日志服务

实践教程

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

实践教程
- 日志采集
  - 采集 Windows 环境日志至 CLS
  - 采集/查询主机文件日志
  - 使用 SDK 上传日志
    - 配置环境变量
    - Python SDK 上传日志
    - GO SDK 上传日志
    - Java SDK 上传日志
    - C SDK 上传日志
    - C++ SDK 上传日志
    - PHP SDK 上传日志
    - NodeJS SDK 上传日志
    - 浏览器 JavaScript SDK上传日志
    - 小程序 JavaScript SDK 上传日志
    - HarmonyOS SDK 上传日志
    - Android SDK 上传日志
    - IOS SDK 上传日志
    - Flutter SDK 上传日志
  - 通过 Kafka 数据订阅跨账号同步日志
- 检索分析
  - 云产品账单数据分析
  - CDN 访问日志分析
  - CLB 访问日志分析
  - Nginx 访问日志分析
  - COS 访问日志分析
  - Flowlog 网络流日志分析
  - TKE 审计日志分析
  - TKE 事件日志分析
  - CSS 云直播日志分析
- 仪表盘
  - 把 Grafana 的 ES 数据源迁移为 CLS 数据源
- 监控告警
  - 按时间段分别设置告警触发条件
  - 使用同环比作为告警触发条件
  - 按日志所属服务将告警发送到不同的团队
  - 告警对接 PagerDuty/Slack 等第三方平台
- 投递和消费
  - 使用 Flink 消费 CLS 日志
  - 使用 DLC（Hive）分析 CLS 日志
- 定时 SQL 分析
  - 使用定时 SQL 解决检索分析超时
  - 从日志中提取指标(Metric)

实践教程
日志采集
采集 Windows 环境日志至 CLS

最近更新时间：2025-02-05 17:42:13

操作场景

本文指导您使用 Winlogbeat 或者 Filebeat 采集 Windows 环境日志上传至 CLS。

前提条件

- 已开通日志服务（Cloud Log Service，CLS），并创建对应资源（如 日志集 和 日志主题 ）。
- 已在 腾讯云访问管理控制台 获取到 SecretId 和 SecretKey。

操作步骤

使用 LogListener Windows 版采集 Windows 事件日志

详情请参见 采集 Windows 事件日志 。

使用 Filebeat 采集 Windows 文件日志

安装 Filebeat

1. 前往 官网，选择对应版本，下载安装包。
2. 将安装包上传至目标 Windows 主机某个盘的根目录，并解压缩。
3. 以管理员身份打开 powershell， `cd` 至解压后的文件目录中，并执行如下命令安装 Filebeat。

```
#执行安装脚本，安装 filebeat 服务
.\install-service-filebeat.ps1

#安装模板文件
.\filebeat.exe setup -e

#启动 filebeat 服务
start-service filebeat
```

上传日志至 CLS

在 Filebeat 安装目录中，找到 `filebeat.reference.yml` 文件，复制并命名为 `filebeat.yml` 。打开 `filebeat.yml` 并将其中的 output.kafka 修改为如下内容。该配置将指定 Filebeat 将日志发送到 CLS Kafka 生产端。

```
output.kafka:
  enabled: true
  hosts: ["${region}-producer.cls.tencentyun.com:9095"] # TODO 服务地址；外网端口9096，内网端口9095
  topic: "${topicID}" #  TODO topicID
  version: "0.11.0.2"
  compression: "${compress}"   # 配置压缩方式，支持gzip，snappy，lz4；例如"lz4"
  username: "${logsetID}"
  password: "${secret_id}#${secret_key}"
```

参数	说明
鉴权机制	当前支持 SASL_PLAINTEXT。
hosts	CLS Kafka 地址。根据目标写入日志主题所在地域配置。请参见 服务入口 。

topic	CLS Kafka topic名称。配置为日志主题 ID。例如：76c63473-c496-466b-XXXX-XXXXXXXXXXXX。
username	CLS Kafka 访问用户名。配置为日志集 ID。例如：0f8e4b82-8adb-47b1-XXXX-XXXXXXXXXXXX。
password	CLS Kafka 访问密码。格式为 `${secret_id}#${secret_key}` 。例如：XXXXXXXXXXXXXXX#YYYYYYYY。密钥信息获取请前往 密钥获取 。

服务入口

地域	网络类型	端口号	服务入口
广州	内网	9095	gz-producer.cls.tencentyun.com:9095
	外网	9096	gz-producer.cls.tencentcs.com:9096

注意：
本文档以广州地域为例，内外网域名需用不同端口标识，其他地域请替换地址前缀。详情请参见 可用域名-Kafka 上传日志 。

采集/查询主机文件日志

最近更新时间：2024-05-29 14:28:51

操作场景

在简单的运维场景下，日志通常先直接输出到服务器本地文件中，再使用 Linux 系统下常用的 grep 命令查找符合要求的日志。此方式在业务系统较为复杂时，会由于日志分散在不同的服务器、命令行操作不直观、服务器权限管理限制等原因导致日志查找困难，严重影响运维效率。当需要基于日志做一些统计分析或监控告警，更是难上加难。
本文将介绍如何快速通过 grep 命令查找的本地日志迁移至日志服务（Cloud Log Service，CLS），以获取如下优势：

- 数据集中存储及检索，无需登录多台服务器分别进行查询，在负载均衡、微服务等架构下尤为关键。
- 简单点击即可快速检索日志，告别命令行及繁琐的服务器权限管理。
- 基于日志进行统计分析，获取关键业务指标，例如 PV、接口响应时间、接口错误率等。
- 实时检测异常日志，通过短信、邮件和微信等多种方式获取通知。

> **说明**
> 如果您的日志已经采集到了 CLS，可跳过日志采集和配置索引步骤，直接执行 步骤3：检索日志 。

操作步骤

步骤1：日志采集

针对服务器本地日志，可使用 LogListener 将原始日志采集至 CLS，LogListener 安装详情请参见 LogListener 安装指南 。
如果您的服务器为腾讯云云服务器（Cloud Virtual Machine，CVM），还可以通过控制台自动安装 LogListener，详情请参见 CVM 批量部署 LogListener 。
与服务器本地日志不同，为了后续对日志进行更方便的检索和统计分析，在采集时可将非格式化的原始日志转换为格式化的数据。例如原始日志为：

```
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/
```

可使用分割符 `:::` 切分为八个字段，并为每个字段定义名称：

```
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]
```

具体操作请参见 分隔符格式 。除了使用分隔符切分日志，CLS 还支持正则、JSON、全文等多种日志切分方式，详情请参见 采集文本日志 。

步骤2：配置索引

配置索引的目的在于定义哪些字段需要检索，字段的类型是什么，以便于后续检索日志。对于大多数使用场景，可使用自动配置索引功能，一键完成配置，详情请参见 配置索引 。

步骤3：检索日志

本文以常用的 grep 命令为例，介绍如何通过 CLS 实现类似的日志检索效果。CLS 检索方式操作步骤参见 语法规则 > 操作步骤 。
示例原始日志为：

```
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/
```

在 CLS 对应的格式化后日志为：

```
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]
```

案例1

检索 request 为/online/sample 的日志。

- 使用 grep 命令：

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

- 使用 CLS 检索方式：

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

案例2

检索状态码 status 不为200的日志。

- 使用 grep 命令：

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

实际上，此方式可能会把一些日志也排除掉（出现了200，但 status 不是200的字段）。如需准确检索，则需要编写正则表达式。

- 使用 CLS 检索方式：

```
NOT status:200
```

CLS 还支持更加灵活的检索方式，例如检索状态码 status 大于等于500的日志。

```
status:>=500
```

案例3

统计 status 不为200的日志条数。

- 使用 grep 命令：

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

- 使用 CLS 检索方式：

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

案例4

检索状态码 status 为200，且 request 为 /online/sample 的日志。

- 使用 grep 命令：

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

- 使用 CLS 检索方式：

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

案例5

检索 request 为 /online/sample 或 /offline/sample 的日志。

- 使用 grep 命令：

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

- 使用 CLS 检索方式：

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

案例6

检索 request 为 /online/sample，但日志文件不是 test.log 的日志。

- 使用 grep 命令：

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

- 使用 CLS 检索方式：

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

案例7

检索 time 为 [Tue Jan 22 14:49:45 CST 2019 +0800] 的日志的前10行日志。

- 使用 grep 命令：

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

- 使用 CLS 检索方式：

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

检索到匹配的日志后，使用 上下文检索 功能，即可查看该条日志附近的日志。

使用 SDK 上传日志
配置环境变量

最近更新时间：2024-12-16 10:02:22

环境变量是操作系统用来存储系统信息的变量，通常用于存储配置信息，在使用腾讯云 CLS 的 SDK 时，推荐通过环境变量的方式来指定云 API 密钥信息。云 API 密钥信息获取请前往 API 密钥管理 。

> **注意：**
> - 不建议通过明文的方式，将密钥信息存储在执行的工程代码文件中，否则可能造成密钥信息泄露，威胁您的账号安全。
> - 配置环境变量之后，您可以在不修改代码的情况下，将动态的鉴权参数传递到对应的函数，实现便捷安全的身份认证。

Linux 和 macOS 系统设置环境变量

配置环境变量后，在当前会话期间，后端服务会动态读取环境变量中指定的参数值，并将其应用于相应的函数。请将 YOUR_SECRET_ID 和 YOUR_SECRET_KEY 替换成您实际的云 API 密钥。

```
export TENCENTCLOUD_SECRET_ID="YOUR_SECRET_ID"
export TENCENTCLOUD_SECRET_KEY="YOUR_SECRET_KEY"
```

Windows 系统设置环境变量

配置环境变量后，在当前会话期间，后端服务会动态读取环境变量中指定的参数值，并将其应用于相应的函数。请将 YOUR_SECRET_ID 和 YOUR_SECRET_KEY 替换成您实际的云 API 密钥。

通过 Windows Command Prompt 设置

执行以下命令设置环境变量：

```
set TENCENTCLOUD_SECRET_ID="YOUR_SECRET_ID"
set TENCENTCLOUD_SECRET_KEY="YOUR_SECRET_KEY"
```

通过 Windows PowerShell 设置

执行以下命令设置环境变量：

```
$Env:TENCENTCLOUD_SECRET_ID="YOUR_SECRET_ID"
$Env:TENCENTCLOUD_SECRET_KEY="YOUR_SECRET_KEY"
```

通过 Windows 图形界面设置

1. 在桌面右击单击此电脑，选择**属性** > **高级系统设置** > **环境变量** > **系统变量/用户变量** > **新建**。
2. 添加云 API 密钥相关的环境变量，并单击**确定**。

> **说明：**
> 后续如需密钥的值，直接修改环境变量对应的参数值即可。

Python SDK 上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的 Python SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-python 。

前提条件

- 创建并获取云 API 密钥信息 accessKeyId 和 accessKey，密钥信息获取请前往 API 密钥管理 。
- 请确保密钥关联的账号具有相应的 SDK上传日志权限 。

准备开发环境

- 请参见 Python 官网 下载并安装 Python 开发环境或使用 conda 创建 Python 虚拟环境。
- 日志服务 Python SDK 支持 Pypy2、3和 Python2.7、3.3、3.4、3.5、3.6、3.7、3.8、3.9版本。

您可执行以下命令检查当前 Python 的版本信息

```
pip -V
```

安装 Python SDK

在命令行工具中，执行以下命令安装 Python SDK。

```
pip install git+https://github.com/TencentCloud/tencentcloud-cls-sdk-python.git@v1.0.4
```

验证 SDK 安装

安装 SDK 后，执行以下步骤验证已安装的 Python SDK。

```
pip show tencentcloud-cls-sdk-python
```

如果返回类型以下的信息，则代表安装成功。

```
Name: tencentcloud-cls-sdk-python
Version: 1.0.4
Summary: TencentCloud cls log service Python client SDK
Home-page: https://github.com/TencentCloud/tencentcloud-cls-sdk-python
Author: farmerx
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
accessKeyId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
accessKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topic_id	String	是	日志主题的 ID 信息。

日志上传示例代码

以下代码以 Python SDK 为例，展示通过调用 SDK 完成日志上传的操作，示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```python
# 导入所需的库
import time
from tencentcloud.log.logclient import LogClient
from tencentcloud.log.logexception import LogException
from tencentcloud.log.cls_pb2 import LogGroupList
import os

def upload(topic_id, client):
    LogLogGroupList = LogGroupList()
    LogGroup = LogLogGroupList.logGroupList.add()
    # 自定义本次日志上传的文件名来源，会作为元数据字段在日志中
    LogGroup.filename = "pyXXXX.log"
    # 自定义本次上传的地址来源，会作为元数据字段在日志中
    LogGroup.source = "192.XX.XX.XX"
    # 自定义元数据字段
    LogTag = LogGroup.logTags.add()
    LogTag.key = "key"
    LogTag.value = "value"

    Log = LogGroup.logs.add()
    Log.time = int(round(time.time() * 1000000))  # 获取当前时间作为日志的时间戳

    # 定义日志的内容
    Content = Log.contents.add()
    Content.key = "Hello"
    Content.value = "World"

    try:
        request = client.put_log_raw(topic_id, LogLogGroupList)
        print("Request ID:", request.get_request_id())
    except LogException as e:
        print("Error uploading log:", e)
if name == 'main':

    # 填入域名信息，填写指引：https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D，请参见
链接中 API 上传日志 Tab 中的域名
    endpoint = 'https://ap-xxxxxxxx.cls.tencentcs.com';

    # 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
    # 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
    # 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
    accessKeyId = os.environ["TENCENTCLOUD_SECRET_ID"]
    accessKey = os.environ["TENCENTCLOUD_SECRET_KEY"]

    # 填入日志主题的 ID
    topic_id = '59XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    # 构建日志客户端
    client = LogClient(endpoint, accessKeyId, accessKey)
    # 调用上传函数
    upload(topic_id, client)
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 Python SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

GO SDK 上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的 GO SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-go 。

前提条件

- 创建并获取云 API 密钥信息 AccessKeyID 和 AccessKeySecret，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK 上传日志权限。

准备开发环境

- 请参见 Go 官网 下载并安装 Go 开发环境。
- Go 安装完毕后请新建系统变量 GOPATH，并将其指向您的代码目录。

安装 GO SDK

在命令行工具中，执行以下命令安装 GO SDK。

```
go get github.com/tencentcloud/tencentcloud-cls-sdk-go
```

引入日志服务 Go SDK

后续实际写代码脚本时，需在您的 Go 代码中需要引入日志服务 Go SDK。

```
import (
    "fmt"
    "github.com/TencentCloud/tencentcloud-cls-sdk-go"
)
```

请求参数

变量	类型	是否必填	说明
Endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
AccessKeyID	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
AccessKeySecret	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topicId	String	是	日志主题的 ID 信息。

日志上传示例代码

以下代码以 Go SDK 为例，展示通过调用 SDK 完成日志上传的操作，示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```
package main
// 引入日志服务 Go SDK
import (
        "fmt"
        "github.com/tencentcloud/tencentcloud-cls-sdk-go"
        "sync"
        "os"
        "time"
)
```

```go
func main() {
	producerConfig := tencentcloud_cls_sdk_go.GetDefaultAsyncProducerClientConfig()
	// 填入域名信息，填写指引：https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D，
请参见链接中 API 上传日志 Tab 中的域名
	producerConfig.Endpoint = "ap-XXXXXXXXX.cls.tencentcs.com"

	// 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
	// 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
	// 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
	producerConfig.AccessKeyID = os.Getenv("TENCENTCLOUD_SECRET_ID")
	producerConfig.AccessKeySecret = os.Getenv("TENCENTCLOUD_SECRET_KEY")

	// 设置要上传日志的主题 ID，替换为您的 Topic ID
	topicId := "8bbXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
	// 创建异步生产者客户端实例
	producerInstance, err := tencentcloud_cls_sdk_go.NewAsyncProducerClient(producerConfig)
	if err != nil {
		fmt.Println(err)
		return
	}

	// 启动异步发送程序
	producerInstance.Start()

	var m sync.WaitGroup
	callBack := &Callback{}
	for i := 0; i < 10; i++ {
		m.Add(1)
		go func() {
			defer m.Done()
			for i := 0; i < 1000; i++ {
				// 创建新的日志，包含当前时间戳和日志内容
				log := tencentcloud_cls_sdk_go.NewCLSLog(time.Now().Unix(),
map[string]string{"content": "hello world| I'm from XXX", "content2": fmt.Sprintf("%v", i)})
				err = producerInstance.SendLog(topicId, log, callBack)
				if err != nil {
					fmt.Println(err)
					continue
				}
			}
		}()
	}
	m.Wait()
	producerInstance.Close(60000)
}

type Callback struct {
}

func (callback *Callback) Success(result *tencentcloud_cls_sdk_go.Result) {
	attemptList := result.GetReservedAttempts()
	for _, attempt := range attemptList {
		fmt.Printf("%+v \n", attempt)
	}
}
```

```
func (callback *Callback) Fail(result *tencentcloud_cls_sdk_go.Result) {
        fmt.Println(result.IsSuccessful())
        fmt.Println(result.GetErrorCode())
        fmt.Println(result.GetErrorMessage())
        fmt.Println(result.GetReservedAttempts())
        fmt.Println(result.GetRequestId())
        fmt.Println(result.GetTimeStampMs())
}
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 Go SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

Java SDK 上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的 Java SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库：tencentcloud-cls-sdk-java 。

前提条件

- 创建并获取云 API 密钥信息 secretId 和 secretKey，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。

准备开发环境

- 请参见 Java 官方网站 下载并安装 Java 开发环境。
- 日志服务 Java SDK 支持 JRE 6.0及以上的 Java 运行环境。您可以执行以下命令查看当前 Java 版本。

```
java -version
```

安装 Java SDK

推荐通过在 Maven 项目中加入依赖项的方式引入 Java SDK。
在 Maven 工程中使用日志服务 Java SDK，只需在 pom.xml 中加入相应依赖即可，Maven 项目管理工具会自动下载相关 JAR 包。

```
    <dependency>
      <groupId>com.tencentcloudapi.cls</groupId>
      <artifactId>tencentcloud-cls-sdk-java</artifactId>
      <version>1.0.15</version>
    </dependency>
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
secretId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
secretKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topicId	String	是	日志主题的 ID 信息。

日志上传示例代码

以使用 Java SDK 为例，展示通过调用 SDK 完成日志上传的操作， 示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```
import com.tencentcloudapi.cls.sdk.AsyncProducerClient;
import com.tencentcloudapi.cls.sdk.AsyncProducerConfig;
import com.tencentcloudapi.cls.sdk.LogItem;
import com.tencentcloudapi.cls.sdk.LogContent;
import com.tencentcloudapi.cls.sdk.NetworkUtils;

import java.util.ArrayList;
import java.util.List;

public class TencentClsDemo {
    public static void main(String[] args) {
```

```
        // 填入域名信息，请参见链接中 API 上传日志 Tab 中的域名：
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D
        String endpoint = "ap-xxxxxx.cls.tencentcs.com";
        // 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
        // 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
        // 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
        String secretId = System.getenv("TENCENTCLOUD_SECRET_ID");
        String secretKey = System.getenv("TENCENTCLOUD_SECRET_KEY");

        // 填入日志主题ID
        String topicId = "5912XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

        final AsyncProducerConfig config = new AsyncProducerConfig(endpoint, secretId, secretKey,
NetworkUtils.getLocalMachineIP());

        // 构建一个客户端实例
        final AsyncProducerClient client = new AsyncProducerClient(config);

        for (int i = 0; i < 10000; ++i) {
            List<LogItem> logItems = new ArrayList<>();
            int ts = (int) (System.currentTimeMillis() / 1000);
            LogItem logItem = new LogItem(ts);
            logItem.PushBack(new LogContent("__CONTENT__", "hello world"));
            logItem.PushBack(new LogContent("city", "guangXXXXX"));
            logItem.PushBack(new LogContent("logNo", Integer.toString(i)));
            logItem.PushBack(new LogContent("__PKG_LOGID__",
(String.valueOf(System.currentTimeMillis()))));
            logItems.add(logItem);
            client.putLogs(topicId, logItems, result -> System.out.println(result.toString()));
        }
        client.close();
    }
}
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 Java SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

C SDK 上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的 C SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-c 。

前提条件

- 创建并获取云 API 密钥信息 accessKeyId 和 accessKey，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。

准备开发环境

安装 C 语言开发环境

根据您的操作系统，安装 C 语言的编译环境和必要的开发工具。例如，在 CentOS 上，您可以使用以下命令安装开发工具和依赖库：

```
sudo yum groupinstall "Development Tools"
sudo yum install glibc-devel cmake openssl-devel git
sudo yum install libcurl-devel
```

安装 C SDK

1. 下载 C SDK：克隆 C SDK 的仓库到本地。

   ```
   git clone https://github.com/TencentCloud/tencentcloud-cls-sdk-c.git
   ```

2. 修改 demo 中代码。

   在 SDK 的 demo 目录中，您会找到示例代码 log_post.c，代码解析见下方。请根据您的需求修改示例代码，并确保替换其中的 Endpoint、AccessId、AccessKey 和 Topic 参数为您的实际信息。

   ```
   cd tencentcloud-cls-sdk-c/demo
   ```

3. 回到进入 SDK 目录，使用 cmake 和 make 工具编译并安装 SDK。

   ```
   cd tencentcloud-cls-sdk-c
   # 生成构建文件
   cmake .
   # 编译 SDK
   make
   ```

编译并运行示例程序

执行编译好的程序，上传日志。

```
# 进入可执行文件目录
cd build/bin
# 运行程序
./post_log_demo
```

请求参数

变量	类型	是否必填	说明
Endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。

AccessId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
AccessKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
Topic	String	是	日志主题的 ID 信息。

日志上传示例代码

以下是一个简化的示例代码，展示了如何使用 C SDK 上传日志。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```
#include "log_producer_config.h"
#include "log_producer_client.h"
#include "log_error.h"
#include <stdio.h>
#include <stdlib.h>

void callback(const char *config_name, int result, size_t log_bytes, size_t compressed_bytes, const
char *req_id, const char *message, const unsigned char *raw_buffer)
{
    if (result == LOG_PRODUCER_OK)
    {
        if (req_id == NULL)
        {
            req_id = "";
        }
        printf("send success, config : %s, result : %d, log bytes : %d, compressed bytes : %d, request
id : %s \n",
                config_name, (result),
                (int)log_bytes, (int)compressed_bytes, req_id);
    }
    else
    {
        if (message == NULL)
        {
            message = "";
        }
        if (req_id == NULL)
        {
            req_id = "";
        }
        printf("send fail, config : %s, result : %d, log bytes : %d, compressed bytes : %d, request id
: %s, error message : %s\n",
                config_name, (result),
                (int)log_bytes, (int)compressed_bytes, req_id, message);
    }
}

clslogproducer *ConstructorLogProducer(SendCallBackFunc notifyFunc)
{
    //调用malloch函数开辟内存，并初始化默认的配置
    ProducerConfig *config = ConstructLogConfig();
    // 填入域名信息，请参见链接中 API 上传日志 Tab 中的域名：
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D
    SetEndpoint(config, "ap-xxxxxxxxx.cls.tencentcs.com");

    // 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
```

```
    // 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
    // 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
    SetAccessId(config, getenv("TENCENTCLOUD_SECRET_ID"));
    SetAccessKey(config, getenv("TENCENTCLOUD_SECRET_KEY"));

    // 设置要上传日志的主题 ID，替换为您的 Topic ID
    SetTopic(config, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX");
    SetPackageLogBytes(config, 4 * 1024 * 1024);
    setPackageTimeout(config, 3000);
    SetMaxBufferLimit(config, 64 * 1024 * 1024);
    set_send_thread_count(config, 4);
    SetCompressType(config, 1);
    SetConnectTtimeoutSec(config, 10);
    SetSendTimeoutSec(config, 15);
    return ConstructorClsLogProducer(config, notifyFunc, NULL);
}

void post_logs()
{
    if (ClsLogProducerInit(LOG_GLOBAL_ALL) != LOG_PRODUCER_OK)
    {
        printf("ClsLogProducerInit init fail \n");
        exit(1);
    }
    clslogproducer *producer = ConstructorLogProducer(callback);
    if (producer == NULL)
    {
        printf("create log producer by config fail \n");
        exit(1);
    }

    clslogproducerclient *client = GetClsLogProducer(producer, NULL);
    if (client == NULL)
    {
        printf("create log producer client by config fail \n");
        exit(1);
    }
    int i = 0;
    for (; i < 10; ++i)
    {
        char indexStr[32];
        sprintf(indexStr, "%d", i);

        int rst = PostClsLog(client, 20, "key1", "value_1",
                             "key2", "value_2",
                             "key3", "value_3",
                             "key4", "value_4",
                             "key5", "value_5",
                             "key6", "value_6",
                             "key7", "value_7",
                             "key8", "value_8",
                             "key9", "value_9",
                             "index", indexStr);
        if (rst != LOG_PRODUCER_OK)
        {
            printf("add log error %d \n", rst);
        }
```

```
    }

    DestructorClsLogProducer(producer);

    ClsLogProducerDestroy();
}

int main(int argc, char *argv[])
{
    post_logs();
    return 0;
}
```

总结

通过以上步骤，您可以快速开始使用 C SDK 上传日志到云服务。确保您已正确设置所有必要的配置，并根据需要调整示例代码。如遇到任何问题，请 联系我们 获取帮助。

C++ SDK 上传日志

最近更新时间：2025-01-09 14:10:42

本文介绍如何快速使用日志服务的 C++ SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-c++ 。

前提条件

- 创建并获取云 API 密钥信息 accesskeyid 和 accessKeysecret，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。

准备开发环境

在安装 C++ SDK 之前，您需要准备 C++ 语言的开发环境，可参考如下命令安装开发环境。

```
# 安装编译工具和依赖（若未安装 C++可以使用）
sudo yum install gcc gcc-c++ automake autoconf libtool make
sudo yum install cmake
```

安装 C++ SDK

安装依赖

C++ SDK 的安装需要依赖于 protobuf，请执行以下命令安装 protobuf。

```
# 手动下载 protobuf
wget https://github.com/protocolbuffers/protobuf/archive/v2.6.1.tar.gz
# 解压源代码
tar -xzf v2.6.1.tar.gz
# 进入解压后的目录
cd protobuf-2.6.1
# 生成配置文件，若执行该步骤一直不动 or 时间超时错误，请参考下方【常见报错处理】
./autogen.sh
# 配置安装
./configure
#编译，这一步可能有点久请耐心等待
 make
 sudo make install
 sudo ldconfig
```

下载并安装 SDK

下载并安装腾讯云日志服务的 C++ SDK。

```
# 下载日志服务 SDK
git clone https://github.com/TencentCloud/tencentcloud-cls-sdk-cpp.git
# 进入目录
cd tencentcloud-cls-sdk-cpp
# 安装依赖
sudo yum install boost-devel
sudo yum install openssl-devel
sudo yum install libcurl-devel
# 生成构建文件
cmake .
# 编译 SDK
make
sudo make install
```

运行示例

以运行日志服务 SDK Demo 中 cls 文件夹下的日志上传代码 `sample.cpp` 为例，代码解析见下方。**实际使用时需替换代码中的密钥等信息方可运行。**

1. 编译示例代码。

```
# 编译示例代码
g++ -o sample ./cls/sample.cpp -std=c++11 -O2 -L/root/tencentcloud-cls-sdk-cpp-main -lclssdk -lcurl -
lprotobuf -lssl -lcrypto -lboost_thread
```

2. 执行以下运行编译后的示例文件，上传日志。若执行过程报 `error while.....` 错，可参见 常见报错处理 。

```
./sample
```

如果看到类似以下的输出，即代表成功上传日志：

```
statusCode:200 requestId: content: bodyBytes:49 header:key:Content-Length value:0 header:key:Date
value:Wed, 20 Nov 2024 03:37:25 GMT header:key:X-Cls-Requestid value:e7329d6d-9a48-4091-bb0c-
5cea8d1c6f48 header:key:X-Cls-Trace-Id value: header:key:x-cls-requestid value:
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
accesskeyid	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
accessKeysecret	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topic	String	是	日志主题的 ID 信息。

日志上传示例代码

以下代码以 SDK 的 demo 文件中的 sample.cpp 为例，通过调用 SDK 完成日志上传的操作，示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```
#include "cls/producerclient.h"
#include "cls/common.h"
#include "cls/cls_logs.pb.h"
#include "cls/logproducerconfig.pb.h"
#include <string>
#include <iostream>
#include <unistd.h>
#include <cstdlib>
#include "cls/result.h"
#include "cls/error.h"
using namespace tencent_log_sdk_cpp_v2;
using namespace std;


class UserResult : public CallBack
{
public:
    UserResult() = default;
```

```
    ~UserResult() = default;
    void Success(PostLogStoreLogsResponse result) override { std::cout << result.Printf() << std::endl;
}
    void Fail(PostLogStoreLogsResponse result) override
    {
        std::cout << result.Printf() << std::endl;
        // std::cout<<result.loggroup_.ShortDebugString().c_str()<<std::endl;
    }
};


int main()
{
    cls_config::LogProducerConfig config;
    // 填入域名信息，填写指引：https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D，请参
见链接中 API 上传日志 Tab 中的域名
    config.set_endpoint("ap-xxxxxxx.cls.tencentcs.com");

    // 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
    // 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
    // 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
    config.set_acceskeyid(getenv("TENCENTCLOUD_SECRET_ID"));
    config.set_accesskeysecret(getenv("TENCENTCLOUD_SECRET_KEY"));

    // 设置要上传日志的主题 ID，替换为您的 Topic ID
    std::string topic = "xxxxxxxxxxxxxxxxxxxxxxxxx";
    auto client = std::make_shared<ProducerClient>(config);
    auto callback = std::make_shared<UserResult>();

    client->Start();
    // 创建日志对象
    cls::Log log;
    // 设置当前时间为日志时间
    log.set_time(time(NULL));

    // 构造日志数据内容
    auto content = log.add_contents();
    // 设置日志内容的键值信息
    content->set_key("content");
    content->set_value("this my test");

    // 发送日志并获取返回结果
    PostLogStoreLogsResponse ret = client->PostLogStoreLogs(topic, log, callback);
    if(ret.statusCode != 0){
        cout<<ret.content<<endl;
    }
    client->LogProducerEnvDestroy();

    return 0;
}
```

常见报错处理

执行 ./autogen.sh 时，时间超时卡住不动

1. 在 protobuf 目录中，手动下载 googletest 文件。

```
# 下载文件，解压并改名（适应执行代码）
wget https://github.com/google/googletest/archive/release-1.5.0.tar.gz
tar -xzf googletest-release-1.5.0
mv googletest-release-1.5.0 gtest
```

2. 在 protobuf 目录中，继续执行 `./autogen.sh` 即可。

```
# 执行命令
./autogen.sh
```

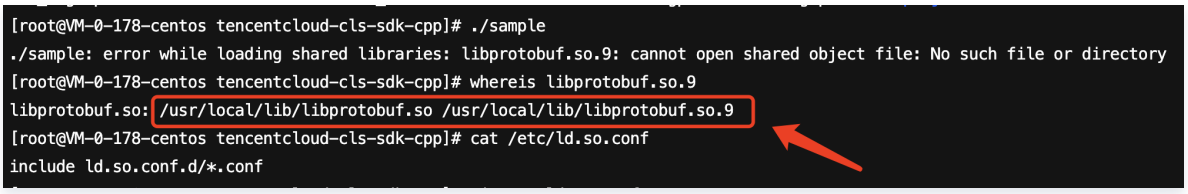
执行可运行文件 `./sample` 时候报错

```
./sample: error while loading shared libraries: libprotobuf.so.9: cannot open shared object file: No such file or directory
```

1. 查看 `protobuf.so` 文件的位置，执行以下命令：

```
whereis libprotobuf.so.9
```

2. 执行 `cat /etc/ld.so.conf` 查看是否有第一步的路径。



3. 如果发现没有的话，把第一步输出的路径，则添加到 `ld.so.conf` 。以上图为例，则需要把 `/usr/local/lib` 添加到 `ld.so.conf` 。
4. 命令行执行 ldconfig 加载文件。

```
ldconfig
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 C++ SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

PHP SDK 上传日志

最近更新时间：2025-05-07 17:18:12

本文介绍如何快速使用日志服务的 PHP SDK 实现日志上传的操作。更多 SDK 使用的详细内容，请参见代码仓库 tencentcloud-cls-sdk-php 。

前提条件

- 创建并获取云 API 密钥信息 accessKeyId 和 accessKey，密钥信息获取请前往 API 密钥管理 。
- 请确保密钥关联的账号具有相应的 SDK 上传日志权限 。

准备开发环境

- 请先安装 PHP，详情请参见 PHP 官网 。
- 日志服务 PHP SDK 支持 PHP 5.6.0及以上版本。您可以执行如下命令检查您已安装的 PHP 版本。

```
php -V
```

执行如下命令安装环境依赖：

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
```

安装 PHP SDK

使用 Composer 安装腾讯云 CLS 的 PHP SDK。

```
composer require tencentcloud/tencentcloud-cls-sdk-php
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API上传日志 Tab 中的域名。
accessKeyId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
accessKey	String	是	云API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topicId	String	是	日志主题的 ID 信息。
token	String	否	临时密钥 token，如需使用临时密钥可填入此参数。临时密钥 token 获取请参见 临时密钥配置指引 。

日志上传示例代码

以 PHP SDK 为例，展示通过调用 SDK 完成日志上传的操作，示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作请参见 配置环境变量 。

```
<?php

require_once __DIR__ . '/vendor/autoload.php'; // 引入 Composer 自动加载器

use TencentCloud\Cls\Models\Request\PutLogsRequest;
use TencentCloud\Cls\Models\LogItem;
use TencentCloud\Cls\Client;
```

```php
use TencentCloud\Cls\TencentCloudLogException;

function putLogs($client, $topicId) {
    $contents = array(
    //设置日志的 key 和 value 内容，前面是 key，后面是 value
        'TestKey' => 'TestContent',
        'test2' => 'helloworld'
    );

    $logItem = new LogItem();
    $logItem->setTime(time()); // 设置日志时间
    $logItem->setContents($contents); // 设置日志内容
    $logItems = array($logItem); // 创建日志项数组

    $request = new PutLogsRequest($topicId, null, $logItems); // 创建请求

    try {
        $response = $client->putLogs($request); // 发送请求
        var_dump($response->getRequestId()); // 输出请求 ID
    } catch (TencentCloudLogException $ex) {
        var_dump($ex); // 捕获并输出腾讯云日志异常
    } catch (Exception $ex) {
        var_dump($ex); // 捕获并输出其他异常
    }
}

// 填入域名信息，请参见链接中 API 上传日志 Tab 中的域名：
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D
$endpoint = 'ap-xxxxxxxx.cls.tencentcs.com';

// 填入云 API 密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
// 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
// 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
$accessKeyId = getenv('TENCENTCLOUD_SECRET_ID');
$accessKey = getenv('TENCENTCLOUD_SECRET_KEY');

// 设置要上传日志的主题 ID，替换为您的 Topic ID
$topicId = 'YOUR_TOPIC_ID';

// 临时密钥的 token，如需使用临时密钥可填入。临时密钥配置指
引:https://cloud.tencent.com/document/product/614/87777
$token = "";

// 创建客户端
$client = new Client($endpoint, $accessKeyId, $accessKey, $token);

// 调用函数发送日志
putLogs($client, $topicId);
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 PHP SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

NodeJS SDK 上传日志

最近更新时间：2025-07-03 17:35:32

本文介绍如何快速使用日志服务的 NodeJS SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-js 。

前提条件

- 创建并获取云 API 密钥信息 accessKeyId 和 accessKey，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK上传日志权限 。

准备开发环境

- 开始接入前，请先下载并安装 Node.js。下载地址及相关操作请参见 Node.js 官网。
- 以下以 CentOS 7系统为例，可使用如下命令安装开发环境。

