

全方位強化 Python 服務可觀測性： 以 FastAPI 和 Grafana Stack 為例

PyCon Taiwan 2024
劉義瑋 Blueswen

自我介绍



- 劉義瑋 Blueswen
- [blueswen](#) @ GitHub
- Berry AI DevOps Engineer
- 技術交流與分享
- 領域
 - DevOps
 - Observability
 - Developer Experience

Outline

- Observability Intro
- Observability Signals Data Flow
 - Metrics
 - Logs
 - Traces
- 資訊交互應用
- Recap



圖片來源：[Grafana](https://grafana.com/)



你也是被冠上天才之名的通靈奇才嗎？

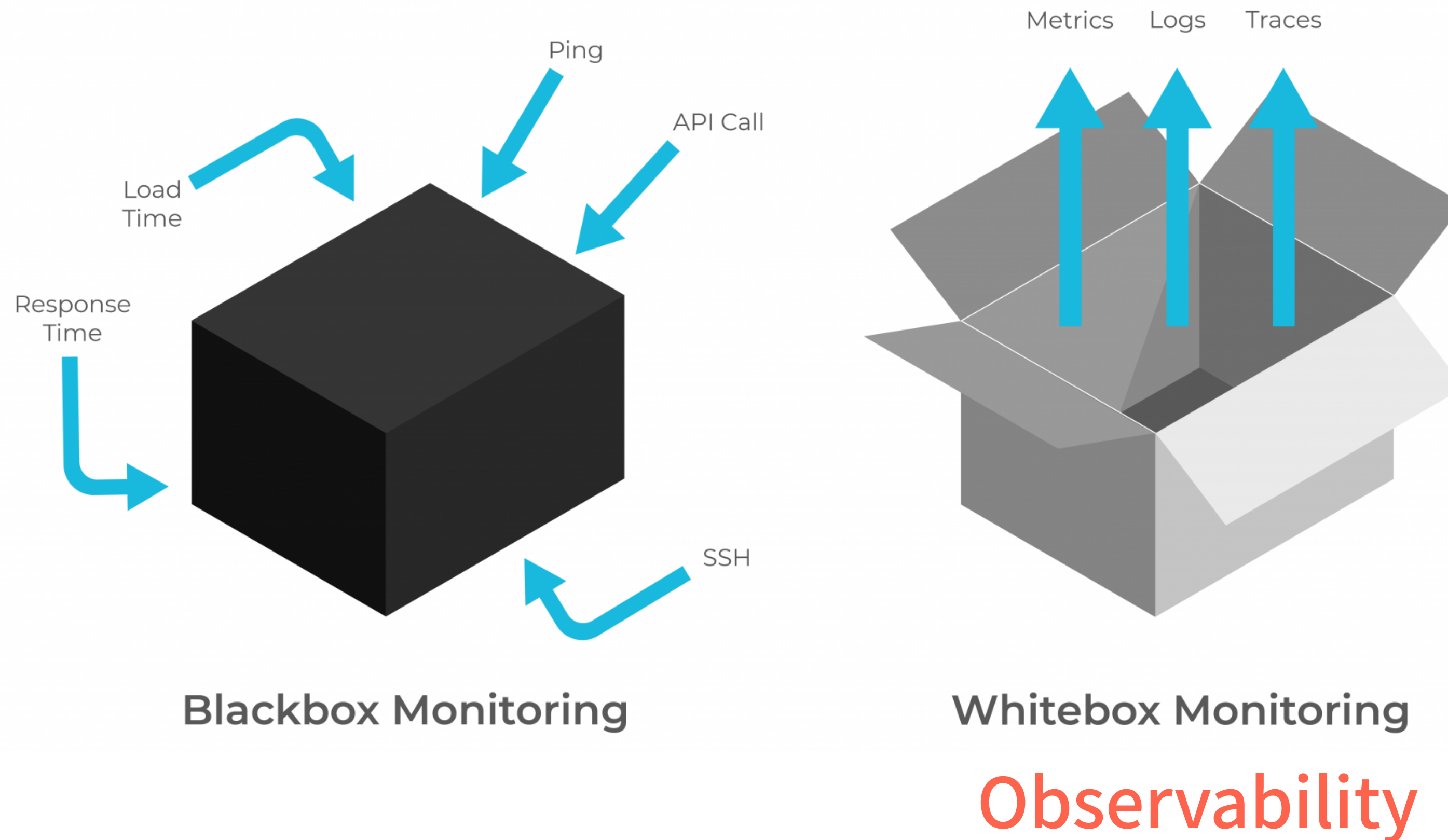
有聽過 Observability 可觀測性嗎？

已經將可觀測性應用在 Production 了嗎？

Observability Intro

Observability Intro

系統可被觀測或測量的一種特性



圖片來源：[Multi-Cloud Monitoring: A Cloud Security Essential](#)

