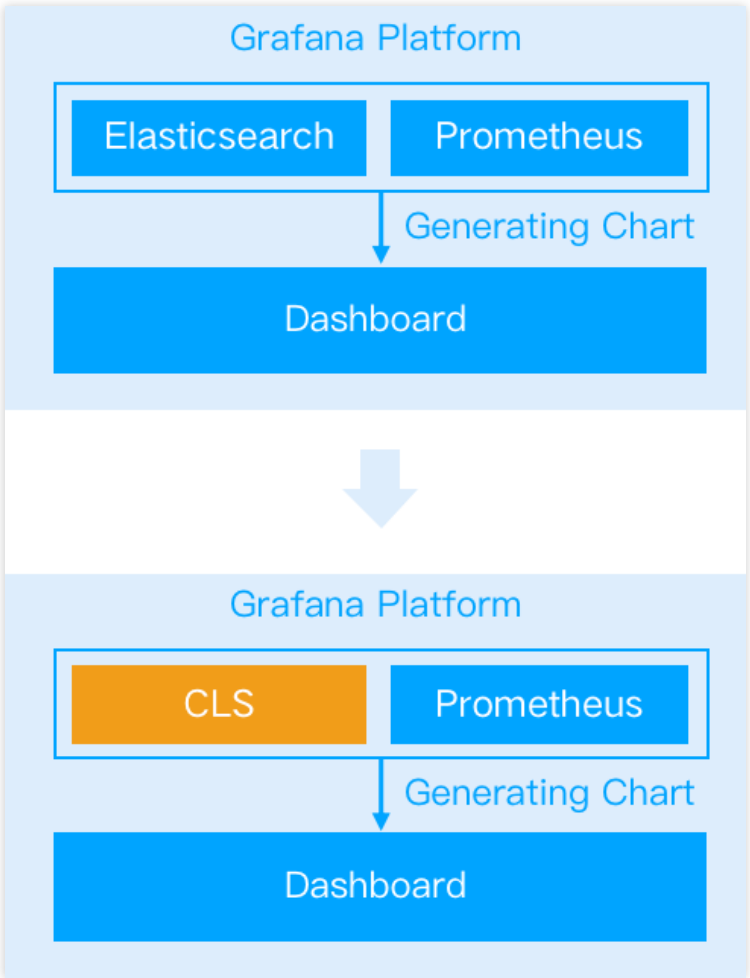
1. Log in to the TKE console.
2. On the left sidebar, choose **Cluster OPS** > **Auditing Search**.
3. On the **Auditing Search** page, click the **Node Operation Overview** tab, and enter the name of the cordoned node on the tab page.
The following figure shows an example of the query result.
As shown in the above figure, account `10001****7138` cordoned off the node `172.16.18.13` at `2020-11-30T06:22:18` .

Dashboard
Migrating the ES data source of Grafana to the CLS data source

Last updated：2024-01-20 17:28:40

Background

In CLS use cases, it is very common to migrate the data source from other log tools to CLS. If you use ES as the data source and Grafana as the visual monitoring tool, after you migrate the data source to CLS, various dashboard resources and Ops tools and platforms created based on Grafana will become useless. To avoid building this system again, CLS needs to be connected to Grafana to replace the ES data source.



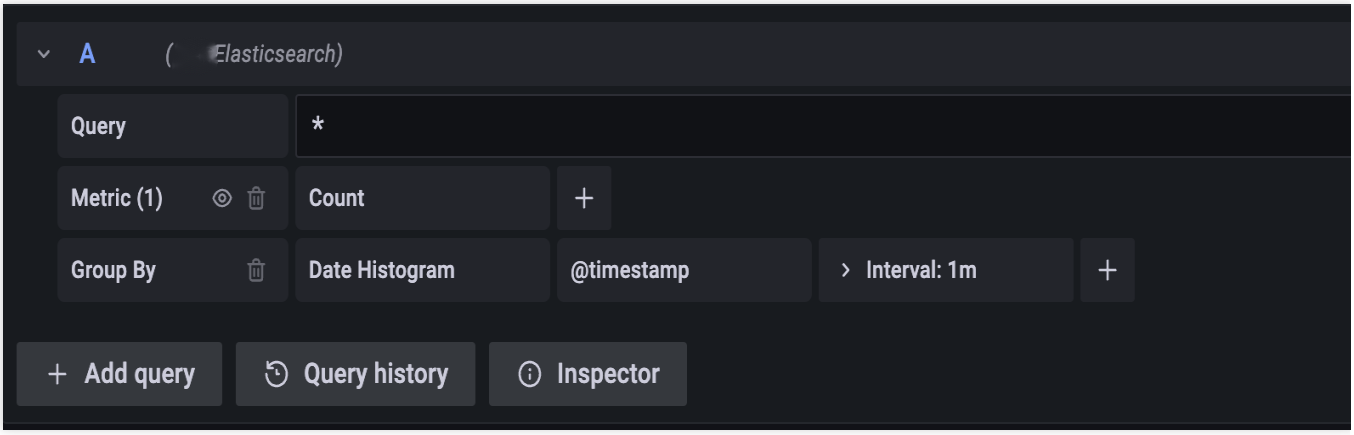
Installing Tencent Cloud Monitor

The Tencent Cloud Monitor plugin (CLS data source) is maintained by the CLS team and has been officially signed by Grafana. You can quickly install it on the Grafana settings page.
For detailed directions, see CLS Connection to Grafana.

Replacing ES with CLS as Data Source

Comparing data source configuration sections

ES data source plugin: The query statement UI consists of the **query input box** at the top and **auxiliary input features** in other places. You can enter Lucene statements in the **Query** input box to filter logs. In the auxiliary input section, you can click buttons and enter values to generate DSL content for data aggregation, which is similar to SQL statements in CLS.

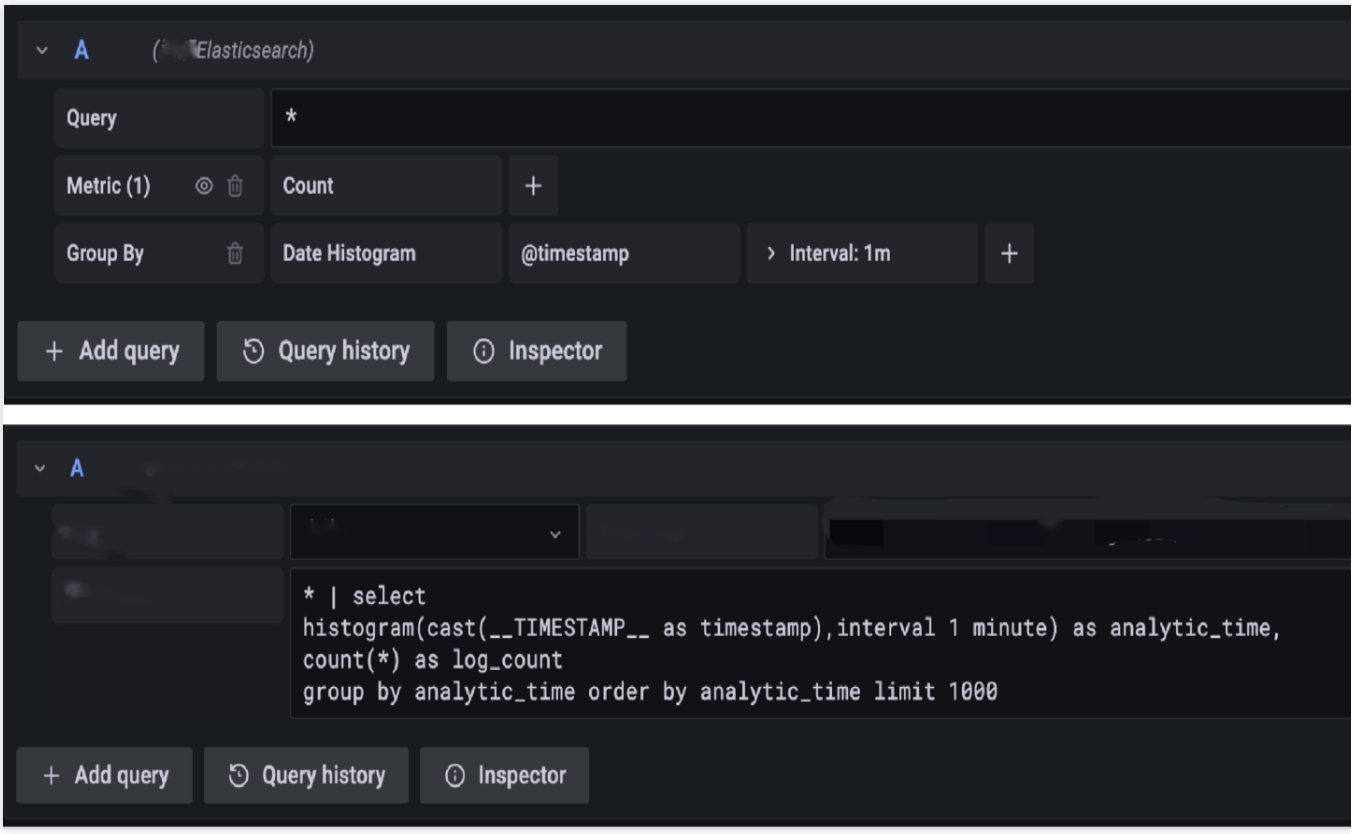


CLS data source plugin: The query statement UI consists of the **region and log topic selection** section and **search and analysis statement** section. They are used to quickly switch the log topic and enter CLS query statements respectively.
A CLS query statement consists of a Lucene statement and a SQL statement, which are separated by a pipe symbol "|". Here, the Lucene statement is the same as the content in the **Query** input box for ES. The entered SQL statement supports not only standard SQL syntax but also diversified SQL functions to offer the same features as in the auxiliary input section for ES. For more information, see Overview and Syntax Rules.