```
# 安装EPEL仓库
sudo yum install epel-release -y
# 安装Node.js和npm
sudo yum install nodejs npm -y
```

安装 NodeJS SDK

1. 创建并进入项目目录。

```
git clone https://github.com/TencentCloud/tencentcloud-cls-sdk-js.git
cd tencentcloud-cls-sdk-js
```

2. 通过 npm 安装 SDK。

```
npm install
npm install tencentcloud-cls-sdk-js
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
secretId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
secretKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
sourceIp	String	否	来源 IP 地址。
retry_times	integer	是	重试次数。
topicID	String	是	日志主题的 ID 信息。

日志上传代码示例

以下代码以 NodeJS SDK 为例，展示通过调用 SDK 完成日志上传的操作。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```
const { AsyncClient, LogItem, LogGroup, Content, PutLogsRequest } = require('tencentcloud-cls-sdk-js');

// CLS日志服务日志主题ID； 必填参数
```

```
let topicID = "591XXXXXXXXXXXXXXXXXXXXXXXXXXXX"; // 替换为你的日志主题ID

let client = new AsyncClient({
    // 填入域名信息，请参见链接中 API 上传日志 Tab 中的域名：
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D
    endpoint: "ap-xxxxxx.cls.tencentcs.com",

    // 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
    // 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
    // 本示例从环境变量中获取，环境变量配置指引：https://cloud.tencent.com/document/product/614/113851
    secretId: process.env.TENCENTCLOUD_SECRET_ID,
    secretKey: process.env.TENCENTCLOUD_SECRET_KEY,

    // 源IP地址： 选填参数， 为空则自动填充本机IP
    sourceIp: "127.XX.XX.1",
    // 重试次数： 必填参数
    retry_times: 10,
});

async function sendLog() {
    let item = new LogItem();
    item.pushBack(new Content("__CONTENT__", "这是发送的日志内容")); // 修改为你想发送的日志内容
    item.setTime(Math.floor(Date.now() / 1000));

    let loggroup = new LogGroup();
    loggroup.addLogs(item);
    let request = new PutLogsRequest(topicID, loggroup);

    try {
        let data = await client.PutLogs(request);
        console.log("发送日志成功:", data);
    } catch (error) {
        console.error("发送日志失败:", error);
    }
}

// 调用发送日志的函数
sendLog();
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 NodeJS SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

浏览器 JavaScript SDK上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的浏览器 JavaScript SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-js-web 。

前提条件

请确保日志主题开启 匿名访问 。

准备开发环境

- 开始接入前，请先下载并安装 Node.js。下载地址及相关操作请查看 Node.js 官网。
- 以下以 CentOS 系统为例，可使用如下命令安装开发环境。

```
# 安装EPEL仓库
sudo yum install epel-release -y
# 安装Node.js和npm
sudo yum install nodejs npm -y
```

安装 浏览器 JavaScript SDK

1. 创建并进入项目目录。

```
git clone  https://github.com/TencentCloud/tencentcloud-cls-sdk-js.git
cd tencentcloud-cls-sdk-js
```

2. 通过npm 安装 SDK。

```
npm install
npm install tencentcloud-cls-sdk-js-web
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API上传日志 Tab 中的域名。
retry_times	integer	是	重试次数。
Source	String	否	来源 IP 地址。
topicID	String	是	日志主题的 ID 信息。

日志上传代码示例

以下代码以浏览器 JavaScript 为例，展示通过调用 SDK 完成日志上传的操作。

```
const { Log, LogGroup, AsyncClient, PutLogsRequest } = require('tencentcloud-cls-sdk-js-web');

// 创建 AsyncClient 实例
let client = new AsyncClient({
    endpoint: "ap-xxxxxxx.cls.tencentcs.com", // 填入域名信息，填写指引:
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D，请参见链接中 API 上传日志 Tab 中的
域名
    retry_times: 10,
});
```

```
// 创建 LogGroup 实例
let logGroup = new LogGroup();
logGroup.setSource("1.X.XX.XX"); // 替换为您的 IP 地址

// 创建 Log 实例
let log = new Log(Date.now());
log.addContent("hello", "hello world1");
log.addContent("world", "hello world2");

// 将 Log 添加到 LogGroup
logGroup.addLog(log);

// 创建 PutLogsRequest 实例
let request = new PutLogsRequest("8bbXXXXXXXXXXXXXXXXXXXXXXXXXXXX", logGroup); // 替换为您的 topicID

// 上传日志
async function uploadLogs() {
    try {
        let data = await client.PutLogs(request);
        console.log("Logs uploaded successfully:", data);
    } catch (error) {
        console.error("Error uploading logs:", error);
    }
}

// 调用上传日志的函数
uploadLogs();
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 浏览器 JavaScript SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

小程序 JavaScript SDK 上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的小程序 JavaScript SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-js-mini 。

前提条件

请确保日志主题开启 匿名访问 。

准备开发环境

- 开始接入前，请先下载并安装 Node.js。下载地址及相关操作请查看 Node.js 官网。
- 以下以 CentOS 系统为例，可使用如下命令安装开发环境。

```
# 安装EPEL仓库
sudo yum install epel-release -y
# 安装Node.js和npm
sudo yum install nodejs npm -y
```

安装 小程序 JavaScript SDK

1. 创建并进入项目目录。

```
git clone https://github.com/TencentCloud/tencentcloud-cls-sdk-js.git
cd tencentcloud-cls-sdk-js
```

2. 通过 npm 安装 SDK。

```
npm install
npm install tencentcloud-cls-sdk-js-mini
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	域名信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
retry_times	integer	是	重试次数。
Source	String	否	来源 IP 地址。
topicID	String	是	日志主题的 ID 信息。

日志上传代码示例

以下代码以小程序 JavaScript SDK为例，展示通过调用 SDK 完成日志上传的操作。

```
const { Log, LogGroup, AsyncClient, PutLogsRequest } = require('tencentcloud-cls-sdk-js-mini');

// 创建 AsyncClient 实例
let client = new AsyncClient({
    endpoint: "ap-xxxxxx.cls.tencentcs.com", // 填入域名信息，请参见链接中 API 上传日志 Tab 中的域名：
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D
    retry_times: 10,
});
```

```
// 创建 LogGroup 实例
let logGroup = new LogGroup();
logGroup.setSource("1.XX.XX.XX"); // 可替换为您的 IP 地址

// 创建 Log 实例
let log = new Log(Date.now());
log.addContent("hello", "hello world1");
log.addContent("world", "hello world2");

// 将 Log 添加到 LogGroup
logGroup.addLog(log);

// 创建 PutLogsRequest 实例
let request = new PutLogsRequest("8bbfXXXXXXXXXXXXXXXXXXXXXXXXXXXX", logGroup); // 替换为您的日志主题 ID

// 上传日志
async function uploadLogs() {
    try {
        let data = await client.PutLogs(request);
        console.log("Logs uploaded successfully:", data);
    } catch (error) {
        console.error("Error uploading logs:", error);
    }
}

// 调用上传日志的函数
uploadLogs();
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 小程序 JavaScript SDK完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

HarmonyOS SDK 上传日志

最近更新时间：2024-12-16 10:02:23

本文介绍如何快速使用日志服务的 HarmonyOS SDK 实现日志上传的操作。

前提条件

- 已安装 Harmony OS 应用开发环境。更多信息，请参见 HarmonyOS 开发者指南 。
- 创建并获取云 API 密钥信息 access_secret 和 access_key，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。

安装 HarmonyOS SDK

1. 创建 Harmony OS 应用项目。
2. 导入日志服务 Harmony OS 模块，在项目下执行如下命令。

```
ohpm install @farmerx/tencntcloud-cls-sdk-ohos
```

3. 执行完成后，可以在指定的 ets 文件中导入日志服务模块，进行引用并编写代码。

```
import { TencntcloudLog, LogCallback } from "@farmerx/tencntcloud-cls-sdk-ohos"
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	地域信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
access_secret	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
access_key	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
access_token	String	否	临时密钥的 token，如使用临时密钥可填入。
topic_id	String	是	日志主题的 ID 信息。

日志上传示例代码

在您的 Harmony 项目中，可以使用如下的示例代码实现日志上传的能力，示例代码如下所示。
在代码中直接明文使用云 API 密钥（access_key、access_secret）风险较高，为确保安全性，建议使用 临时密钥 进行鉴权。

```
import {LogProducer} from "@farmerx/tencntcloud-cls-sdk-ohos";

// 初始化
let producer = new LogProducer(
    "#[topic_id]",
    "#[access_key]",
    "#[access_secret]",
    "#[access_token]",
    "",
    "#[endpoint]", 100, 100, 2000)

// 设置callback 回掉函数
producer.setCallback((err: Error | null, topicID: string, code: number, request_id: string,
errorMessage: string) => {
```

```
    if (err==null) {
        hilog.info(0x0000,
            'testTagCLS',
            'onLogCallback.topic: %{public}s , code: %{public}d, request_id: %{public}s, errorMessage:
%{public}s',
            topicID, code, request_id, errorMessage
        );
    } else {
        hilog.info(0x0000,
            'testTagCLS',
            'onLogCallback.topic: %{public}s, code: %{public}d, request_id: %{public}s, errorMessage: %
{public}s',
            topicID, code, request_id, err.message
        );
    }
})

// 设置来源 IP
producer.setSourceIp("127.0.0.3")


// reset access key
producer.setAccessKey("#[access_key]", "#[access_secret]", "#[access_token]");

// 同步发送
function syncsencLog() {
    let logs: Record<string, string>[] = [];
    let log: Record<string, string> = {};
    // 添加键值对
    log["key1"] = "value1";
    log["key2"] = "value2";
    log["key3"] = "value3";
    logs.push(log)
    try {
        let rsp = producer.syncSendLogs( "127.0.0.1", logs)
        hilog.info(0x0000, 'testTag', "========%{public}s, %{public}s", rsp.requestId, rsp.rspText)
    } catch (e) {
        // ....
    }
}
// 异步发送
function asyncSendLog() {
    let log: Record<string, string> = {};
    // 添加键值对
    log["key1"] = "value1";
    log["key2"] = "value2";
    log["key3"] = "value3";
    try {
        producer.addLog(log)
    } catch (e) {
        // ....
    }

}

asyncSendLog()
sencLog()
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 HarmonyOS SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

Android SDK 上传日志

最近更新时间：2025-01-25 17:08:12

本文介绍如何快速使用日志服务的 Android SDK 实现日志上传的操作。

前提条件

- 创建并获取云 API 密钥信息 secretId 和 secretKey，密钥信息获取请前往 API 密钥管理 。
- 请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
- 已安装 Android 开发环境。

安装 Android SDK

1. 创建 Android 项目。
2. 在 Android Studio 工程对应模块下的 build.gradle 文件的 dependencies 块中增加以下依赖（build.gradle 通常在 APP 目录下）。

```
implementation(group: 'com.tencentcloudapi.cls', name: 'tencentcloud-cls-sdk-android', version:
'1.0.13')
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	地域信息，填写请参考 可用地域 中 API 上传日志 Tab 中的域名。
secretId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
secretKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topicId	String	是	日志主题的 ID 信息。

日志上传示例代码

在您的 Android 项目中，可以使用如下的示例代码实现日志上传的能力，示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作请参见 配置环境变量 。

```
public static void main(String[] args) {
        String endpoint = "ap-xxxxxx.cls.tencentcs.com";
        // API密钥 secretId，必填
        String secretId = "";
        // API密钥 secretKey，必填
        String secretKey = "";
        // 日志主题ID，必填
        String topicId = "";

        // NetworkUtils.getLocalMachineIP() 获取本地网卡ip，如果不指定，默认填充服务端接收到的网络出口ip
        final AsyncProducerConfig config = new AsyncProducerConfig(endpoint, secretId, secretKey, "",
NetworkUtils.getLocalMachineIP());

        // 构建一个客户端实例
        final AsyncProducerClient client = new AsyncProducerClient(config);

        for (int i = 0; i < 10000; ++i) {
            List<LogItem> logItems = new ArrayList<>();
            int ts = (int) (System.currentTimeMillis() / 1000);
            LogItem logItem = new LogItem(ts);
```

```
            logItem.PushBack(new LogContent("__CONTENT__", "hello world"));
            logItem.PushBack(new LogContent("city", "xxxxxxx"));
            logItem.PushBack(new LogContent("logNo", Integer.toString(i)));
            logItem.PushBack(new LogContent("__PKG_LOGID__",
(String.valueOf(System.currentTimeMillis())));
            logItems.add(logItem);
            client.putLogs(topicId, logItems, result -> System.out.println(result.toString()));
        }
        client.close();
}
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 Android SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

IOS SDK 上传日志

最近更新时间：2025-01-08 17:30:22

本文介绍如何快速使用日志服务的 IOS SDK 实现日志上传的操作。更多 SDK 使用的详细内容，可见代码仓库 tencentcloud-cls-sdk-ios 。

前提条件

- 创建并获取云 API 密钥信息 accessKeyId 和 accessKey，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK 上传日志权限
- 已安装 iOS 开发环境。更多信息，请参见 Apple Developer 。

安装 iOS SDK

1. 创建 iOS 项目。
2. 导入头文件。

```
#import <TencentCloudLogProducer.h>
```

3. 在项目文件夹中，创建 Podfile 文件，并输入导入日志服务 iOS SDK 依赖包的命令。

```
pod 'TencentCloudLogProducer/Core', '1.1.2'
```

请求参数

变量	类型	是否必填	说明
endpoint	String	是	地域信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
secretId	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
secretKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topicId	String	是	日志主题的 ID 信息。

日志上传示例代码

在您的项目中，可以使用如下的示例代码实现日志上传的能力，示例代码如下所示。
不建议将云 API 密钥信息明文存储至工程代码中，可通过环境变量动态获取云 API 密钥信息，具体操作，请参见 配置环境变量 。

```
// 填入域名信息，请参见链接中 API 上传日志 Tab 中的域名：
https://cloud.tencent.com/document/product/614/18940#.E5.9F.9F.E5.90.8D
NSString* endpoint = @"project's_endpoint";

// 填入云API密钥信息。密钥信息获取请前往：https://console.cloud.tencent.com/cam/capi
// 并请确保密钥关联的账号具有相应的日志上传权限，权限配置指引：
https://cloud.tencent.com/document/product/614/68374#.E4.BD.BF.E7.94.A8-
api-.E4.B8.8A.E4.BC.A0.E6.95.B0.E6.8D.AE
// 建议从环境变量中获取密钥信息
NSString* accesskeyid = @"your_accesskey_id";
NSString* accesskeysecret = @"your_accesskey_secret";

// 设置要上传日志的主题 ID，替换为您的 Topic ID
NSString* topic_id = @"your_topic";
```

```
    LogProducerConfig *config = [[LogProducerConfig alloc] initWithCoreInfo:[endpoint] accessKeyID:
[accesskeyid] accessKeySecret:[accesskeysecret];
    [config SetTopic:topic_id];
    [config SetPackageLogBytes:1024*1024];
    [config SetPackageLogCount:1024];
    [config SetPackageTimeout:3000];
    [config SetMaxBufferLimit:64*1024*1024];
    [config SetSendThreadCount:1];
    [config SetConnectTimeoutSec:10];
    [config SetSendTimeoutSec:10];
    [config SetDestroyFlusherWaitSec:1];
    [config SetDestroySenderWaitSec:1];
    [config SetCompressType:1];

        //callback若传入空则不会回调
    LogProducerClient *client; = [[LogProducerClient alloc] initWithClsLogProducer:config
callback:nil];
        Log* log = [[Log alloc] init];
    [log PutContent:@"cls_key_1" value:@"cls_value_1"];
    [log PutContent:@"cls_key_1" value:@"cls_value_2"];
    LogProducerResult result = [client PostLog:log];
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 iOS SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

Flutter SDK 上传日志

最近更新时间：2024-12-16 10:02:23

本文介绍如何快速使用日志服务的Flutter SDK 实现日志上传的操作。

前提条件

- 创建并获取云 API 密钥信息 accessKey 和 accessSecret，密钥信息获取请前往 API 密钥管理 。
- 并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
- 已安装 Flutter 环境。更多信息请参见 Install Flutter 。

安装 Flutter SDK

1. 创建 Flutter 项目。
2. 在项目的根目录下执行如下命令添加依赖。

```
flutter pub add tencentcloud_cls_sdk_dart
```

3. 安装完成后，在您的 Dart 文件中导入日志服务模块。

```
import 'package:tencentcloud_cls_sdk_dart/tencentcloud_cls_sdk_dart.dart';
```

请求参数

变量	类型	是否必填	说明
host	String	是	地域信息，填写请参见 可用地域 中 API 上传日志 Tab 中的域名。
accessKey	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
accessSecret	String	是	云 API 密钥信息，密钥信息获取请前往 API 密钥管理 。并请确保密钥关联的账号具有相应的 SDK 上传日志权限 。
topicId	String	是	日志主题的 ID 信息。
accessToken	String	否	临时密钥的 token，如使用临时密钥可填入。

日志上传示例代码

在您的 Flutter 项目中，可以使用如下的示例代码实现日志上传的能力，示例代码如下所示。
在代码中直接明文使用云 API 密钥（accessKey、accessSecret）风险较高，为确保安全性，建议使用 临时密钥 进行鉴权。

```
import 'package:flutter/material.dart';
import 'package:tencentcloud_cls_sdk_dart/tencentcloud_cls_sdk_dart.dart';

Future<void> main() async {
  await RustLib.init();

  runApp(const MyApp());
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
```

```
}

class _MyAppState extends State<MyApp> {
  LogProducer? _logProducer;
  String _consoleText = '';


  @override
  void initState() {
    super.initState();

    if (!mounted) return;
    print("application started.");
  }

  void print(String message) {
    setState(() {
      _consoleText += message;
      _consoleText += '\n';
    });
  }

  void _initProducer() async {
    _logProducer = LogProducer(
      topicId: '',
      accessKey: '',
      accessSecret: '',
      accessToken: '',
      host: 'ap-XXXXXX.cls.tencentcs.com',
      waitSendLogQueue: 1000, addLogQueue: 1000, lingerMs: 2000);

    print('init producer client success');
  }

  void _callback() async {
    if (!check()) {
      return;
    }
    _logProducer?.setCallback(dartCallback: (topicId, requestId, status, errorMessage) =>
print('$requestId!, $status, $topicId, $errorMessage'));
    print('init callback success');
  }

  void _sendLog() async {
    if (!check()) {
      return;
    }
    _logProducer?.addLog(log: { 'hello': 'world' });
  }

  bool check() {
    if (null == _logProducer) {
      print('you should init producer first.');
      return false;
    }
    return true;
  }

  @override
```

```
Widget build(BuildContext context) {
  Color color = Theme.of(context).primaryColor;

  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text('tencent cloud cls flutter sdk demo')),
      body: Column(
        children: [
          _buildConsoleText(),
          _buildButton(color, 'init', _initProducer),
          _buildButton(color, 'set callback', _callback),
          _buildButton(color, 'send', _sendLog),
        ],
      ),
    ),
  );
}

Widget _buildConsoleText() {
  return Row(mainAxisSize: MainAxisSize.max, mainAxisAlignment: MainAxisAlignment.start, children: [
    Expanded(
        flex: 1,
        child: Container(
          margin: const EdgeInsets.only(bottom: 18),
          padding: const EdgeInsets.all(6),
          height: 140,
          decoration: BoxDecoration(
              border: Border.all(
                color: Colors.grey,
                width: 0.67,
              ),
              color: Colors.black),
          child: SingleChildScrollView(
            scrollDirection: Axis.vertical,
            child: Text(
              _consoleText,
              style: const TextStyle(
                  color: Colors.white,
                  fontSize: 12,
                  letterSpacing: 2,
                  wordSpacing: 2,
                  fontFeatures: [FontFeature.tabularFigures()]),
            ),
          ),
        ))
  ]);
}

Widget _buildButton(Color color, String label, VoidCallback? onPressed) {
  return Row(
    mainAxisSize: MainAxisSize.max,
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Expanded(
          flex: 1,
          child: Container(
            margin: const EdgeInsets.only(left: 16, top: 8, right: 16),
            child: TextButton(
                onPressed: onPressed,
```

```
                    style: ButtonStyle(
                        shape: WidgetStateProperty.all(RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12))),
                        side: WidgetStateProperty.all(BorderSide(color: color, width: 0.67)),
                        backgroundColor: WidgetStateProperty.all(Colors.transparent),
                        padding:
                        WidgetStateProperty.all(const EdgeInsets.only(left: 12, top: 6, right: 12,
bottom: 6))),
                    child: Text(
                      label,
                      style: TextStyle(fontSize: 22, fontWeight: FontWeight.w400, color: color),
                    )),
            )),
      ],
    );
  }

}
```

结语

通过以上步骤，您可以快速使用腾讯云 CLS 的 Flutter SDK 完成日志的上传操作。如遇到任何问题，请 联系我们 获取帮助。

通过 Kafka 数据订阅跨账号同步日志

最近更新时间：2024-11-05 12:10:52

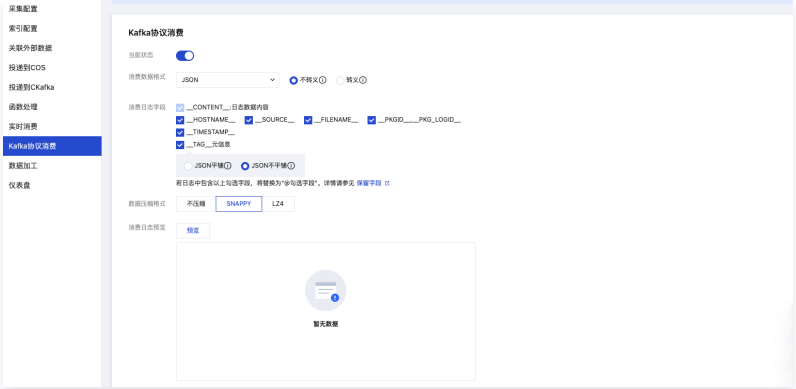
简介

在一些场景中， 我们可能需要将日志在不同账号的日志主题之间进行迁移与同步。 本文将介绍如何通过 Kafka 数据订阅功能实现将账号 A 的 a 日志主题中的日志同步至账号 B 的 b 日志主题。

操作步骤

1. 登录 CLS 控制台，在**日志主题**管理页中，找到需要被迁移或同步的 a 日志主题， 单击 a 日志主题的名称进入日志主题详情页。
2. 在日志主题详情页中，找到并单击 **Kafka 协议消费**页签，如下图所示：

3. 将 Kafka 协议消费当前状态开启。
4. 按照下图配置 Kafka 协议消费。



5. 完成配置后，单击**确定**后，您会看到该日志主题作为 Kafka 消费端的 Kafka Topic ID，Kafka 服务域名。将 Kafka Topic ID 与 Kafka 服务域名记录下来。

> **注意：**
> 若 a 日志主题与 b 日志主题地域不同，请记录外网域名； 若地域相同，建议记录内网域名。

6. 查看 a 日志主题所在的日志集， 并记录下日志集 ID。
7. 前往 API 密钥管理，查看 A 账号的密钥 ID 与密钥 KEY，并记录下来。
8. 登录 B 账号， 找到需要迁移或同步的 b 日志主题， 单击 b 日志主题的名称进入日志主题详情页。
9. 在日志主题详情页中，找到并单击**采集配置**页签。
10. 单击 **Kafka 数据订阅配置**入口，如下图所示：

11. 在集群配置步骤，进行如下配置：
    - 集群类型：自建 Kafka
    - 服务地址：a 日志主题消费端的 Kafka 服务域名。 若 a 日志主题与 b 日志主题不同地域，请使用外网域名； 若地域相同，建议使用内网域名。
    - 协议类型：sasl_plaintext
    - 鉴权机制：plaintext
    - 用户名：a 日志主题的日志集 ID
    - 密码：A 账号密钥 ID#A 账号密钥 KEY
    - Kafka Topic 列表：a 日志主题消费端的 Kafka Topic ID
    - 消费组：为空
    - 起始位置：最晚
12. 完成集群配置后，可单击预览查看是否成功消费到 a 日志主题的日志。
13. 单击**下一步**，按需配置订阅规则，索引配置。详情可参见 配置 Kafka 数据订阅任务 。
14. 完成 b 日志主题的 Kafka 数据订阅任务创建后， 即可进行以下日志检索分析，仪表盘，监控告警等功能。
    - 检索分析
    - 监控告警
    - 仪表盘
    - 数据加工

检索分析
云产品账单数据分析

最近更新时间：2025-01-24 18:10:52

简介

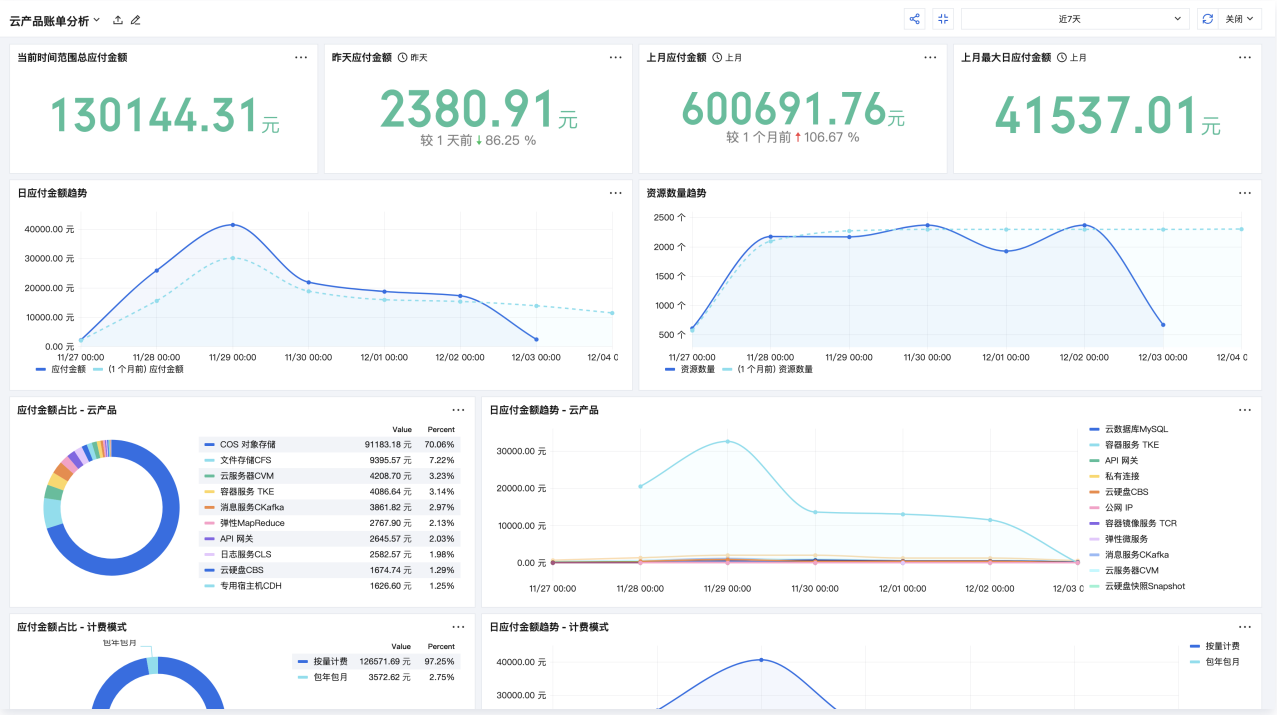
腾讯云费用中心支持将账单存储到 COS，此后可以从 COS 导入账单到 CLS。账单数据接入后，可使用 CLS 提供的预置仪表盘分析全部云产品的成本分布与趋势。本文档将说明从账单接入到查看分析仪表盘的流程，基础步骤如下：

- 存储账单到 COS 桶
- 导入账单数据到 CLS
- 使用预置仪表盘分析账单数据
- 配置云产品费用监控告警

您也可通过 云产品账单分析 Demo，快速体验云产品账单分析。

预置仪表盘

CLS 已提供预置仪表盘分析云产品的花费。云产品账单分析 包括昨天应付金额、上月应付金额、各云产品应付金额占比等。
在仪表盘右上角单击**编辑仪表盘**可基于预置仪表盘进行编辑。



操作步骤

存储账单到 COS 桶

1. 登录腾讯云控制台，在 费用中心 选择 成本管理 > 消耗账单，开启右上角的**账单存储**。

2. 账单存储配置中选择**消耗账单-日明细**或**消耗账单-月明细**（推荐），选择存储桶（建议新建一个空存储桶），配置后以天或月为周期，自动存储昨天或上月账单。

> **注意：**
> - **消耗账单-日明细**：每天生成前一天的账单文件，**账单数据不完整**，不包含后付费日结、后付费月结产品的账单。
> - **消耗账单-月明细**：每月2号左右生成前一个月的账单文件，**账单数据完整**，包含所有计费类型。

3. （非必要，如果不需要导入历史账单则跳过）在 成本管理 > 消耗账单 页面，单击右上角**账单包下载**。

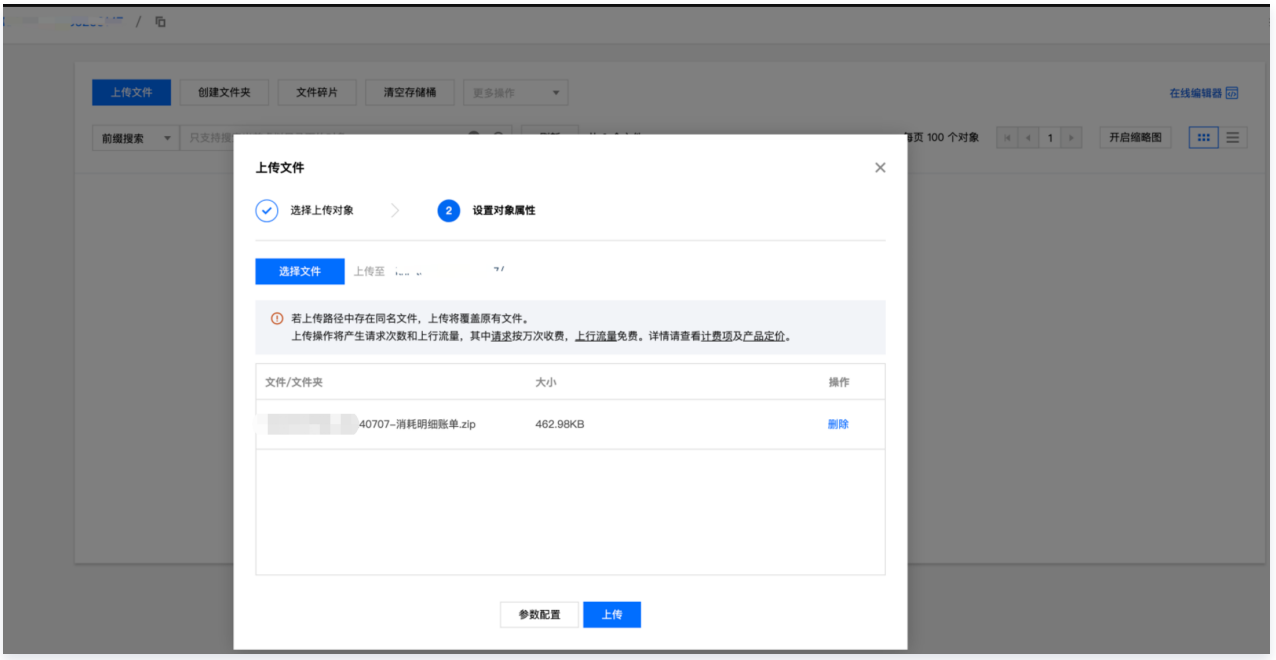
> **注意：**
> 使用日明细和月明细时，导入历史账单的范围有区别，例如：10月15号配置了存储新增的消耗账单到 COS 桶，并打算上传历史账单，参考下表。

选择存储的账单	自动存储到 COS 的账单	需手动上传 COS 的历史月份账单
消耗账单-日明细	从第二天开始生效，包含10月15号及以后的账单数据	需下载10月的历史账单包（内容包含1-14号的账单）
消耗账单-月明细	从第二个月开始生效，包含10月1号及以后的数据	无需下载10月的历史账单

选择历史月份的账单下载。

4. （可选，如果不需要导入历史账单则跳过）打开步骤2中选择的 COS 桶，上传步骤3中下载的历史账单包。

> **注意：**
> 历史账单包里包含 L0 – L2三种账单，仅需要挑选出所有 L2账单汇总后打包为 zip 格式上传。



导入账单数据到 CLS

1. 登录 日志服务控制台，新建账单 日志主题。主题类型选择标准主题，保存时间根据想要分析的时间范围决定，建议180天。

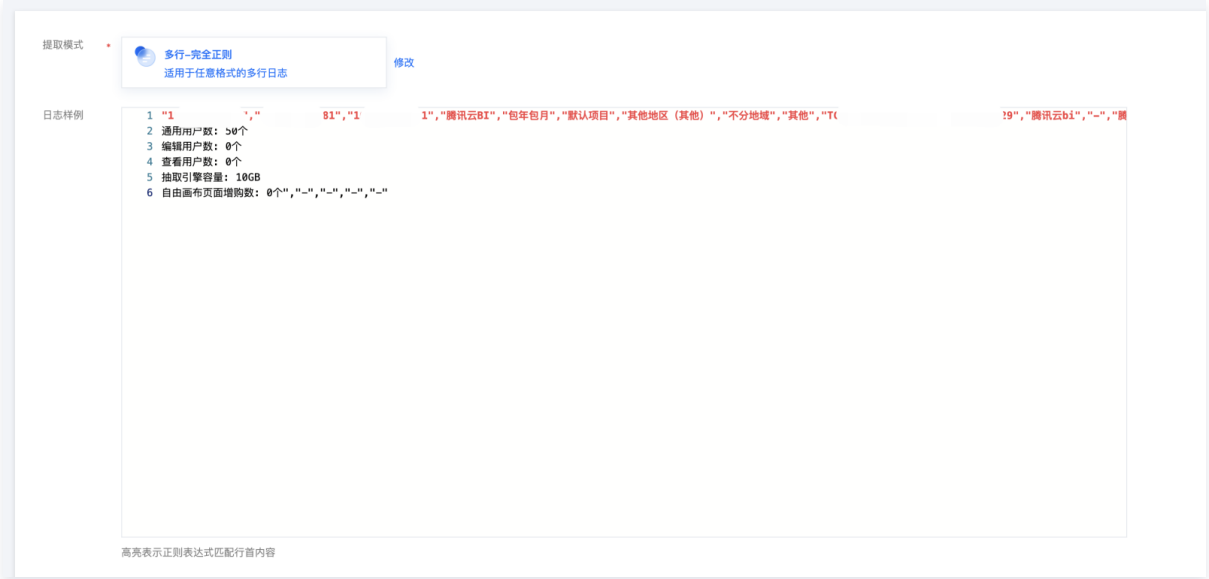
2. 创建完成后单击主题名称，进入详情。选择**采集配置**，新增 **COS 数据导入配置**。

3. 任务类型选择**持续导入**，选择存储账单的 COS 桶，选择 zip 压缩后预览。预览后建议复制一条账单记录，单击**下一步**。

> **注意：**
> COS 持续导入目前为白名单功能，请联系 技术支持 。

- 任务类型：选择持续导入。
- 存储桶地域：选择 COS 存储桶的地域。
- COS 存储桶：选择账单的 COS 存储桶。
- 文件前缀：前缀可为空，默认采集存储桶中的所有文件。
- 压缩模式：选择 zip 压缩。

4. 解析规则配置。
   - 提取模式：多行-完全正则。
   - 日志样例：粘贴预览时复制的记录。



   - 行首正则：选择手动输入。