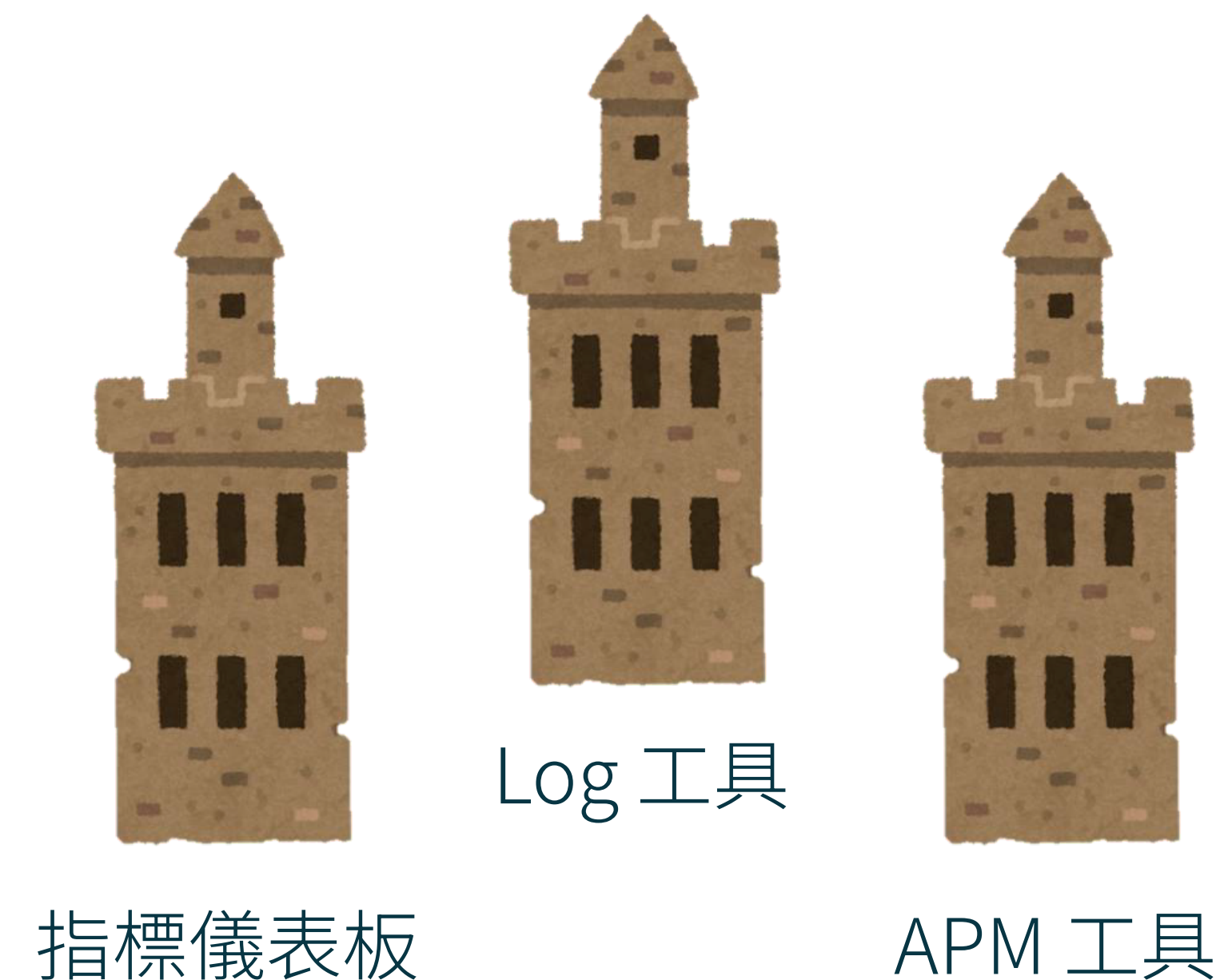
Observability Intro

透過各種資訊，清楚了解系統狀態

有足夠的資訊嗎？
他們都有被好好保存嗎？



有好好利用這些資訊嗎？
是否散落個處，形成 Data Silo？



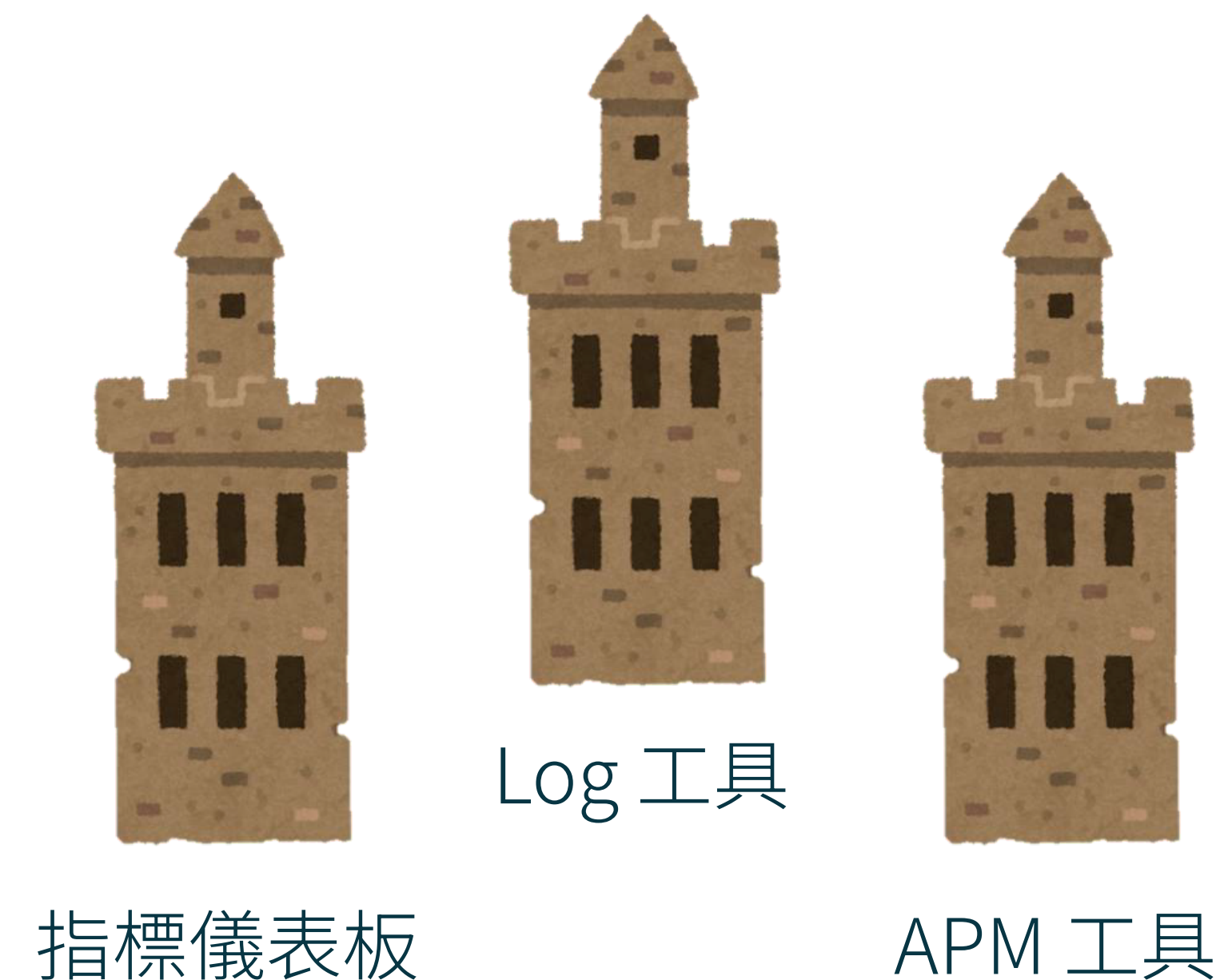
