端到端追蹤 ID 實現

此博客文章是在 ChatGPT-4o 的協助下撰寫的。

我開發了一個端到端的追蹤 ID 解決方案，以確保我們系統中的每個請求和回應都能在前端和後端之間一致地追蹤。這個解決方案通過將每個操作與唯一的追蹤 ID 關聯起來，幫助進行調試、監控和日誌記錄。以下是該解決方案的詳細解釋，以及代碼示例。

工作原理

前端

這個解決方案的前端部分涉及為每個請求生成一個追蹤 ID，並將其與客戶端信息一起發送到後端。這個追蹤 ID 用於在後端的各個處理階段追蹤請求。

1. **客戶端信息收集**：我們從客戶端收集相關信息，例如屏幕尺寸、網絡類型、時區等。這些信息會與請求頭一起發送。
2. **追蹤 ID 生成**：為每個請求生成一個唯一的追蹤 ID。這個追蹤 ID 會包含在請求頭中，使我們能夠追蹤請求的生命週期。
3. **API 請求**：使用 `apiFetch` 函數進行 API 調用。它會在每個請求的頭部包含追蹤 ID 和客戶端信息。

後端

解決方案的後端部分涉及在每個日誌消息中記錄追蹤 ID，並在回應中包含追蹤 ID。這使我們能夠追蹤請求在後端的處理過程，並將回應與請求匹配。

1. **追蹤 ID 處理**：後端從請求頭中獲取追蹤 ID，如果未提供則生成一個新的追蹤 ID。追蹤 ID 存儲在 Flask 的全局對象中，以便在請求的生命週期內使用。
2. **日誌記錄**：使用自定義日誌格式化器將追蹤 ID 包含在每個日誌消息中。這確保了與請求相關的所有日誌消息都可以通過追蹤 ID 進行關聯。
3. **回應處理**：追蹤 ID 包含在回應頭中。如果發生錯誤，追蹤 ID 也會包含在錯誤回應體中，以幫助調試。

Kibana

Kibana 是一個強大的工具，用於可視化和搜索存儲在 Elasticsearch 中的日誌數據。通過我們的追蹤 ID 解決方案，您可以輕鬆使用 Kibana 追蹤和調試請求。追蹤 ID 包含在每個日誌條目中，可用於過濾和搜索特定日誌。

要搜索具有特定追蹤 ID 的日誌，您可以使用 Kibana 查詢語言（KQL）。例如，您可以使用以下查詢搜索與特定追蹤 ID 相關的所有日誌：

```
trace_id:"Lc6t"
```

此查詢將返回包含追蹤 ID“Lc6t” 的所有日誌條目，使您能夠追蹤請求在系統中的路徑。此外，您可以將此查詢與其他條件結合，以縮小搜索結果範圍，例如按日誌級別、時間戳或日誌消息中的特定關鍵字進行過濾。

通過利用 Kibana 的可視化功能，您還可以創建基於追蹤 ID 的儀表板，顯示指標和趨勢。例如，您可以可視化處理的請求數量、平均回應時間和錯誤率，這些都與其各自的追蹤 ID 相關聯。這有助於識別應用程序性能和可靠性中的模式和潛在問題。

將 Kibana 與我們的追蹤 ID 解決方案結合使用，提供了一種全面的方法來監控、調試和分析系統行為，確保每個請求都能有效地被追蹤和調查。

前端

api.js

```
const BASE_URL = process.env.REACT_APP_BASE_URL;

// 獲取客戶端信息的函數
const getClientInfo = () => {
    const { language, platform, cookieEnabled, doNotTrack, onLine } = navigator;
    const { width, height } = window.screen;
    const connection = navigator.connection || navigator.mozConnection || navigator.webkitConnection;
    const networkType = connection ? connection.effectiveType : 'unknown';
    const timeZone = Intl.DateTimeFormat().resolvedOptions().timeZone;
    const referrer = document.referrer;
    const viewportWidth = window.innerWidth;
    const viewportHeight = window.innerHeight;
```

```
    return {
        screenWidth: width,
        screenHeight: height,
        networkType,
        timeZone,
        language,
        platform,
        cookieEnabled,
        doNotTrack,
        onLine,
        referrer,
        viewportWidth,
        viewportHeight
    };
};

// 生成唯一追蹤 ID 的函數
export const generateTraceId = (length = 4) => {
    const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    let traceId = '';
    for (let i = 0; i < length; i++) {
        const randomIndex = Math.floor(Math.random() * characters.length);
        traceId += characters.charAt(randomIndex);
    }
    return traceId;
};

export const apiFetch = async (endpoint, options = {}) => {
    const url = `${BASE_URL}${endpoint}`;
    const clientInfo = getClientInfo();

    const traceId = options.traceId || generateTraceId();

    const headers = {
        'Content-Type': 'application/json',
        'X-Client-Info': JSON.stringify(clientInfo),
```

```
        'X-Trace-Id': traceId,
        ...(options.headers || {})
    };

    const response = await fetch(url, {
        ...options,
        headers
    });

    return response;
};
```