```
^"\d+".*
```

   - 提取正则：

```
"([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","
([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","
([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","
([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","
([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","([^"]*)","
([^"]*)",?(.*)
```

- 抽取结果（复制下表 Key 列）。

Key	Value	解释
PayerUIN	1000000111	支付者 UIN
UserUIN	1000000111	使用者 UIN
OperatorUIN	5275122700	操作者 UIN
BusinessCodeName	云硬盘快照 Snapshot	产品名称
PayMode	按量计费	计费模式
ProjectName	默认项目	项目名称
RegionName	华北地区（北京）	地域
ZoneName	其他	可用区
RegionTypeName	国内	国内国际
ResourceId	snap-r626n9dm	资源 ID
ResourceName	未命名	实例名称
InstanceTypeName	-	实例类型
ProductCodeName	云硬盘快照	子产品名称
ActionType	按量计费小时结	消耗类型
BillId	202409123163992352162	交易 ID
OrderId	-	订单 ID
Month	2024-09	消耗月份
PayTime	2024-09-13 23:43:00	扣费时间
FeeBeginTime	2024-09-13 21:00:00	开始使用时间
FeeEndTime	2024-09-13 21:59:59	结束使用时间
ComponentCodeName	存储空间	组件类型
ItemCodeName	云硬盘快照-存储空间	组件名称
SinglePrice	0.00000005	刊例价
ContractPrice	0.00000002	优惠后单价

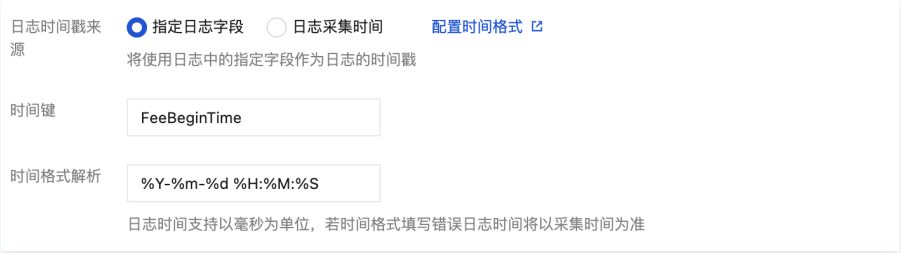
SinglePriceUnit	元/GiB/秒	价格单位
UsedAmount	5.31000000	用量
UsedAmountUnit	GiB	用量单位
TimeSpan	3600.00000000	使用时长
TimeUnitName	秒	时长单位
ReserveDetail	-	备注属性
Cost	0.00095580	原价(元)
OriginalCostWithRI	0.00000000	预留实例抵扣原价(元)
RiTimeSpan	0.00000000	预留实例抵扣时长
OriginalCostWithSP	0.00000000	节省计划抵扣原价(元)
Discount	0.44870800	折扣率
BlendedDiscount	0.44870800	混合折扣率
RealTotalCost	0.00042888	优惠后总价
VoucherPayAmount	0.00000000	优惠券支付(元)
TransferPayAmount	0.00000000	分成金支付(元)
IncentivePayAmount	0.00042888	赠送金支付(元)
CashPayAmount	0.00000000	现金支付(元)
ConfigDesc	-	配置描述
RealTotalMeasure	\N	原始用量/时长
DeductedMeasure	\N	抵扣用量/时长(含资源包)
PriceInfo	-	价格属性
AssociatedOrder	-	关联单据 ID
Tag	-	分账标签的字符串

- **日志时间戳来源：选择指定日志字段。**
- **时间键：**

```
FeeBeginTime
```

- **时间格式解析：**

```
%Y-%m-%d %H:%M:%S
```



5. 索引配置，使用默认配置，自动开索引。保存后完成账单数据采集。

使用预置仪表盘分析账单数据

登录 CLS 控制台，打开预置仪表盘 云产品账单分析，开始分析账单。

配置云产品费用监控告警

监控云产品每天的账单，对比今天与昨天的账单，如果费用增加超过阈值，则发送告警邮件。邮件内容展示费用增加的云产品以及今天的账单、昨天的账单、同比增长率和增长的费用。操作步骤如下：

步骤1：创建告警通知内容模板

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警 > 通知内容模板**，进入通知内容模板管理页面。
3. 单击**新建**，并在“新建通知内容模板”中，填写如下信息：

- 模板名称：输入模板名称。
- 语言：支持中文及英文，将决定内容模板中**固定部分**使用的语言。
- 通知内容：按渠道分别配置告警触发及恢复时的内容模板，可使用 告警通知变量 动态渲染通知内容。通知内容邮件示例如下：
  - **告警邮件标题：**

```
云产品账单突增告警-{{range .QueryResult[0]}} ●{{.BusinessCodeName}} {{- end}}
```

- **告警邮件内容：**

```
费用增加的云产品如下：
{{range .QueryResult[0]}}
● 云产品：{{.BusinessCodeName}} | 今日：{{.today}} 元 | 昨日：{{.yesterday}} 元 | 同比增长：{{.ratio}}% | 费用差值：{{.data}} 元
{{- end}}
```

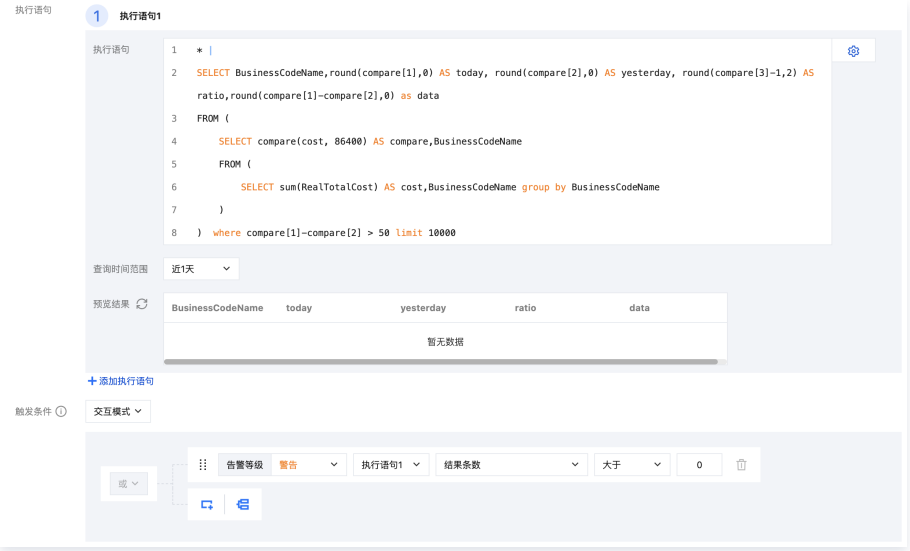
4. 单击**保存**，完成通知内容模板配置。
5. 在**通知渠道组**中选择需要使用的内容模板，操作步骤请参见 管理通知渠道组 。

步骤2：创建告警策略

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警 > 告警策略**，进入告警策略管理页面。
3. 单击**新建**，创建告警策略。
   - **监控对象**选择云产品账单日志主题。
   - **监控任务**的执行语句如下，查询时间范围选择近一天。

```
* | SELECT BusinessCodeName,round(compare[1],0) AS today, round(compare[2],0) AS yesterday, round(compare[3]-1,2) AS ratio,round(compare[1]-compare[2],0) as data FROM ( SELECT compare(cost, 86400) AS compare,BusinessCodeName FROM ( SELECT sum(RealTotalCost) AS cost,BusinessCodeName group by BusinessCodeName ) ) where compare[1]-compare[2] > 50 limit 10000
```

   - **触发条件**为执行语句结果条数大于0，即存在增长数额大于50元的云产品。



其余配置可根据需求选择，详情请参见 配置告警策略 。

CDN 访问日志分析

最近更新时间：2025-02-14 11:21:53

简介

内容分发网络（Content Delivery Network，CDN） 是非常重要的互联网基础设施，用户可以通过 CDN，快速地访问网络中各种图片，视频等资源。在访问过程中，CDN 会产生大量的访问日志数据。这些日志详细记录了用户的访问请求，包括**访问时间**、**访问的资源**、**用户的地理位置**、**设备信息**以及**请求的状态**等信息（完整日志字段说明请参见 日志字段说明 ）。CLS 日志服务联合 CDN，支持将 CDN 域名访问日志实时采集至 CLS，并基于 CLS 的日志分析能力，帮助您完成 CDN 的**质量和性能的分析**、**错误诊断**、**客户端分布**、**用户行为分析**等。通过本篇实践，您可以详细的了解如下内容：

- 采集 CDN 域名访问日志
- 查看 CDN 域名访问分析仪表盘
- 检索分析 CDN 域名访问日志
- 实践场景案例

您也可通过以下 Demo，快速体验 CLS 与 CDN 的强大结合。

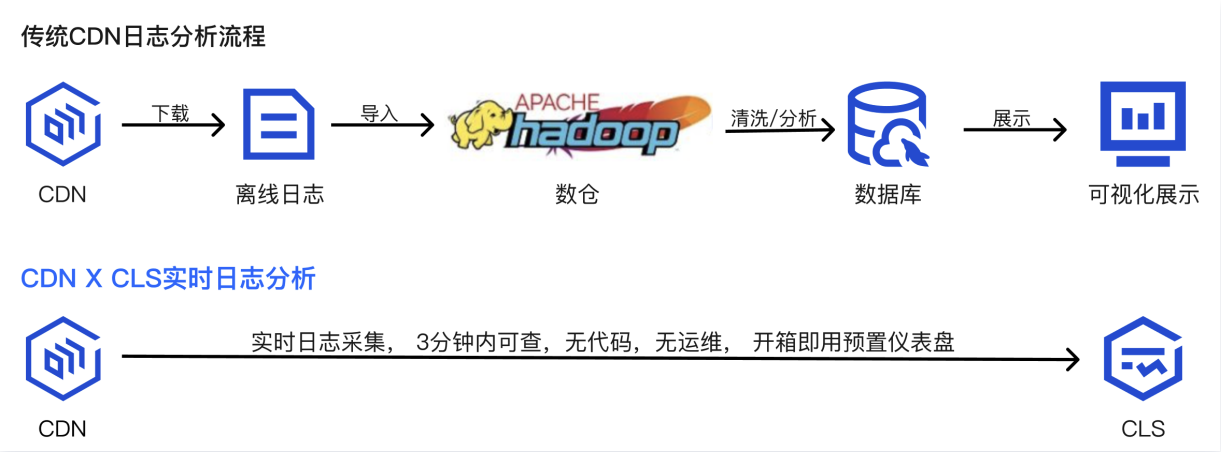
- 访问质量监控分析仪表盘
- 用户行为分析仪表盘
- 检索分析 CDN 访问日志

功能优势

- 日志实时分析：

  CLS 实时采集 CDN 的域名访问日志，日志数据延迟不超过3分钟。而传统 CDN 日志分析场景中，往往需要下载日志进行离线分析，延迟通常在24小时以内。
- 分析报表开箱即用：

  CLS 为 CDN 访问日志分析场景提供了开箱即用的访问分析仪表盘，仪表中包含 CDN **质量和性能**、**错误诊断**、**客户端分布**，以及**用户行为**等分析看板。而在传统 CDN 日志分析场景中，需下载日志至离线，再上传至数据仓库，然后在数据仓库进行一系列的数据清洗和数据模型定义。这一过程繁琐又消耗较多人力成本。



采集 CDN 域名访问日志

步骤1：进入 CDN 云产品中心

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**云产品中心**，进入云产品中心页面。
3. 在**云产品日志中心**，选择并单击**内容分发网络 CDN 中心**。

步骤2：开启域名访问日志采集

1. 在域名列表中，找到并勾选一个或多个目标域名。

2. 单击**开启日志采集**，在下拉选项中选择开启境内或境外域名的访问日志。

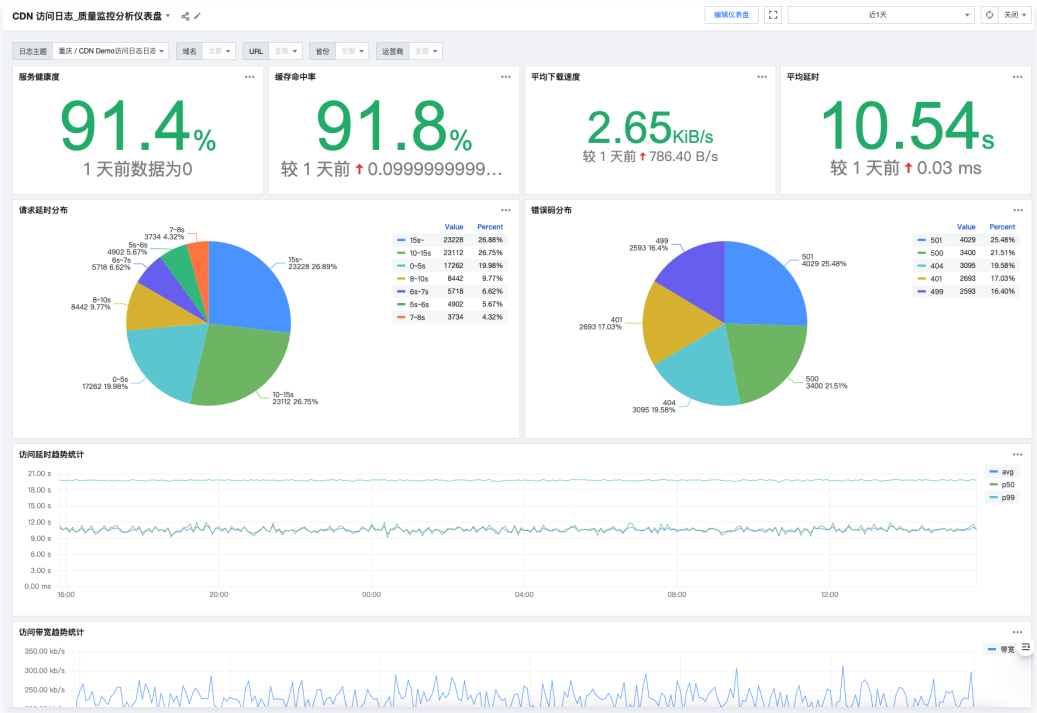
> **注意：**
> - ECDN 域名与境内域名不支持开启境外访问日志采集。
> - 境外域名不支持开启境内访问日志采集。

3. 在开启日志采集弹窗中，选择或 创建目标日志主题 ，并单击**确认**。

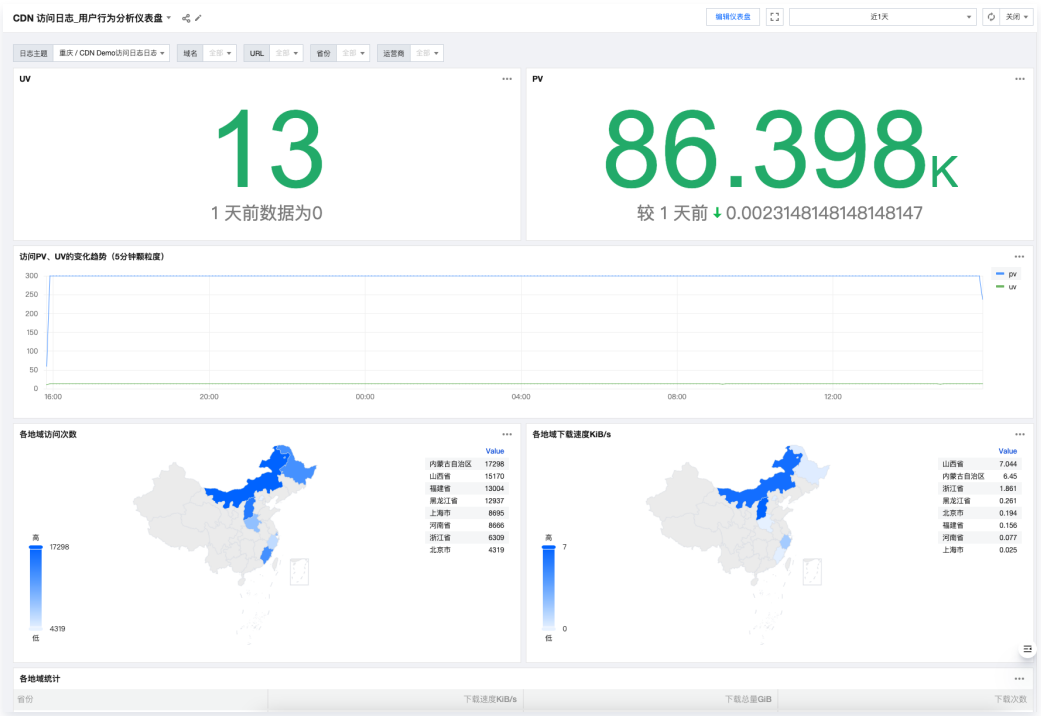
查看 CDN 域名访问分析仪表盘

完成 开启域名访问日志采集 后，CLS 提供两款开箱即用的预置仪表盘：

- **访问质量监控分析仪表盘：**可视化展示 CDN 域名的访问质量情况（如缓存命中率、延时等），助力运维排障场景。



- **用户行为分析仪表盘：**可视化展示用户的访问行为（如 PV、UV 等），助力运营分析场景。

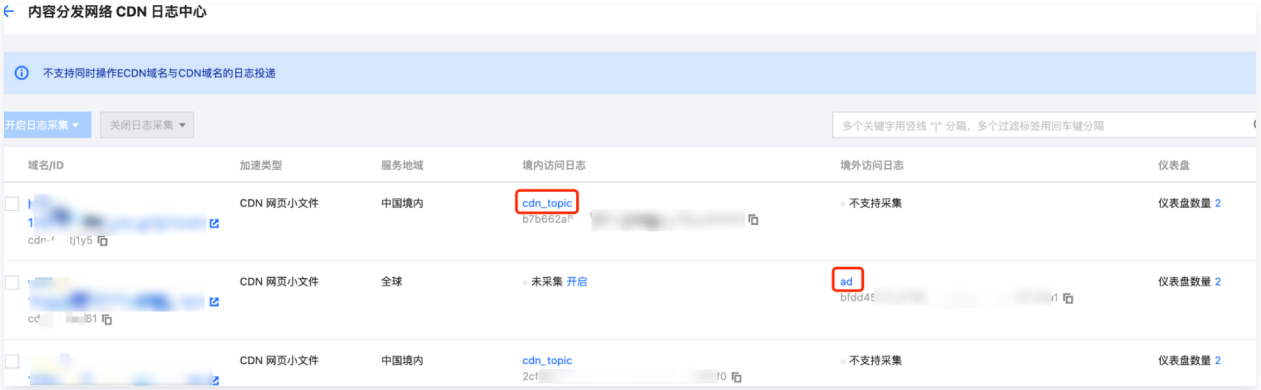


您可通过以下操作操作查看以上 CDN 预置仪表盘：

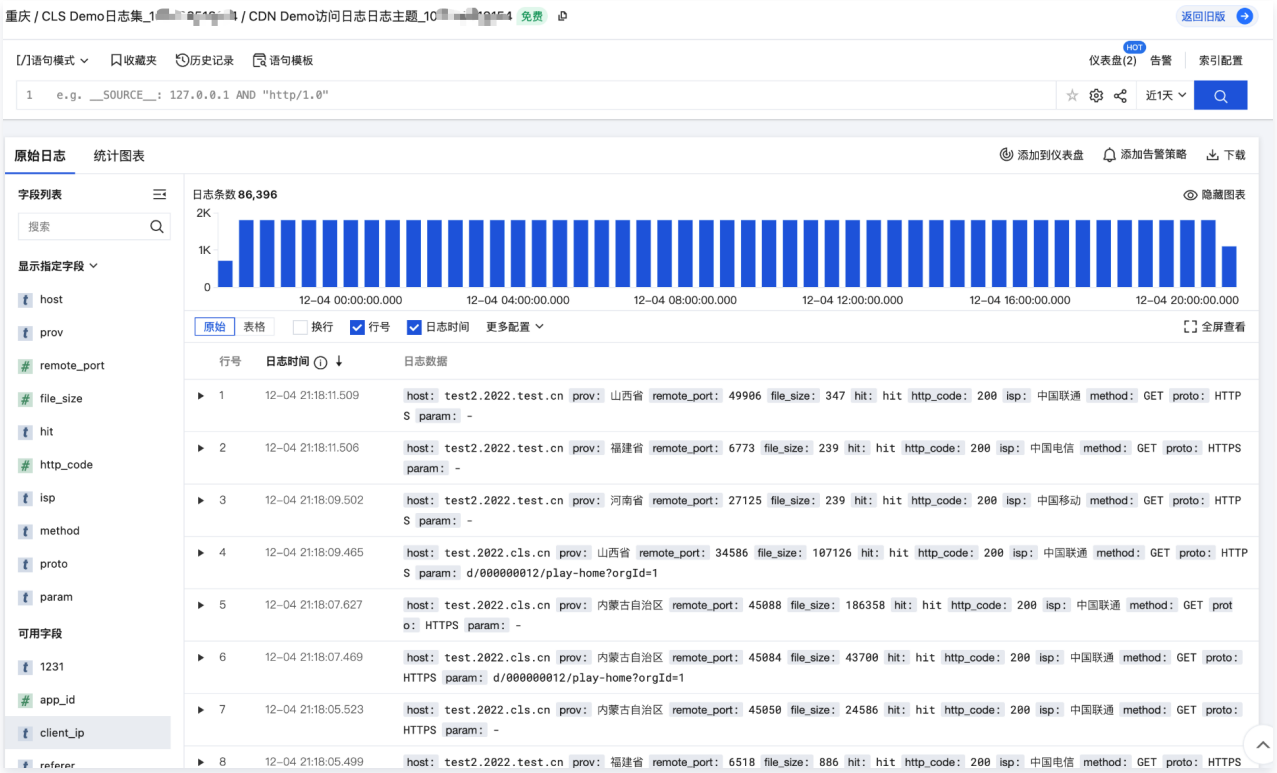
1. 在域名列表中，找到已开启访问日志采集的域名。
2. 单击**仪表盘数量**，在下拉列表中选择**质量监控分析仪表盘**或**用户行为分析仪表盘**，单击后跳转仪表盘。

检索分析 CDN 域名访问日志

1. 完成开启域名访问日志采集后，在域名列表中，找到已开启访问日志采集的域名，单击境内或境外访问日志所在的目标日志主题名称。



2. 单击后将跳转至日志检索页，即可检索分析访问日志。日志字段的详细介绍请参见 日志字段说明 。



日志字段说明

针对 CDN 访问日志中的字段解释，可参见下表：

字段名	说明
app_id	腾讯云账号 APPID
client_ip	客户端 IP
file_size	文件大小
hit	缓存 HIT / MISS，在 CDN 边缘节点命中、父节点命中均标记为 HIT
host	域名
http_code	HTTP 状态码
isp	运营商
method	HTTP Method
param	URL 携带的参数
proto	HTTP 协议标识
prov	运营商省份
referer	Referer 信息，HTTP 来源地址
request_range	Range 参数，请求范围
request_time	响应时间（毫秒），指节点从收到请求后响应回包所花费的时间
request_port	客户端与 CDN 节点建立连接的端口。若无，则为 –
rsp_size	返回字节数

time	请求时间，UNIX 时间戳，单位为：秒
ua	User-Agent 信息
url	请求路径
uuid	请求的唯一标识
version	CDN 实时日志版本

实践场景案例

您可以基于 CDN 访问日志配置异常监控告警，实时监控 CDN 访问流量中发生的异常。以下提供了两个案例：

案例1：针对99%的延时大于100ms 进行告警，并且在告警信息中展示受影响域名、url、client_ip，以便快速判断错误情况。

1. 登录 日志服务控制台 ，并进入 告警策略 管理页面，单击**新建**，进入告警策略创建页。

2. 在告警策略页中，配置如下内容：
   - 基本信息
     - 告警策略名称：CDN 访问延迟告警
     - 启用状态：启用
   - 监控对象：选中在 采集 CDN 访问日志 步骤中创建的日志主题。
   - 监控任务
     - 执行语句：输入以下语句，时间范围选择15分钟，统计近15分钟内的99%延时。

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

     - 触发条件： 配置如下，即99%延时大于100ms 时，满足告警条件。

```
$1.p99 > 100
```

     - 执行周期：固定频率，每1分钟执行一次。
   - 多维分析：在告警信息中展示受影响的域名、客户端 IP、url，帮助开发人员快速定位问题。

- 通知渠道组： 通过关联通知渠道组，设置发送通知的方式及对象，支持短信、邮件、电话、微信、企业微信、钉钉、飞书、自定义接口回调（webhook）等通知方式。详情参见 管理通知渠道组 。

3. 告警触发后，将通过微信、企业微信、短信第一时间获取关键信息。

案例2：资源访问错误激增告警，当同比增数超过一定阈值时，触发告警通知。

1. 参考 案例1，登录 日志服务控制台，并进入 告警策略 管理页面，单击**新建**，进入告警策略创建页。
2. 在执行语句中输入以下语句，时间范围选择1分钟，统计最近1分钟相比上1分钟的错误数增量。

```
http_code:>=400 | select compare[1]-compare[2] as errorCNTRise from (select compare(errorCNT,60) as
compare from(select count(*) as errorCNT))
```

3. 触发条件如下，即错误数增量大于100时，满足告警条件。

```
$1.errorCNTRise > 100
```

CLB 访问日志分析

最近更新时间：2024-12-13 15:05:42

简介

负载均衡（Cloud Load Balancer，CLB）是腾讯云提供的高性能网关产品，用于帮助用户通过流量分发扩展应用系统的对外服务能力。CLB 在处理用户请求时，会产生大量的访问数据日志，这些日志详细记录了用户的访问请求，包括**访问时间**、**请求方式**、**源IP地址**、**响应状态**等关键信息（完整日志字段说明请参见 日志字段说明）。

腾讯云 CLS 日志服务联合 CLB，支持将 CLB 访问日志 实时采集至 CLS，并基于 CLS 的日志分析能力，帮助您完成从 CLB 访问日志中**监控客户端请求**、**辅助排查异常问题**、**分析梳理用户行为**，从而为排障和运营决策提供数据支持。通过本篇实践，您可以详细了解以下信息：

- 采集 CLB 负载均衡访问日志
- 查看 CLB 负载均衡访问分析仪表盘
- 检索分析 CLB 负载均衡访问日志
- 实践场景案例

您也可通过 CLB 访问分析仪表盘 Demo，快速体验 CLS 与 CLB 的强大结合。

功能优势

CLS 为 CLB 访问日志分析场景提供了开箱即用的访问分析仪表盘，仪表中包含 **CLB 访问流量分析**、**状态码分布**，以及**用户行为**等分析看板。

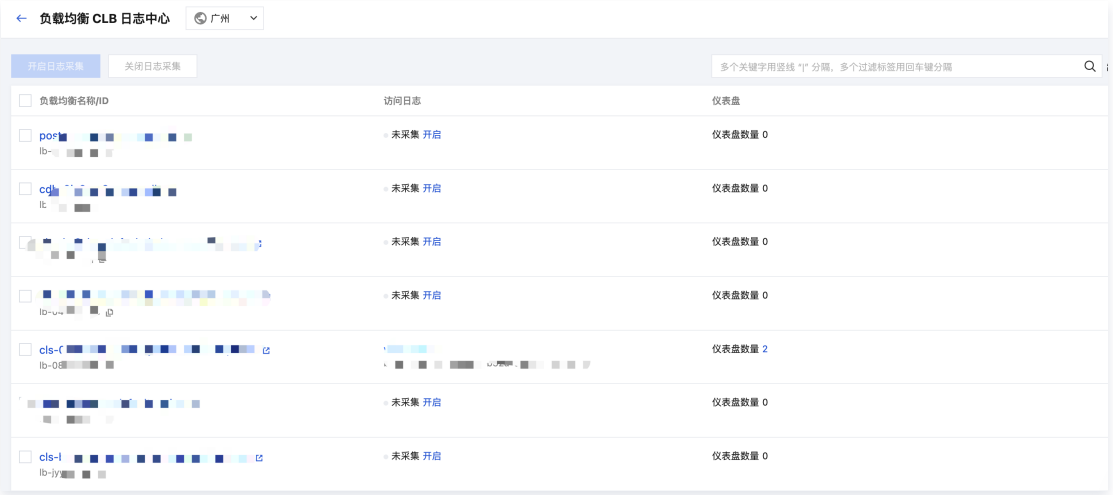
采集 CLB 负载均衡访问日志

步骤1：进入 CLB 云产品中心

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，单击**云产品中心**，进入云产品中心页面。
3. 在**云产品日志中心**，找到**负载均衡 CLB**，并单击进入**负载均衡 CLB 日志中心**。

步骤2：开启负载均衡访问日志采集

1. 在负载均衡实例列表中，找到并勾选一个或多个目标负载均衡实例。



2. 单击**开启日志采集**。

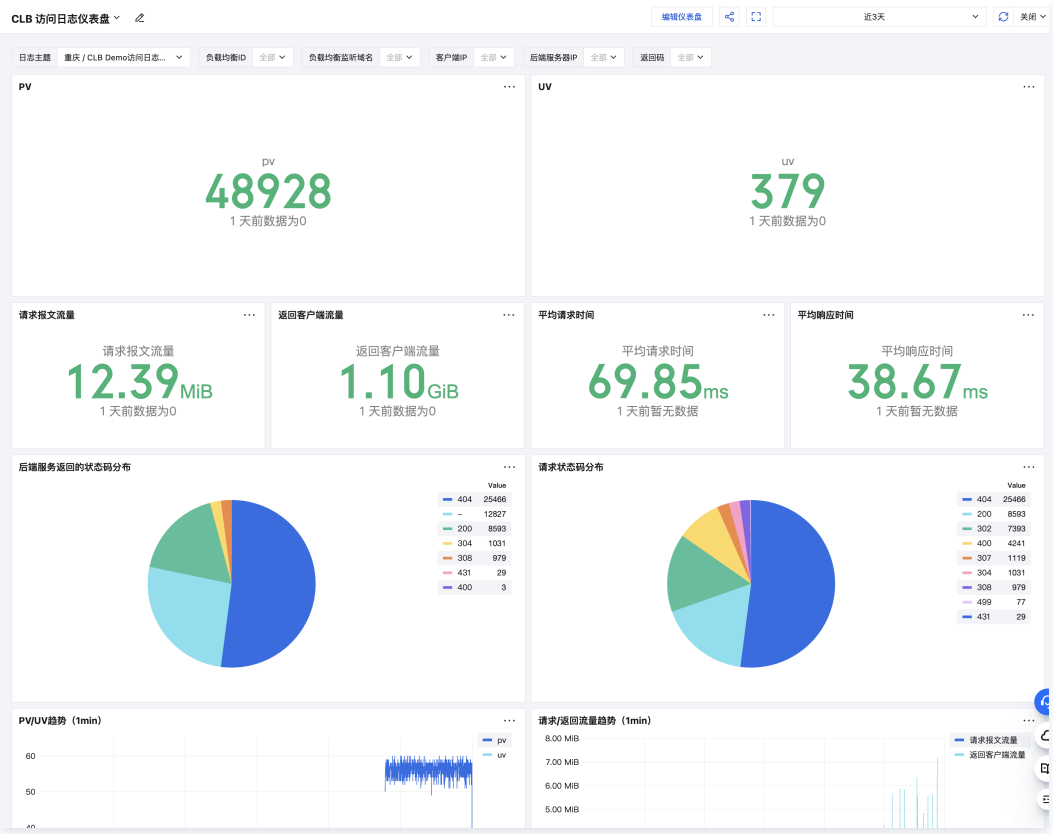
3. 在开启日志采集弹窗中，选择或 创建 目标日志主题，并单击**确认**。

步骤3：抽样采集（可选）

在一些场景下，CLB 请求量较大时，产生的日志量也会很大，全量日志上报可能会导致日志成本较高。CLB 支持抽样采集部分日志，减少数据上报量，从而降低日志成本，需详情请参见 抽样采集日志 。

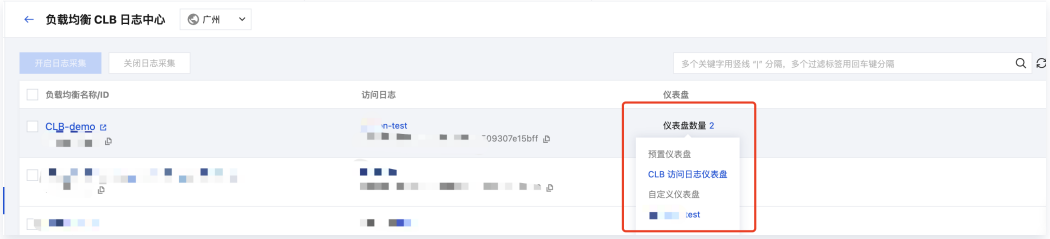
查看 CLB 负载均衡访问分析仪表盘

完成 开启 CLB 负载均衡访问日志采集 后，CLS 提供了预置的**访问分析仪表盘**，帮助您快速了解当前 CLB 访问请求情况。
CLB 访问日志仪表盘：可视化展示 **CLB 访问流量分析、状态码分布，以及用户行为**等分析看板，助力运维排障场景。



您可通过以下操作查看以上 CLB 预置仪表盘：

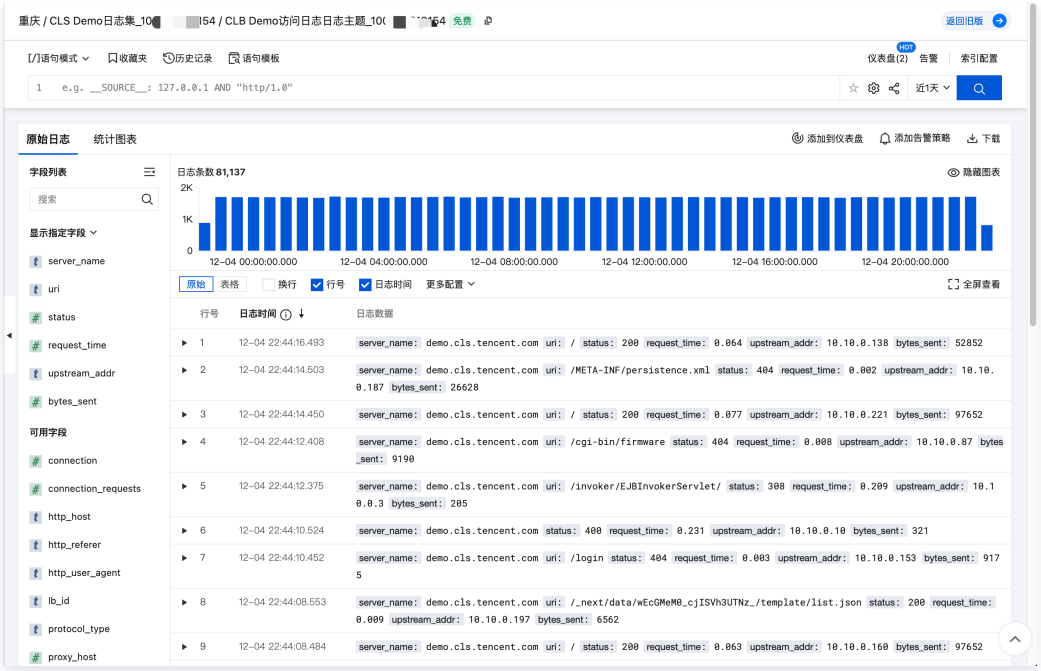
1. 在负载均衡实例列表中，找到已开启访问日志采集的域名。
2. 单击**仪表盘数量**，在下拉列表中选择 **CLB 访问日志仪表盘**，单击后跳转仪表盘。



检索分析 CLB 负载均衡访问日志

1. 完成开启负载均衡访问日志采集后，在负载均衡实例列表中，找到已开启访问日志采集的负载均衡实例，然后单击访问日志所在的目标日志主题名称。

2. 单击后将跳转日志检索页，即可检索分析访问日志。日志字段的详细介绍请参见 日志字段说明 。



日志字段说明

针对 CLB 访问日志中的字段解释， 可参见下表：

字段名	说明
stgw_request_id	请求 ID。
time_local	访问的时间与时区，例如，“01/Jul/2019:11:11:00 +0800”，最后的“+0800”表示所处时区为 UTC 之后的8小时，即为北京时间。
protocol_type	协议类型（HTTP/HTTPS/SPDY/HTTP2/WS/WSS）。
server_addr	CLB 的 VIP。
server_port	CLB 的 VPort，即监听端口。
server_name	规则的 server_name，CLB 的监听器中配置的域名。
remote_addr	客户端 IP。
remote_port	客户端端口。
status	CLB 返回给客户端的状态码。
upstream_addr	RS 地址。
upstream_status	RS 返回给 CLB 的状态码。
proxy_host	stream ID。
request	请求行。
request_length	从客户端收到的请求字节数。
bytes_sent	发送到客户端的字节数。
http_host	请求域名，即 HTTP 头部中的 Host。
http_user_agent	HTTP 协议头的 user_agent 字段。
http_referer	HTTP 请求来源。

http_x_forwarded_for	HTTP 请求中 x-forwarded-for header 的内容。
request_time	请求处理时间：从收到客户端的第一个字节开始，直到给客户端发送的最后一个字节为止，包括客户端请求到 CLB、CLB 转发请求到 RS、RS 响应数据到 CLB、CLB 转发数据到客户端的总时间。单位：秒。
upstream_response_time	整个后端请求所花费时间：从开始 CONNECT RS 到从 RS 接收完应答的时间。单位：秒。
upstream_connect_time	和 RS 建立 TCP 连接所花费时间：从开始 CONNECT RS 到开始发送 HTTP 请求的时间。
upstream_header_time	从 RS 接收完 HTTP 头部所花费时间：从开始 CONNECT RS 到从 RS 接收完 HTTP 应答头部的时间。
tcpinfo_rtt	TCP 连接的 RTT。
connection	连接 ID。
connection_requests	连接上的请求个数。
ssl_handshake_time	记录 SSL 握手各阶段耗时，格式：x:x:x:x:x:x:x。其中，冒号分隔的字符串，单位是ms，每个阶段耗时若小于1ms则显示为0。 • 第1个字段表示是否 SSL 会话复用。 • 第2个字段表示完整的握手时间。 • 3~7表示 SSL 各阶段耗时。 • 第3个字段表示 CLB 从收到 client hello 到发送 server hello done 的时间。 • 第4个字段表示 CLB 从发送 server 证书开始到发送 server 证书完成的时间。 • 第5个字段表示 CLB 从计算签名到发送 server key exchange 完成的时间。 • 第6个字段表示 CLB 从收到 client key exchange 开始到收完 client key exchange 的时间。 • 第7个字段表示 CLB 从收到 client key exchange 到发送 server finished 的时间。
ssl_cipher	SSL 加密套件。
ssl_protocol	SSL 协议版本。
vip_vpcid	负载均衡实例所属的私有网络 ID，公网 CLB 的取值为-1。
request_method	请求方式，支持 POST 和 GET 请求。
uri	资源标识符。
server_protocol	CLB 的协议。

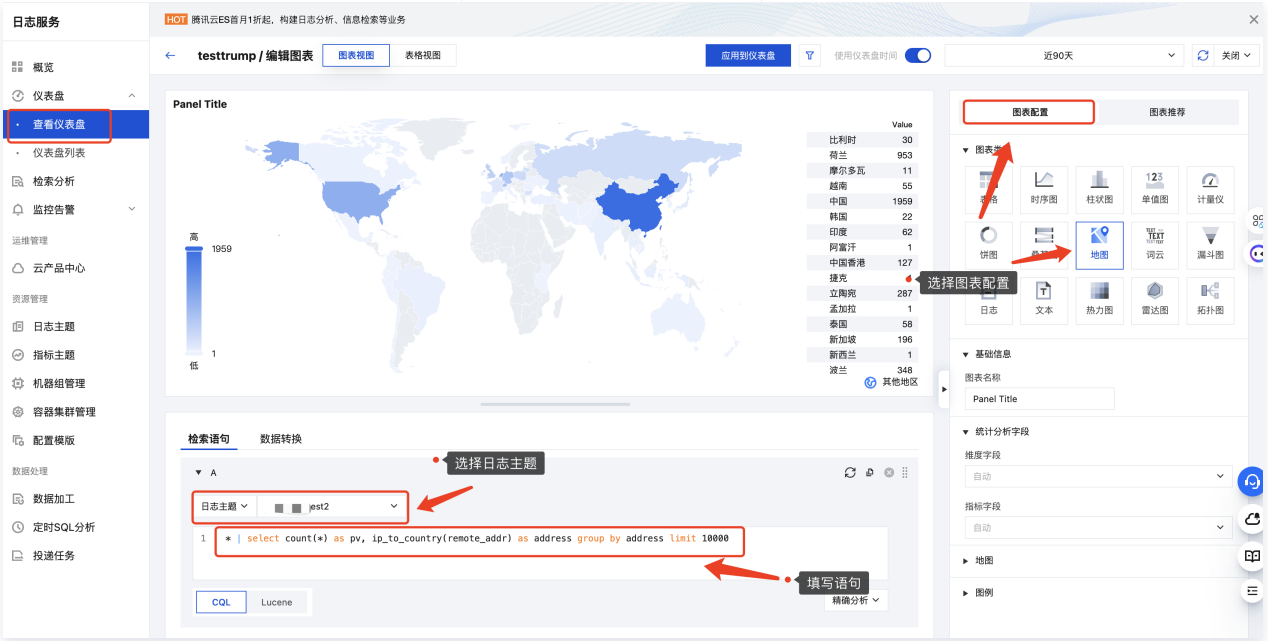
实践场景案例

您可以基于 CLB 访问日志配置进行检索分析和配置仪表盘，实时的分析 CLB 访问日志中的数据。以下提供了两个案例：

案例 1：分析 CLB 访问日志中所有请求客户的地理分布情况

1. 登录 日志服务控制台，并进入 仪表盘列表 页面，单击**创建仪表盘**，进入仪表盘创建页。

2. 在仪表盘创建页，选择空白仪表盘，并填入仪表盘名称为：CLB 访问日志地理分布。
3. 仪表盘完成创建后，单击自定义图表配置如下内容。



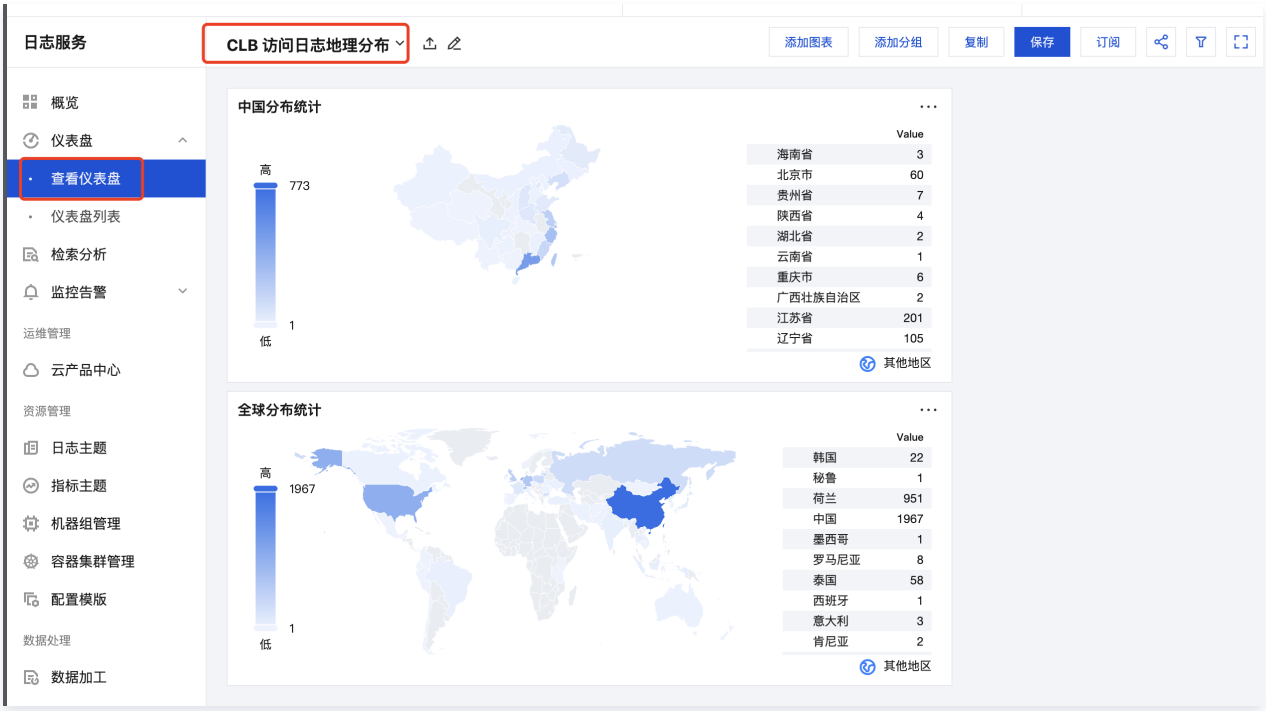
- 日志主题：选择在 采集 CLB 访问日志 步骤中所创建日志主题。
- 图表配置：图表类型选择“地图”。
- 执行语句：输入以下语句，即可呈现统计分布结果。
  - 中国分布：