Directions

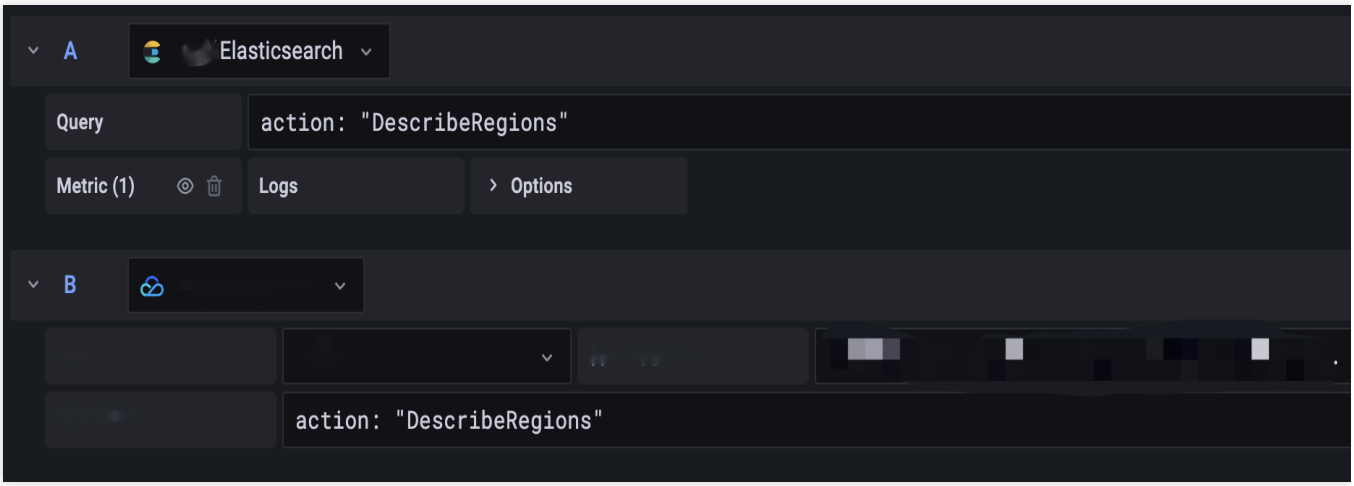
Counting logs

To draw a histogram of the number of logs changing over time, in the ES data source, select **Count** for **Metric** and **Data Histogram** for **Group By**; in the CLS source data, you can combine the histogram and the aggregate function `Count` to write a search statement. You can also use other general aggregate functions such as `Max` , `Min` , and `Distinct` in the same way by directly replacing the `Count` function.



Viewing raw logs

To directly view logs meeting the search criteria, select **Logs** for **Metric** in the ES data source, or enter the corresponding Lucene statement in the CLS data source. The statements entered are as compared below:



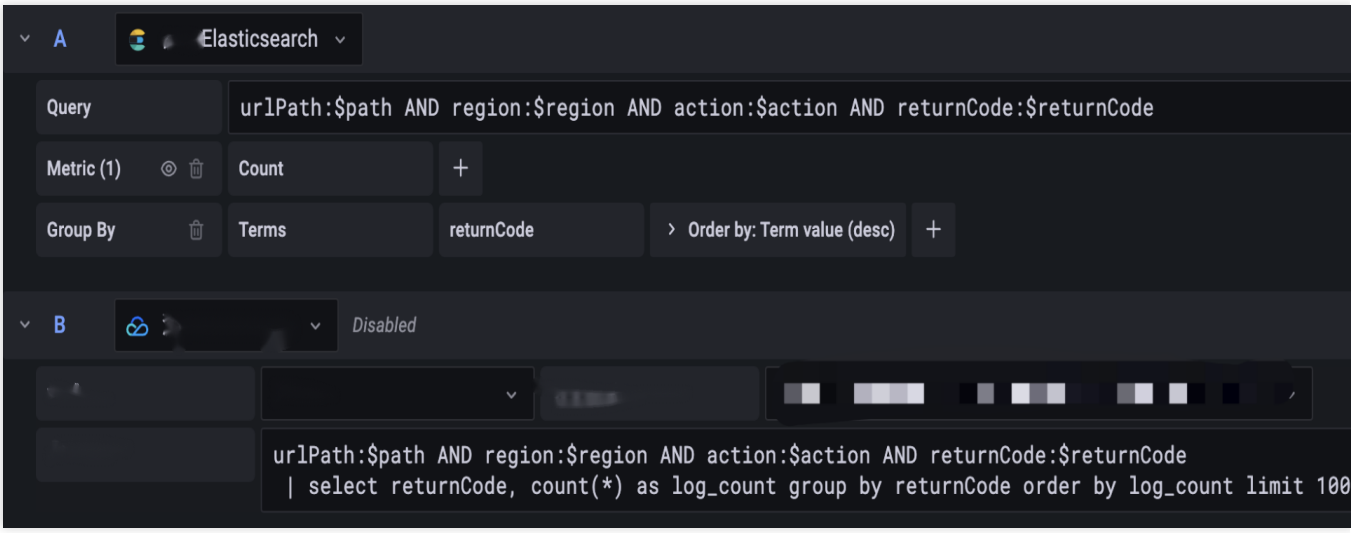
Display effect:

Aggregate statistics - error code proportion

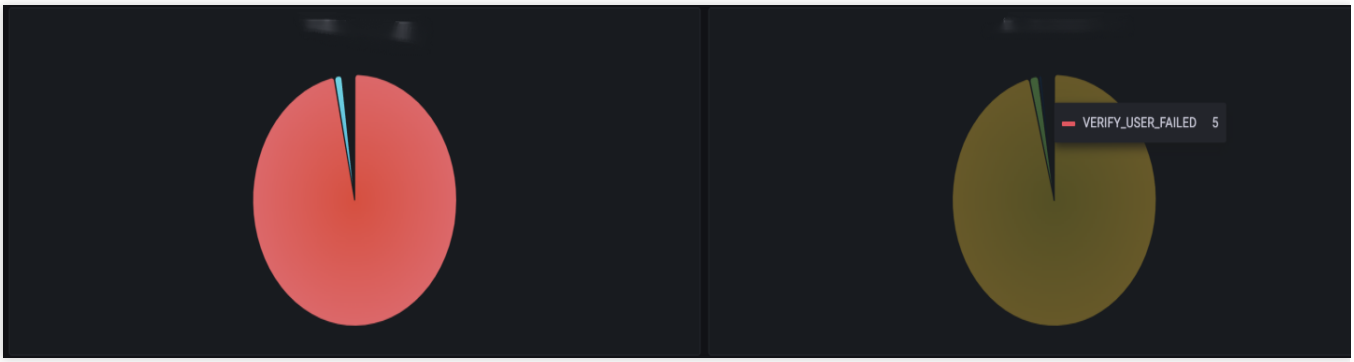
You can aggregate error codes and display the numbers of logs with each error code. As can be seen here, the statement contains the `$path` variable. The CLS data source plugin adapts to the variable capabilities of Grafana for direct use.

Note:

To draw a pie chart, select **ValueOptions-AllValues** in the chart options on the right.



Display effect:

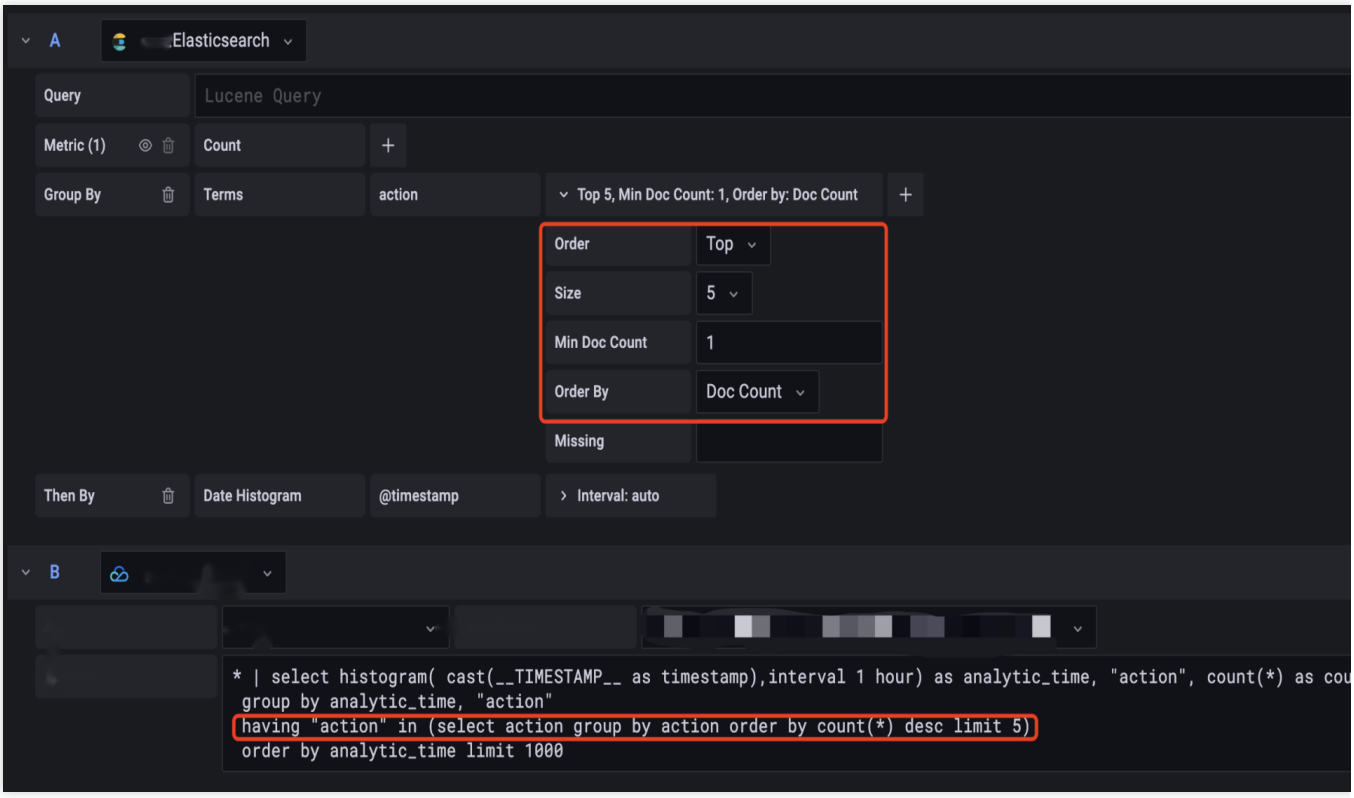


Aggregate statistics - changes in numbers of top five requests

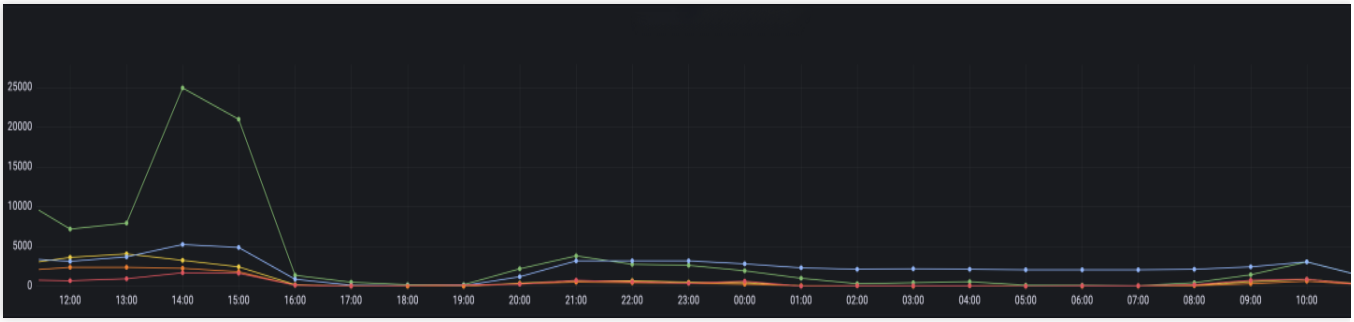
In the ES data source plugin, you can enter a `Size` value for the **Group By** aggregate option to select the top N most frequent values and aggregate them.