Observability Intro

透過各種資訊，清楚了解系統狀態

有足夠的資訊嗎？
他們都有被好好保存嗎？



有好好利用這些資訊嗎？
是否散落個處，形成 Data Silo？



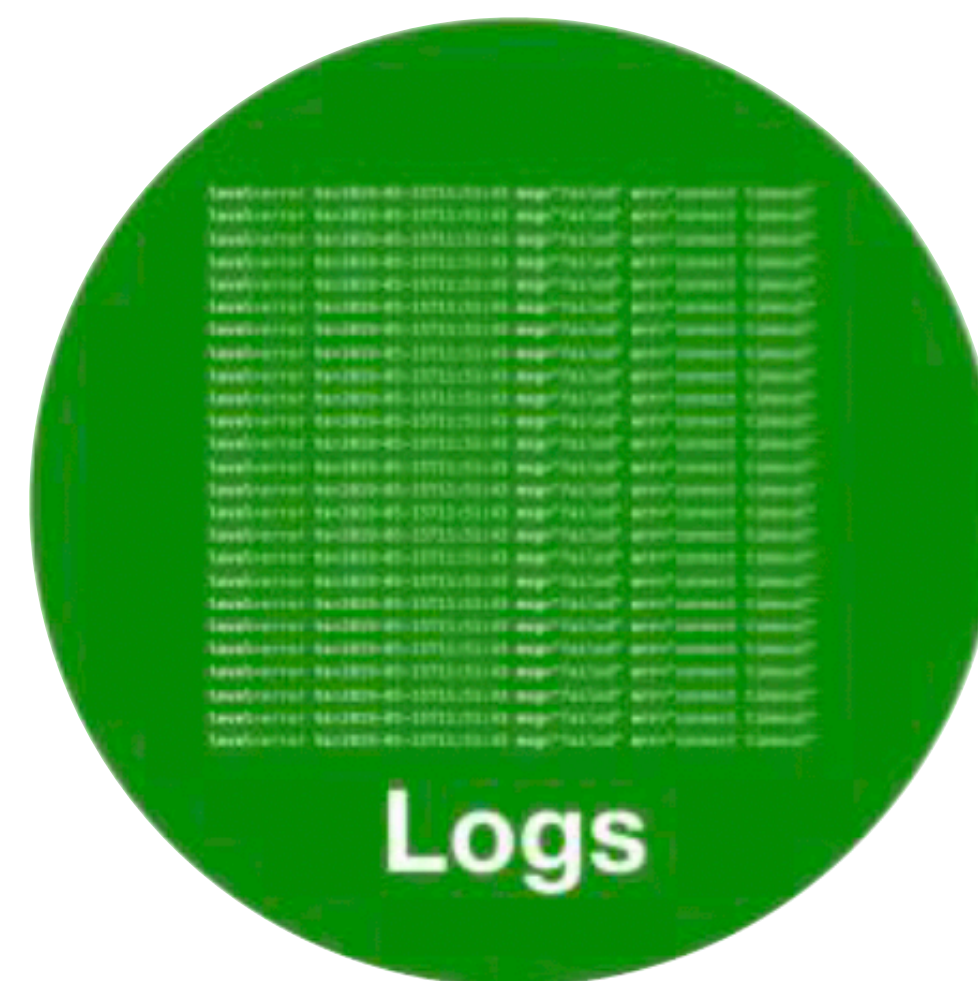
Observability Intro - Observability Signals