```
* | select count(*) as pv, ip_to_province(remote_addr) as address group by address limit
10000
```

  - 全球分布：

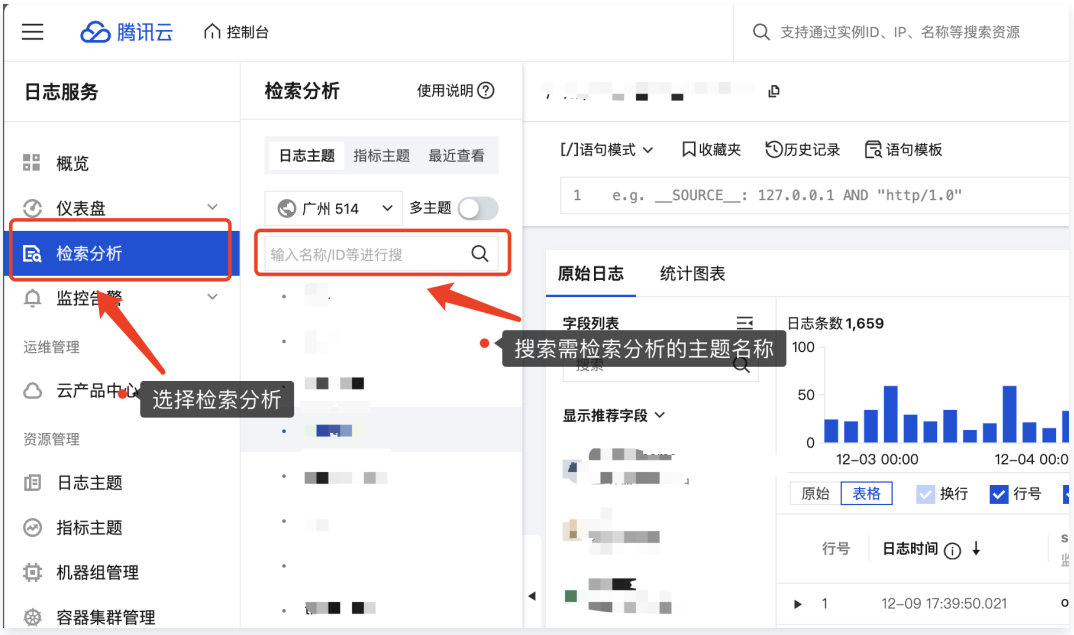
```
* | select count(*) as pv, ip_to_country(remote_addr) as address group by address limit 10000
```

4. 单击应用到仪表盘后，即可在**查看仪表盘**视图中看到已保存的仪表盘。



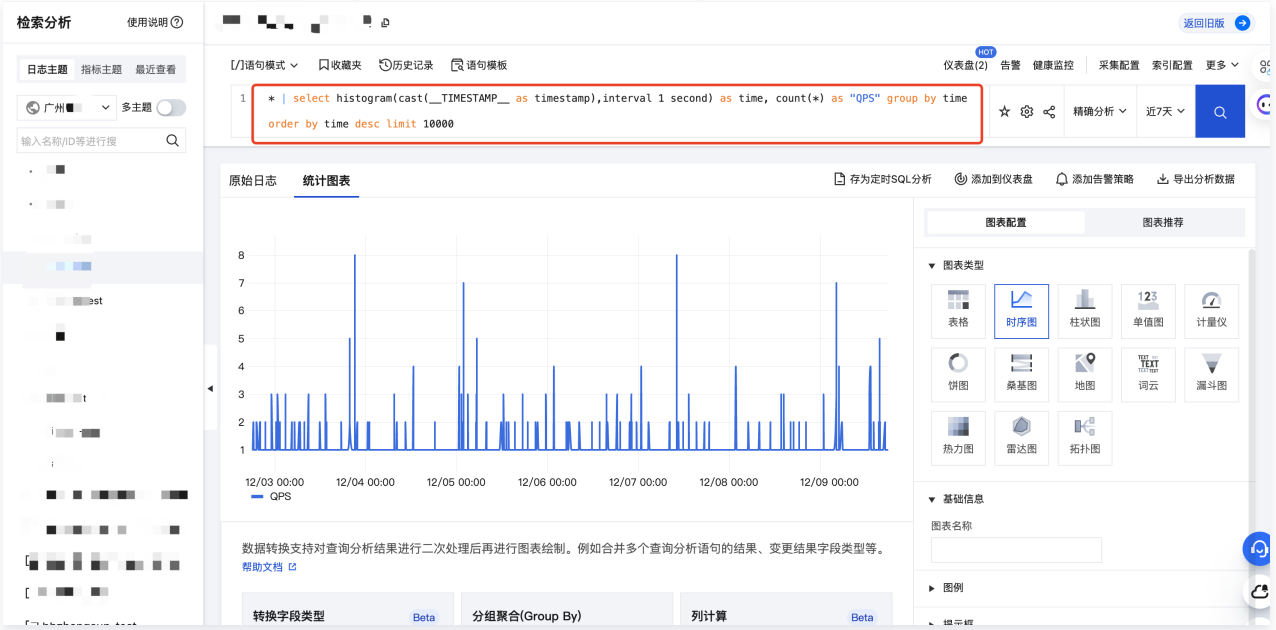
案例 2：CLB 日志访问请求 QPS 趋势分析

1. 登录 日志服务控制台，在左侧导航栏中，单击**检索分析**，进入检索分析页面。
2. 在输入框中，搜索在 采集 CLB 访问日志 步骤中所创建的日志主题。



3. 在执行语句中输入以下语句，用于统计所选时间范围内 QBS 的请求次数。并在图表配置的图表类型选择“时序图”。

```
* | select histogram(cast(__TIMESTAMP__ as timestamp),interval 1 second) as time, count(*) as "QPS"
group by time order by time desc limit 10000
```


检索分析
使用说明
日志主题
指标主题
最近查看
广州
多主题
输入名称/ID等进行搜
返回旧版
语句模式
收藏夹
历史记录
语句模板
仪表盘(2)
告警
健康监控
采集配置
索引配置
更多
* | select histogram(cast(__TIMESTAMP__ as timestamp),interval 1 second) as time, count(*) as "QPS" group by time
order by time desc limit 10000
精确分析
近7天
原始日志
统计图表
存为定时SQL分析
添加到仪表盘
添加告警策略
导出分析数据
图表配置
图表推荐
图表类型
表格
时序图
柱状图
单值图
计量仪
饼图
桑基图
地图
词云
漏斗图
热力图
雷达图
拓扑图
基础信息
图表名称
图例
12/03 00:00
12/04 00:00
12/05 00:00
12/06 00:00
12/07 00:00
12/08 00:00
12/09 00:00
QPS
数据转换支持对查询分析结果进行二次处理后再进行图表绘制。例如合并多个查询分析语句的结果、变更结果字段类型等。
帮助文档
转换字段类型
Beta
分组聚合(Group By)
列计算
Beta

Nginx 访问日志分析

最近更新时间：2024-10-24 20:59:22

Nginx 是一个高性能的 HTTP 和反向代理 Web 服务器，透过 Nginx 日志可以挖掘非常大的价值，例如诊断调优网站，监控网站稳定性，运营数据统计等。本文介绍如何通过日志服务（Cloud Log Service，CLS）对 Nginx 进行全方位日志数据挖掘。

前提条件

- 已将 Nginx 日志采集至 CLS，详见 操作指南 。
- 本文采取标准 Nginx 日志配置：

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

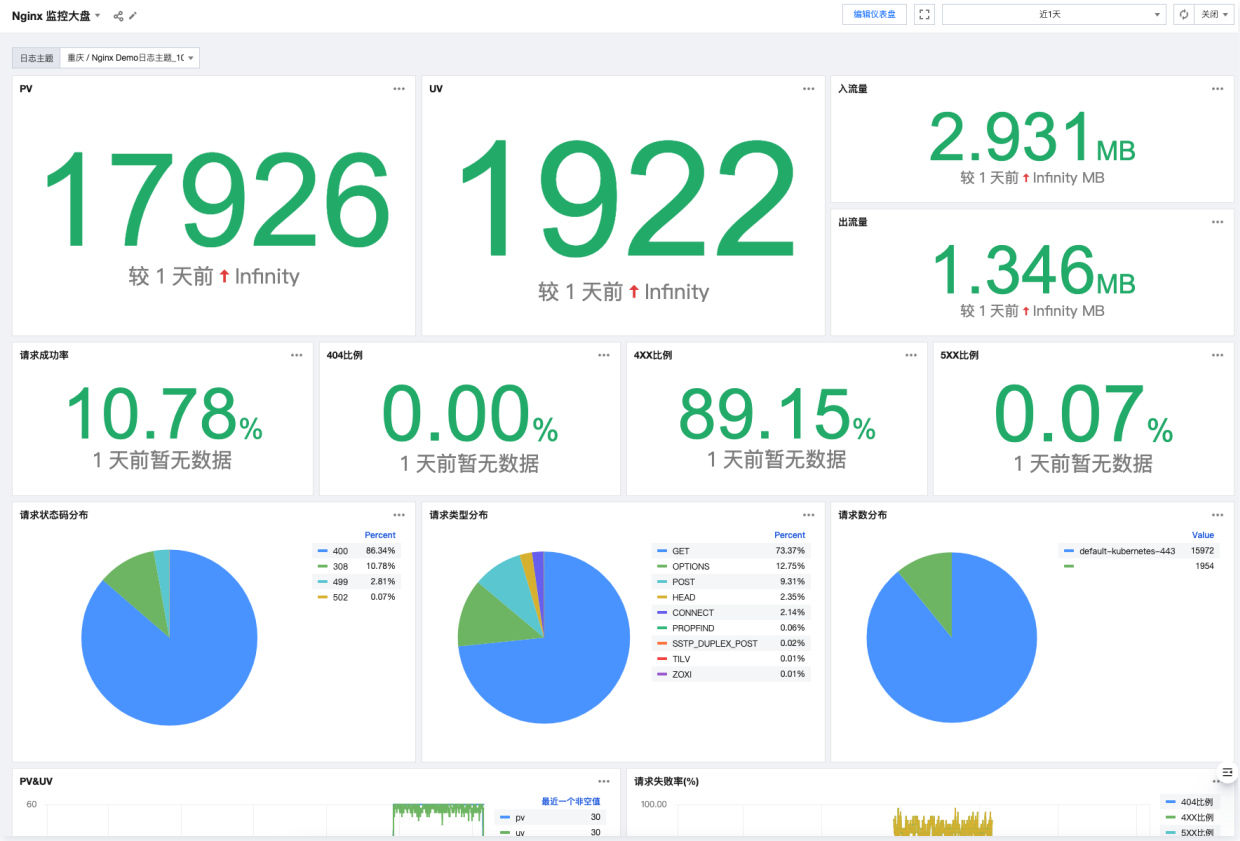
- 如果您暂无 Nginx 日志，还可以使用日志服务免费提供的 Demo 日志主题来体验该功能，操作步骤详见 使用 Demo 日志快速体验 CLS 。

预置仪表盘

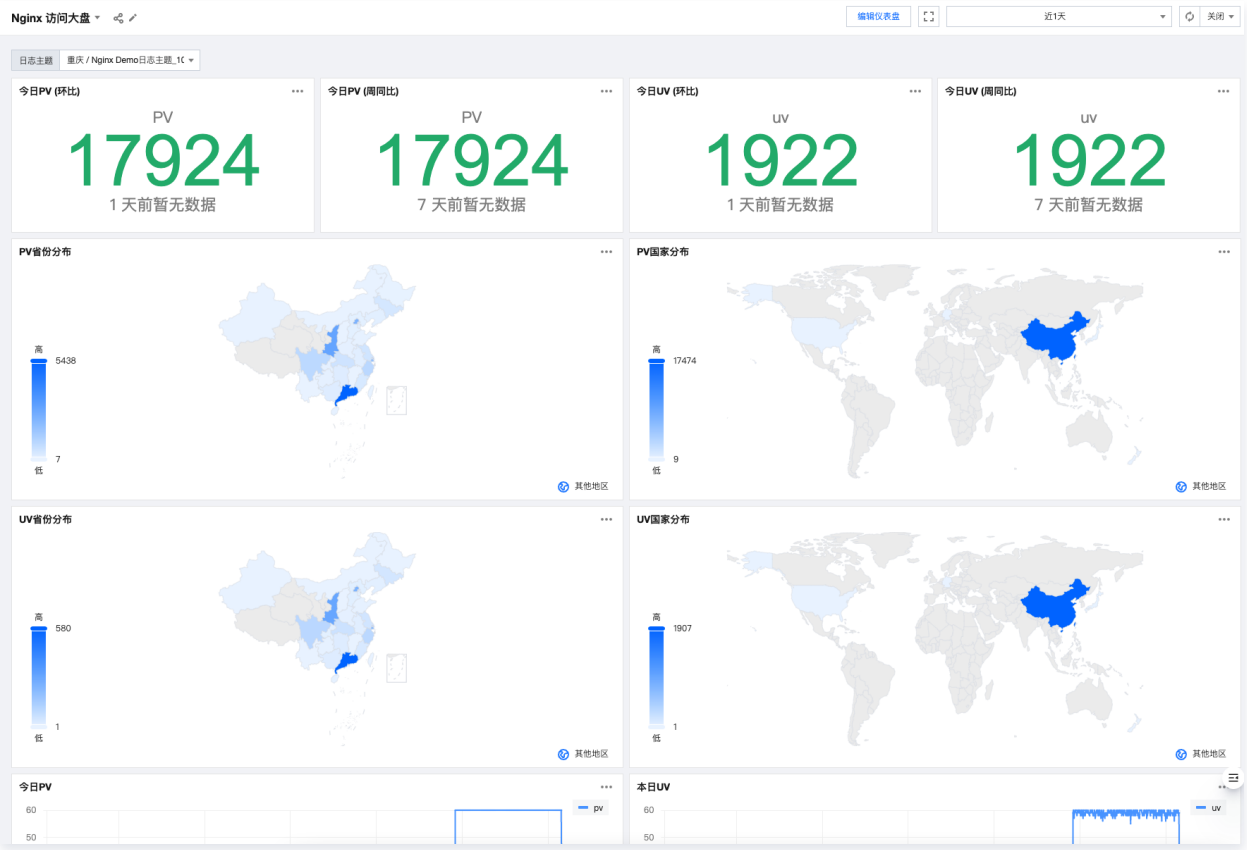
CLS 已将常用的Nginx日志统计预置为仪表盘，您可通过这些仪表盘快速了解当前 Nginx 运行状态及终端用户访问行为。

- Nginx 监控大盘，包括请求成功率、请求状态码分布、请求延迟等。
- Nginx 访问大盘，包括 PV、UV 及请求地域分布等。

在仪表盘右上角单击**编辑仪表盘**可基于预置仪表盘进行编辑。



可以构建更适用您的专属仪表盘。


Nginx 访问大盘
编辑仪表盘
近1天
关闭
日志主题
重庆 / Nginx Demo日志主题_1
今日PV (环比)
PV
17924
1 天前暂无数据
今日PV (周同比)
PV
17924
7 天前暂无数据
今日UV (环比)
uv
1922
1 天前暂无数据
今日UV (周同比)
uv
1922
7 天前暂无数据
PV省份分布
高
5438
7
低
其他地区
PV国家分布
高
17474
9
低
其他地区
UV省份分布
高
580
1
低
其他地区
UV国家分布
高
1907
1
低
其他地区
今日PV
60
50
pv
本日UV
60
50
uv

COS 访问日志分析

最近更新时间：2024-11-12 17:09:22

简介

对象存储（Cloud Object Storage，COS）访问日志记录了用户对 COS 资源的访问信息，包括上传对象（PUT），删除对象（DELETE），访问对象（GET）等。通过分析访问日志，用户可以完成审计回溯，如删除资源记录，同时也可以完成资源热门相关的资源统计等能力。

前提条件

已将 COS 日志采集至日志服务（Cloud Log Service，CLS），详情请参见 COS 开启实时日志。
如果您当前暂未启用上述功能，还可以使用日志服务免费提供的 Demo 日志主题来体验该功能，操作步骤详见 使用 Demo 日志快速体验 CLS。

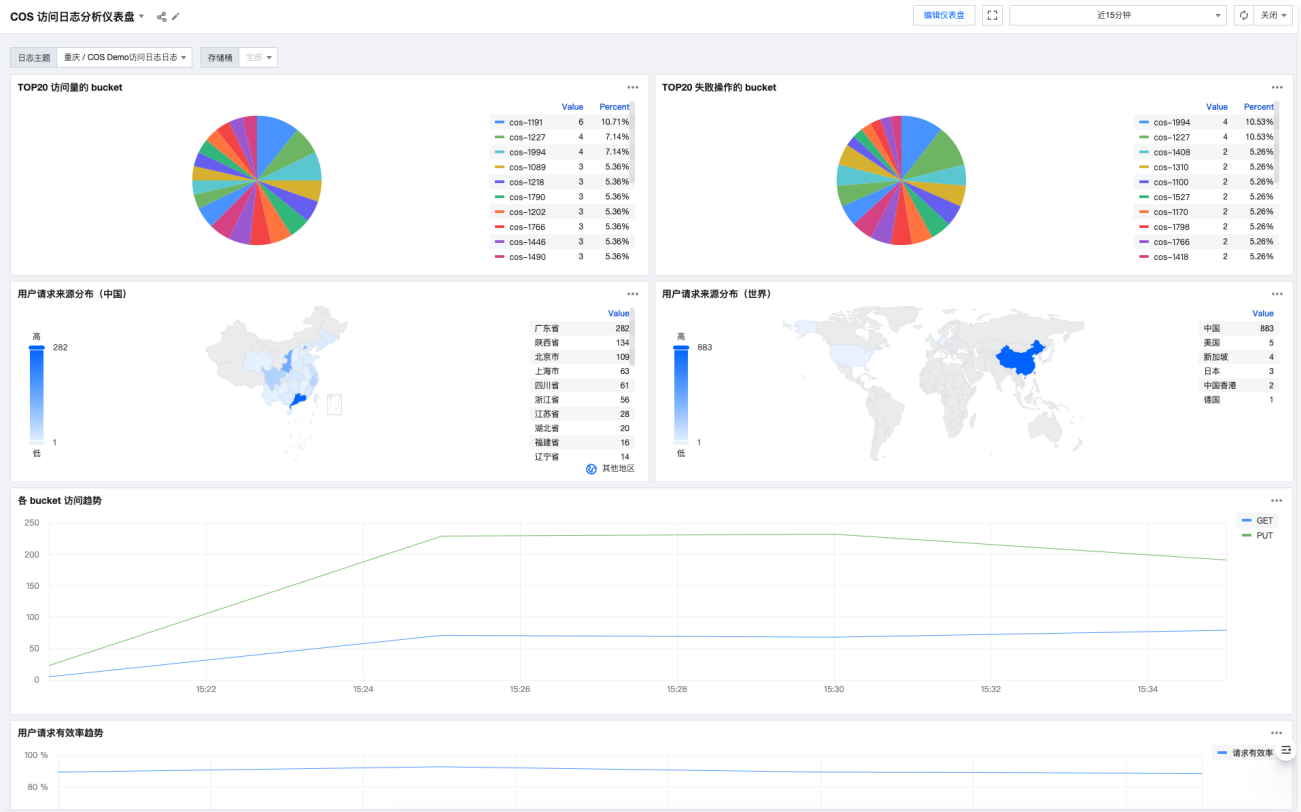
日志字段说明

字段序号	名称	含义	示例
1	eventVersion	记录版本	1.0
2	bucketName	存储桶名称	examplebucket-1250000000
3	qcsRegion	请求地域	ap-beijing
4	eventTime	事件时间（请求结束时间，UTC 0时 时间戳）	2018-12-01T11:02:33Z
5	eventSource	用户访问的域名	examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
6	eventName	事件名称	UploadPart
7	remoteIp	来源 IP	192.168.0.1
8	userSecretKeyId	用户访问 KeyId	AKID**********
9	reservedFiled	保留字段	保留字段，显示为 `-`
10	reqBytesSent	请求字节数（Bytes）	83886080
11	deltaDataSize	请求对存储量的改变（Bytes）	808
12	reqPath	请求的文件路径	/folder/text.txt
13	reqMethod	请求方法	put
14	userAgent	用户 UA	cos-go-sdk-v5.2.9
15	resHttpCode	HTTP 返回码	404
16	resErrorCode	错误码	NoSuchKey
17	resErrorMsg	错误信息	The specified key does not exist.
18	resBytesSent	返回字节数（Bytes）	197
19	resTotalTime	请求总耗时（毫秒，等于响应末字节的时间-请求首字节的时间）	4295
20	logSourceType	日志源类型	USER（用户访问请求），CDN（CDN 回源请求）
21	storageClass	存储类型	STANDARD，STANDARD_IA，ARCHIVE
22	accountId	存储桶所有者 ID	100000000001

23	resTurnAroundTime	请求服务端耗时（毫秒，等于响应首字节的时间-请求末字节的时间）	4295
24	requester	访问者	主账号 ID：子账号 ID，如果是匿名访问则显示 `-` 。
25	requestId	请求 ID	NWQ1ZjY4MTBfMjZiMjU4NjRfOWI1N180NDBiYTY=
26	objectSize	对象大小（Bytes）	808，如果您使用分块上传，objectSize 字段只会在完成上传的时候显示，各个分块上传期间该字段显示 `-`
27	versionId	对象版本 ID	随机字符串
28	targetStorageClass	目标存储类型，发起复制操作的请求会记录该字段	STANDARD，STANDARD_IA，ARCHIVE
29	referer	请求的 HTTP referer	`*.example.com` 或者111.111.111.1
30	requestUri	请求 URI	"GET /fdgfdgsf%20/%E6%B5%AE%E7%82%B9%E6%95%B0 HTTP/1.1"

预置仪表盘

CLS 已将常用的 COS 日志分析方式预置为仪表盘，您可通过该 仪表盘 快速了解当前 COS 请求状态。
在仪表盘右上角单击**编辑仪表盘**可基于预置仪表盘进行编辑。



场景示例

需求场景

某个对象文件访问不了，定位原因：

This XML file does not appear to have any style information associated with it. The document tree is shown below.

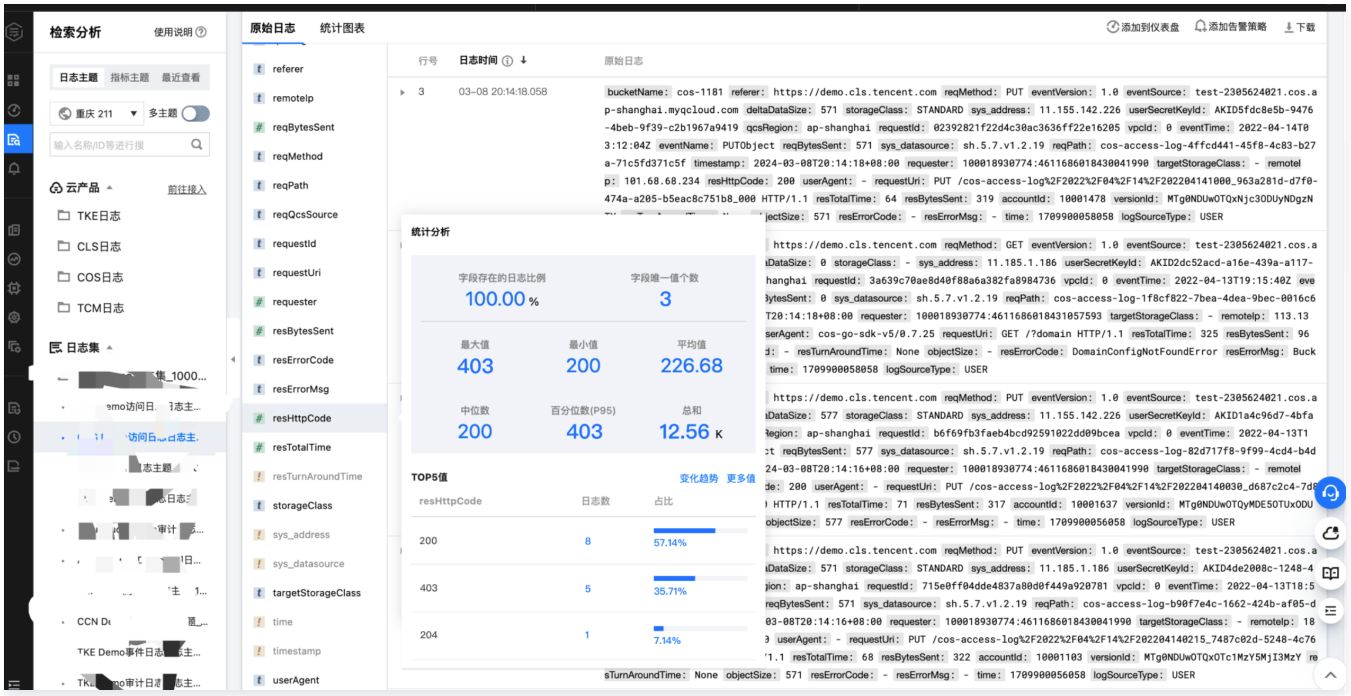
```
▼<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied.</Message>
  <Resource>ajaxhe-cls-1256238147.cos.ap-guangzhou.myqcloud.com/json-log2019-05-09_00645d9a-1118-4d69-8411-cfd57ede9ea1_000</Resource>
  <RequestId>NjEyNGUzMDhfNzJjYzNiMGJfNGY3Y18xNDc0M2VhNg==</RequestId>
  <TraceId>OGVmYzZiMmQzYjA2OWNhODk0NTRkMTBiOWVmMDAx0Dc0OWRkZjk0ZDM1NmI1M2E2MTR1Y2MzZDhmNmI5MWI10WI2NmQ0YjJkZWE3NjcxYTUzN2Q1NDQzNjY0ZmI3ZTMyN2MyODM0NzI1NDI4MWRkZjdiZjJhMDVkM2Y4MzdiZTk=</TraceId>
</Error>
```


解决方案

进入 COS 访问日志检索页面，输入对象名称作为关键词检索日志。

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

通过时间柱状图，得知近1天有14条日志记录。点击左侧字段名称，打开快速分析，可进一步对这14条日志按 resHttpCode 字段进行统计分析。



通过快速分析得知，6条非200的请求信息，其中5条 resHttpCode 为403的日志信息和一条 resHttpCode 为204日志信息，单击日志数，即可检索这两个 httpcode 的日志。

行号	日志时间 ↓	原始日志
1	08-24 20:16:03.000	reqQcsSource: - bucketName: [redacted] referer: - reqMethod: DELETE eventVersion: 1.0 eventSource: [redacted].ap-guangzhou.myqcloud.com deltaDataSize: -17054 storageClass: STANDARD userSecretKeyId: AKIDOInWCcDpQY90YAvLMZyE7H7lIgaqcvg2W_D_pAPfkfmIvPmF[redacted] qcsRegion: ap-guangzhou requestId: NjEyNGUzMDNfMWIzNTJjMGJfMmMxNTNfMTYwOTc4YjA= vpcId: 0 eventTime: 2021-08-24T12:16:03Z eventName: DELETEObject reqBytesSent: 0 reqPath: /json-log2019-05-09_00645d9a-1118-4d69-8411-cfd57ede9ea1_000 requester: 100004375281:100005586635 targetStorageClass: - remoteIp: 113.108.77.55 resHttpCode: 204 userAgent: tencentcloud-cos-console requestUri: DELETE /json-log2019-05-09_00645d9a-1118-4d69-8411-cfd57ede9ea1_000 HTTP/1.1 resTotalTime: 33 resBytesSent: 148 accountId: 100004[redacted] versionId: - objectSize: 17054

由日志可以得知，5条错误码为 Access Deny 日志均为访问对象失败日志，通过 resHttpCode 为204的日志发现，用户 `1000******` 在8月24日20点16分，通过 COS 控制台执行了删除 object 操作，导致对象访问失败。

Flowlog 网络流日志分析

最近更新时间：2024-07-15 09:57:01

网络流日志（Flow Logs，FL） 为您提供全时、全流、非侵入的流量采集服务，可将采集的网络流量进行实时的存储、分析，适用于故障排查、合规审计、架构优化、安全检测等场景，让您的云上网络更加稳定、安全和智能。
您可以创建指定采集范围（例如弹性网卡、NAT 网关、云联网跨地域流量）的网络流日志，来采集该范围内传入/传出的流量。

前提条件

已将网络流日志采集至日志服务（Cloud Log Service，CLS），详见 操作详情 。
如果您当前暂未将网络流日志采集至日志服务，还可以使用日志服务免费提供的Demo日志主题来体验该功能，操作步骤详见 使用 Demo 日志快速体验 CLS 。

日志字段说明

云联网跨地域流量的网络流日志

流日志将记录特定捕获窗口中，按“五元组 + 流量源地域 + 流量目的地域”规则过滤的网络流，即只有在捕获窗口中符合规则的网络流日志，才能记录为云联网跨地域流量的网络流日志记录。

五元组 + 流量源地域 + 流量目的地域

- 五元组即源 IP 地址、源端口、目的 IP 地址、目的端口和传输层协议这五个量组成的一个集合。
- 流量源地域指云联网跨地域流量发出的地域。
- 流量目的地域指云联网跨地域流量到达的地域。

捕获窗口

- 即一段持续时间，在这段时间内流日志服务会聚合数据，然后再发布流日志记录。捕获窗口大约为1分钟，推送时间约为5分钟。

字段	数据类型	说明
version	text	流日志版本。
region-id	text	记录日志的地域。
ccn-id	text	云联网唯一标识，如需确定云联网的信息请 联系我们 。
srcaddr	text	源 IP。
dstaddr	text	目标 IP。
srcport	text	流量的源端口。该字段仅对 UDP/TCP 协议生效，当流量为其他协议时，该字段显示为“-”。
dstport	long	流量的目标端口。该字段仅对 UDP/TCP 协议生效，当流量为其他协议时，该字段显示为“-”。
protocol	long	流量的 IANA 协议编号。更多信息，请转到分配的 Internet 协议 编号。
srcregionid	text	流量源地域。
dstregionid	text	流量目的地域。
packets	long	捕获窗口中传输的数据包的数量。当“log-status”为“NODATA”时，该字段显示为“-”。
bytes	long	捕获窗口中传输的字节数。当“log-status”为“NODATA”时，该字段显示为“-”。
start	long	当前捕获窗口收到第一个报文的时间戳，如果在捕获窗口内没有报文，则显示为该捕获窗口的起始时间，采用 Unix 秒的格式。
end	long	当前捕获窗口收到最后一个报文的时间戳，如果在捕获窗口内没有报文，则显示为该捕获窗口的结束时间，采用 Unix 秒的格式。

版权所有：腾讯云计算（北京）有限责任公司

action	text	与流量关联的操作： • ACCEPT：通过云联网正常转发的跨地域流量。 • REJECT：因限速被阻止转发的跨地域流量。
log-status	text	流日志的日志记录状态： • OK：表示数据正常记录到指定目标。 • NODATA：表示捕获窗口中没有传入或传出网络流量，此时“packets”和“bytes”字段会显示为“-1”。

其他类型的网络流日志

流日志将记录特定捕获窗口中，按五元组规则过滤的网络流。

五元组

即源 IP 地址、源端口、目的 IP 地址、目的端口和传输层协议这五个量组成的一个集合。

捕获窗口

即一段持续时间，在这段时间内流日志服务会聚合数据，然后再发布流日志记录。捕获窗口大约为5分钟，推送时间约为5分钟。流日志记录是以空格分隔的字符串，采用以下格式，字段无固定顺序：