In the CLS data source, you can write a `having` SQL statement nesting subqueries to achieve the same purpose.

```
*|select histogram( cast(__TIMESTAMP__ astimestamp),interval1hour)as
analytic_time,"action",count(*)as countgroupby
analytic_time,"action"having"action"in(selectactiongroupbyactionorderbycount(*)
desclimit5)orderby analytic_time limit1000
```



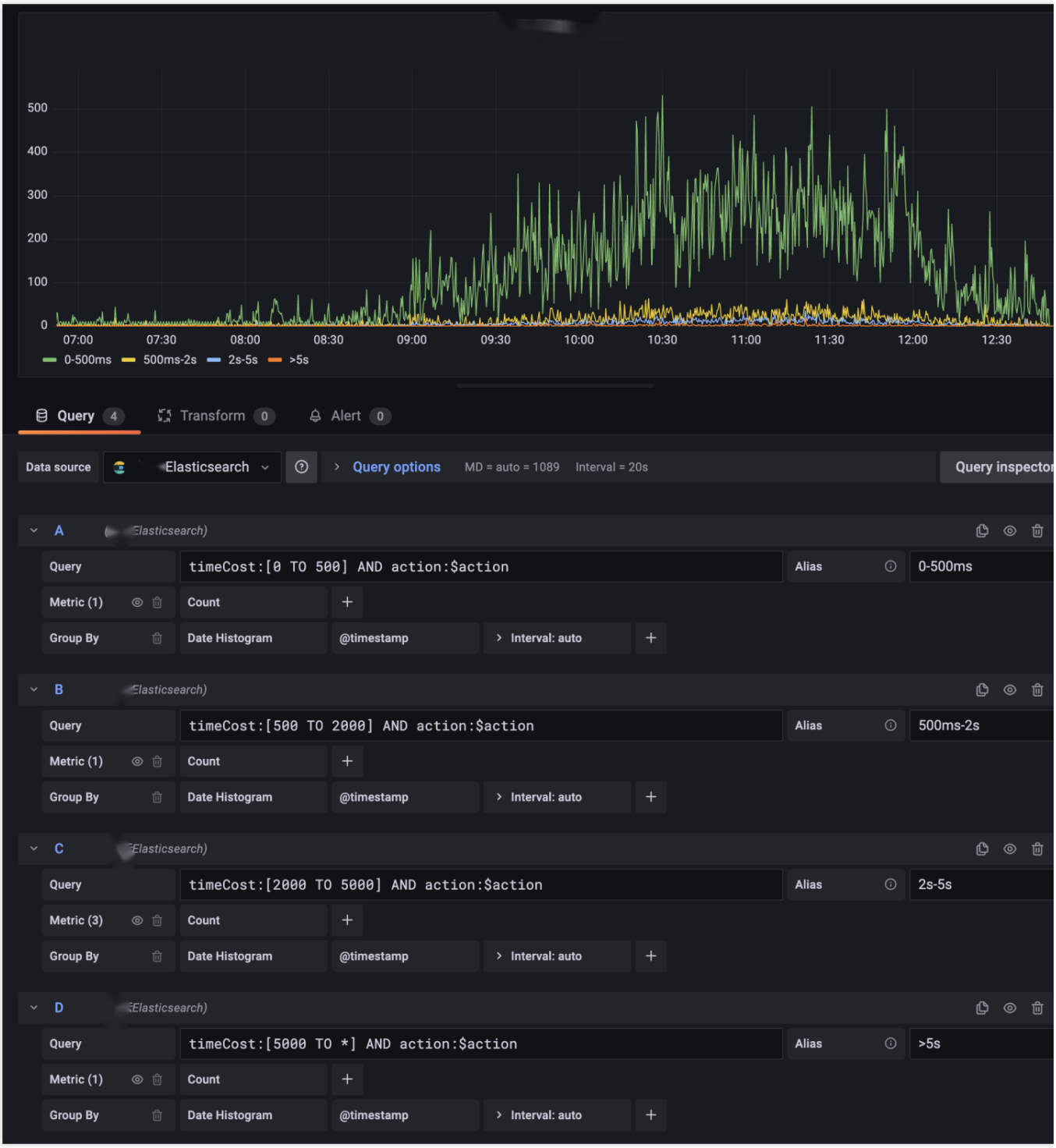
The chart of query results displays five curves as shown below:



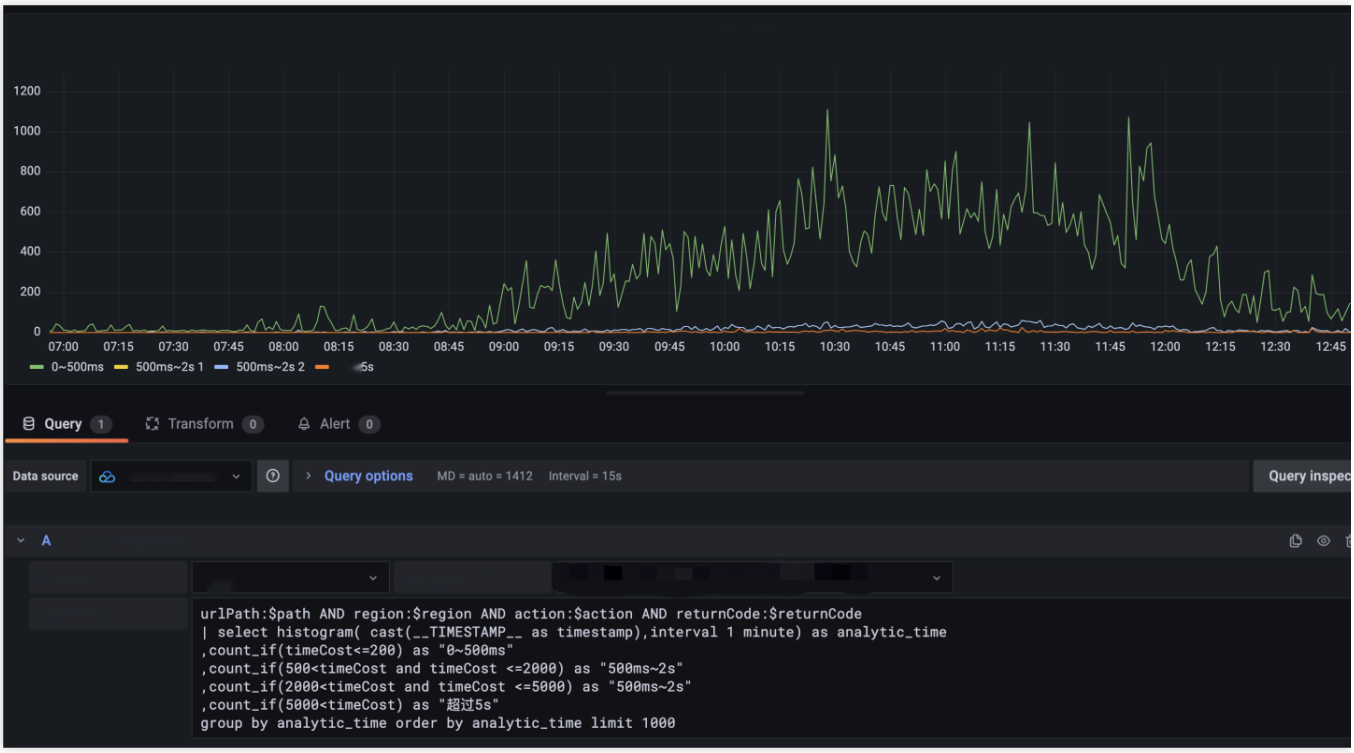
By combining the above statements, you can meet the needs in most search and analysis scenarios.

Ranged statistics of API call time

In the dashboard of the ES data source, there is an example with complicated configuration items but a wide applicability, that is, to draw a chart of numbers of requests distributed in different time ranges.
In this example, the numbers of requests with an API call time of 0–500 ms, 500 ms–2s, 2–5s, and above 5s are presented respectively.



Accordingly, after migrating the data source to CLS, you can also use multiple statements to draw a similar chart. However, CLS has more powerful SQL capabilities, so you can merge relevant statistics collection statements into one SQL statement:




```
urlPath:$path AND region:$region AND action:$action AND returnCode:$returnCode
| select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 minute) as
analytic_time ,count_if(timeCost<=200) as "0~500ms" ,count_if(500<timeCost and
timeCost <=2000) as "500ms~2s" ,count_if(2000<timeCost and timeCost <=5000) as
"2s~5s" ,count_if(5000<timeCost) as "Above 5s" group by analytic_time order by
analytic_time limit 1000
```