系統**主動揭露**讓我們能夠**更理解**它的資訊



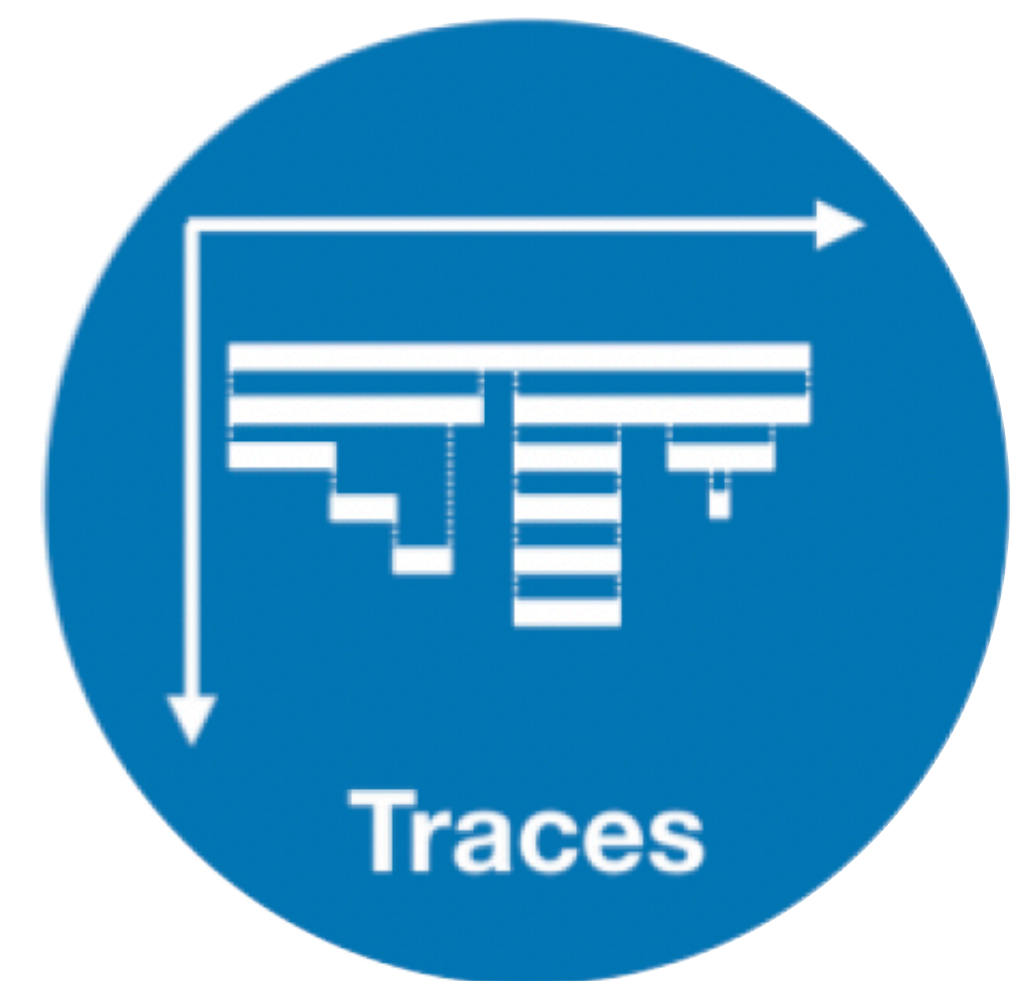
指標

不同時間採樣的系統量化指標
如：CPU 使用率、API 回應時間



日誌

紀錄系統中發生的事情
如：Debug 訊息、Exception



分散式追蹤

紀錄行為在不同服務中的歷程