```
version account-id interface-id srcaddr dstaddr srcport dstport protocol packets bytes start end action log-status
```

字段	说明
version	流日志版本。
account-id	流日志的账户 AppID。
instance-id	弹性网卡所属实例 ID。
interface-id	弹性网卡 ID。
region-id	地域 ID。
az-id	可用区 ID。
vpc-id	私有网络 ID。
subnet-id	子网 ID。
direction	流量访问方向（云服务器访问外部为 out，外部访问云服务器为 in）。
cross-region	若为跨地域上报，此处为实际采集地域，非跨地域上报，显示为0。
srcaddr	源 IP。
dstaddr	目标 IP。
srcport	流量的源端口。当流量为 ICMP 协议时，该字段表示 ICMP 的 id。
dstport	流量的目标端口。当流量为 ICMP 协议时，该字段表示 ICMP 的 type（高8bit）+code（低8bit）组合。
protocol	流量的 IANA 协议编号。更多信息，请转到分配的 Internet 协议 编号。
packets	捕获窗口中传输的数据包的数量。
bytes	捕获窗口中传输的字节数。
start	捕获窗口启动的时间，采用 Unix 秒的格式。
end	捕获窗口结束的时间，采用 Unix 秒的格式。

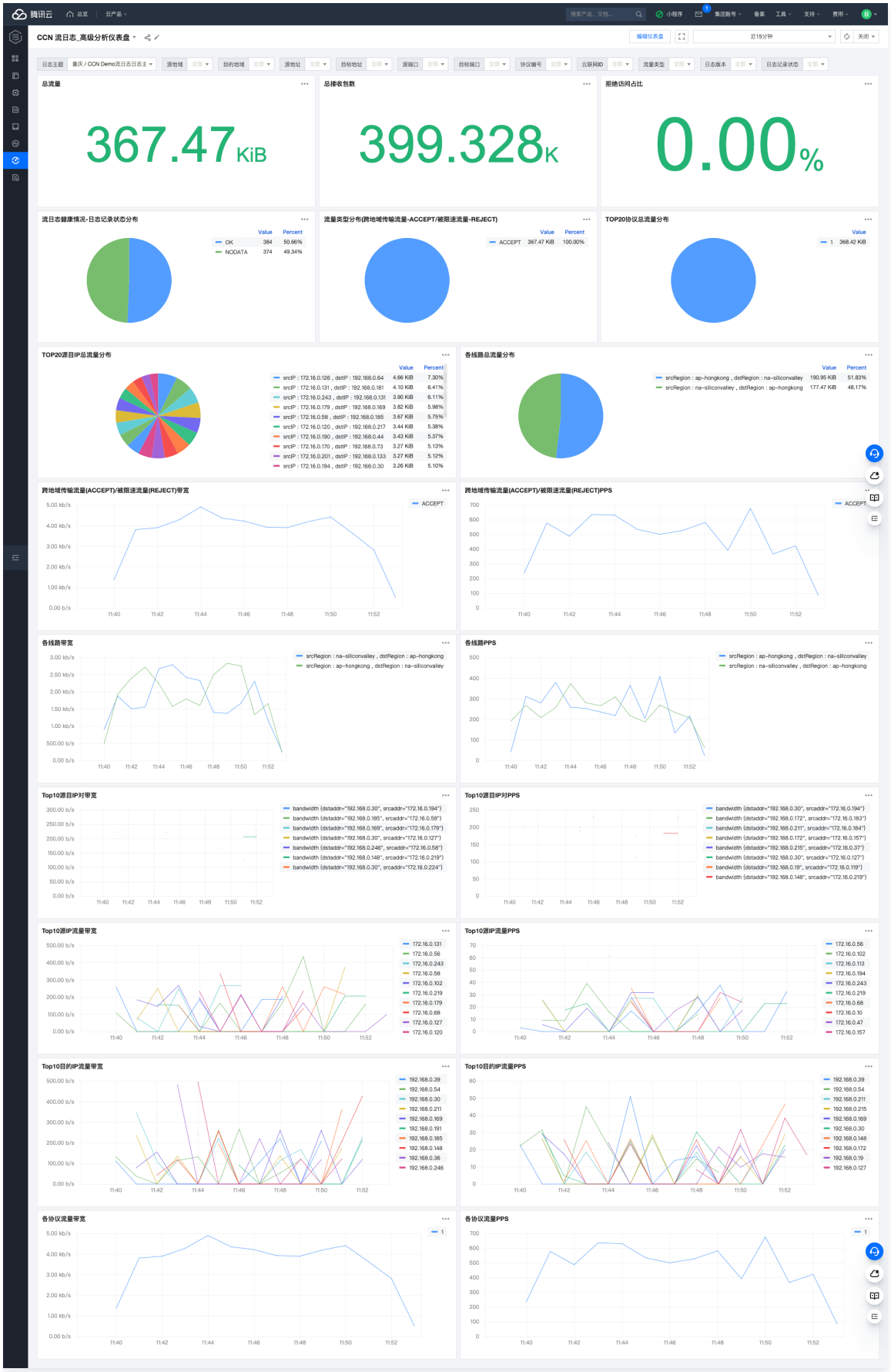
action	与流量关联的操作： • ACCEPT：安全组或网络 ACL 允许记录的流量。 • REJECT：安全组或网络 ACL 未允许记录的流量。
log-status	流日志的日志记录状态： • OK：表示数据正常记录到指定目标。 • NODATA：表示捕获窗口中没有传入或传出网络流量，此时“packets”和“bytes”字段会显示为“-1”。 • SKIPDATA：表示捕获窗口中跳过了一些流日志记录。可能是内部容量限制或内部错误引起的。

预置仪表盘

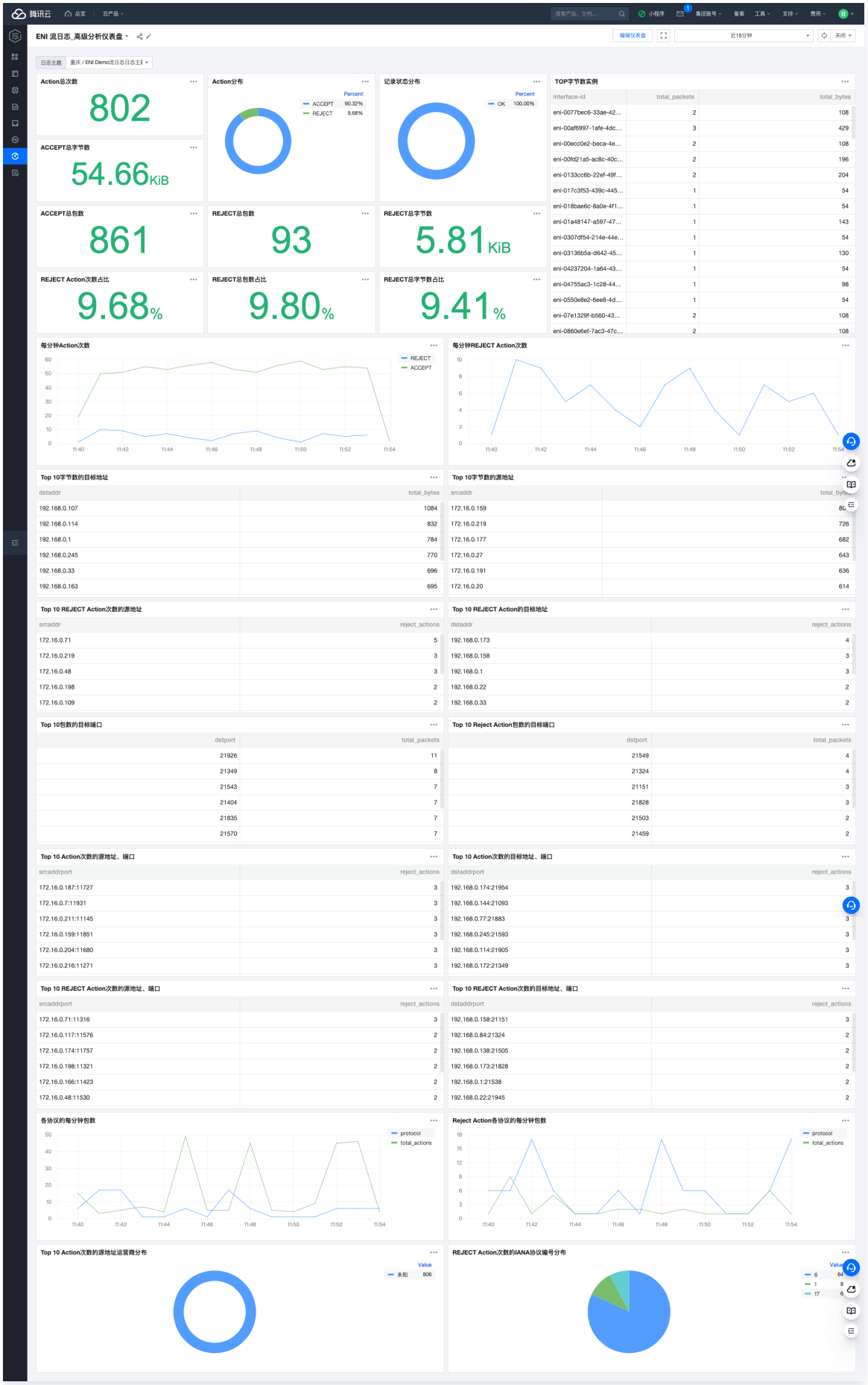
CLS 已将常用的云联网及弹性网卡流日志统计预置为仪表盘，您可通过这些仪表盘快速了解当前网络状态。

- 云联网：https://console.cloud.tencent.com/cls/dashboard/d?templateId=flow-log-ccn-analysis-dashboard
- 弹性网卡：https://console.cloud.tencent.com/cls/dashboard/d?templateId=flow-log-eni-analysis-dashboard

在仪表盘右上角单击**编辑仪表盘**可基于预置仪表盘进行编辑。



构建更适用您的专属仪表盘。


腾讯云
ENI 流日志_高级分析仪表盘
Action总次数
802
Action分布
Percent
ACCEPT 90.32%
REJECT 9.68%
记录状态分布
Percent
OK 100.00%
TOP字节数实例
interface-id total_packets total_bytes
eni-0077bec6-33ae-42... 2 108
eni-00af6997-1afe-4dc... 3 429
eni-00ecc0e2-beca-4e... 2 108
eni-00fd21a5-ac8c-40c... 2 196
eni-0133cc6b-22ef-49f... 2 204
eni-017c3f53-439c-445... 1 54
eni-018bae6c-8a0e-4f1... 1 54
eni-01a48147-a597-47... 1 143
eni-0307df54-214e-44e... 1 54
eni-03136b5a-d642-45... 1 130
eni-04237204-1a64-43... 1 54
eni-04755ac3-1c28-44... 1 98
eni-0550e8e2-6ee8-4d... 1 54
eni-07e1329f-b560-43... 2 108
eni-0860e6ef-7ac3-47c... 2 108
ACCEPT总字节数
54.66KiB
ACCEPT总包数
861
REJECT总包数
93
REJECT总字节数
5.81KiB
REJECT Action次数占比
9.68%
REJECT总包数占比
9.80%
REJECT总字节数占比
9.41%
每分钟Action次数
REJECT
ACCEPT
每分钟REJECT Action次数
Top 10字节数的目标地址
dstaddr total_bytes
192.168.0.107 1084
192.168.0.114 832
192.168.0.1 784
192.168.0.245 770
192.168.0.33 696
192.168.0.163 695
Top 10字节数的源地址
srcaddr
172.16.0.159
172.16.0.219 726
172.16.0.177 682
172.16.0.27 643
172.16.0.191 636
172.16.0.20 614
Top 10 REJECT Action次数的源地址
srcaddr reject_actions
172.16.0.71 5
172.16.0.219 3
172.16.0.48 3
172.16.0.198 2
172.16.0.109 2
Top 10 REJECT Action的目标地址
dstaddr reject_actions
192.168.0.173 4
192.168.0.158 3
192.168.0.1 3
192.168.0.22 2
192.168.0.33 2
Top 10包数的目标端口
dstport total_packets
21926 11
21349 8
21543 7
21404 7
21835 7
21570 7
Top 10 Reject Action包数的目标端口
dstport total_packets
21549 4
21324 4
21151 3
21828 3
21503 2
21459 2
Top 10 Action次数的源地址、端口
srcaddrport reject_actions
172.16.0.187:11727 3
172.16.0.7:11931 3
172.16.0.211:11145 3
172.16.0.159:11851 3
172.16.0.204:11680 3
172.16.0.216:11271 3
Top 10 Action次数的目标地址、端口
dstaddrport reject_actions
192.168.0.174:21954 3
192.168.0.144:21093
192.168.0.77:21883 3
192.168.0.245:21593 3
192.168.0.114:21905 3
192.168.0.172:21349 3
Top 10 REJECT Action次数的源地址、端口
srcaddrport reject_actions
172.16.0.71:11316 3
172.16.0.117:11576 2
172.16.0.174:11757 2
172.16.0.198:11321 2
172.16.0.166:11423 2
172.16.0.48:11530 2
Top 10 REJECT Action次数的目标地址、端口
dstaddrport reject_actions
192.168.0.158:21151 3
192.168.0.84:21324 2
192.168.0.138:21505 2
192.168.0.173:21828 2
192.168.0.1:21538 2
192.168.0.22:21945 2
各协议的每分钟包数
protocol
total_actions
Reject Action各协议的每分钟包数
protocol
total_actions
Top 10 Action次数的源地址运营商分布
Value
未知 806
REJECT Action次数的IANA协议编号分布
Value

配置告警

例如为云联网中国香港-硅谷线路配置了带宽上限100Mbps，需监控当前带宽使用情况，连续10分钟带宽大于等于95Mbps 时触发告警，以便在必要时对带宽上限进行调整。

1. 进入创建告警策略页面，操作步骤详见 配置告警策略 。
2. 在执行语句中输入以下语句，时间范围选择1分钟，统计近1分钟内的中国香港-硅谷线路带宽使用情况。该执行语句的结果中 bandwidth 即为1分钟带宽，单位为 Mbps。

```
log-status:OK AND srcregionid:ap-hongkong AND dstregionid:na-siliconvalley | select
sum(bytes)/60.00*8/1000/1000 as bandwidth
```

3. 触发条件如下，即带宽大于等于95Mbps 时，满足告警条件。

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

4. 执行周期：固定频率，每1分钟执行一次。
5. 告警通知-告警频率：持续10个周期满足触发条件则始终触发告警，即连续10分钟带宽大于等于95Mbps 时触发告警。

   针对预置仪表盘中的图表，还可以单击右上角的“**添加到监控告警**”将该图表中的指标添加到告警策略中。

TKE 审计日志分析

最近更新时间：2024-10-24 20:59:22

生产环境中的 Kubernetes 集群通常是一个相当复杂的系统，底层是各种异构的主机、网络、存储等云基础设施，上层承载着大量的应用负载，中间运行着各种原生（Scheduler、Kubelet等）和第三方（例如各种Operator）的组件对基础设施和应用进行管理和调度。此外不同角色的人员频繁地在集群上进行部署应用、添加节点等各种操作。因此在集群运维场景中， 用户常见的问题有：

- 集群中的某个应用被删除了，谁操作的？
- Apiserver 的负载突然变高，大量访问失败，集群中到底发生了什么？
- 集群节点被封锁了，是谁在什么时候操作的？

日志服务（Cloud Log Service，CLS）与 容器服务（Tencent Kubernetes Engine，TKE） 已打通。 通过 Kubernetes 集群审计日志可以帮助用户快速解决以上这些问题。

前提条件

已购买 TKE，并开启集群审计功能，详情请参见 操作指南 。
如果您当前暂无 TKE 集群或未开启集群审计功能，还可以使用日志服务免费提供的 Demo 日志主题来体验该功能，操作步骤详见 使用 Demo 日志快速体验 CLS 。

审计日志简介

审计日志记录了对 kube-apiserver 的访问事件，会按顺序记录每个用户、管理员或系统组件影响集群的活动。每一条审计日志是一个 JSON 格式的结构化记录，包括元数据（metadata）、请求内容（requestObject）和响应内容（responseObject）三个部分。其中元数据（包含了请求的上下文信息，例如谁发起的请求、从哪里发起的、访问的 URI 等信息）一定会存在，请求和响应内容是否存在取决于审计级别。通过日志可以了解到以下内容：

- 集群里发生的活动。
- 活动的发生时间及发生对象。
- 活动的触发时间、触发位置及观察点。
- 活动的结果以及后续处理行为。

审计日志字段说明

```
{
  "kind":"Event",
  "apiVersion":"audit.k8s.io/v1",
  "level":"RequestResponse",
  "auditID":0a4376d5-307a-4e16-a049-24e017******,
  "stage":"ResponseComplete",
  // 发生了什么
  "requestURI":"/apis/apps/v1/namespaces/default/deployments",
  "verb":"create",
  // 谁发起的
  "user":{
    "username":"admin",
      "uid":"admin",
      "groups":[
        "system:masters",
        "system:authenticated"
      ]
  },
  // 从哪里发起
  "sourceIPs":[
    "10.0.6.68"
  ],
  "userAgent":"kubectl/v1.16.3 (linux/amd64) kubernetes/ald64d8",
  // 发生了什么
  "objectRef":{
    "resource":"deployments",
```

```
    "namespace":"default",
    "name":"nginx-deployment",
    "apiGroup":"apps",
    "apiVersion":"v1"
  },
  // 结果是什么
  "responseStatus":{
    "metadata":{
    },
    "code":201
  },
  // 请求及返回具体信息
  "requestObject":Object{...},
  "responseObject":Object{...},
  // 什么时候开始/结束
  "requestReceivedTimestamp":"2020-04-10T10:47:34.315746Z",
  "stageTimestamp":"2020-04-10T10:47:34.328942Z",
  // 请求被接收/拒绝的原因是什么
  "annotations":{
    "authorization.k8s.io/decision":"allow",
    "authorization.k8s.io/reason":""
  }
}
```

预置仪表盘

CLS 已将常用的审计日志分析方式预置为仪表盘，您可通过这些仪表盘快速了解当前集群状态。

- 审计总览，观测整个集群 APIserver 操作。
- 节点操作概览，主要用于排查节点相关问题。
- K8S对象操作概览，主要用于排查 K8S 对象（例如某个工作负载）的相关问题。
- 聚合检索，观测某个维度下审计日志的分布趋势。

在仪表盘右上角单击**编辑仪表盘**可基于预置仪表盘进行编辑，构建更适用您的专属仪表盘。

排查问题场景示例

场景1：集群中的某个应用被删除了，谁操作的？

1. 打开 K8S 对象操作概览仪表盘，在日志主题中选择需要审计日志存储的日志主题，指定操作类型为 delete 和资源对象 nginx。

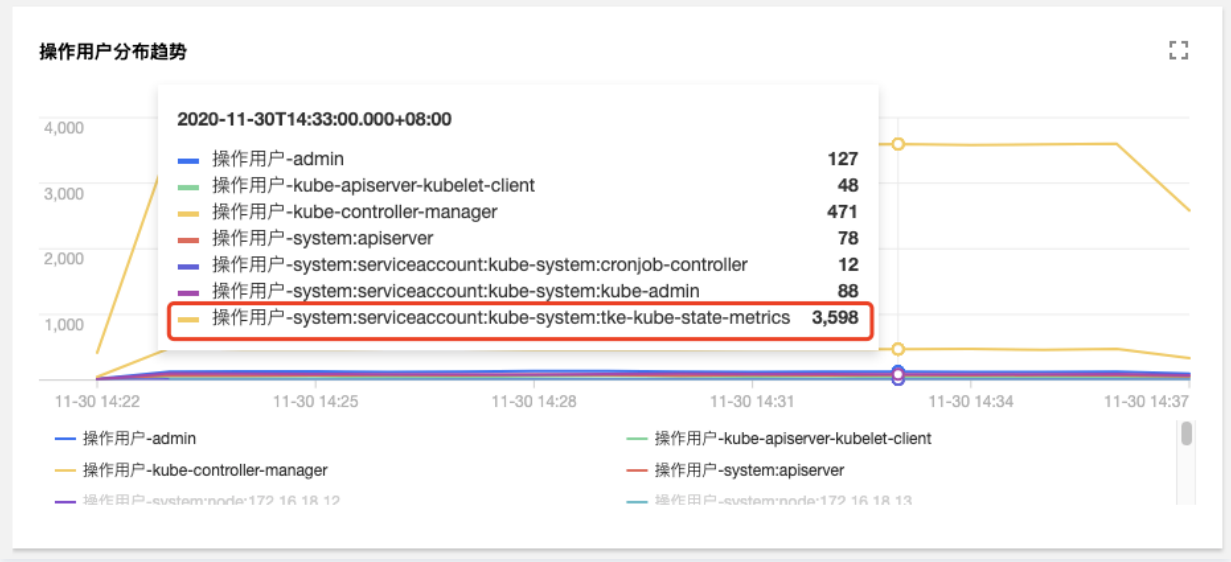
2. 查询结果如下图所示：

由图可见，是 `10001****7138` 这个账号，对应用「nginx」进行了删除。可根据账号 ID 在**访问管理** > **用户** > **用户列表**中找到关于此账号的详细信息。

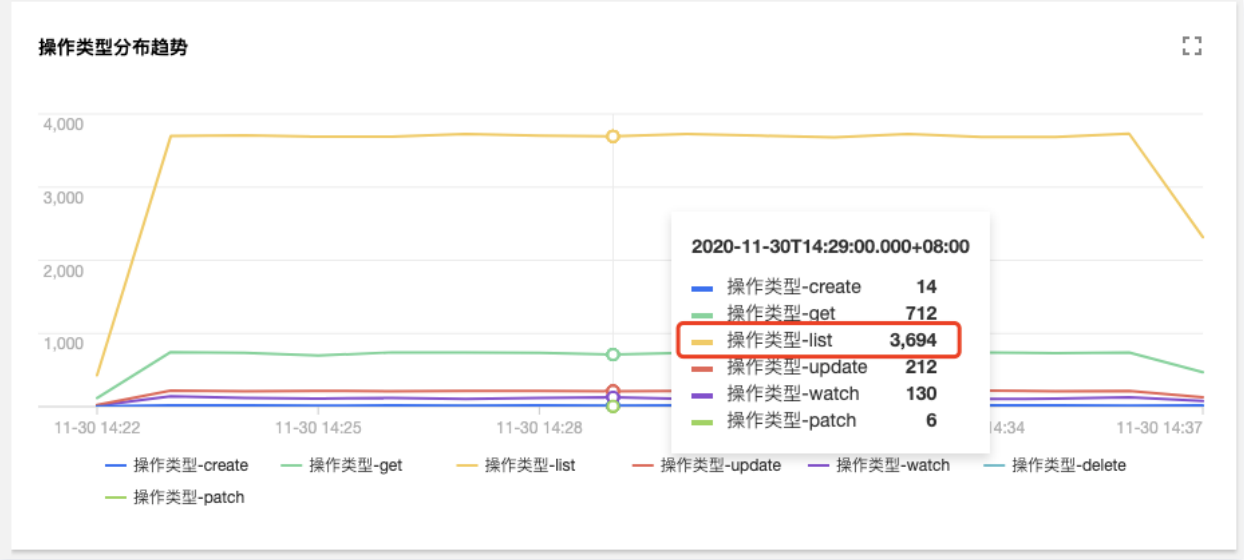
场景2：Apiserver 的负载突然变高，大量访问失败，集群中到底发生了什么？

1. 打开 聚合检索仪表盘，该仪表盘提供了从用户、操作类型、返回状态码等多个维度对于 Apiserver 访问聚合趋势图。

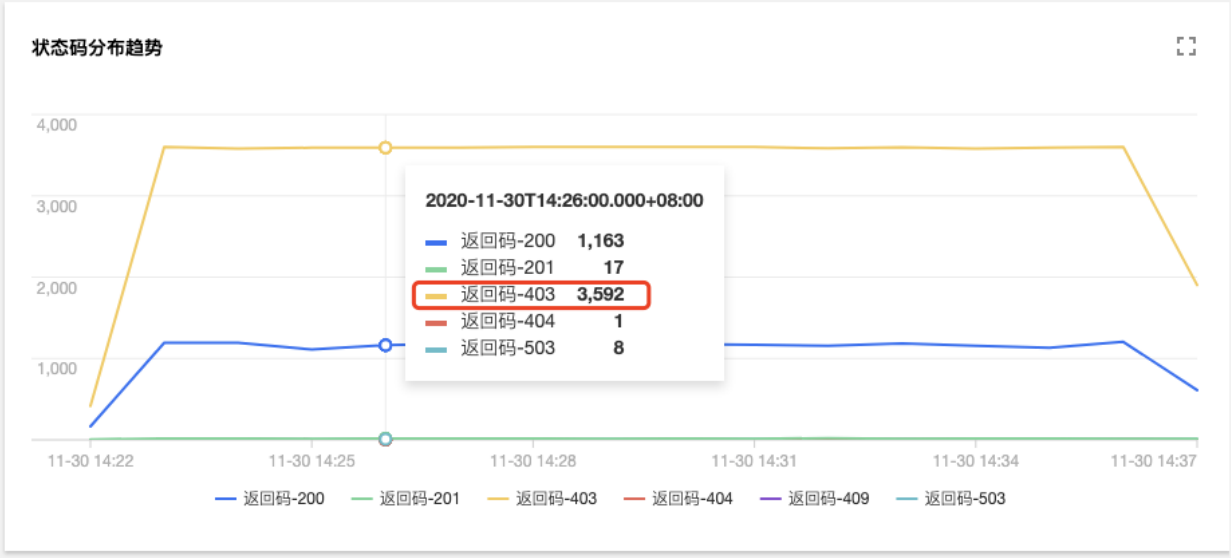
- 操作用户分布趋势图：



- 操作类型分布趋势图：



- 状态码分布趋势图：



2. 通过以上图表得知，用户 `tke-kube-state-metrics` 的访问量远高于其他用户，并且在“操作类型分布趋势”图中可以看出大多数都是 list 操作，在“状态码分布趋势”图中可以看出，状态码大多数为403，根据 `tke-kube-state-metrics` 关键词，检索日志。

结合业务日志可知，由于 RBAC 鉴权问题导致 `tke-kube-state-metrics` 组件不停地请求 Apiserver 重试，导致 Apiserver 访问剧增。

场景3：集群节点被封锁了，是谁在什么时候操作的？

1. 打开 节点操作概览仪表盘 ，填写被封锁的节点名。

2. 查询结果如下图所示：

由图可见，是 `10001****7138` 这个账号在 `2020-11-30T06:22:18` 时对 `172.16.18.13` 这台节点进行了封锁操作。

TKE 事件日志分析

最近更新时间：2024-08-28 11:07:42

集群内的状况层出不穷，变化莫测，如节点状态异常，Pod 重启等，如果无法第一时间感知状况，会错过最佳的问题处理时间，待问题扩大，影响到业务时才发现往往已经为时已晚。而事件日志（Event）记录了全面的集群状态变更信息，不仅可以帮助用户第一时间发现问题，也是排查问题的最佳帮手。

前提条件

已购买 TKE，并开启事件存储功能，详情请参见 操作指南 。
如果您当前暂无TKE集群或未开启事件存储功能，还可以使用日志服务免费提供的Demo日志主题来体验该功能，操作步骤详见 使用 Demo 日志快速体验 CLS 。

什么是事件日志

Event 是 Kubernetes 中众多资源对象中的一员，通常用来记录集群内发生的状态变更，大到集群节点异常，小到 Pod 启动、调度成功等等。我们常用的 `kubectl describe` 命令就可以查看相关资源的事件信息。

事件日志字段说明