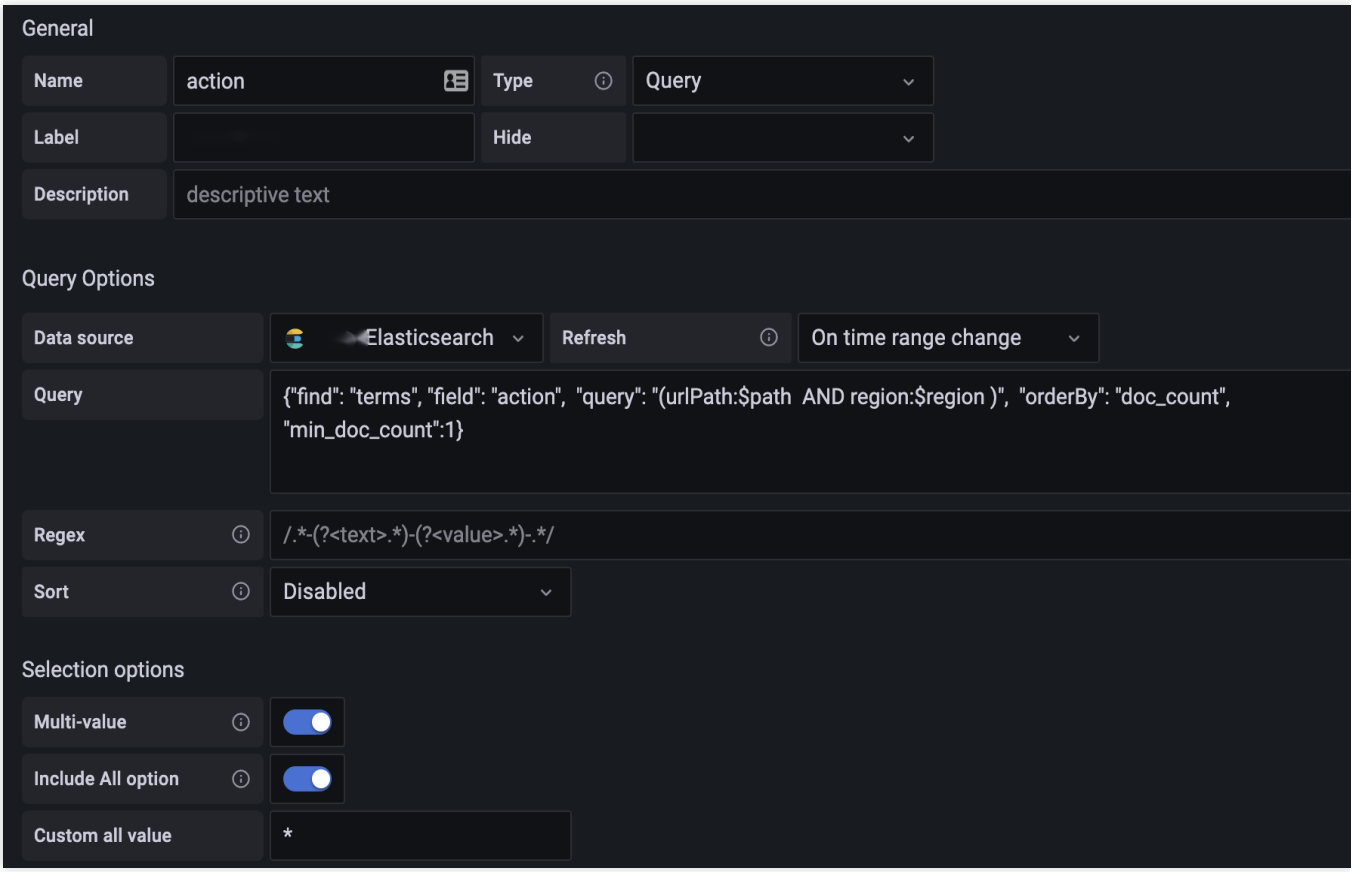
In similar scenarios, you can write a statement by using the estimation function `approx_percentile` to analyze the consumed time.


```
urlPath:$path AND region:$region AND action:$action AND returnCode:$returnCode
| select time_series(__TIMESTAMP__, '$__interval', '%Y-%m-%dT%H:%i:%s+08:00',
'0') as time ,avg(timeCost) as avg ,approx_percentile(timeCost, 0.50) as P50
,approx_percentile(timeCost, 0.90) as P90 ,approx_percentile(timeCost, 0.95) as
P95 group by time order by time limit 10000
```

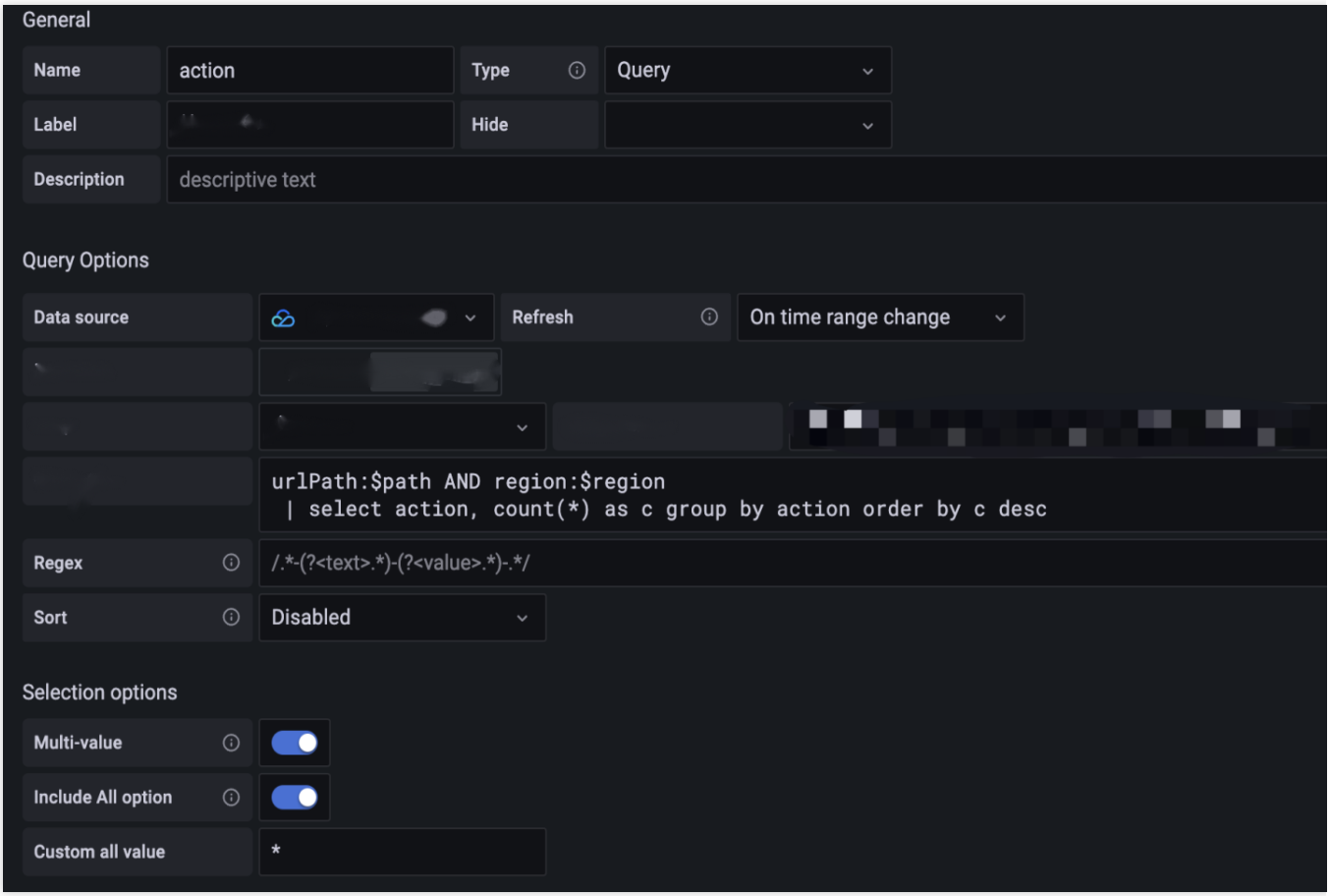
Template variable capabilities

The Grafana variable feature is used in all of the above examples to different degrees. Grafana has diverse variable types. For constant and textbox types, they are completely the same for different data sources and don't require additional configuration for migration. This section describes how to migrate variables of query type.

`$action` **variable for ES**: It indicates the API category and is described in DSL in the ES data source. The following syntax is to find the content matching the query condition `urlPath:$path AND region:$region`, select the `action` field, and sort API categories by the number of occurrences.



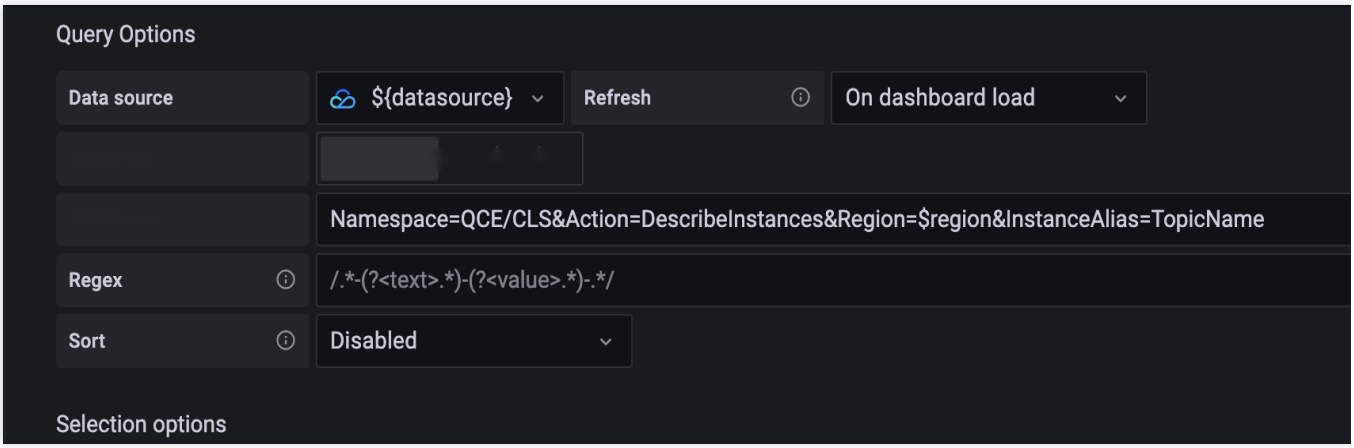
`$action` **variable for CLS**: For this variable, the CLS data source has the same user experience and input behavior in chart editing as ES. You can select CLS as the service type, select the corresponding log topic, and enter a SQL statement to achieve the same effect.



In addition to using search statements of CLS to query variables, you can also use the resource query feature of CM to display service resources in Tencent Cloud as a list. For more information, see Template Variables.