App.js

```
try {
  const response = await apiFetch('api', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(content),
    traceId: traceId
  });

  if (response.ok) {
    const data = await response.json();
    //...
  } else {
    const errorData = await response.json();
    const errorMessage = errorData.message || 'An unknown error occurred';
    let errorToastMessage = errorMessage;
    errorToastMessage += ` (Trace ID: ${traceId})`;
    toast.error(errorToastMessage, {
      autoClose: 8000
    });
    setError(errorToastMessage);
```

```
  }
} catch (error) {
  let errorString = error instanceof Error ? error.message : JSON.stringify(error);

  const duration = (Date.now() - startTime) / 1000;

  if (error.response) {
    // 請求已發出，但服務器回應的狀態碼不在 2xx 範圍內
    errorString += ` (HTTP ${error.response.status}: ${error.response.statusText})`;
    console.error('Response error data:', error.response.data);
  } else if (error.request) {
    // 請求已發出，但未收到回應
    errorString += ' (No response received)';
    console.error('Request error data:', error.request);
  } else {
    // 在設置請求時發生了錯誤，觸發了 Error
    errorString += ` (Error setting up request: ${error.message})`;
  }

  errorString += ` (Trace ID: ${traceId})`;

  if (error instanceof Error) {
    errorString += `\nStack: ${error.stack}`;
  }

  errorString += JSON.stringify(error);

  errorString += ` (Duration: ${duration} seconds)`;

  toast.error(`Error: ${errorString}`, {
    autoClose: 8000
  });
  setError(errorString);
} finally {
  toast.dismiss(toastId);
}
```

後端

__init__.py

```
# -*- encoding: utf-8 -*-

import os
import json
import time
import uuid
import string
import random

from flask import Flask, request, Response, g, has_request_context
from flask_cors import CORS

from .routes import initialize_routes
from .models import db, insert_default_config
import logging
from logging.handlers import RotatingFileHandler
from prometheus_client import Counter, generate_latest, Gauge
from flask_migrate import Migrate
from logstash_formatter import LogstashFormatterV1

app = Flask(__name__)

app.config.from_object('api.config.BaseConfig')

db.init_app(app)
initialize_routes(app)

CORS(app)

migrate = Migrate(app, db)


class RequestFormatter(logging.Formatter):
```

```
    def format(self, record):
        if has_request_context():
            record.trace_id = getattr(g, 'trace_id', 'unknown')
        else:
            record.trace_id = 'unknown'
        return super().format(record)


class CustomLogstashFormatter(LogstashFormatterV1):
    def format(self, record):
        if has_request_context():
            record.trace_id = getattr(g, 'trace_id', 'unknown')
        else:
            record.trace_id = 'unknown'
        return super().format(record)


def setup_loggers():
    logstash_handler = RotatingFileHandler(
        'app.log', maxBytes=100000000, backupCount=1)
    logstash_handler.setLevel(logging.DEBUG)
    logstash_formatter = CustomLogstashFormatter()
    logstash_handler.setFormatter(logstash_formatter)

    txt_handler = RotatingFileHandler(
        'plain.log', maxBytes=100000000, backupCount=1)
    txt_handler.setLevel(logging.DEBUG)
    txt_formatter = RequestFormatter(
        '%(asctime)s %(levelname)s: %(message)s [in %(pathname)s:%(lineno)d] [trace_id: %(trace_id)s]')
    txt_handler.setFormatter(txt_formatter)

    root_logger = logging.getLogger()
    root_logger.setLevel(logging.DEBUG)
    root_logger.addHandler(logstash_handler)
    root_logger.addHandler(txt_handler)
```

```
    app.logger.addHandler(logstash_handler)
    app.logger.addHandler(txt_handler)

    werkzeug_logger = logging.getLogger('werkzeug')
    werkzeug_logger.setLevel(logging.DEBUG)
    werkzeug_logger.addHandler(logstash_handler)
    werkzeug_logger.addHandler(txt_handler)


setup_loggers()


def generate_trace_id(length=4):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))


@app.before_request
def before_request():
    request.start_time = time.time()
    trace_id = request.headers.get('X-Trace-Id', generate_trace_id())
    g.trace_id = trace_id

    client_info = request.headers.get('X-Client-Info')
    if client_info:
        try:
            client_info_json = json.loads(client_info)
            logging.info(f"Client Info: {client_info_json}")
        except json.JSONDecodeError:
            logging.warning("Invalid JSON format for X-Client-Info header")


@app.after_request
def after_request(response):
    response.headers['X-Trace-Id'] = g.trace_id
```

```
    if response.status_code != 200:
        logging.error(f'Response status code: {response.status_code}')
        logging.error(f'Response body: {response.get_data(as_text=True)}')

        if response.content_type == 'application/json':
            try:
                response_json = response.get_json()
                response_json['trace_id'] = g.trace_id
                response.set_data(json.dumps(response_json))
            except Exception as e:
                logging.error(f"Error adding trace_id to response: {e}")

    return response
```