```
"event" : {
  "firstTimestamp" : "2020-11-25T14:10:17Z"
  "reason" : "EvictionThresholdMet"                                  内容
  "metadata" : {
    "uid" : "3ba47a24-135d-4eef-bbca-5b94a5d0b83e"
    "managedFields" : [...]
    "resourceVersion" : "1219076"
    "creationTimestamp" : "2020-11-25T14:10:25Z"
    "name" : "172.16.18.13.164ac58ada1dec47"
    "namespace" : "default"
    "selfLink" : "/api/v1/namespaces/default/events/172.16.18.13.164ac58ada1dec47"
  }
  "involvedObject" : {
    "uid" : "172.16.18.13"
    "kind" : "Node"                                                  资源类型/对象
    "name" : "172.16.18.13"
  }
  "reportingInstance" : ""
  "lastTimestamp" : "2020-11-28T07:45:22Z"
  "count" : 23538                                                    次数
  "source" : {
    "component" : "kubelet"                                          事件源
    "host" : "172.16.18.13"
  }
  "message" : "Attempting to reclaim ephemeral-storage"              详细描述
  "type" : "Warning"                                                 级别
  "reportingComponent" : ""
}
```


- 级别（Type）： 目前仅有 “Normal” 和 “Warning”，但是如果需要，可以使用自定义类型。
- 资源类型/对象（Involved Object）：事件所涉及的对象，例如 Pod，Deployment，Node 等。
- 事件源（Source）：报告此事件的组件；例如 Scheduler、Kubelet等。
- 内容（Reason）：当前发生事件的简短描述，一般为枚举值，主要在程序内部使用。
- 详细描述（Message）：当前发生事件的详细描述信息。
- 出现次数（Count）：事件发生的次数。

预置仪表盘

CLS已将常用的事件日志分析方式预置为仪表盘，您可通过这些仪表盘快速了解当前集群状态。

- 事件总览，可根据时间、命名空间、级别、原因、资源类型、资源对象等维度过滤事件，查看核心事件的汇总统计信息，并展示一个周期内的数据对比。例如，事件总数及分布情况、节点异常、Pod OOM、重要事件趋势等仪表盘以及异常 TOP 事件列表。
- 异常事件聚合检索，查看某个时间段内各类异常事件的 reason 和 object 分布趋势。

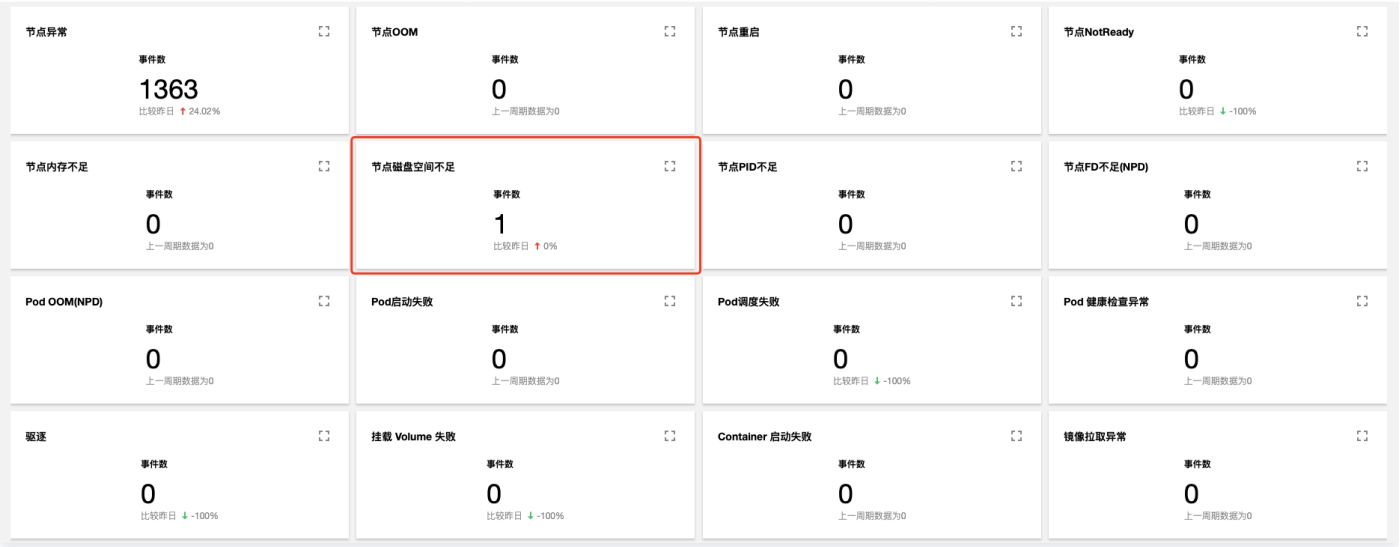
在仪表盘右上角单击**编辑仪表盘**可基于预置仪表盘进行编辑，构建更适用您的专属仪表盘。

排查问题场景示例

场景1：一台 Node 节点出现异常，定位原因

1. 打开事件总览仪表盘，在资源对象中输入异常节点名称。

2. 查询结果显示，有一条节点磁盘空间不足：



场景2：节点触发扩容了，用户需要对扩容过程进行回溯，以确定具体原因

开启 节点池「弹性伸缩」的集群，CA（cluster-autoscaler）组件会根据负载状况自动对集群中节点数量进行增减。如果集群中的节点发生了自动扩（缩）容，用户可通过事件检索对整个扩（缩）容过程进行回溯。

1. 登录 TKE 控制台。
2. 在左侧导航栏中，单击**日志管理** > **事件日志**。
3. 在事件检索页面，单击**全局检索**页签，并输入以下检索命令：

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

4. 结果如下图所示：

日志时间 ↑	event.reason	event.message	event.involvedObject.name	event.involvedObject.namespace
2020-11-25 20:35:43	ScaledUpGroup	Scale-up: setting group asg-qyi22zfi size to 1	cluster-autoscaler-status	kube-system
2020-11-25 20:35:45	ScaledUpGroup	Scale-up: group asg-qyi22zfi size set to 1	cluster-autoscaler-status	kube-system
2020-11-25 20:35:45	TriggeredScaleUp	pod triggered scale-up: [{asg-qyi22zfi 0->1 (max: 3)}]	nginx-5dbf784b68-tq8rd	default
2020-11-25 20:35:45	TriggeredScaleUp	pod triggered scale-up: [{asg-qyi22zfi 0->1 (max: 3)}]	nginx-5dbf784b68-fpvbx	default
2020-11-25 20:57:15	ScaledUpGroup	Scale-up: setting group asg-qyi22zfi size to 3	cluster-autoscaler-status	kube-system
2020-11-25 20:57:15	TriggeredScaleUp	pod triggered scale-up: [{asg-qyi22zfi 1->3 (max: 3)}]	nginx-5dbf784b68-v9jv5	default
2020-11-25 20:57:15	NotTriggerScaleUp	pod didn't trigger scale-up (it wouldn't fit if a new node is added): 1 node(s) didn't match node selector	ccs-log-collector-55nw9	kube-system
2020-11-25 20:57:15	ScaledUpGroup	Scale-up: setting group asg-qyi22zfi size to 3	cluster-autoscaler-status	kube-system
2020-11-25 20:57:15	ScaledUpGroup	Scale-up: group asg-qyi22zfi size set to 3	cluster-autoscaler-status	kube-system
2020-11-25 20:57:15	ScaledUpGroup	Scale-up: group asg-qyi22zfi size set to 3	cluster-autoscaler-status	kube-system
2020-11-25 20:57:15	NotTriggerScaleUp	pod didn't trigger scale-up (it wouldn't fit if a new node is added): 1 node(s) didn't match node selector	ccs-log-collector-dg9rc	kube-system
2020-11-25 20:57:15	TriggeredScaleUp	pod triggered scale-up: [{asg-qyi22zfi 1->3 (max: 3)}]	nginx-5dbf784b68-v7dn2	default
2020-11-25 20:57:15	TriggeredScaleUp	pod triggered scale-up: [{asg-qyi22zfi 1->3 (max: 3)}]	nginx-5dbf784b68-fdjhm	default
2020-11-25 20:57:36	NotTriggerScaleUp	pod didn't trigger scale-up (it wouldn't fit if a new node is added): 1 max limit reached	nginx-5dbf784b68-v7dn2	default

通过上图的事件流水，可以看到节点扩容操作在 `2020-11-25 20:35:45` 左右，分别由三个 nginx Pod(nginx-5dbf784b68-tq8rd、nginx-5dbf784b68-fpvbx、nginx-5dbf784b68-v9jv5) 触发，最终扩增了3个节点，后续的扩容由于达到节点池的最大节点数没有再次触发。

CSS 云直播日志分析

最近更新时间：2025-04-02 15:20:42

简介

通过本篇实践，您可以详细的了解如下内容：

- 采集 CSS 云直播日志
- 查看 CSS 云直播分析仪表盘
- 检索分析 CSS 云直播访问日志
- 监控告警配置案例

您也可通过以下 Demo，快速体验使用 CLS 分析 CSS。

- CSS 云直播日志分析仪表盘
- 检索分析 CSS 云直播日志

功能优势

- 日志实时分析：

  CLS 通过对云直播日志的实时采集、投递，实现对日志数据的快速检索、分析及存储，通过对日志数据的挖掘来实现数据驱动运维及运营，从而快速准确地制定运营策略。
- 分析报表开箱即用：

  CLS 为 CSS 云直播日志分析场景提供了开箱即用的访问分析仪表盘，仪表中包含 CSS **用户访问分布**、**流量分析**、**请求错误率分析**、**请求耗时分析**以及**资源分布**等内容。而在传统 CSS 日志分析场景中，需下载离线日志，再上传至数据仓库，然后在数据仓库进行一系列的数据清洗和数据模型定义。这一过程繁琐又消耗较多人力成本。

采集 CSS 云直播日志

详情请参见 实时采集 CSS 云直播日志 。

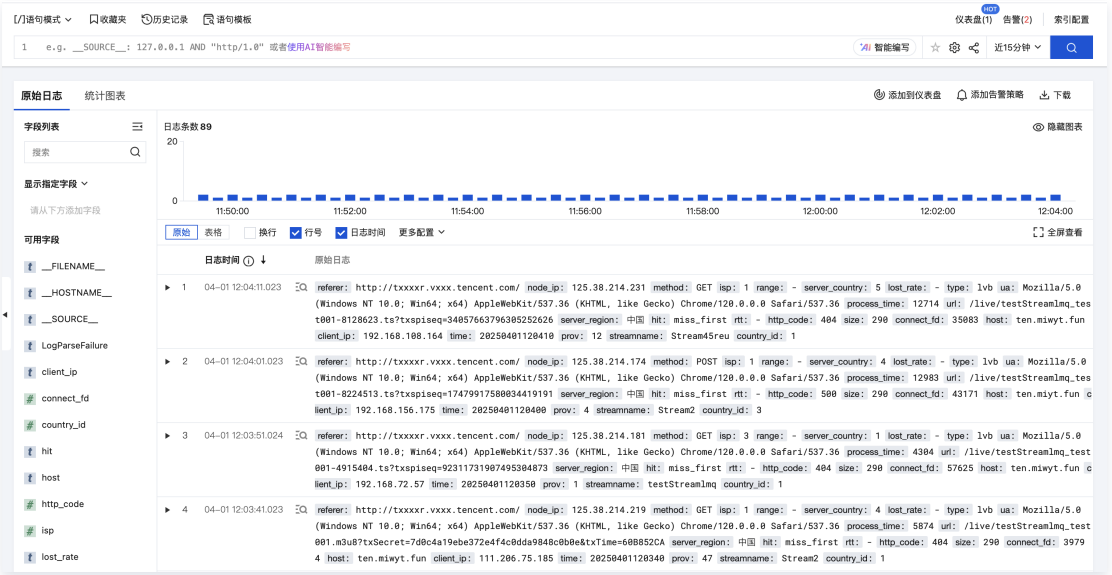
查看 CSS 云直播分析仪表盘

完成 实时采集 CSS 云直播日志 后，CLS 提供开箱即用的预置仪表盘。
CSS 云直播日志分析仪表盘：可视化展示 CSS 的用户访问情况（如用户 UV、PV、用户访问地理分布等）和请求质量情况（如错误率、延时等），助力用户运营和运维排障场景。

检索分析 CSS 云直播日志

1. 完成日志采集后，进入 云直播控制台 ，选择**业务监控** > **日志服务** > 实时日志分析 ，进入实时日志分析。查看日志主题，单击**检索**。

2. 单击检索后将跳转至日志检索页，即可检索分析日志。



日志字段说明

- **推流日志**

顺序	日志字段	说明
1	time	请求时间
2	client_ip	客户端 IP
3	host	被访问的域名
4	url	URL
5	size	推流字节数大小
6	country_id	country_id
7	prov	省份
8	isp	运营商
9	streamname	流 ID
10	node_ip	节点 IP
11	server_region	服务器地区
12	server_country	服务器国家

- **播放日志**

顺序	日志字段	说明
1	type	播放类型，lvb 代表标准直播，leb 代表快直播
2	time	请求时间
3	client_ip	客户端 IP
4	host	被访问的域名
5	url	URL
6	size	本次访问字节数大小
7	country_id	country_id
8	prov	省份
9	isp	运营商
10	http_code	HTTP 状态码
11	referer	Referer 信息
12	process_time	处理时长（单位：毫秒）
13	ua	User - Agent 信息
14	range	Range 参数
15	method	HTTP Method
16	streamname	流 ID
17	hit	缓存 HIT/MISS

版权所有：腾讯云计算（北京）有限责任公司

18	node_ip	节点 IP（因无法获取部分 CDN 集群节点 IP，此字段可能为空）
19	server_region	服务器地区
20	server_country	服务器国家
21	connect_fd	connect_fd（连接端口号）
22	lost_rate	丢包率，仅 type=leb 时才有值
23	rtt	rtt，仅 type=leb 时才有值

注意：
日志中特殊状态码说明如下：
- 0：连接建立。
- 4：请求超时，鉴权超时或者响应超时。
- 5：回源断连接或者流销毁。
- 6：客户端断连接。

国家（地区）、省份、运营商、服务器地区与国家（地区）映射可参见 日志下载 。

监控告警案例

您可以基于 CSS 日志配置异常监控告警，实时监控 CSS 访问流量中发生的异常。以下提供了两个案例。

案例1：针对当 P99的延时大于100ms进行告警，并且在告警信息中展示受影响域名、url、client_ip，以便快速判断错误情况。

1. 登录 日志服务控制台 ，并进入 监控告警 > 告警策略 管理页面，单击**新建**，进入告警策略创建页。
2. 在告警策略页中，配置如下内容。
   - **基本信息**：
     - **告警名称**：CSS 访问延迟告警。
     - **启用状态**：启用
   - **监控对象**：选中创建的 CSS 日志主题。
   - **监控任务**：
     - **执行语句**：输入以下语句，时间范围选择15分钟，统计近15分钟内的99%延时。

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

     - **触发条件**：配置如下，即99%延时大于100ms 时，满足告警条件。

```
$1.p99 > 100
```

     - **执行周期**：固定频率，每1分钟执行一次。
   - **多维分析**：在告警信息中展示受影响的域名、客户端 IP、url，帮助开发人员快速定位问题。

- **通知渠道组**：通过关联通知渠道组，设置发送通知的方式及对象，支持短信、邮件、电话、微信、企业微信、钉钉、飞书、自定义接口回调（webhook）等通知方式。详情请参见 管理通知渠道组 。

案例2：资源访问错误率或延迟超过一定阈值时，触发告警通知。

参考 案例1，登录 日志服务控制台，并进入 监控告警 > 告警策略 管理页面，单击**新建**，进入告警策略创建页。

- **执行语句**：
  - 输入语句1，时间范围选择近15分钟。

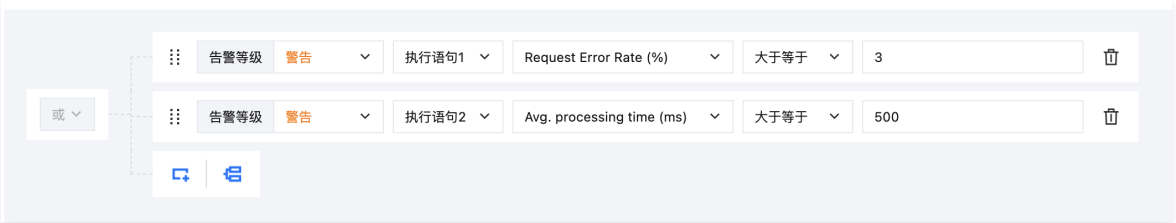
```
type:* | select url_extract_path(url) as url_path , round(count_if(try_cast( "http_code" as bigint) >= 400)*100.0/count(*),2) as "Request Error Rate (%)"  group by "url_path" order by "Request Error Rate (%)" desc limit 100
```

  - 输入语句2，时间范围选择近15分钟。

```
type:* | select url_extract_path(url) as url_path , round(avg( "process_time" ), 1) as "Avg. processing time (ms)" group by "url_path" order by "Avg. processing time (ms)" desc limit 100
```

- **触发条件**：

根据业务情况自行定义，如案例设置资源访问的错误率大于3%或耗时超过500ms时，触发告警。



- **多维分析：**
  - 展示错误率超过阈值的资源 URL。

```
type:* | select url_extract_path(url) as url_path , round(count_if(try_cast( "http_code" as bigint) >= 400)*100.0/count(*),2) as "Request Error Rate (%)"  group by "url_path" having "Request Error Rate (%)" >=3 order by "Request Error Rate (%)" desc limit 100
```

  - 展示耗时超过阈值的资源 URL。

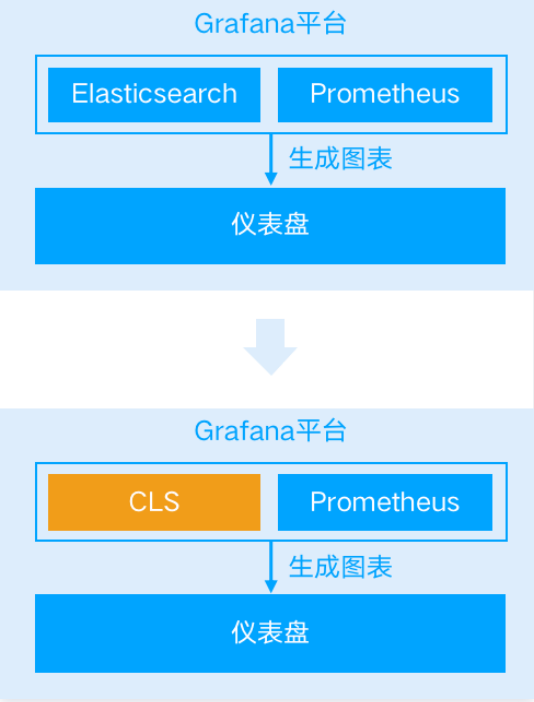
```
type:* | select url_extract_path(url) as url_path , round(avg( "process_time" ), 1) as "Avg. processing time (ms)" group by "url_path" having "Avg. processing time (ms)" >=500 order by "Avg. processing time (ms)" desc limit 100
```

仪表盘
把 Grafana 的 ES 数据源迁移为 CLS 数据源

最近更新时间：2025-02-28 14:46:22

背景

在日志服务（Cloud Log Service，CLS）使用场景里，从其他日志工具迁移到 CLS 是非常常见的情况。其中，存在用户使用 Grafana 做可视化监控工具，例如 ES + Grafana 的组合。当数据源迁移到 CLS 后，用户依托 Grafana 制作的各种仪表盘资源，搭建的运维工具和平台就都失去了作用。为了避免重建这套体系，需要 CLS 对接 Grafana，替换 ES 数据源。



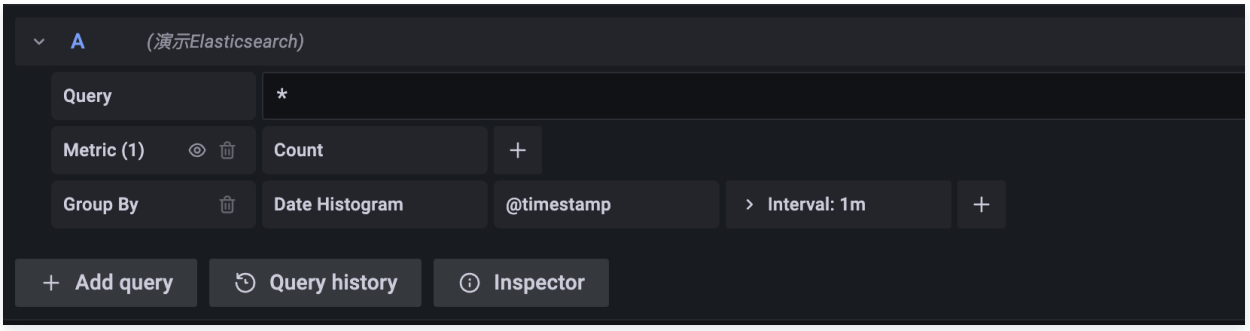
安装 CLS-Grafana 插件

CLS 数据源由腾讯云日志服务团队进行维护，已经通过 官方签名认证，可以在 Grafana 设置页面一键安装。具体接入步骤请参见 CLS 对接 Grafana。

使用 CLS 数据源替换 ES 数据源

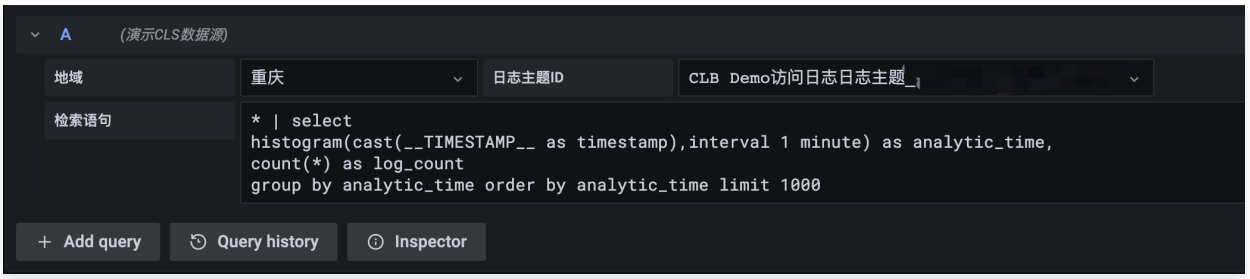
数据源配置区域对比

- **ES 数据源**：查询语句界面分为顶部的 **Query 输入区**和其余的**辅助输入功能区**。Query 输入区可输入 Lucene 语句，用于对日志进行过滤。辅助输入区通过单击填写，生成 DSL 内容，用于数据聚合，相当于 CLS 的 SQL。



- **CLS 数据源**：查询语句界面分为**地域与日志主题**选择和**检索分析语句**两个部分。地域与日志主题选择模块可以快速进行日志主题切换，而检索分析语句则用于输入 CLS 查询语句。

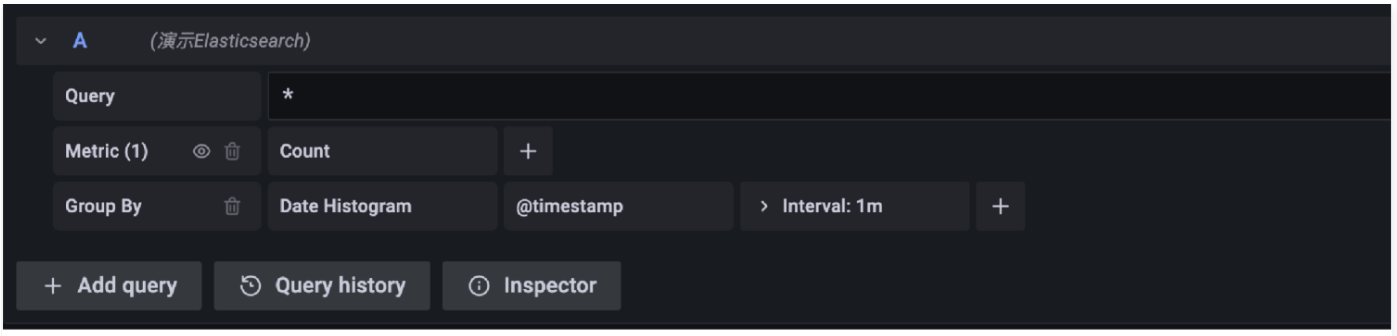
CLS 查询语句分为 Lucene 和 SQL 两个部分，两个部分之间使用管道符 “|” 进行分隔。其中 Lucene 部分和 ES 的 Query 输入区内容相同。SQL 输入内容除了支持标准的 SQL 语法外，还支持大量的 SQL 函数，SQL 区域内容和 ES 输入区的辅助输入模块完成对标。更多请参见 CLS 语法规则 。



实践示例

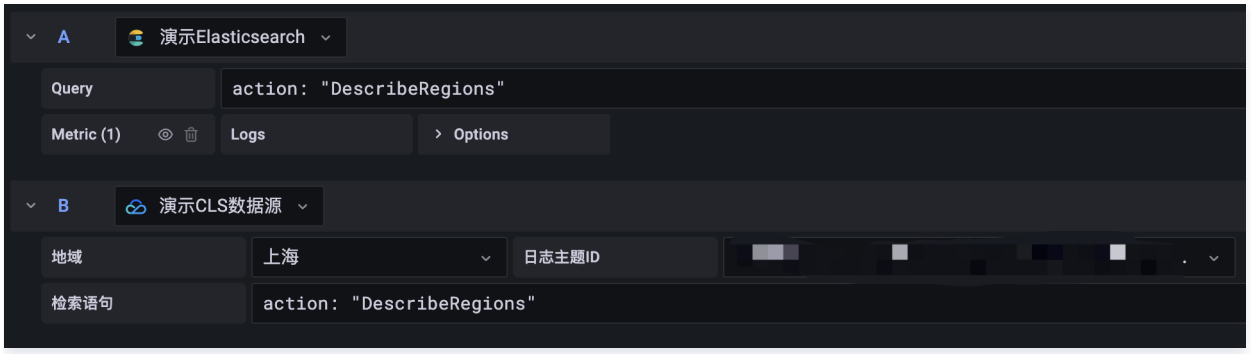
统计日志条数

对于想要绘制随时间变化的日志条数，ES 数据源将 Metric 选中 Count，GroupBy 选中 Histogram。CLS 的检索语句可以使用 Histogram 结合聚合函数 Count 完成。 类似的，对于 Max、Min、Distinct 等其他 通用聚合函数 使用上也完全一致，直接将 Count 函数进行替换即可。



查看原始日志

想要直接查看符合条件的日志，ES 数据源需要将 Metric 选中 Logs 模式，而 CLS 只需要输入对于的 Lucene 语句即可。输入语句比对：



展示效果：

聚合统计 – 错误码占比

根据错误码进行聚合，展示各个错误码的日志数量。此处可以看到，语句中包含变量 $path。CLS 数据源插件进行了变量功能的相关适配，允许直接使用 Grafana 的变量能力。

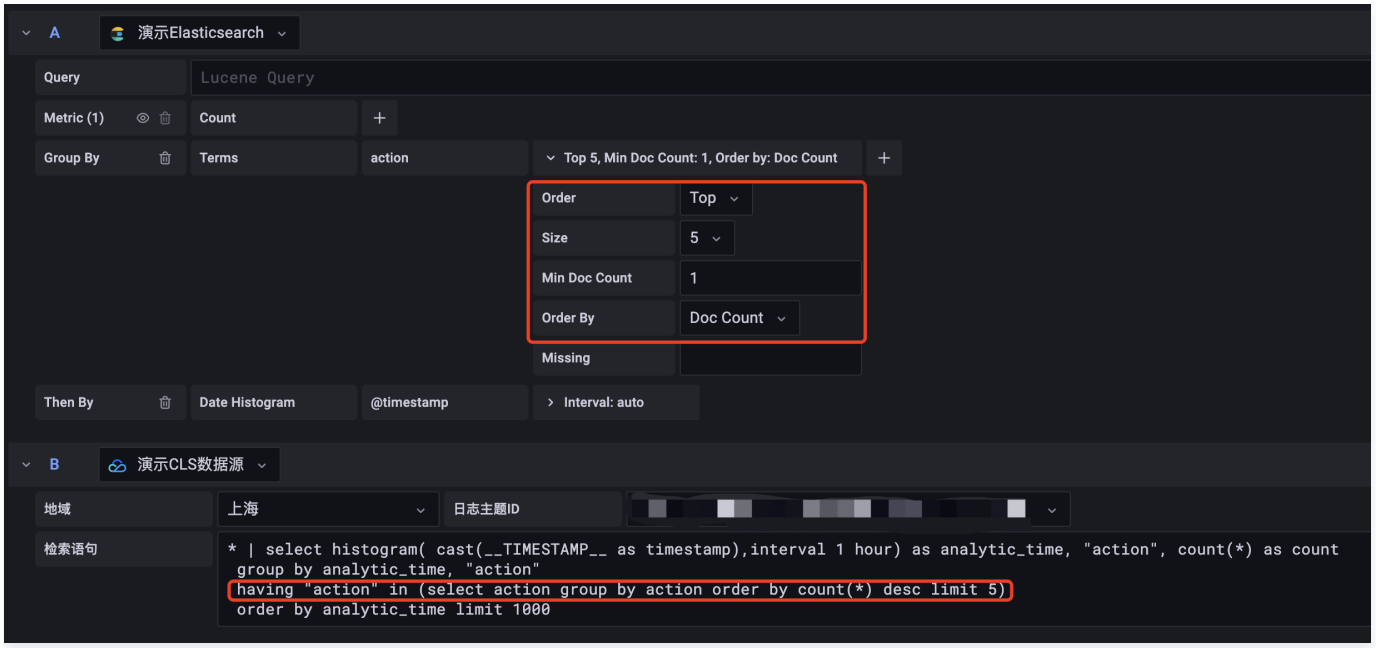
> **注意：**
> 绘制饼图时右侧图表选项请选择 ValueOptions-AllValues。

展示效果：

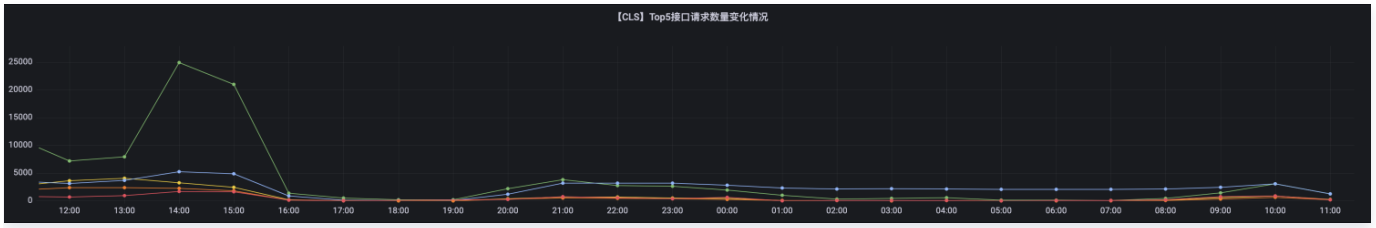
聚合统计 – Top5请求的数量变化情况

ES 数据源中，GroupBy 聚合选项允许填写 Size 值，支持选中出现频率最高的 N 个值，再进行聚合。
此情况在 CLS 数据源 SQL 中，可以通过 having 语句搭配嵌套子查询实现。

```
*|select histogram( cast(__TIMESTAMP__ as timestamp),interval 1 hour) as
analytic_time,"action",count(*) as count group by analytic_time, "action" having "action" in (select
action group by action order by count(*) desc limit 5) order by analytic_time limit 1000
```



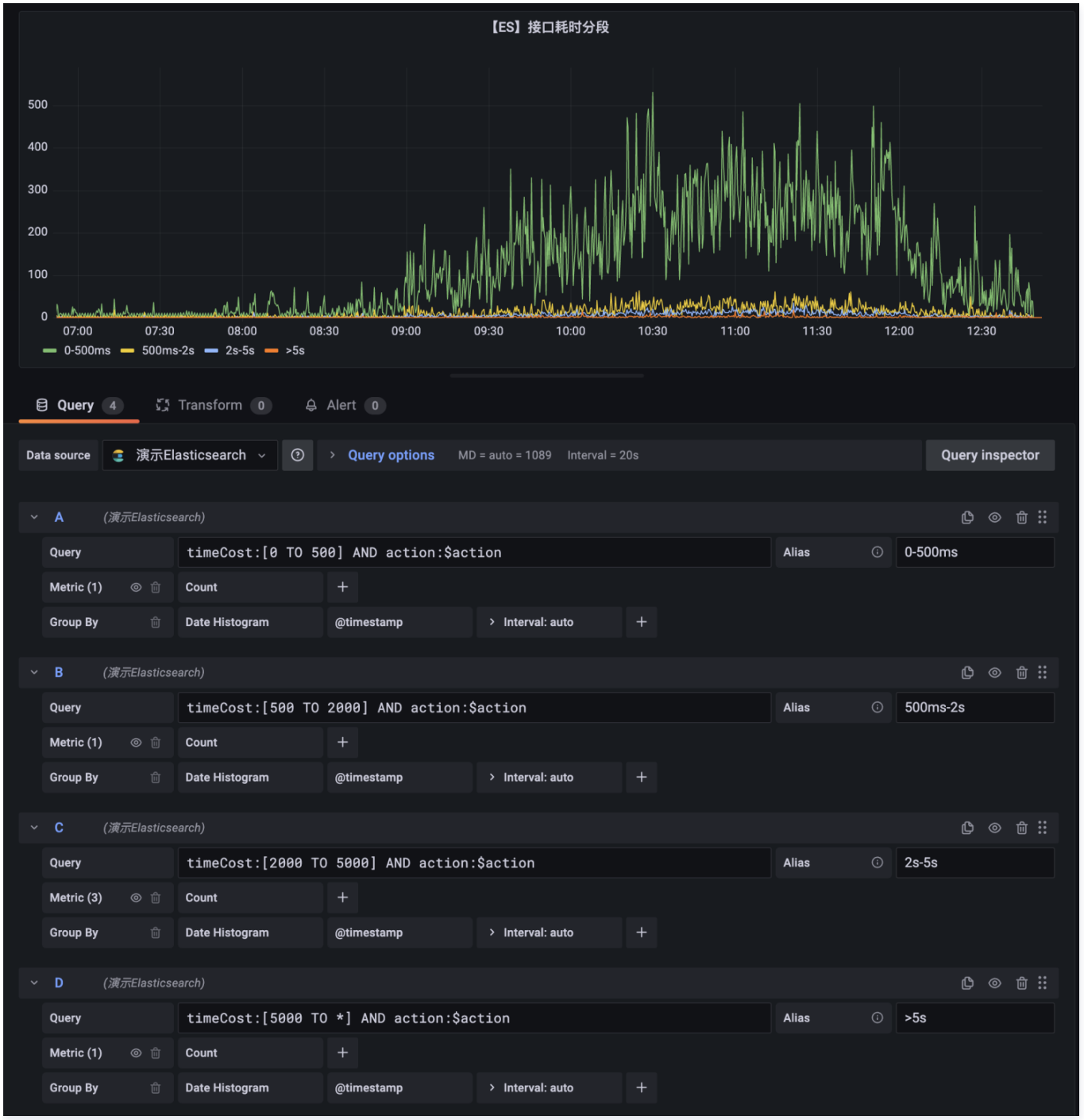
查询结果可以看到，图中共有5条曲线。



通过以上的语句搭配使用，已经可以满足大部分的检索分析场景。

统计接口耗时的分段情况

在 ES 数据源仪表盘中，有一个配置项繁多，但场景适用广的示例：根据不同的时间范围，绘制在这个时间范围的请求数量。
这个案例，统计了接口在0到500ms，500ms到2s，2s到5s，以及大于5秒的请求个数。



对应的，迁移到 CLS 数据源，也可以使用类似的多条语句进行绘制。但 CLS 本身的 SQL 能力更强，可以将相关的统计处理合并成一条 SQL 语句：

```
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<=500) 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 "超过5s" group by analytic_time order by
analytic_time limit 1000
```

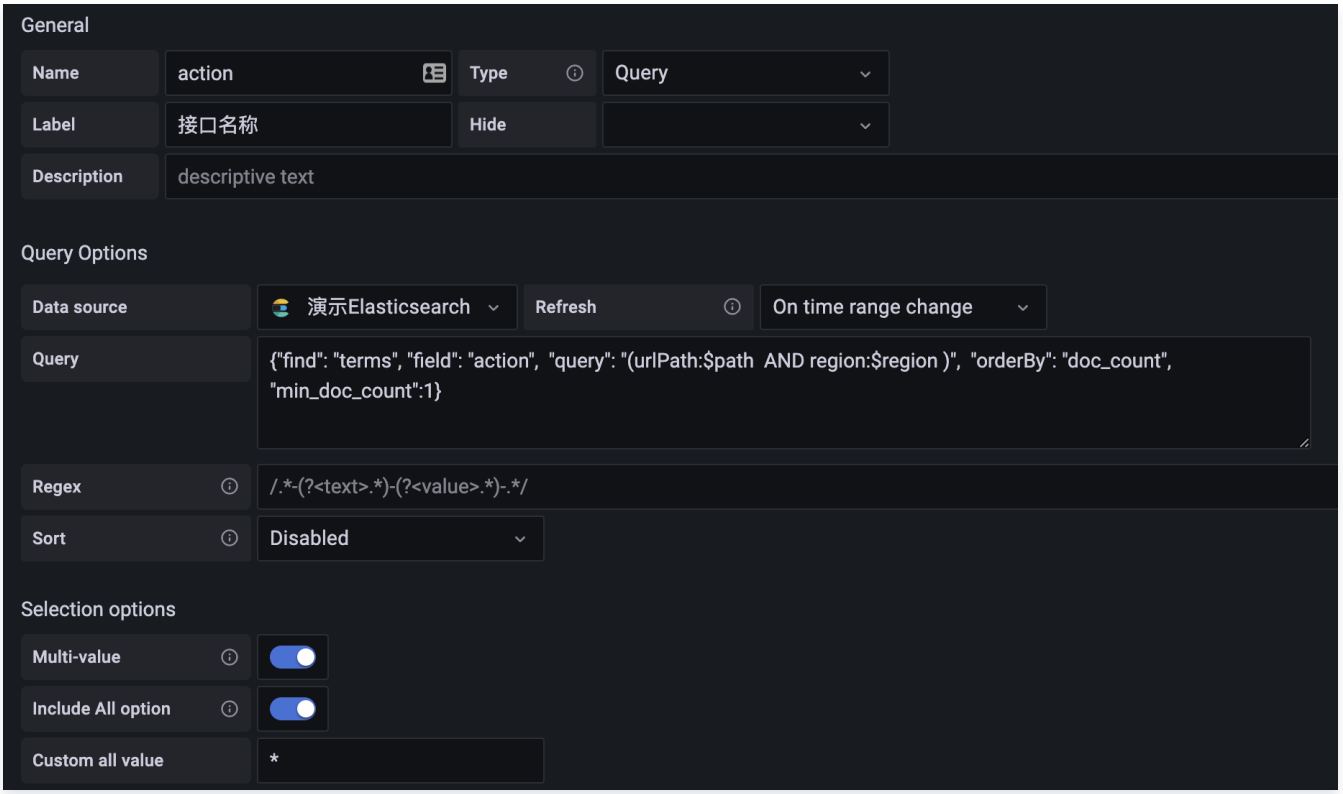
类似的场景，我们也可以写出使用估算函数 approx_percentile 分析得出的耗时相关情况。