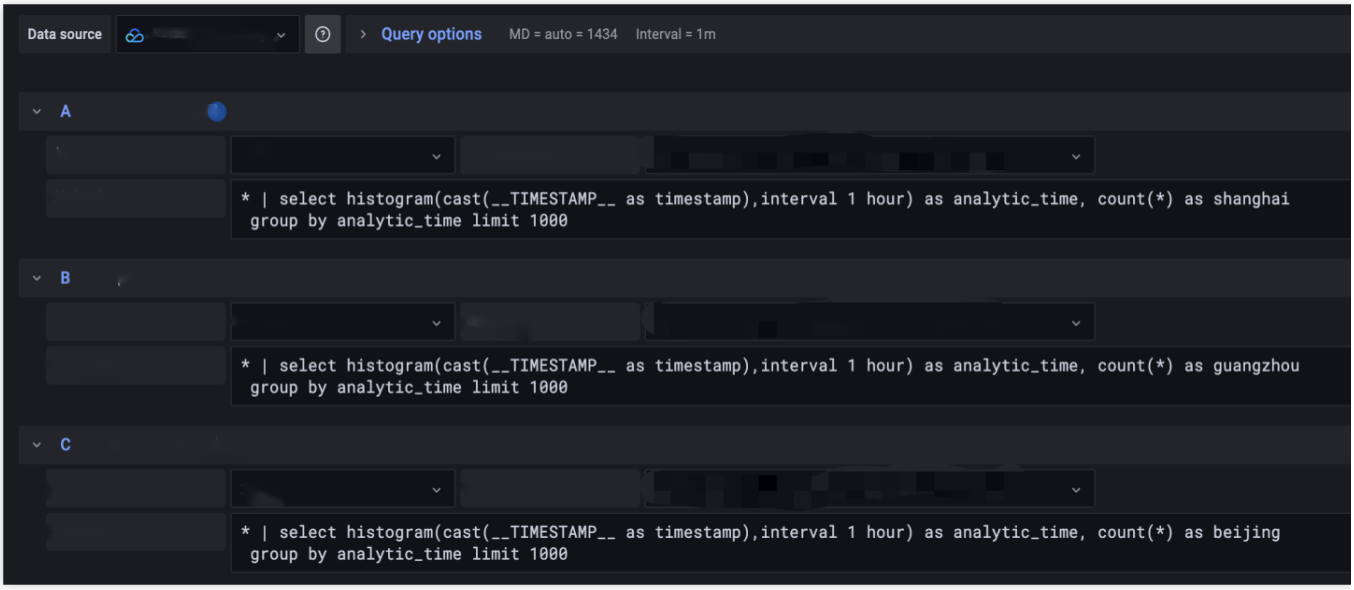
```
Namespace=QCE/CLS&Action=DescribeInstances&Region=$region&display=${TopicName}/
${TopicId}
```

Query the log topic list:

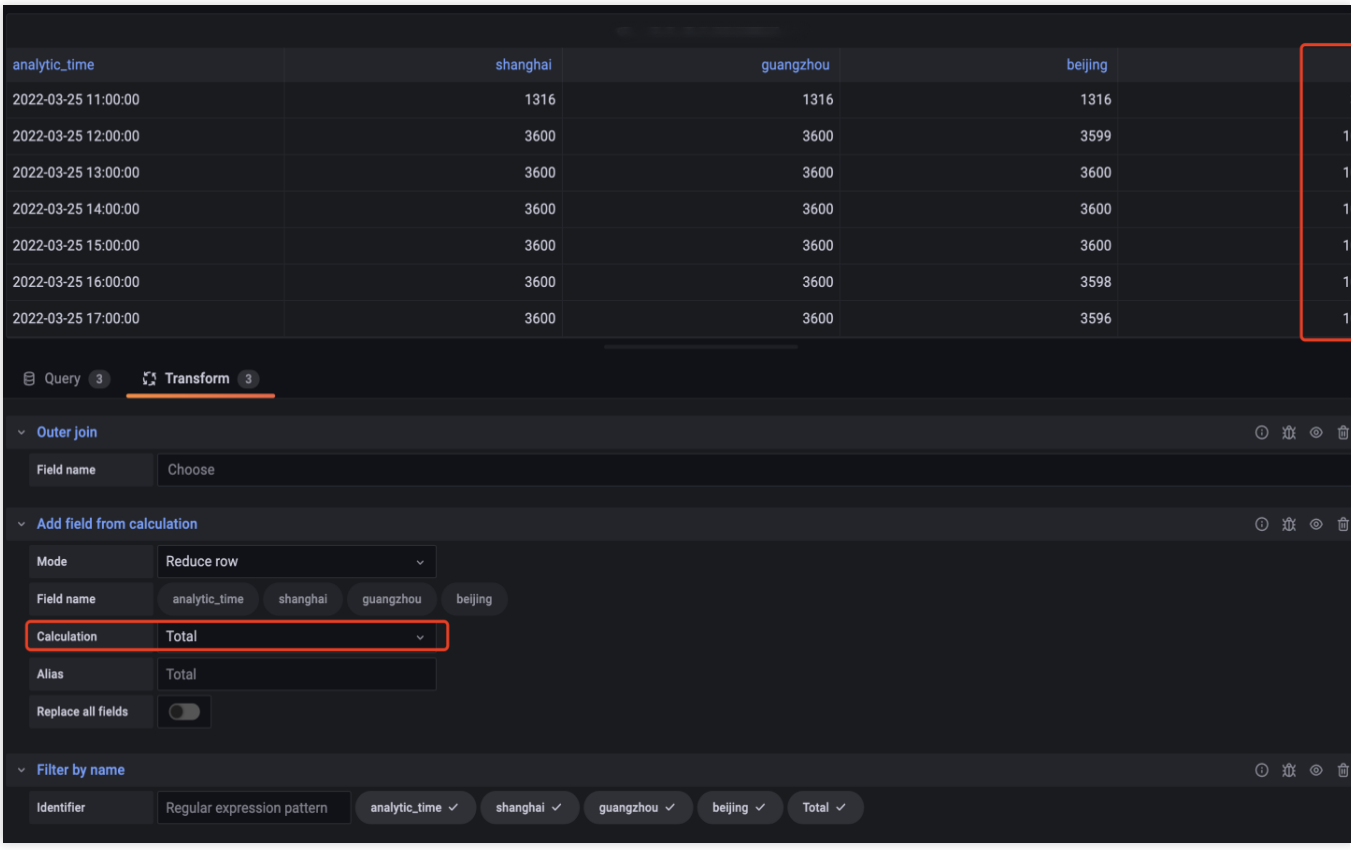


Merging data content of requests from different regions

In the original implementation, if you store all data in the same ES instance, after CLS is used, you may want to merge the content of those log tops into the same chart.
For three statements querying logs from different regions:



You can calculate the total numbers in the **Transform** module and select an appropriate chart to display them.



Summary

You can repeat the above migration steps to completely convert an existing ES data source dashboard into a CLS one.

By migrating the data source from ES to CLS, you can continue to use the accumulated visual resources after migrating your business from self-built ELK to CLS.

A converted dashboard not only has all capabilities of the ES data source, but also supports other capabilities of the CLS data source plugin, such as CM template variables, to better integrate with the Tencent Cloud ecosystem.

Monitoring Alarm
Setting Alarm Trigger Conditions by Time Period

Last updated：2024-01-20 17:28:40

Overview

For business nature reasons, different alarm trigger conditions need to be set for different time periods. For example, an alarm should be triggered when the number of logs containing errors exceeds 100 during work hours (9:00 AM to 6:00 PM) or when this number exceeds 10 during off hours (7:00 PM to 8:00 AM).

Configuration method

When configuring an alarm policy, enter the following query statements and trigger conditions:

Query statement 1:

```
error | select count(*) as error_count
```

Count logs containing errors

Query statement 2:

```
* | select hour(now()) as hour limit 1
```

Use date and time functions to get the alarm hour, i.e., at which hour the alarm is triggered.

Trigger conditions:

```
($1.error_count>100 && $2.hour>=9 && $2.hour<=18 ) || ($1.error_count>10 &&
($2.hour<9 || $2.hour>18))
```

Use the trigger condition expression to specify a trigger condition and threshold. Here, `$1.error_count>100 && $2.hour>=9 && $2.hour<=18` indicates to trigger an alarm when `error_count` exceeds 100 between 9:00 AM and 6:00 PM, while `$1.error_count>10 && ($2.hour<9 || $2.hour>18)` indicates to trigger an alarm when `error_count` exceeds 10 between 7:00 PM and 8:00 AM.

Setting Interval-Valued Comparison and Periodically-Valued Comparison as Alarm Trigger Conditions

Last updated：2024-01-20 17:28:40

Overview

Setting an alarm trigger condition usually involves interval-valued comparison of metrics due to business characteristics. For example, you can set to trigger an alarm when API response time is over 50% longer than that in the same time period yesterday.

Configuration method

When configuring an alarm policy, enter the following query statements and trigger conditions:

Query statements:

```
* | select
  round(compare[3], 4) as ratio,
  compare[1] as current_avg_request_time,
  compare[2] as yesterday_avg_request_time
from
  (
    select compare(avg_request_time, 86400) as compare
    from
      (
        select avg("request_time") as avg_request_time
      )
  )
```

In the execution result of the above statements:

`ratio` indicates the ratio of the current average API response time to the value yesterday (86,400 seconds earlier).

`current_avg_request_time` indicates the current average API response time.

`yesterday_avg_request_time` indicates the average API response time in the same period yesterday.

The `compare` function is used in the above statement. For more information, see Interval-Valued Comparison and Periodicity-Valued Comparison Functions.

Trigger conditions:

```
$1.ratio > 1.5
```

An alarm will be triggered if `ratio` exceeds `1.5`, that is, the time is over 50% longer than that yesterday.

Integrating Alarms with PagerDuty/Slack/Microsoft Teams

Last updated：2025-05-29 20:01:46

Overview

Alarms can be integrated with PagerDuty/Slack/Microsoft Teams via webhook, making it easy to unify alarm notifications.

Configuration Process

Step 1: Creating a Notification Template

1. Log in to the Cloud Log Service console.
2. In the left navigation bar, select **Monitoring Alarm** > **Notification Template** to enter the notification template management page.
3. Click **Create**, in the **Webhook** tag, fill in the following information according to the channels that need to be integrated and save.

PagerDuty

Slack

Microsoft Teams

Note:

CLS integrates with PagerDuty via Events API V2. If the current Service has not added Events API V2 Integration, follow the steps below: Add Integrations to an Existing Service.

Replace `routing_key` in the following request content with the Integration Key in PagerDuty.

Note: Record the Integration URL in PagerDuty for subsequent steps.