日誌

您可以使用以下查詢搜索與特定追蹤 ID 相關的所有日誌：

```
trace_id:"Lc6t"
```


```
{
  "_index": "flask-logs-2024.07.05",
  "_type": "_doc",
  "_id": "Ae9zgZABqOMSOpxCZC5X",
  "_version": 1,
  "_score": 1,
  "_source": {
    "tags": [
      "_grokparsefailure"
    ],
    "filename": "generate.py",
    "funcName": "post",
    "message": "Request processed successfully",
    "@version": 1,
    "name": "root",
    "host": "ip-172-31-35-xxx.ec2.internal",
```

```
    "relativeCreated": 685817.8744316101,
    "levelname": "INFO",
    "created": 1720158740.894831,
    "thread": 139715118360128,
    "threadName": "Thread-5",
    "levelno": 20,
    "pathname": "/home/project/project-name/api/routes/generate.py",
    "msecs": 894.8309421539307,
    "processName": "MainProcess",
    "lineno": 287,
    "path": "/home/project/project-name/app.log",
    "args": [],
    "source_host": "ip-172-31-35-xxx.ec2.internal",
    "module": "generate",
    "trace_id": "Lc6t",
    "stack_info": null,
    "process": 107613,
    "@timestamp": "2024-07-05T05:52:20.894Z"
  },
  "fields": {
    "levelname.keyword": [
      "INFO"
    ],
    "tags.keyword": [
      "_grokparsefailure"
    ],
    "relativeCreated": [
      685817.9
    ],
    "processName.keyword": [
      "MainProcess"
    ],
    "filename.keyword": [
      "generate.py"
    ],
    "funcName": [
```

```
    "post"
  ],
  "path": [
    "/home/project/project-name/app.log"
  ],
  "processName": [
    "MainProcess"
  ],
  "@version": [
    1
  ],
  "host": [
    "ip-172-31-35-xxx.ec2.internal"
  ],
  "msecs": [
    894.83093
  ],
  "source_host.keyword": [
    "ip-172-31-35-xxx.ec2.internal"
  ],
  "host.keyword": [
    "ip-172-31-35-xxx.ec2.internal"
  ],
  "levelname": [
    "INFO"
  ],
  "process": [
    107613
  ],
  "threadName.keyword": [
    "Thread-5"
  ],
  "trace_id": [
    "Lc6t"
  ],
  "source_host": [
```

```
    "ip-172-31-35-xxx.ec2.internal"
  ],
  "created": [
    1720158700
  ],
  "module": [
    "generate"
  ],
  "module.keyword": [
    "generate"
  ],
  "name.keyword": [
    "root"
  ],
  "thread": [
    139715118360128
  ],
  "message": [
    "Request processed successfully"
  ],
  "levelno": [
    20
  ],
  "trace_id.keyword": [
    "Lc6t"
  ],
  "threadName": [
    "Thread-5"
  ],
  "pathname": [
    "/home/project/project-name/api/routes/generate.py"
  ],
  "tags": [
    "_grokparsefailure"
  ],
  "pathname.keyword": [
```

```
      "/home/project/project-name/api/routes/generate.py"
    ],
    "@timestamp": [
      "2024-07-05T05:52:20.894Z"
    ],
    "filename": [
      "generate.py"
    ],
    "lineno": [
      287
    ],
    "message.keyword": [
      "Request processed successfully"
    ],
    "name": [
      "root"
    ],
    "funcName.keyword": [
      "post"
    ],
    "path.keyword": [
      "/home/project/project-name/app.log"
    ]
  }
}
```


如上所示，您可以在日誌中看到追蹤 ID。