```
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
```

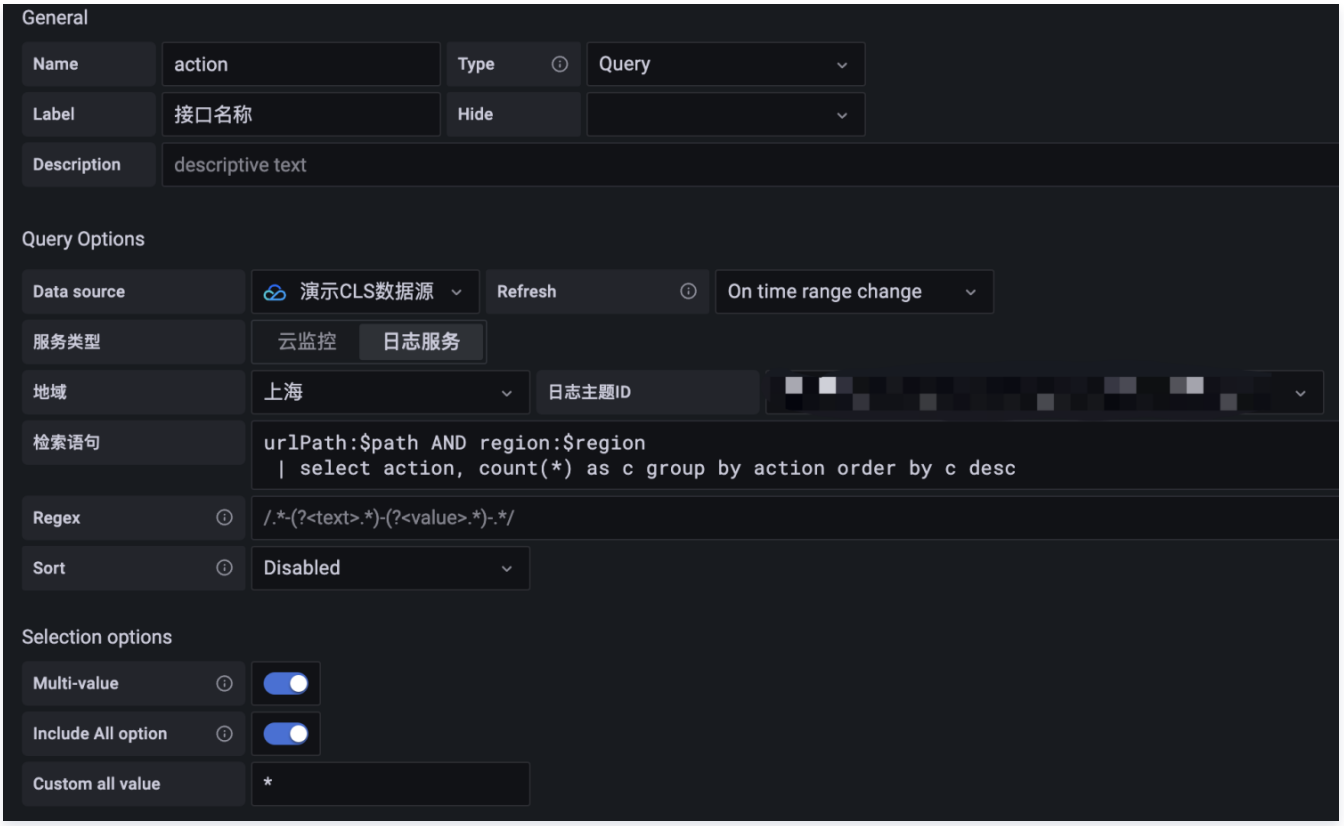
模板变量能力

在以上的案例中，不同程度的出现了 Grafana 变量功能的身影。对于变量功能，Grafana 变量的类型种类繁多。常量类型、Textbox 输入框类型对各类数据源来说，是完全相同的，无需进行迁移。这里主要介绍如何迁移 Query 类型变量。

- **ES 版本的 $action 变量**：用于展示出现的接口种类，ES 数据源的版本使用 DSL 进行描述，语义上是找到符合 query 条件为 `urlPath:$path AND region:$region` 的内容，再选取 action 字段，并按照出现次数排序。



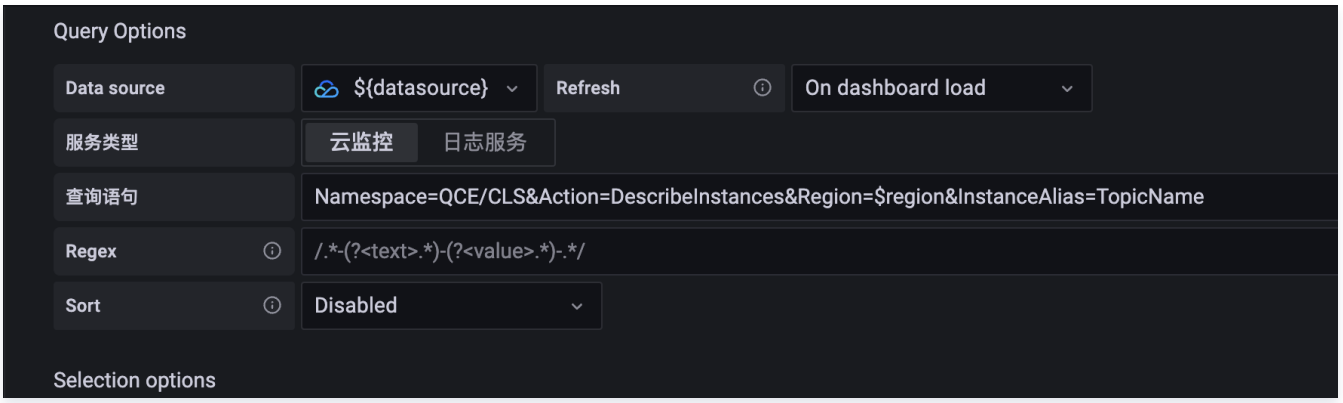
- **CLS 版本的 $action 变量**：使用体验上与在图表编辑的输入行为上，保持一致。选择服务类型为日志服务并选中对应的日志主题后，输入 SQL 语句，即可达到相同效果。详情请参见 在 Grafana 配置变量 。



- 除了使用 CLS 的检索语句进行变量查询，还可以使用腾讯云可观测平台的资源查询功能，将腾讯云上的服务资源，作为列表内容进行展示。功能文档请参见 腾讯云可观测平台数据源模板变量功能 。 如使用语句：

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

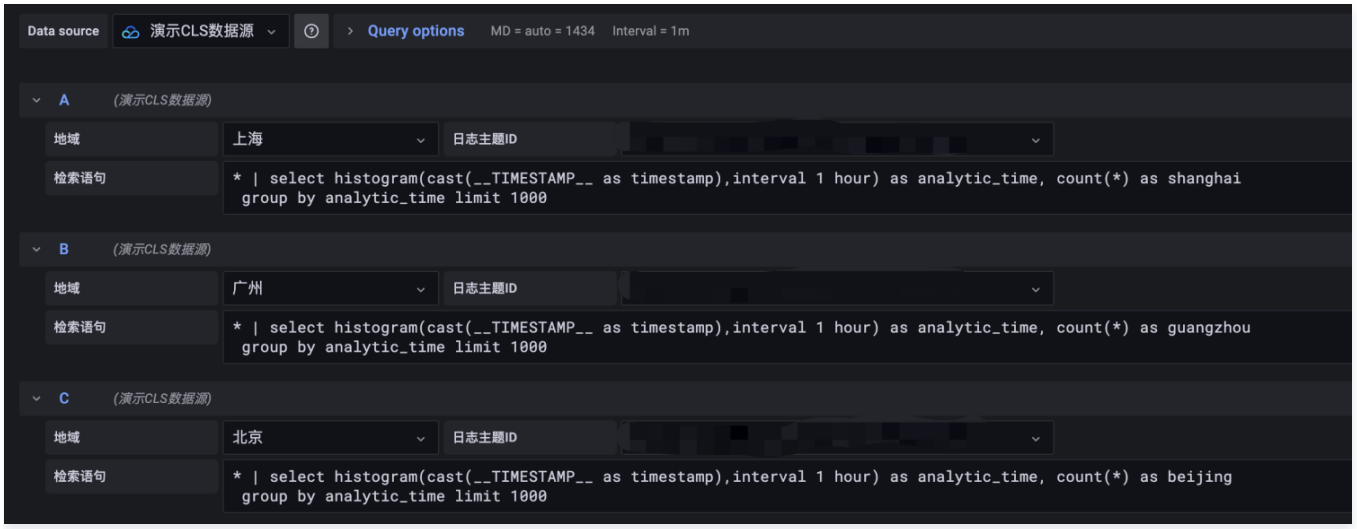
查询日志主题列表：



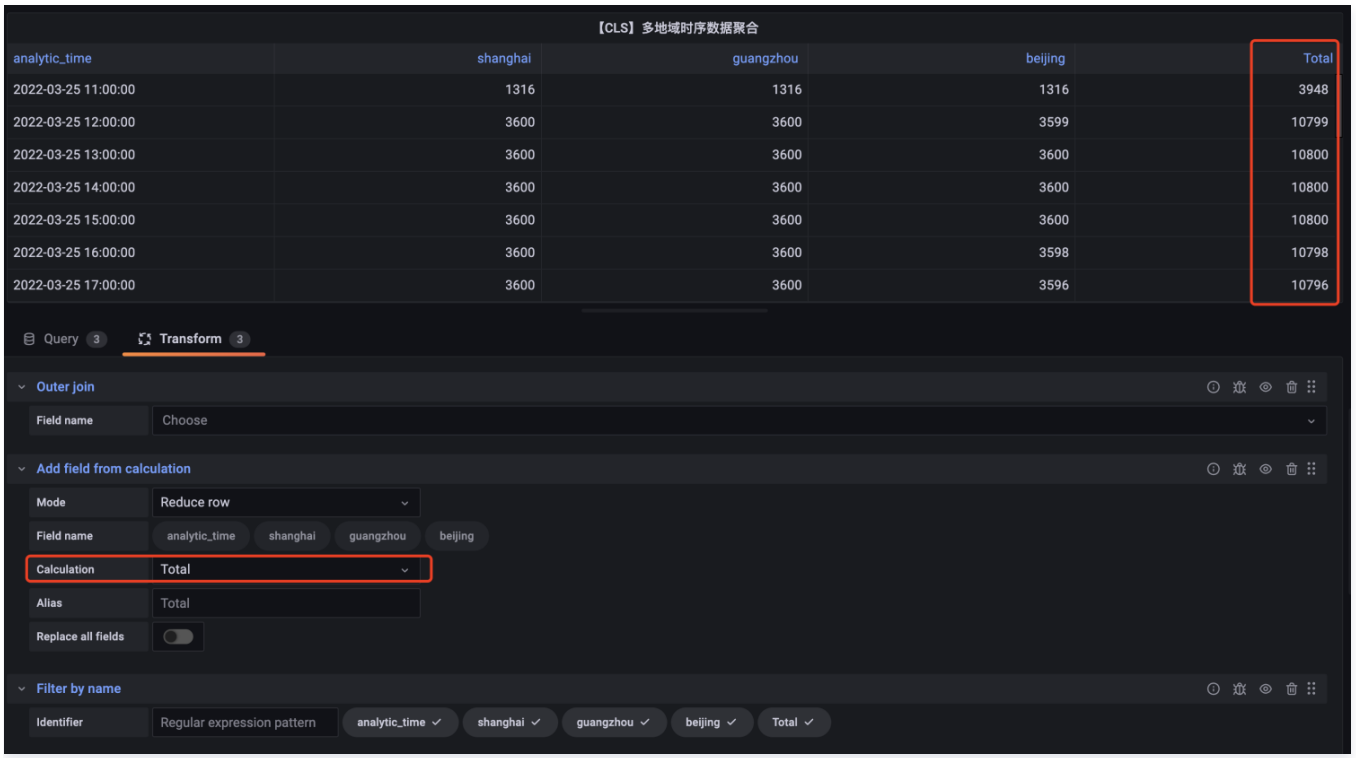
合并不同地域的请求数据内容

在原本的实现中，部分用户会存在所有数据都存储在同一台 ES 实例上的情况。在使用 CLS 之后，采用就近原则创建了多个日志主题。此时，用户可能会想要将多个日志主题内容合并到图表中。

对于3条来自不同地域的日志查询：



我们可以使用 Transform 模块，实现数据求和的效果，并选用需要的图表进行展示。



总结

对于存量的 ES 仪表盘，重复以上的迁移步骤，就可以将一个 ES 数据源的仪表盘，完全转化成为 CLS 数据源的仪表盘。
ES 到 CLS 数据源的迁移，可以让用户从自建 ELK 迁移到腾讯云日志服务后，积累的可视化资源得到继续的利用。
转化之后的仪表盘，不仅在能力上完全对标 ES 数据源版本，还可以结合数据源插件的一些其他能力（如腾讯云可观测平台模板变量），更好地与腾讯云生态进行融合。

监控告警
按时间段分别设置告警触发条件

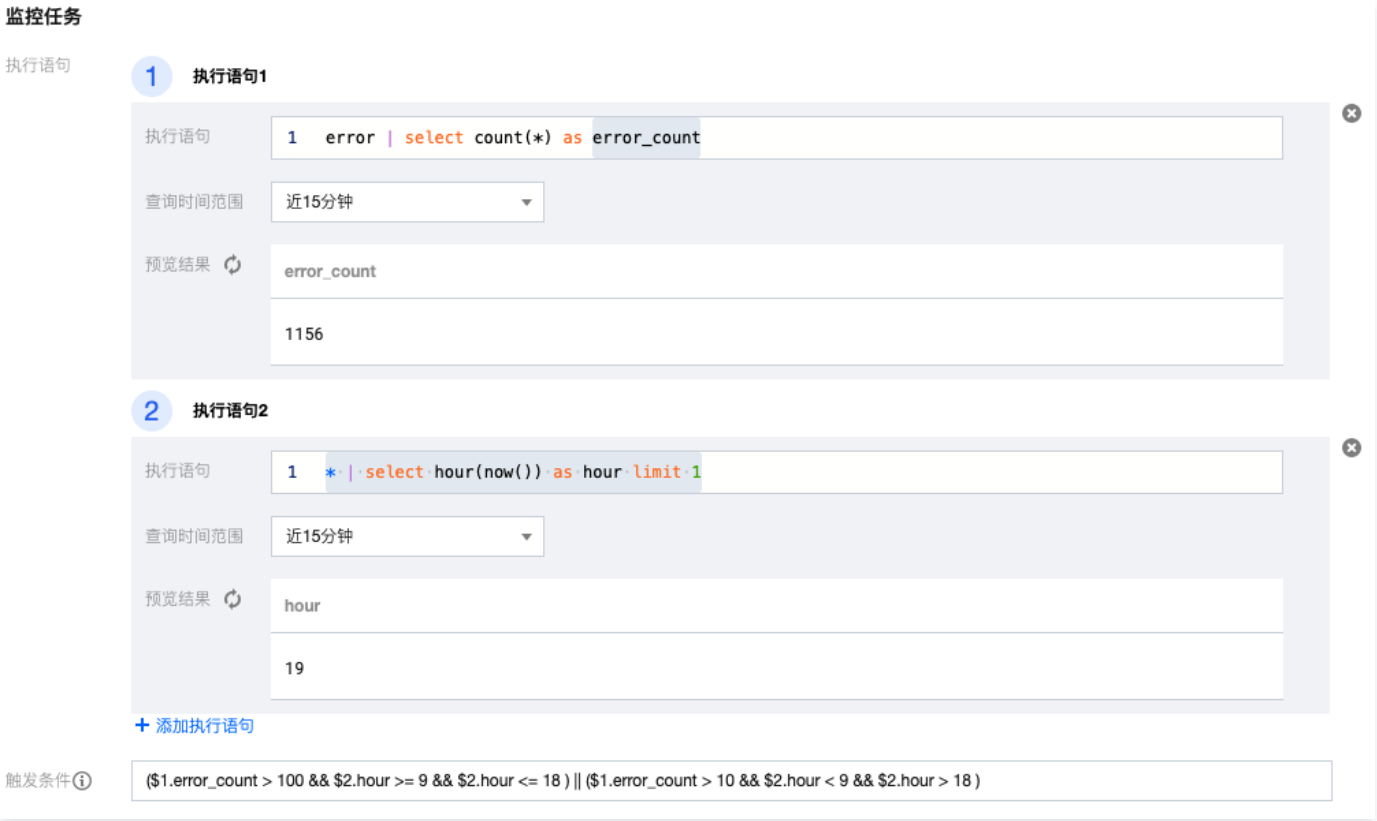
最近更新时间：2024-05-29 14:28:51

简介

因为业务性质等原因，配置告警策略时，需要针对不同时间段分别设置不同的告警触发条件。例如：工作时间段（09点 – 18点）包含 “error” 的日志超过100条触发告警，非工作时间（19点 – 次日08点）包含 “error” 的日志超过10条触发告警。

配置方式

配置告警策略 过程中，填写如下执行语句及触发条件：



执行语句1:

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

统计包含 “error” 的日志条数。

执行语句2:

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

使用 日期和时间函数 获取告警执行时刻的小时部分，即告警时属于几点。

触发条件:

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

使用 触发条件表达式 指定具体的告警触发条件及阈值。其中，`$1.error_count>100 && $2.hour>=9 && $2.hour<=18` 表示告警执行时间在09点 – 18点时，`error_count` 大于100才会触发告警；`$1.error_count>10 && ($2.hour<9 || $2.hour>18)` 表示告警时间在9点前及18点后（即19点

- 次日08点）时，`error_count` 大于10即会触发告警。

使用同环比作为告警触发条件

最近更新时间：2024-05-29 11:46:11

简介

设定告警触发条件时，由于业务特点，常需要对指标进行环比，变化超过一定阈值再触发告警。例如接口响应时间相比昨天同时间段上升超过50%即触发告警。

配置方式

配置告警策略 过程中，填写如下执行语句及触发条件：

执行语句：

```
* | 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
      )
  )
```

上述语句的执行结果中：

- ratio 代表：当前接口平均响应时间相比昨天（即86400秒前）的比值
- current_avg_request_time 代表：当前接口平均响应时间
- yesterday_avg_request_time 代表：昨天同时间段接口平均响应时间

上述语句中主要使用了 compare 函数，详细说明请参见 同环比函数 。

触发条件：

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

ratio 大于1.5即触发告警，即相比昨天上升超过50%。

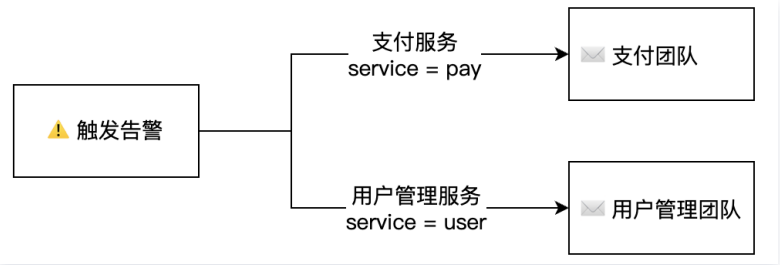
按日志所属服务将告警发送到不同的团队

最近更新时间：2024-04-25 16:32:11

简介

访问日志中包含多个服务(service)的日志，告警时需要按服务将告警分别发送至所属的团队。例如以下日志，期望每个服务出现5条以上错误日志时将告警发送至所属团队。

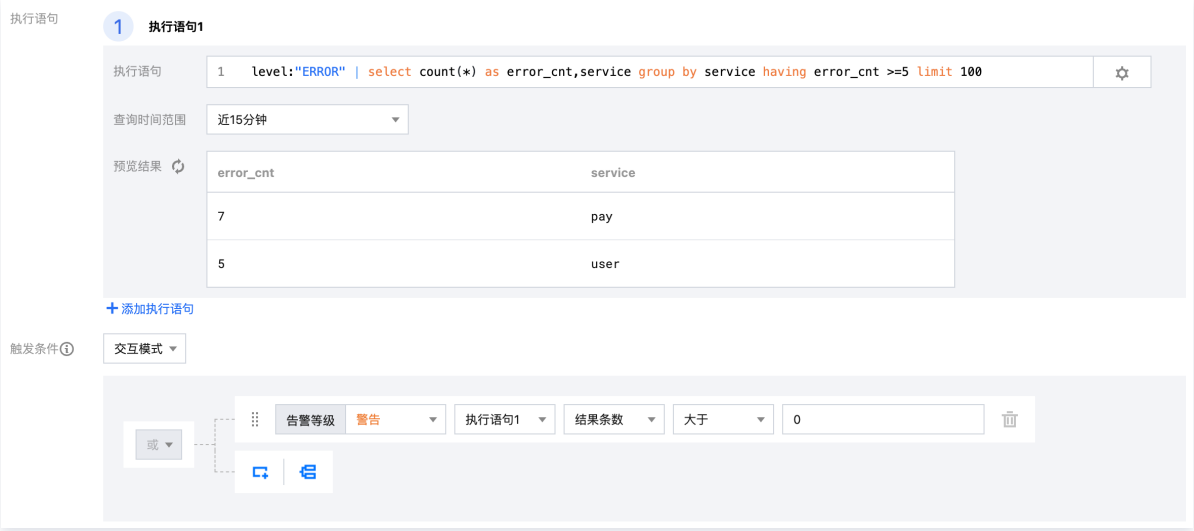
行号	日志数据
1	service： user level： ERROR msg： xxxxxx
2	service： pay level： ERROR msg： xxxxxx



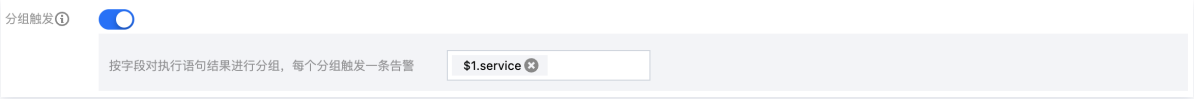
配置步骤

步骤一：配置告警策略，填写如下配置：

1. 执行语句及触发条件：使用 SQL 分别统计各个 service 的错误日志条数，并过滤出错误日志条数大于等于5的 service。



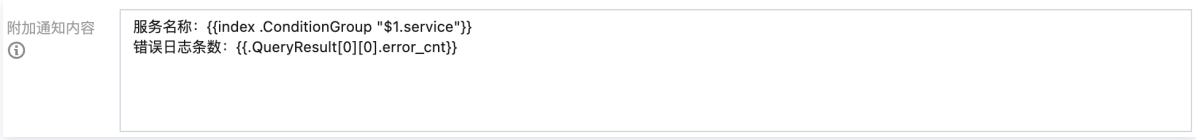
2. 启用分组触发：按照 service 对告警进行分组，每个 service 单独发送告警。



3. 添加多维分析：触发告警时，查看该 service 的日志详情(msg)。其中 `{{index .ConditionGroup "$1.service"}}` 表示本次告警对应的 service。

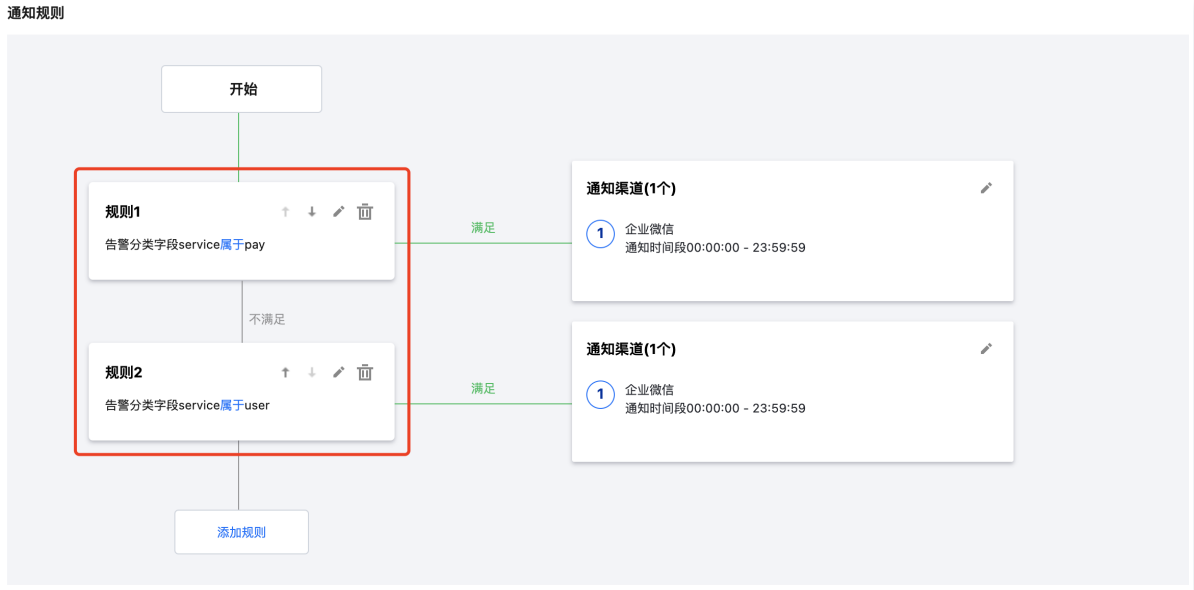
4. 添加告警分类：其中 `{{index .ConditionGroup "$1.service"}}` 表示本次告警对应的 service

5. 添加自定义告警通知内容：将触发告警的服务名称及对应的错误日志条数展示在告警通知中



步骤二：配置通知渠道组

使用如下通知规则，按服务(service)将告警分别发送至所属的团队。



步骤三：接收告警通知

alarm BOT

【告警】[警告]账号 1[redacted] 1) 的日志服务触发告警:
告警策略：test
监控对象：test
触发条件：[$1.__QUERYCOUNT__]> 0 [$1.service=user]
触发时间：2024-04-24 11:04:39
持续时间：223 分钟
附加通知内容：服务名称：user
错误日志条数：6
多维分析：
错误详情

> **msg**
> **xxxxxx**
> **xxxxxx**
> **xxxxxx**
> **xxxxxx**
> **xxxxxx**
> **xxxxxx**

详细报告　查询数据　屏蔽告警

告警对接 PagerDuty/Slack 等第三方平台

最近更新时间：2025-06-18 19:38:01

简介

告警可通过自定义回调的方式对接 PagerDuty、Slack、Microsoft Teams、Jira Service Management、Google Chat，便于统一接收告警通知。

配置步骤

步骤1：新建通知内容模板

1. 登录 日志服务控制台 。
2. 在左侧导航栏中，选择**监控告警** > **通知内容模板**，进入通知内容模板管理页面。
3. 单击**新建**，在**自定义回调**标签中，根据需要对接的渠道，填写如下信息并保存。

PagerDuty

说明：
- CLS 通过 Events API V2与 PagerDuty 对接，如果当前 Service 未添加 Events API V2 Integration，请执行 Add Integrations to an Existing Service 。
- 请替换以下请求内容中的 `routing_key` 为 PagerDuty 中的 Integration Key。
- 请同时记录 PagerDuty 中的 Integration URL，便于后续步骤使用。

告警触发

- 请求头

```
Accept: application/json
```

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

- 请求内容

```
{
  "payload": {
    "summary": "{{escape .Alarm}}",
    "timestamp": "{{fromUnixTime .NotifyTimeUnix}}",
    "severity": "{{if eq .Level "Critical"}}critical{{else if eq .Level "Warn"}}warning{{else if
eq .Level "Info"}}info{{end}}",
    "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": "R03ECCMUCxxxxxxxxxxxNGFE87CT",
  "dedup_key": "{{.RecordGroupId}}",
  "event_action": "trigger",
  "client": "{{escape .Topic}}",
  "client_url": "{{.QueryUrl}}",
```

```
  "links": [
    {
      "href": "{{.DetailUrl}}",
      "text": "Alert Detail"
    }
  ]
}
```

告警恢复

- **请求头**

```
Accept: application/json
```

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

- **请求内容**

```
{
  "payload": {
    "summary": "{{escape .Alarm}}",
    "timestamp": "{{fromUnixTime .NotifyTimeUnix}}",
    "severity": "{{if eq .Level "Critical"}}critical{{else if eq .Level "Warn"}}warning{{else if
eq .Level "Info"}}info{{end}}",
    "source": "Tencent Cloud Log Service"
  },
  "routing_key": "R03ECCMUCxxxxxxxxxxxNGFE87CT",
  "dedup_key": "{{.RecordGroupId}}",
  "event_action": "resolve"
}
```

Slack

告警触发

- **请求头**

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

- **请求内容**

```
{{- 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}}
<{{.DetailUrl}}|ClaimAlarm>  <{{.SilentUrl}}|SilenceAlarm>{{end}}
{{- end -}}
payload={"username": "CLS Alert","icon_emoji": ":rotating_light:","blocks":
[{"type":"header","text":{"type":"plain_text","text":":rotating_light:Alarm:{{escape
.Alarm}}","emoji":true}},{"type": "section","text": {"type": "mrkdwn","text": "{{escape (substr
(renderTemplate "subTemplate")  0 3500)}}"}}]}
```

告警恢复

- **请求头**

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

- **请求内容**

```
{{- 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":"header","text":{"type":"plain_text","text":":large_green_circle:Resolved:{{escape
.Alarm}}","emoji":true}},{"type": "section","text": {"type": "mrkdwn","text": "{{escape (substr
(renderTemplate "subTemplate")  0 3500)}}"}}]}
```

Microsoft Teams

告警触发

- **请求头**

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

- **请求内容**

```
{{- 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"}}warning{{else
if eq .Level "Info"}}accent{{end}}",
        "size": "Large"
      },
      {
        "type": "TextBlock",
        "text": "{{- escape (substr (renderTemplate "subTemplate")  0 3500)}}",
        "wrap": true
      },
      {
        "type": "ActionSet",
        "actions": [{"type":"Action.OpenUrl","title":"DetailedReport","url":"{{.DetailUrl}}"},
{"type":"Action.OpenUrl","title":"QueryData","url":"{{.QueryUrl}}"}{{if.CanSilent}},
{"type":"Action.OpenUrl","title":"ClaimAlarm","url":"{{.DetailUrl}}"},
{"type":"Action.OpenUrl","title":"SilenceAlarm","url":"{{.SilentUrl}}"}{{end}}]
      }]
    }
  }]
}
```

告警恢复

- **请求头**

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

- **请求内容**

```
{{- 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
      }]
    }
  }]
}
```

Jira Service Management

> **说明：**
> - CLS 通过 API Integration 与 JSM(Jira Service Management) 对接，如果当前未添加 API Integration，请执行 Add a global integration 。添加完成后启用该 Integration，并获取其中的 API key。
> - 请替换以下请求头中的 `Authorization` 为 JSM 中的 API Key，格式为 `Basic` + 空格 + `Base64` 编码后的 `API Key` 。

告警触发

- 请求头

```
Accept: application/json
```

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

```
Authorization: Basic YjdiYjI5MWXXXXXXXXXXXXXXXXXXdhNDUyYjVmYjQw
```

- 请求内容

```
{{- define "subTemplate" -}}
Current Data:{{.TriggerParams}}
Additional Message:{{.Message}}
{{.AnalysisResultFormat}}
Detail Report:{{.DetailUrl}}
{{- end -}}
{
  "message": "{{escape .Alarm}}",
  "alias": "{{.RecordGroupId}}",
  "description": "Alarm Policy:{{escape .Alarm}}\nTrigger Condition:{{escape .Condition}}",
```

```
  "entity": "{{escape .Topic}}",
  "source": "Tencent Cloud Log Service",
  "priority": "{{if eq .Level "Critical"}}P1{{else if eq .Level "Warn"}}P2{{else if eq .Level
"Info"}}P5{{end}}",
  "note": "{{escape (substr (renderTemplate "subTemplate") 0 3500)}}"
}
```

告警恢复

- **请求头**

```
Accept: application/json
```

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

```
Authorization: Basic YjdiYjI5MWXXXXXXXXXXXXXXXXXXdhNDUyYjVmYjQw
```

- **请求内容**

```
{{- define "subTemplate" -}}
Alarm was resolved
Trigger Time:{{.StartTime}}
Resolved Time:{{.NotifyTime}}
Duration: {{.Duration}} minutes"
{{- end -}}
{
  "message": "{{escape .Alarm}}",
  "alias": "{{.RecordGroupId}}",
  "description": "Alarm Policy:{{escape .Alarm}}\nTrigger Condition:{{escape .Condition}}",
  "entity": "{{escape .Topic}}",
  "source": "Tencent Cloud Log Service",
  "priority": "{{if eq .Level "Critical"}}P1{{else if eq .Level "Warn"}}P2{{else if eq .Level
"Info"}}P5{{end}}",
  "note": "{{escape (substr (renderTemplate "subTemplate") 0 3500)}}"
}
```

Google Chat

> **注意：**
> 中国大陆地域（北京、广州、上海、重庆等）不支持直接对接 Google Chat。

告警触发

- **请求头**

```
Content-Type: application/json; charset=UTF-8
```

- **请求内容**

```
{{- 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}}
<{{.DetailUrl}}|ClaimAlarm>  <{{.SilentUrl}}|SilenceAlarm>{{end}}
{{- end -}}
{"text": "{{escape (substr (renderTemplate "subTemplate")  0 3500)}}"}
```

告警恢复

- **请求头**

```
Content-Type: application/json; charset=UTF-8
```

- **请求内容**

```
{{- 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 -}}
{"text": "{{escape (substr (renderTemplate "subTemplate")  0 3500)}}"}
```

步骤2：新建通知渠道组

1. 在左侧导航栏中，选择**监控告警** > **通知渠道组**，进入通知渠道组管理页面。
2. 单击**新建**，在**通知规则**中单击**添加规则**，在**设置通知渠道**中填写如下信息并保存：
   - 渠道类型：自定义接口回调
   - Webhook地址：
     - PagerDuty：Add Integrations to an Existing Service 步骤中的 Integration URL。
     - Slack：CLS 通过 Incoming Webhooks 与 Slack 对接，如当前无合适的 Incoming Webhooks，请执行 Sending messages using incoming webhooks，创建 Incoming Webhook，获取其中的 Webhook URL。
     - Microsoft Teams：CLS 通过 Workflows 与 Teams 对接，请执行 Create incoming webhooks with Workflows for Microsoft Teams，以模板方式创建 workflow，获取其中的 URL。模板请选择 Post to a channel when a webhook request is received。
     - Jira Service Management：https://api.atlassian.com/jsm/ops/integration/v2/alerts。
     - Google Chat：CLS 通过 webhook 与 Google Chat 对接，如当前无合适的 webhook，请执行 Register the incoming webhook，创建 Webhook，获取其中的 webhook URL。
   - 请求方法：POST
   - 内容模板：选择 步骤1 新建的通知内容模板。

> **说明：**
```

> 详细的配置说明可参考 管理通知渠道组 。

步骤3：在告警策略中选择通知渠道组

1. 在左侧导航栏中，选择**监控告警** > **告警策略**，进入告警策略管理页面。
2. 单击**新建**，开始配置告警策略，详细的配置说明可参考 配置告警策略 。其中关联通知渠道组时选择 步骤2 新建的通知渠道组。

投递和消费
使用 Flink 消费 CLS 日志

最近更新时间：2025-05-29 19:08:21

操作场景

本文详细描述了如何使用 Flink 实时消费和分析 CLS 中的 Nginx 日志数据，计算 Web 端的 PV/UV 值，并将结果数据实时写入到自建的 MySQL 数据库。
文中使用的组件/应用及版本如下：

技术组件	版本
Nginx	1.22
CLS 日志服务	-
Java OpenJDK	openjdk version 1.8.0_452
Scala	2.11.12
Flink	1.14.5
MySQL	5.7

操作步骤

步骤1：安装腾讯云 Nginx 网关

1. 购买腾讯云主机 CVM，请参见 创建 CVM 实例 。
2. 安装 Nginx 1.22版本。

步骤2：采集 Nginx 日志到腾讯云 CLS 日志服务

1. 将 Nginx 日志采集到 CLS 日志主题 。
2. CLS 日志服务采集终端 Loglistener 的安装 ，Loglistener 类似于开源组件 Beats，用来采集日志数据的 Agent。
3. 日志主题开启索引后，可以正常查询到 Nginx 的日志数据。
4. 最后，在 CLS 控制台 开启 kafka 消费，使用 Kafka 协议消费功能，您可以将一个日志主题，当作一个 Kafka Topic 来消费。本文就是使用流计算框架 Flink，实时消费 Nginx 日志数据，将实时计算的结果写入到 MySQL。

步骤3：搭建 MySQL 数据库

1. 创建 MySQL 实例，登录数据库。

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

2. 新建需要使用的 DB 和表，例子中的 DB 名为 flink_nginx，表名为 mysql_dest。

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

步骤4：部署 Flink

1. 部署 Flink 时，建议使用如下版本，否则可能会安装不成功。
   - 购买腾讯云主机 CVM，资源配置最小8C16G。
   - 安装 JDK v8.0.0。

```
#下载 openjdk
wget https://corretto.aws/downloads/latest/amazon-corretto-8-x64-linux-jdk.rpm
#前往下载目录，运行安装命令
sudo rpm -ivh amazon-corretto-8-x64-linux-jdk.rpm
# 查看版本
java -version
```

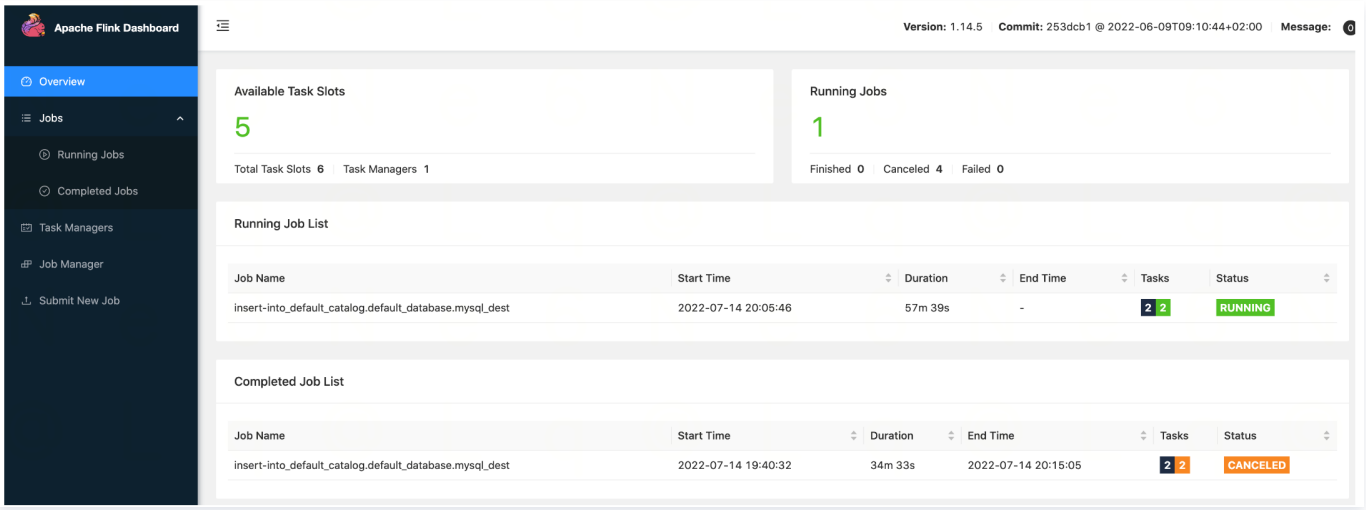
   - 安装 Scala 2.11.12
2. 安装 Flink 1.14.5，并进入 SQL 界面，从 Apache Flink 官网 下载 Flink 二进制代码包并开始安装。

```
# 解压缩 Flink 二进制包
tar -xf flink-1.14.5-bin-scala_2.11.tgz
cd flink-1.14.5

# 下载 kafka 相关依赖
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

# 下载 MySQL 相关依赖
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

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

3. 当出现以下画面则说明安装成功。

```
[[illegible]tos ~/flink-1.14.5]$ bin/start-cluster.sh
Starting cluster.