Alarm triggered

Request Header

```
Accept: application/json
```

```
Content-Type: application/json
```

Request Content

```
{
  "payload": {
```

```
    "summary": "{{escape .Alarm}}",
    "timestamp": "{{fromUnixTime .NotifyTimeUnix}}",
    "severity": "{{if eq .Level "Critical"}}critical{{else if eq .Level "Warn"}}war
    "source": "Tencent Cloud Log Service",
    "custom_details": {
      "Alarm Policy": "{{escape .Alarm}}",
      "Trigger Condition": "{{escape .Condition}}",
      "Current Data": "{{escape .TriggerParams}}",
      "Additional Message": "{{escape .Message}}",
      "Multidimensional Analysis": "{{escape .AnalysisResultFormat}}"
    }
  },
  "routing_key": "R03ECCMUCxxxxxxxxxxNGFE87CT",
  "dedup_key": "{{.RecordGroupId}}",
  "event_action": "trigger",
  "client": "{{escape .Topic}}",
  "client_url": "{{.QueryUrl}}",
  "links": [
    {
      "href": "{{.DetailUrl}}",
      "text": "Alert Detail"
    }
  ]
}
```

Alarm cleared

Request Header

```
Accept: application/json
```

```
Content-Type: application/json
```

Request Content

```
{
  "payload": {
    "summary": "{{escape .Alarm}}",
    "timestamp": "{{fromUnixTime .NotifyTimeUnix}}",
    "severity": "{{if eq .Level "Critical"}}critical{{else if eq .Level "Warn"}}war
    "source": "Tencent Cloud Log Service"
  },
  "routing_key": "R03ECCMUCxxxxxxxxxxNGFE87CT",
  "dedup_key": "{{.RecordGroupId}}",
  "event_action": "resolve"
}
```

Alarm triggered

Request Header

```
Content-Type: application/x-www-form-urlencoded
```

Request Content

```
{{- define "subTemplate" -}}
Alarm triggered for the log service of account {{.UIN}} ({{.Nickname}}):
•  Alarm Policy: {{.Alarm}}
•  Alarm Level: {{.Level}}
•  Monitoring Object: {{.Topic}}
•  Trigger Condition: {{.Condition}}
•  Current Data: {{.TriggerParams}}
•  Trigger Time: {{.StartTime}}
{{- if (gt .Duration 0)}}
•  Duration: {{.Duration}} minutes
{{- end}}
•  Additional Message: {{.Message}}
•  Multidimensional Analysis:
``{{.AnalysisResultFormat}}``
<{{.DetailUrl}}|DetailedReport>  <{{.QueryUrl}}|QueryData>{{if .CanSilent}}  <{{.De
{{- end -}}
payload={"username": "CLS Alert","icon_emoji": ":rotating_light:","blocks": [{"type
```

Alarm cleared

Request header

```
Content-Type: application/x-www-form-urlencoded
```

request content

```
{{- define "subTemplate" -}}
 A CLS alarm was resolved under your account (ID: {{.UIN}}; name: {{.Nickname}}):
•  Alarm Policy: {{.Alarm}}
•  Alarm Level: {{.Level}}
•  Monitoring Object: {{.Topic}}
•  Trigger Condition: {{.Condition}}
•  Trigger Time: {{.StartTime}}
•  Resolved Time: {{.NotifyTime}}
•  Duration: {{.Duration}} minutes"
{{- end -}}
payload={"username": "CLS Alert","icon_emoji": ":green_circle:","blocks": [{"type":
```

Alarm triggered

Request Header

```
Content-Type: application/json
```

Request Content

```
{{- define "subTemplate" -}}
Alarm triggered for the log service of account {{.UIN}} ({{.Nickname}}):
- Alarm Policy: {{.Alarm}}
- Alarm Level: {{.Level}}
- Monitoring Object: {{.Topic}}
- Trigger Condition: {{.Condition}}
- Current Data: {{.TriggerParams}}
- Trigger Time: {{.StartTime}}
{{- if (gt .Duration 0)}}
- Duration: {{.Duration}} minutes
{{- end}}
- Additional Message: {{.Message}}

{{- range $key,$value :=  .AnalysisResult}}

{{$key}}

{{$value}}
{{- end}}
{{- end -}}
{
  "type": "message",
  "attachments": [{
    "contentType": "application/vnd.microsoft.card.adaptive",
    "content": {
      "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
      "type": "AdaptiveCard",
      "version": "1.3",
      "msteams": {"width": "Full"},
      "body": [
          {
        "type": "TextBlock",
        "text": "Alarm:{{escape .Alarm}}",
        "wrap": true,
        "color": "{{if eq .Level "Critical"}}attention{{else if eq .Level "Warn"}}w
        "size": "Large"
      },
      {
        "type": "TextBlock",
        "text": "{{- escape (substr (renderTemplate "subTemplate")  0 3500)}}",
        "wrap": true
      },
```

```
      {
        "type": "ActionSet",
        "actions": [{"type":"Action.OpenUrl","title":"DetailedReport","url":"{{.Det
      }]
    }
  }]
}
```

Alarm cleared

Request Header

```
Content-Type: application/json
```

Request Content

```
{{- define "subTemplate" -}}
A CLS alarm was resolved under your account (ID: {{.UIN}}; name: {{.Nickname}}):
- Alarm Policy: {{.Alarm}}
- Alarm Level: {{.Level}}
- Monitoring Object: {{.Topic}}
- Trigger Condition: {{.Condition}}
- Trigger Time: {{.StartTime}}
- Resolved Time: {{.NotifyTime}}
- Duration: {{.Duration}} minutes"
{{- end -}}
{
  "type": "message",
  "attachments": [{
    "contentType": "application/vnd.microsoft.card.adaptive",
    "content": {
      "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
      "type": "AdaptiveCard",
      "version": "1.3",
      "msteams": {"width": "Full"},
      "body": [
          {
        "type": "TextBlock",
        "text": "Resolved:{{escape .Alarm}}",
        "wrap": true,
        "color": "good",
        "size": "Large"
      },
      {
        "type": "TextBlock",
        "text": "{{- escape (substr (renderTemplate "subTemplate")  0 3500)}}",
        "wrap": true
      }]
```

```
        }
    }]
}
```

Step 2: Creating a New Notification Channel Group

1. In the left navigation bar, select **Monitoring Alarm** > **Notification Group** to access the Notification Group Management Page.
2. Click **Create**, then click **Add rule** in **Notification rules**, and fill in the following information in **Set notification channel** and save:

Type: Custom webhooks.

Webhook address:

PagerDuty: Integration URL in the Add Integrations to an Existing Service procedure.

Slack: CLS integrates with Slack through Incoming Webhooks. If there is no suitable Incoming Webhook currently, follow these steps: Sending messages using incoming webhooks, create an Incoming Webhook, and obtain the Webhook URL.

Microsoft Teams: CLS integrates with Teams through Workflows. Follow these steps: Create incoming webhooks with Workflows for Microsoft Teams, create a workflow using the template method, and retrieve the URL. For the template, select "Post to a channel when a webhook request is received".

Request Method: POST

Notification template: Select the one created in Step 1.

Note:

Detailed configuration instructions can be found in Manage Notification Group.

Step 3: Selecting a Notification Channel Group in the Alarm Policy

1. In the left navigation bar, select **Monitoring Alarm** > **Alarm Policy** to enter the alarm policy management page.
2. Click **Create** to start configuring the alarm policy. For detailed configuration instructions, refer to Configure Alarm Policy. When associating a notification group, select the Step 2 newly created notification channel group.

Shipping and Consumption
Consumption of CLS Log with Flink

Last updated：2024-01-20 17:28:40

This document describes how to consume CLS logs with Flink in real time, analyze Nginx log data with Flink SQL, calculate web PVs/UVs, and write the result to self-built MySQL databases in real time.

Components/Applications and their versions used in this document are as described below:

Technical Component	Version
Nginx	1.22
CLS	-
Java	OpenJDK version "1.8.0_232"
Scala	2.11.12
Flink SQL	Flink 1.14.5
MySQL	5.7

Directions

Step 1. Install the Tencent Cloud Nginx gateway

1. Purchase a CVM instance as instructed in Creating Instances via CVM Purchase Page.
2. Install Nginx as instructed in the directions for installing Nginx on Linux.
3. The installation is successful if you can access Nginx in the browser and see the following page:

Step 2. Collect Nginx logs to CLS

1. Configure Nginx log collection as instructed in Collecting and Searching NGINX Access Logs.
2. Install CLS's log collector LogListener as instructed in LogListener Installation Guide. Similar to the open-source component Beats, LogListener is an agent that collects logs.
3. After index is enabled for log topics, you can query Nginx logs as shown below:
4. Enable consumption over Kafka in the CLS console to use the feature. You can consume a log topic as a Kafka topic. This document describes how to consume Nginx log data in real time with the stream computing framework Flink and then write the real-time computing result to MySQL.

Step 3. Set up a MySQL database

For detailed directions, see Creating MySQL Instance.

1. Log in to the database:

```
mysql -h 172.16.1.1 -uroot
```

2. Create the target database and table. Here, the `flink_nginx` database and `mysql_dest` table are created.

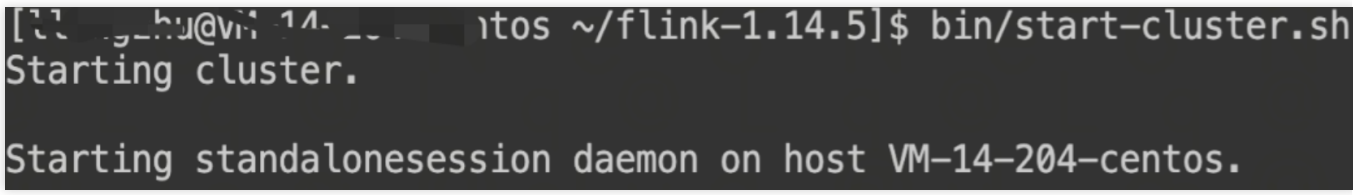
```
create database if not exists flink_nginx;
create table if not exists mysql_dest(
 ts timestamp,
 pv bigint,
 uv bigint
);
```

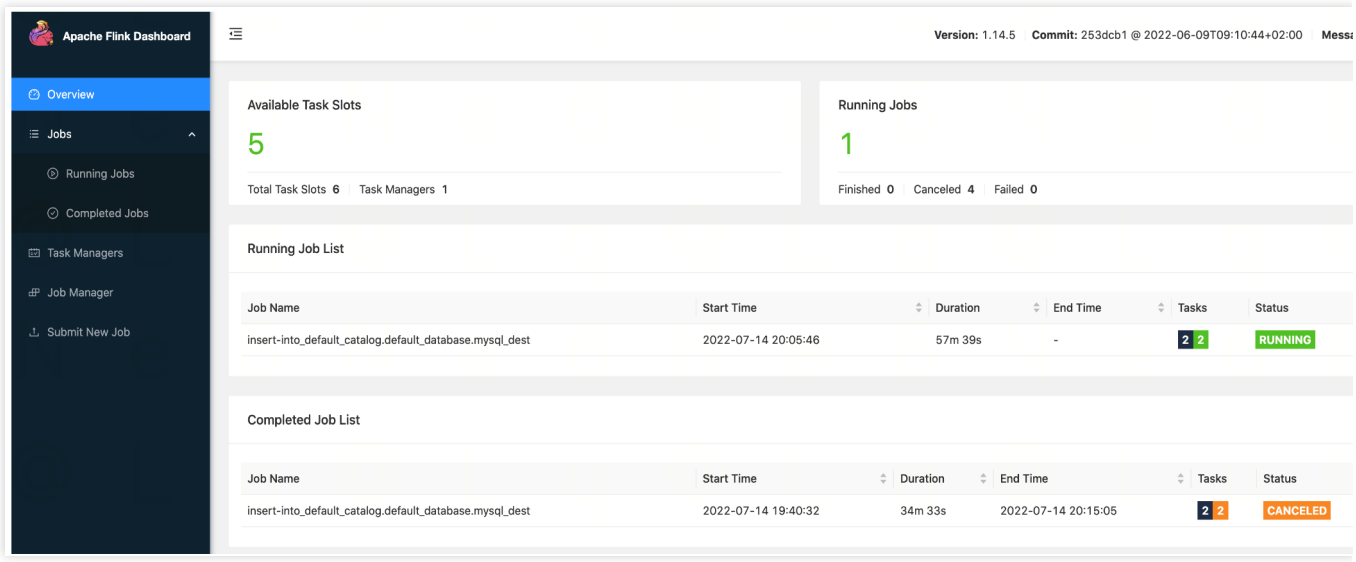
Step 4. Deploy Flink

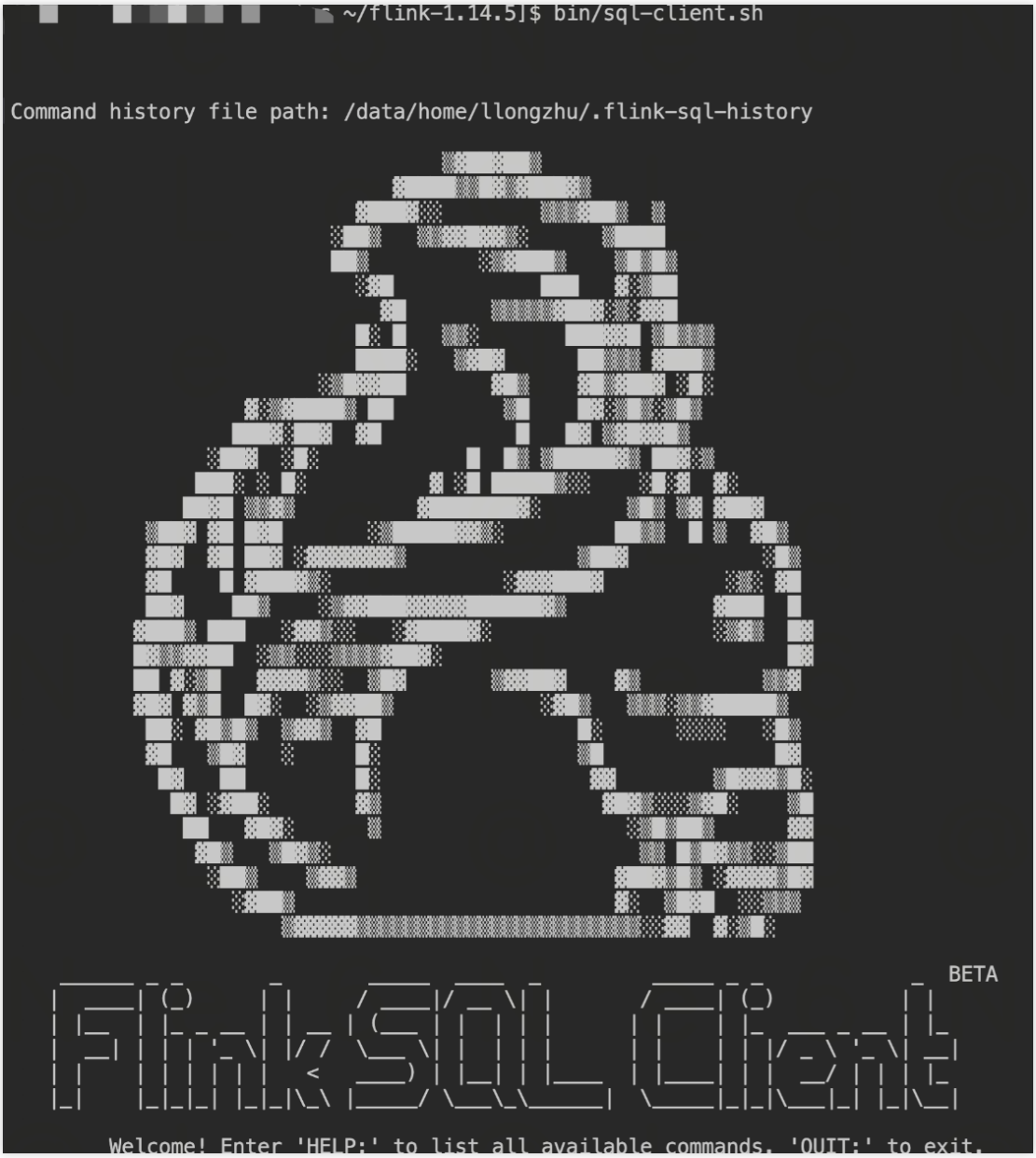
1. We recommend that you use the following versions for Flink deployment; otherwise, the installation may fail.
Purchase a CVM instance as instructed in Creating Instances via CVM Purchase Page.
Install Scala 2.11.12 as instructed in Ways to Install This Release.
2. Install Flink 1.14.15 and go to the SQL UI. Download the binary code package of Flink from the Apache Flink website and start installation.

```
# Decompress the Flink binary package
tar -xf flink-1.14.5-bin-scala_2.11.tgz
cd flink-1.14.5

# Download Kafka dependencies
wget https://repo1.maven.org/maven2/org/apache/flink/flink-connector-
kafka_2.11/1.14.5/flink-connector-kafka_2.11-1.14.5.jar
mv flink-connector-kafka_2.11-1.14.5.jar lib
wget https://repo1.maven.org/maven2/org/apache/kafka/kafka-clients/2.4.1/kafka-
clients-2.4.1.jar
mv kafka-clients-2.4.1.jar lib

# Download MySQL dependencies
wget https://repo1.maven.org/maven2/org/apache/flink/flink-connector-
jdbc_2.11/1.14.5/flink-connector-jdbc_2.11-1.14.5.jar
mv flink-connector-jdbc_2.11-1.14.5.jar lib
wget https://repo1.maven.org/maven2/mysql/mysql-connector-java/8.0.11/mysql-
connector-java-8.0.11.jar
mv mysql-connector-java-8.0.11.jar lib
wget https://repo1.maven.org/maven2/org/apache/flink/flink-table-
common/1.14.5/flink-table-common-1.14.5.jar
mv flink-table-common-1.14.5.jar lib

# Start Flink
bin/start-cluster.sh
bin/sql-client.sh
```

3. The installation is successful if the following information is displayed. Note that the default web port is 8081.




Apache Flink Dashboard
Version: 1.14.5
Commit: 253dcb1 @ 2022-06-09T09:10:44+02:00
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit New Job
Available Task Slots
5
Total Task Slots 6
Task Managers 1
Running Jobs
1
Finished 0
Canceled 4
Failed 0
Running Job List
Job Name
Start Time
Duration
End Time
Tasks
Status
insert-into_default_catalog.default_database.mysql_dest
2022-07-14 20:05:46
57m 39s
-
2 2
RUNNING
Completed Job List
Job Name
Start Time
Duration
End Time
Tasks
Status
insert-into_default_catalog.default_database.mysql_dest
2022-07-14 19:40:32
34m 33s
2022-07-14 20:15:05
2 2
CANCELED



Step 5. Consume CLS log data with Flink

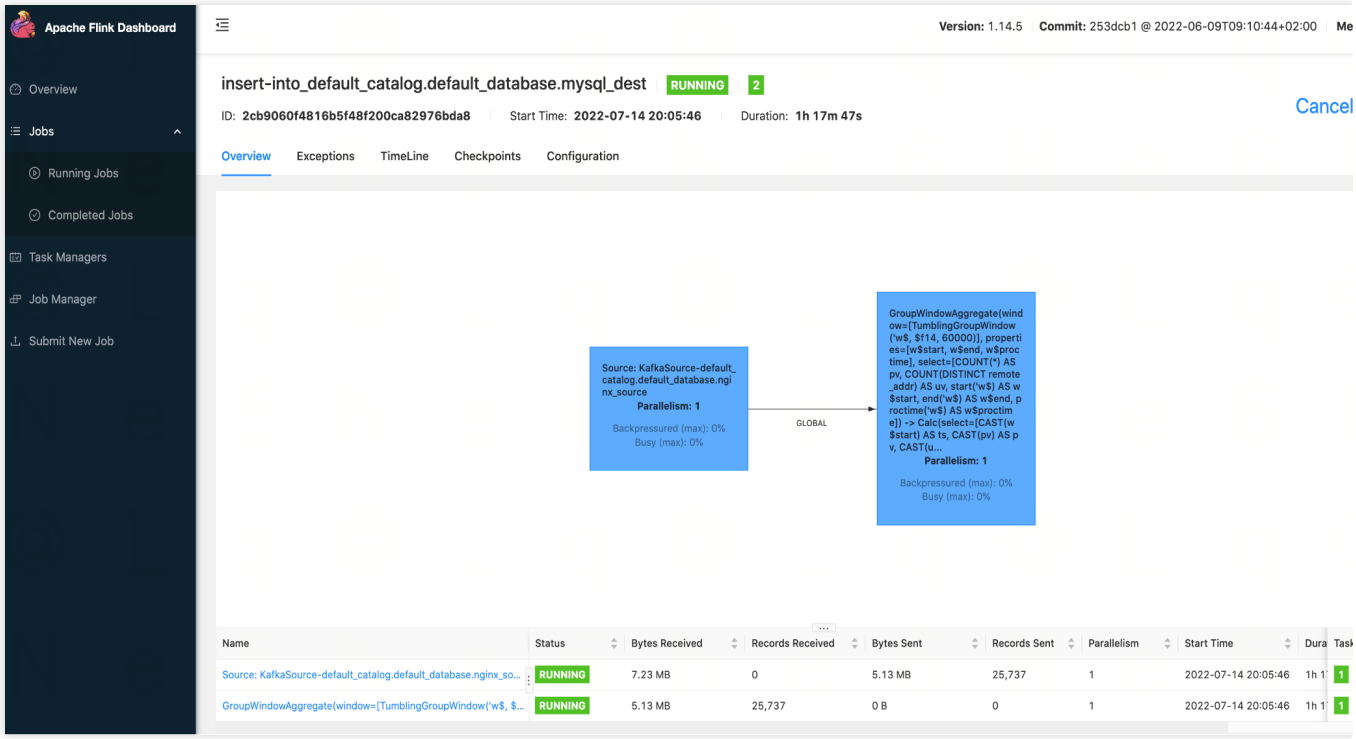
1. On the SQL client UI, execute the following SQL statements:

```
-- Create a data source table to consume Kafka data
CREATE TABLE `nginx_source`
(
    `remote_user` STRING,           -- Field in the log, which indicates the
client name.
    `time_local` STRING,            -- Field in the log, which indicates the
local time of the server.
    `body_bytes_sent` BIGINT,       -- Field in the log, which indicates the
number of bytes sent to the client.
```

```
    `http_x_forwarded_for` STRING,  -- Field in the log, which records the
actual client IP when there is a proxy server on the frontend.
    `remote_addr` STRING,           -- Field in the log, which indicates the
client IP.
    `protocol` STRING,              -- Field in the log, which indicates the
protocol type.
    `status` INT,                   -- Field in the log, which indicates the
HTTP request status code.
    `url` STRING,                   -- Field in the log, which indicates the
URL.
    `http_referer` STRING,          -- Field in the log, which indicates the
URL of the referer.
    `http_user_agent` STRING,       -- Field in the log, which indicates the
client browser information.
    `method` STRING,                -- Field in the log, which indicates the
HTTP request method.
    `partition_id` BIGINT METADATA FROM 'partition' VIRTUAL,    -- Kafka
partition
    `ts` AS PROCTIME()
)  WITH (
  'connector' = 'kafka',
  'topic' = 'YourTopic',  -- Topic name provided in the CLS console for
consumption over Kafka, such as `out-633a268c-XXXX-4a4c-XXXX-7a9a1a7baXXXX`
  'properties.bootstrap.servers' = 'kafkaconsumer-ap-
guangzhou.cls.tencentcs.com:9096',   -- Service address provided in the CLS
console for consumption over Kafka. The public network consumer address in
Guangzhou region is used as an example. You need to enter the actual
information.
  'properties.group.id' = 'kafka_flink', -- Kafka consumer group name
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json',
  'json.fail-on-missing-field' = 'false',
  'json.ignore-parse-errors' = 'true' ,
  'properties.sasl.jaas.config' =
'org.apache.kafka.common.security.plain.PlainLoginModule required
username="your username" password="your password";',--Your username is the
logset ID of the log topic, such as `ca5cXXXX-dd2e-4ac0-af12-92d4b677d2c6`, and
the password is a string of your `secretid#secrectkey`, 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.
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.sasl.mechanism' = 'PLAIN'

);
```

```
--- Create the target table and write it to MySQL
CREATE TABLE `mysql_dest`
(
    `ts` TIMESTAMP,
    `pv` BIGINT,
    `uv` BIGINT
)  WITH (
    'connector' = 'jdbc',
    'url' = 'jdbc:mysql://11.150.2.1:3306/flink_nginx?
&amp;serverTimezone=Asia/Shanghai', -- Note the time zone settings here
    'username'= 'username',     -- MySQL account
    'password'= 'password',     -- MySQL password
    'table-name' = 'mysql_dest' -- MySQL table name
);

--- Query the Kafka data source table and write the computing result to the
MySQL target table
INSERT INTO mysql_dest (ts,uv,pv)
SELECT TUMBLE_START(ts, INTERVAL '1' MINUTE) start_ts, COUNT(DISTINCT
remote_addr) uv,count(*) pv
FROM nginx_source
GROUP BY TUMBLE(ts, INTERVAL '1' MINUTE);
```