Starting standalonesession daemon on host VM-14-204-centos.
```

网页端口是8081，可以查看 Flink Dashboard。



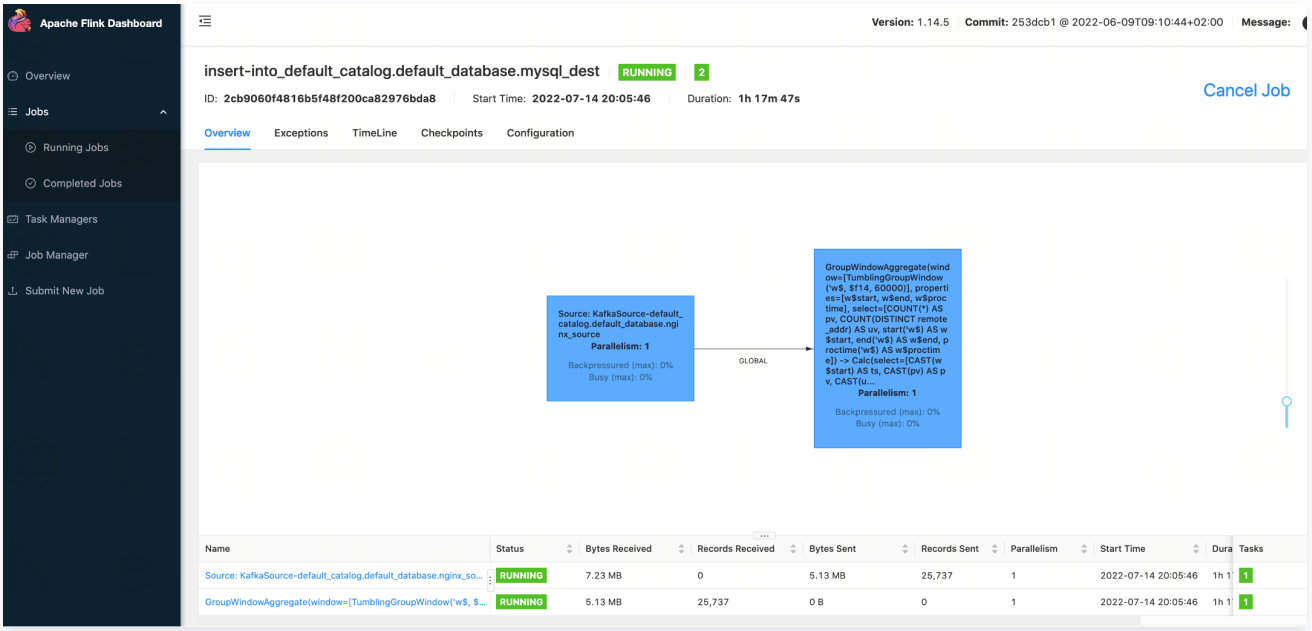
步骤5：使用 Flink 消费 CLS 日志数据

1. 在 SQL Client 界面中，执行如下 SQL：

```
-- 消费CLS日志主题数据 ，建数据源表
CREATE TABLE `nginx_source`
(
    `remote_user` STRING,            -- 日志中字段，客户端名称
    `time_local` STRING,             -- 日志中字段，服务器本地时间
    `body_bytes_sent` BIGINT,        -- 日志中字段，发送给客户端的字节数
    `http_x_forwarded_for` STRING,   -- 日志中字段，当前端有代理服务器时，记录客户端真实 IP 地址的配置
    `remote_addr` STRING,            -- 日志中字段，客户端 IP 地址
    `protocol` STRING,               -- 日志中字段，协议类型
    `status` INT,                    -- 日志中字段，HTTP 请求状态码
    `url` STRING,                    -- 日志中字段，URL 地址
    `http_referer` STRING,           -- 日志中字段，该网页是从哪个页面链接过来的
    `http_user_agent` STRING,        -- 日志中字段，客户端浏览器信息
    `method` STRING,                 -- 日志中字段，HTTP 请求方法
    `partition_id` BIGINT METADATA FROM 'partition' VIRTUAL,    -- CLS日志主题分区
    `ts` AS PROCTIME()
)  WITH (
  'connector' = 'kafka',--将CLS日志主题当作Kafka Topic来消费
  'topic' = 'YourTopic',  -- CLS Kafka协议消费控制台给出的主题名称，例如12345-633a268c-XXXX-4a4c-XXXX-
7a9a1a7baXXXX
  'properties.bootstrap.servers' = 'kafkaconsumer-ap-guangzhou.cls.tencentcs.com:9096',   -- CLS
Kafka协议消费控制台给出的服务地址，例子中是广州地域的外网消费地址，请按照您的实际情况填写
  'properties.group.id' = 'kafka_flink', -- Kafka 消费组名称
  '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";',--用户名是日志主题所属的日志集合ID，例如ca5cXXXX-dd2e-
4ac0-af12-92d4b677d2c6，密码是用户的secretid#secrectkey组合的字符串，比
AKID*******************************#XXXXuXtymIXT0Lac注意不要丢失#。建议使用子账号密钥，主账号为子账号授权时，
遵循最小权限原则，即子账号的访问策略中的action、resource都配置为最小范围，可以满足操作即可.
  'properties.security.protocol' = 'SASL_PLAINTEXT',
  'properties.sasl.mechanism' = 'PLAIN'
);

--- 建立目标表
CREATE TABLE `mysql_dest`
```

```
(
    `ts` TIMESTAMP,
    `pv` BIGINT,
    `uv` BIGINT
)  WITH (
    'connector' = 'jdbc',
    'url' = 'jdbc:mysql://11.150.2.1:3306/flink_nginx', -- 注意这边的时区设置
    'username'= 'username',      -- MYSQL账号
    'password'= 'password',      -- MYSQL密码
    'table-name' = 'mysql_dest' -- MYSQL表名
);

--- 查询 CLS日志主题，对Nginx日志中的PV，UV进行聚合计算，结果写入 MYSQL目标表
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. 在 Flink 的任务监控页，我们可以看到 Flink 任务的监控数据：



3. 进入 MySql 数据库，可看到计算 PV、UV 的结果数据实时写入：

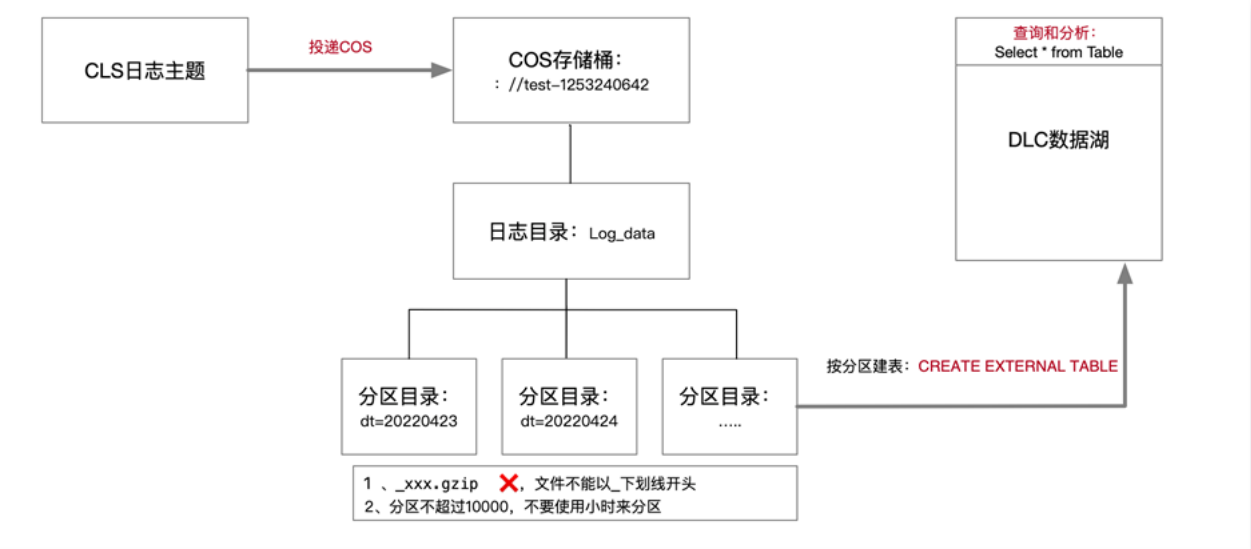
```
MySQL [flink_nginx]> select * from mysql_dest;
+---------------------+------+------+
| ts                  | pv   | uv   |
+---------------------+------+------+
| 2022-07-14 20:16:00 |   60 |    1 |
| 2022-07-14 20:17:00 |   62 |    2 |
| 2022-07-14 20:18:00 |   64 |    3 |
| 2022-07-14 20:19:00 |   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)
```

使用 DLC（Hive）分析 CLS 日志

最近更新时间：2024-10-24 20:59:22

概述

当您需要将日志服务 CLS 中的日志投递到 Hive 进行 OLAP 计算时，可以参见本文进行实践。您可以通过腾讯云数据湖计算 DLC（Data Lake Compute，DLC）提供的数据分析与计算服务，完成对日志的离线计算和分析。示意图如下所示：



操作步骤

CLS 日志投递至 COS

创建投递任务

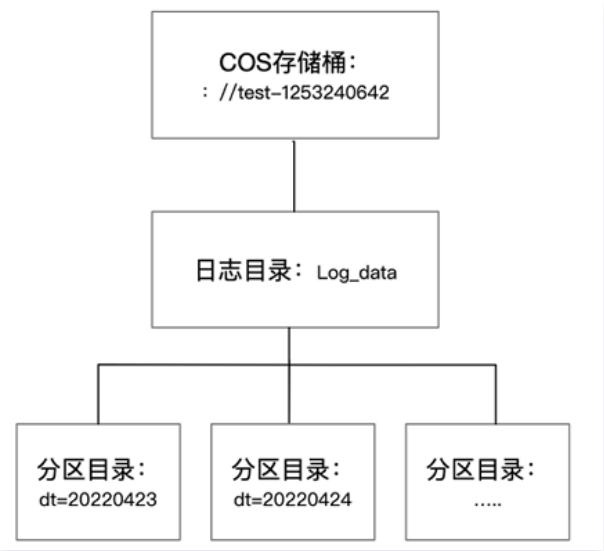
1. 登录 日志服务控制台，选择左侧导航栏中的**投递任务 > 投递至 COS**。
2. 在“投递至 COS” 页面中，单击**添加投递配置**，在弹出的“投递至 COS”窗口中，配置并创建投递任务。
   如下配置项需要注意：

配置项	注意事项
COS 存储桶	日志文件会投递到对象存储桶的该目录下。在数据仓库模型中，一般对应为 Table Location 的地址。
COS 路径	按照 Hive 分区表格式指定。例如，按天分区可以设置为 /dt=%Y%m%d/test，其中 dt= 代表分区字段，%Y%m%d 代表年月日，test 代表日志文件前缀。
文件命名	投递时间命名
投递间隔时间	可在5 – 15分钟范围内选择，建议选择15分钟，250MB，这样文件数量会比较少，查询性能更佳。
投递格式	JSON 格式。

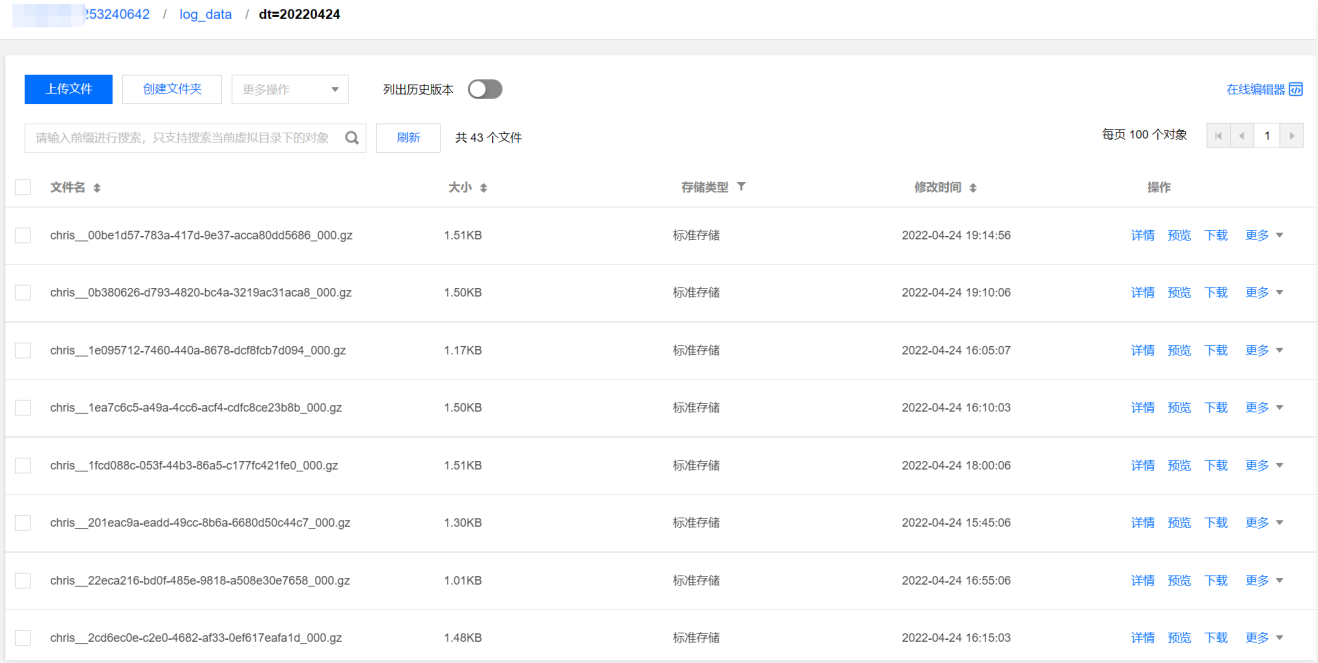
单击下一步，进入高级配置，选择 JSON 和您需要处理的字段。

查看投递任务结果

通常在启动投递任务15分钟后，可以在COS（对象存储）控制台查看到日志数据，目录结构类似下图，分区目录下包含具体的日志文件。



对象存储控制台日志数据如下图所示：



DLC（Hive）分析

DLC 创建外部表并映射到对象存储日志目录

日志数据投递至对象存储后，即可通过 DLC 控制台 → 数据探索功能创建外部表，建表语句可参见如下 SQL 示例，**需特别注意分区字段以及 Location 字段要与目录结构保持一致。**
DLC 创建外表向导提供高级选项，可以帮助您推断数据文件表结构自动快捷生成 SQL，因为是采样推断所以需要您进一步根据 SQL 判断表字段是否合理，例如以下案例，TIMESTAMP 字段推断出为 int，但可能 bigint 才够用。

```
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.JsonSerDe' STORED AS
TEXTFILE LOCATION 'cosn://coreywei-1253240642/log_data/'
```

- 如果是按分区投递，Location 需要指向 `cosn://coreywei-1253240642/log_data/` 目录，而不是 `cosn://coreywei-1253240642/log_data/20220423/` 目录。
- 使用推断功能，需要将目录指向数据文件所在的子目录即：`cosn://coreywei-1253240642/log_data/20220423/` 目录，推断完成后在 SQL 中 Location 修改回 `cosn://coreywei-1253240642/log_data/` 目录即可。
- 适当分区会提升性能，但分区总数建议不超过1万。

添加分区

分区表需要在添加分区数据后，才能通过 select 语句获取数据。您可以通过如下两种方式添加分区：

历史分区添加

该方案可一次性加载所有分区数据，运行较慢，适用首次加载较多分区场景。

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

增量分区添加

在加载完历史分区之后，增量分区还会定期增加。例如，每天新增一个分区，则可以通过该方案进行增量添加。

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

分析数据

添加完分区后，即可通过 DLC 进行数据开发或分析。

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

结果如下图所示：

查询-2022-04-24 16...
运行
保存
刷新
格式化
SQL
1
2 select dt,count(1) from `DataLakeCatalog`.`test`.`log_data` group by dt;
3
查询结果
Task ID 结果条数 1 数据扫描量 31 KB 运行时间 1s SQL语句 select dt,count(1) from `Dat...
dt
EXPR_1
20220424
156

定时 SQL 分析
使用定时 SQL 解决检索分析超时

最近更新时间：2024-06-07 17:23:13

背景

检索分析日志时提示查询超时的处理方法：

1. 如果是由于查询时间范围过长，导致单次查询的数据量过大，推荐您使用定时 SQL 分析，将该查询拆分成多个小范围查询，然后再汇总结果。
2. 如果您的**查询结果都是数值类型**，也可通过定时SQL分析，将 日志转为指标 Metric，再使用 PromQL 对指标计算分析，其查询性能比日志要高，可有效解决日志查询超时的问题。

本文介绍将查询拆分成多个小范围查询，然后再汇总结果的详细操作。

操作步骤

思路：拆分查询时间范围

通过定时 SQL，可将在日志主题 A 一个**长时间**范围查询拆成多个**短时间**范围的查询：例如查询**1天**的数据，可以拆成**12个**2小时的查询，并将查询结果保存至新的日志主题B，您可在日志主题 B 中查看查询结果，或者在日志主题 B 中继续做检索分析，该方案已经在生产中得到应用，可有效解决检索分析超时问题。

案例：

1. 假定用户在**源日志主题**：**cls_service_log**，查询数据范围**2023-12-13 12:00:00 – 2023-12-13 12:15:00**，查询超时，语句如下：

```
select sum(etl_input_line) as input_lines ,sum(etl_failure_line) as failed_lines ,
etl_task_id group by etl_task_id
```

2. 使用定时 SQL 分析**预计算**源日志数据，参考 新建定时 SQL 任务。

   定时 SQL 分析的配置如下：

- **源日志主题**：配置为需要拆分查询的日志主题，示例中为 **cls_service_log**。
- **写入目标**：选择日志主题，示例中为 **sql_result**。
- **调度范围**：2023-12-13 12:05:00 – 2023-12-13 12:15:00（实际查询的是12:00:00 -12:15:00的数据，您可在**调度详情**中查看 SQL 时间窗口，SQL 时间窗口是查询的范围）
- **调度周期**：5分种
- **SQL 时间窗口**：@m-5m,@m
- **SQL 语句（CQL）**：

```
* | select sum(etl_input_line) as input_lines ,sum(etl_failure_line) as failed_lines ,
etl_task_id group by etl_task_id
```

> **说明：**
> 示例中的定时 SQL 分析使用的语句和检索分析的一样，但在某些业务场景下，例如有 **Distinct**、**Avg** 、**TOP N** 计算时，需要您根据业务诉求**调整 SQL** 语句。

可以在定时 SQL 分析的**调度详情**中看到，将原来的1个15分钟范围的查询，拆分成了**3个5分钟**范围的查询，如下图所示。三个查询的 **SQL 时间窗口**加起来刚好是原来的时间范围12:00-12:15。

调度详情

近1天 实例总数3，失败0，成功3，运行中0

实例ID	执行时间 ↓	SQL时间窗口 ⇕	处理的数据量
ba98d817-a6aa-4f57-8d34-edc39af23425	开始时间：2023-12-13 12:15:00 结束时间：2023-12-13 19:16:12 耗时：48ms	开始时间：2023-12-13 12:10:00 结束时间：2023-12-13 12:15:00	输入行数：1052 输出行数：7
f256c8df-7c25-4f9a-beb6-5d6be0b00a32	开始时间：2023-12-13 12:10:00 结束时间：2023-12-13 19:16:07 耗时：45ms	开始时间：2023-12-13 12:05:00 结束时间：2023-12-13 12:10:00	输入行数：1046 输出行数：7
465707f7-287e-4aac-aedb-4bb29ae0c186	开始时间：2023-12-13 12:05:00 结束时间：2023-12-13 19:16:02 耗时：81ms	开始时间：2023-12-13 12:00:00 结束时间：2023-12-13 12:05:00	输入行数：1028 输出行数：7

3. 打开目标主题 sql_result 查看数据：

可以看到定时 SQL 分析处理的数据如下。由于 input_lines 和 failed_lines 在定时 SQL 中，分三次（每次计算5分钟的 sum 值）后写入。因此我们需要将结果数据再做一次 sum，就可以得到原始业务场景中15分钟的 input_lines 和 failed_lines 的 sum 值。

1	12-13 12:10:00.000	input_lines: 8455144 etl_task_id: 1aa7a610-ae08 .bfd-b5b8-d22259597318 failed_lines: 0 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
2	12-13 12:10:00.000	input_lines: 195 etl_task_id: 415d3de7-3f -472a-bef7-e73f6fbd94ab failed_lines: 0 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
3	12-13 12:10:00.000	input_lines: 9 etl_task_id: 65713daa-c34c-49 9771-43830d1ddb6d failed_lines: 9 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
4	12-13 12:10:00.000	input_lines: 4058 etl_task_id: 7558d625-47f7-4f52-8360-4bd393778bd6 failed_lines: 0 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
5	12-13 12:10:00.000	input_lines: 159 etl_task_id: 9777c703-14 -4bd4-bf75-08750c349c16 failed_lines: 0 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
6	12-13 12:10:00.000	input_lines: 2033507 etl_task_id: dcd199a1-65 -4225-bca0-dc5fe7b2bd20 failed_lines: 0 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
7	12-13 12:10:00.000	input_lines: 2035059 etl_task_id: ee371a35- 87ef-25baa78d19a1 failed_lines: 0 __TAG__.process_id: ba98d817-a6aa-4f57-8d34-edc39af23425 __TAG__.trigger_time: 1702466172 __TAG__.schedule_time: 1702440900 __TAG__.job: abcd123
8	12-13 12:05:00.000	input_lines: 8766785 etl_task_id: 1aa7a610-ae0 .ofd-b5b8-d22259597318 failed_lines: 0 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
9	12-13 12:05:00.000	input_lines: 206 etl_task_id: 415d3de7-3f -472a-bef7-e73f6fbd94ab failed_lines: 0 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
10	12-13 12:05:00.000	input_lines: 11 etl_task_id: 65713daa-c34c- 9-9771-43830d1ddb6d failed_lines: 11 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
11	12-13 12:05:00.000	input_lines: 4221 etl_task_id: 7558 -47f7-4f52-8360-4bd393778bd6 failed_lines: 0 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
12	12-13 12:05:00.000	input_lines: 210 etl_task_id: 9777c703- -4bd4-bf75-08750c349c16 failed_lines: 0 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
13	12-13 12:05:00.000	input_lines: 2021065 etl_task_id: dcd199a1-65ef -4225-bca0-dc5fe7b2bd20 failed_lines: 0 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
14	12-13 12:05:00.000	input_lines: 2262777 etl_task_id: ee371a35-fcbd- .dc-87ef-25baa78d19a1 failed_lines: 0 __TAG__.process_id: f256c8df-7c25-4f9a-beb6-5d6be0b00a32 __TAG__.trigger_time: 1702466167 __TAG__.schedule_time: 1702440600 __TAG__.job: abcd123
15	12-13 12:00:00.000	input_lines: 9864571 etl_task_id: 1aa7a610- b8-d22259597318 failed_lines: 0 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123
16	12-13 12:00:00.000	input_lines: 207 etl_task_id: 415d3de7-3f83-47 -bef7-e73f6fbd94ab failed_lines: 0 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123
17	12-13 12:00:00.000	input_lines: 10 etl_task_id: 65713daa-c34c- 9771-43830d1ddb6d failed_lines: 10 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123
18	12-13 12:00:00.000	input_lines: 4308 etl_task_id: 7558d625-47f 4f52-8360-4bd393778bd6 failed_lines: 0 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123
19	12-13 12:00:00.000	input_lines: 207 etl_task_id: 9777c703-147c-4bd4-b 5-08750c349c16 failed_lines: 0 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123
20	12-13 12:00:00.000	input_lines: 2019292 etl_task_id: dcd199a1-65ef 5-bca0-dc5fe7b2bd20 failed_lines: 0 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123
21	12-13 12:00:00.000	input_lines: 2230147 etl_task_id: ee371a35-fcb af-25baa78d19a1 failed_lines: 0 __TAG__.process_id: 465707f7-287e-4aac-aedb-4bb29ae0c186 __TAG__.trigger_time: 1702466162 __TAG__.schedule_time: 1702440300 __TAG__.job: abcd123

4. 在目标日志主题中检索

在目标日志主题 sql_result 中执行 SQL 语句：

```
select sum(input_lines) as input_lines, sum(failed_lines) as failed_lines, etl_task_id group by etl_task_id
```

得到如下图的结果，和在源日志主题：cls_service_log 直接检索分析的结果一致（没有超时的情况下）。

# input_lines	# failed_lines	t etl_task_id
27086500	0	1aa7a610-[illegible]-d22259597318
608	0	415d3de7-[illegible]-472a-bef7-e73f6fbd94ab
30	30	65713daa-c[illegible]-49d8-977[illegible]-43830d1ddb6d
12587	0	7558d625-[illegible]393778bd6
576	0	9777c703-[illegible]-4bd4-[illegible]-08750c349c16
6073864	0	dcd199a1-65e[illegible]-dc5fe7b2bd20
6527983	0	ee371a35-fc[illegible]-4bdc-[illegible]baa78d19a1

结论

当您的查询超时，您可通过**定时 SQL**，对源日志数据预计算（拆分查询范围）之后，然后在**目标日志主题**中进行检索，或者配置仪表盘、告警，可有效提升对日志数据的计算分析能力。

从日志中提取指标(Metric)

最近更新时间：2024-12-05 11:00:22

背景

用户使用 CLS 仪表盘时，偶尔会遇到检索超时，导致图表绘制失败的问题，这个时候，可以考虑将日志转为指标（Metric）数据，然后使用 PromQL 对指标进行计算分析，其分析性能比日志高，可以有效的解决仪表盘超时的问题。

概述

使用定时 SQL，可以从**日志中提取指标**，并将指标保存在指标主题中。相比日志，指标**查询快，存储空间小**，可对接 **Prometheus** 生态。指标的更多信息参见 指标存储概述 。
下述内容以示例的形式来说明如何使用定时 SQL 转 Metric。其业务场景是：从 Nginx 日志中提取3个指标（**PV**、**失败请求数**、**P99_请求时间**）。指标维度是 URL，按照 PV 的大小取 TOP10。
我们将使用定时 SQL，完成**日志转指标**。

操作步骤

1. 登录 CLS 控制台，选择重庆地域，打开 Nginx Demo 日志主题（免费的体验主题）检索分析页面，Demo 日志主题请参见 使用 Demo 日志快速体验 CLS 。
   单条的原始日志如下：

```
{
"remote_addr":"124.78.124.112"
"method":"GET"
"upstream_addr":"169.254.128.14:60002"
"upstream_response_length":"48"
"body_bytes_sent":"59"
"time_local":"2021-12-03T17:16:58+00:00"
"version":"HTTP/1.1"
"url":"/"
"http_user_agent":"-"
"remote_user":"-"
"req_id":"e3d1ae06b30344539bc7e28314d542fa"
"upstream_status":"400"
"request_time":"0.002"
"sys_address":"11.149.155.219"
"request_length":"40"
"http_referer":"-"
"sys_datasource":"gz.1.1.v1.2.19"
"proxy_upstream_name":"default-kubernetes-443"
"upstream_response_time":"0.000"
"time":"1702542388004"
"timestamp":"2023-12-14T16:26:28+08:00"
"status":"400"
}
```

2. 根据上述业务场景，在控制台**配置统计分析**中，输入SQL语句进行验证，语句如下：

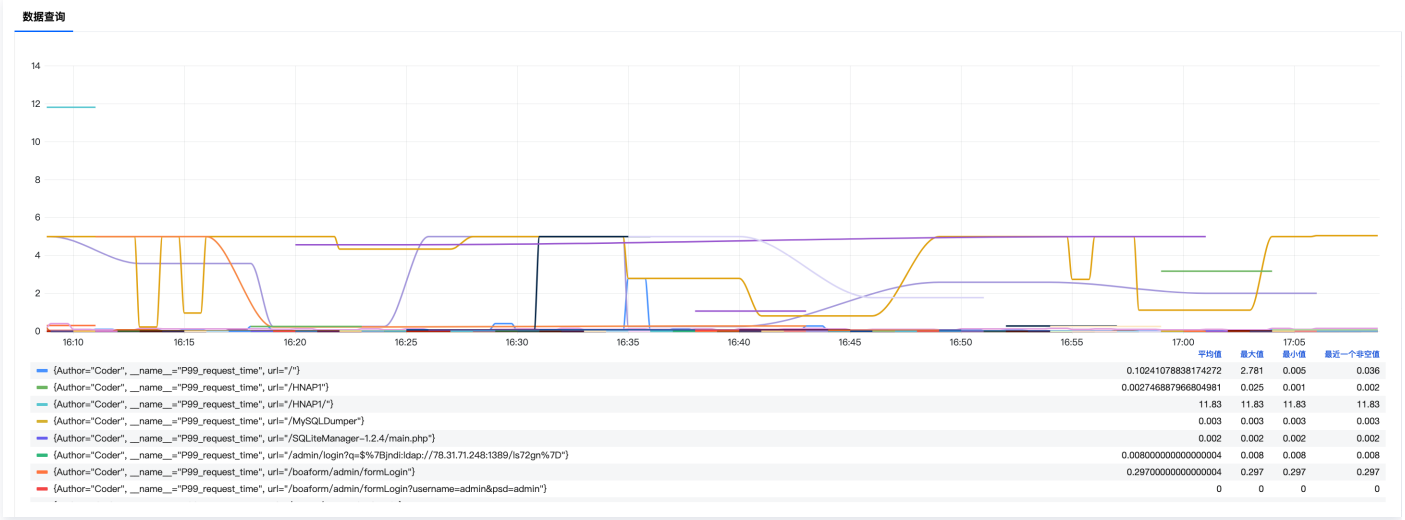
```
select count(*)as pv,  count_if(status>=400)as bad_request_count, approx_percentile(request_time,0.99)
as P99_request_time,url group by url order by pv desc limit 10
```

结果如下图，**确认了检索语句的准确性**：

# pv	# bad_request_count	# P99_request_time	t url
397	197	0.009	/nice%20ports%2C/Tri%6Eity.txt%2ebak
100	100	0.002	http://azenv.net/
98	98	0.232	http://81.71.160.63/cc.php
91	91	5.054	dnspod.qcloud.com:443
40	40	0.048	http://110.242.68.4/
38	38	5.003	/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
26	26	0.003	/?XDEBUG_SESSION_START=phpstorm
25	25	0.009	/config/getuser?index=0
23	23	5.001	/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh
22	22	0.003	/favicon.ico

3. 在**统计图表** TAB 页面的右上方，点击**存为定时SQL分析**。完成以下配置项，点击**确定**。

任务名称	Log2Metric
写入目标	选择指标主题，提前新建好，示例中的名称为 NginxMetric
调度周期	1分钟，意为**每1分钟**发起一次查询，并将结果保存到指标主题
SQL 时间窗口	@m-1m,@m ,意为查询的时间范围近1分钟
SQL 语句	* | select count(*)as pv，count_if(status>=400)as bad_request_count, approx_percentile(request_time,0.99) as P99_request_time,url group by url order by pv desc limit 10
指标名称	pv,bad_request_count,P99_request_time，一般为数值
指标维度	url，一般都是 group by 后面的字段，例子中是按照 URL 去分组统计的意思。
自定义维度	{"Author","Coder"}。添加您自己的维度，实际生产中可能是您所在的开发组、应用、或者集群、命名空间之类的。

4. 查看定时 SQL 分析

   可以查看每分钟会调度一次我们配置的 SQL 语句，并将查询结果输出至指标主题。
5. 在定时 SQL 分析**任务详情**中找到**目标主题** NginxMetric，点击跳转至指标主题页面，点击**浏览指标**，从指标列表中选择一个指标进行查看。
6. 使用 指标检索 对指标数据进行计算分析，并将分析结果**添加至仪表盘**。