2. On the Flink task monitoring page, view the monitoring data of the task:



3. Go to the MySQL database, and you can see that PV and UV results are calculated and written in real time:

```
MySQL [flink_nginx]> select * from mysql_dest;
+---------------------+------+------+
| ts                  | pv   | uv   |
+---------------------+------+------+
| 2022-07-14 20:16:00 |   60 |    1 |
| 2022-07-14 20:17:00 |   62 |    2 |
| 2022-07-14 20:18:00 |   64 |    3 |
| 2022-07-14 20:19:00 |   63 |    2 |
| 2022-07-14 20:20:00 |   59 |    1 |
| 2022-07-14 20:21:00 |   59 |    1 |
| 2022-07-14 20:22:00 |   60 |    1 |
+---------------------+------+------+
7 rows in set (0.05 sec)
```

Using DLC (Hive) to Analyze CLS Log

Last updated：2024-01-20 17:28:40

Overview

This document describes how to ship logs in CLS to Hive for OLAP computing. You can use the data analysis and computing services provided by Tencent Cloud Data Lake Compute to complete offline log computing and analysis.

Directions

Shipping CLS logs to COS

Creating a shipping task

1. Log in to the CLS console and select **Shipping Task Management** > **Ship to COS** on the left sidebar.
2. On the **Ship to COS** page, click **Add Shipping Configuration**. In the **Ship to COS** pop-up window, create a shipping task.

Pay attention to the following configuration items:

Configuration Item	Description
Directory Prefix	Log files will be shipped to the corresponding directory in the COS bucket, which is generally the address of the table location in a data warehouse model.
Partition Format	A shipping task can automatically partition data by creation time. We recommend you specify the partition format according to the Hive partitioned table format.For example, to partition by day, you can set /dt=%Y%m%d/test. Here, dt= indicates the partitioning field, %Y%m%d indicates the year, month, and day, and test indicates the log file prefix. As the name of a shipped file start with an underscore ((_)) by default, the big data computing engine will ignore such files and cause a failure to find the data. Therefore, you need to add a prefix, and the actual partition directory name will be dt=20220424 for example.
Shipping Interval	You can select 5–15 minutes. We recommend you select 15 minutes, 250 MB. In this case, the number of files will be lower, and the query performance will be high.
Shipping Format	The JSON format is recommended.

Viewing the shipping task result

Generally, you can view the log data in the COS console 15 minutes after the shipping task starts. If you set partitioning by day in the `log_data` logset, the directory structure will be as shown below, where specific log files are stored in partition directories.

Data Lake Compute (Hive) analysis

Using Data Lake Compute to create a foreign table and map it to a COS log directory

After log data is shipped to COS, you can use the data exploration feature in the Data Lake Compute console to create a foreign table. For the table creation statement, refer to the following example. **Note that the partitioning field and the `location` field must match the directory structure.**
The Data Lake Compute foreign table creation wizard provides advanced options to infer the data file table structure and quickly and automatically generate SQL statements. As sampled inference is used, you need to further determine whether table fields are appropriate based on the SQL statements. For example, in the sample code below, it is inferred that the `__TIMESTAMP__` field is of `int` type, but maybe the `bigint` type should be used to meet the business requirements.

```
CREATE EXTERNAL TABLE IF NOT EXISTS `DataLakeCatalog`.`test`.`log_data` (
  `__FILENAME__` string,
  `__SOURCE__` string,
  `__TIMESTAMP__` bigint,
  `appId` string,
  `caller` string,
  `consumeTime` string,
  `data` string,
  `datacontenttype` string,
  `deliveryStatus` string,
  `errorResponse` string,
  `eventRuleId` string,
  `eventbusId` string,
  `eventbusType` string,
  `id` string,
  `logTime` string,
  `region` string,
  `requestId` string,
  `retryNum` string,
  `source` string,
  `sourceType` string,
  `specversion` string,
  `status` string,
  `subject` string,
  `tags` string,
  `targetId` string,
  `targetSource` string,
  `time` string,
```

```
    `type` string,
    `uin` string
) PARTITIONED BY (`dt` string) ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.Json
```

For shipping by partition, `location` must point to the `cosn://coreywei-1253240642/log_data/` directory instead of the `cosn://coreywei-1253240642/log_data/20220423/` directory.

To use the inference feature, you need to point the directory to the subdirectory of the data file, i.e., `cosn://coreywei-1253240642/log_data/20220423/` . After inference is completed, change `location` in the SQL statement back to `cosn://coreywei-1253240642/log_data/` .

Appropriate partitioning can improve the performance. However, we recommend you create no more than 10,000 partitions.

Adding a partition

You can use a `SELECT` statement to get data from a partitioned table only after adding partitions in the following two ways:

Adding historical partitions

Adding incremental partitions

This option can load all partition data at a time but is slow, so it is suitable for scenarios where many partitions need to be loaded for the first time.

```
msck repair table DataLakeCatalog.test.log_data;
```

After historical partitions are loaded, incremental partitions will be added periodically. For example, you can use this option to add a partition every day.

```
alter table DataLakeCatalog.test.log_data add partition(dt='20220424')
```

Analyzing data

After adding partitions, you can use Data Lake Compute for data development and analysis.

```
select dt,count(1) from `DataLakeCatalog`.`test`.`log_data` group by dt;
```

Cost Optimization
Saving Product Use Costs

Last updated：2024-01-20 17:28:40

Cost Saving Suggestions

When considering cost saving for CLS, pay more attention to billable items that are more costly. Index traffic and index storage are usually several times more expensive than write traffic and log storage, mainly because they are billed according to the volume of uncompressed log data. If LogListener is used to upload logs, data is compressed. Write traffic and log storage are calculated based on the volume of the compressed log data. Generally, the log compression ratio is 1:4 to 1:10.
You can adjust log topic configuration as follows to save costs:
1. **Reduce the amount of log data to be uploaded**: Most billable items, including index traffic and index storage, are related to the amount of uploaded log data. Reducing the amount of uploaded log data can save costs at the source. You can reduce unnecessary log collection or use the filter in the LogListener collection configuration to collect only logs that meet the filter rules, for example, only Error and Warning logs.
2. **Shorten the log storage period**: The longer the log storage period, the higher the log storage expense and index storage expense. For older logs, if they are no longer needed for daily search and analysis and only need to be backed up as historical data, they can be shipped to COS.
3. **Simplify index configuration**: Simplifying index configuration can reduce index traffic and index storage costs. Note that if you enable both full-text index and key-value index, the index traffic will not be billed twice, but billed in a union-like manner. Therefore, when full-text index is enabled, reducing the number of key-value index fields does not reduce index traffic or index storage. If you do not need full-text search, you can disable full-text index and reduce unwanted key-value index fields.
4. **Use infrequent access (IA) storage**: IA storage is a low-cost offline log storage solution to search and store massive infrequently accessed logs. It applies to scenarios where users do not have real-time log search requirements and logs need to be stored for a long time. The solution can save about 80% of costs. For more information, see here.
5. **Distribute log topics using the data processing feature**: With the data processing feature, you can ship raw logs to different log topics. For example, you can classify logs by log level, ERROR, WARNING, and INFO, and then ship them to different log topics. You can create a source log topic for receiving all log data (you can disable index for the log topic so that it does not incur any index traffic or index storage fees, reducing costs), and then use the data processing feature to ship raw logs to log topics with different storage periods, different index configurations, or different storage types according to log search and analysis requirements to better balance log usage requirements and costs. For more information, see Creating a Processing Task.