如：SSO 行為橫跨多個服務

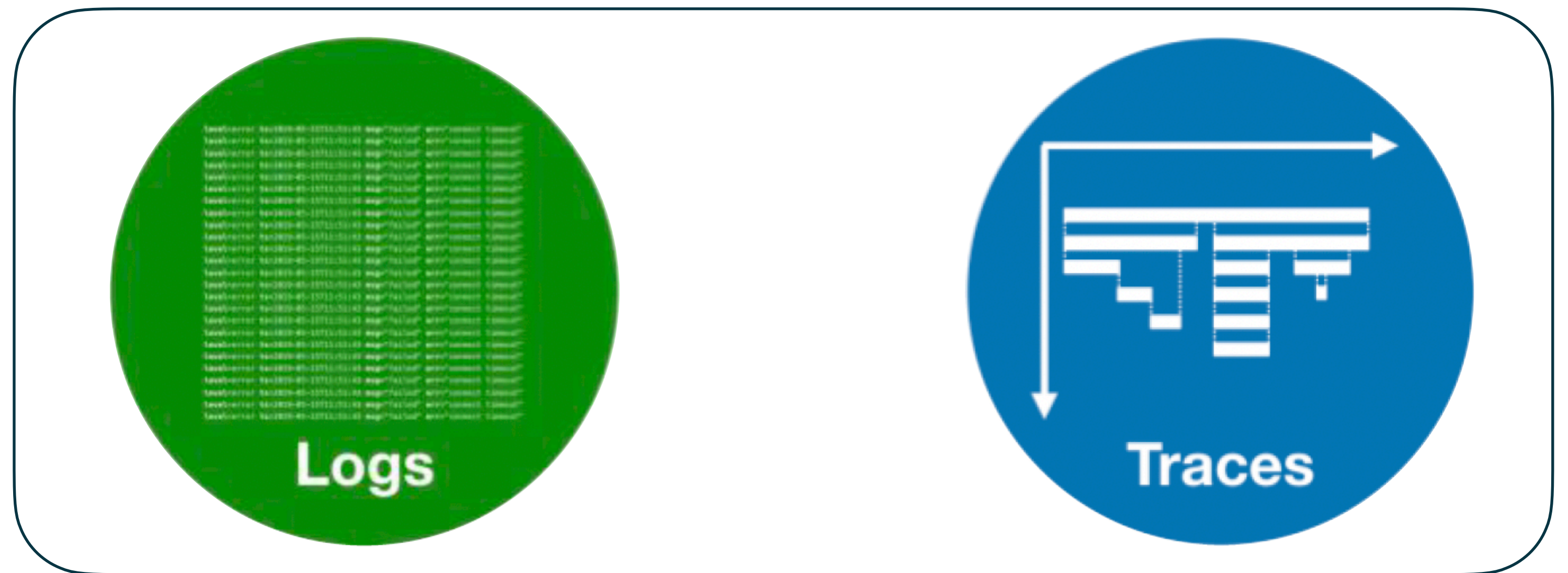
Observability Intro - Observability Signals

系統**主動揭露**讓我們能夠**更理解**它的資訊



狀況如何

徵狀：有問題發生



發生什麼事

脈絡：怎麼發生的



哪裡發生的

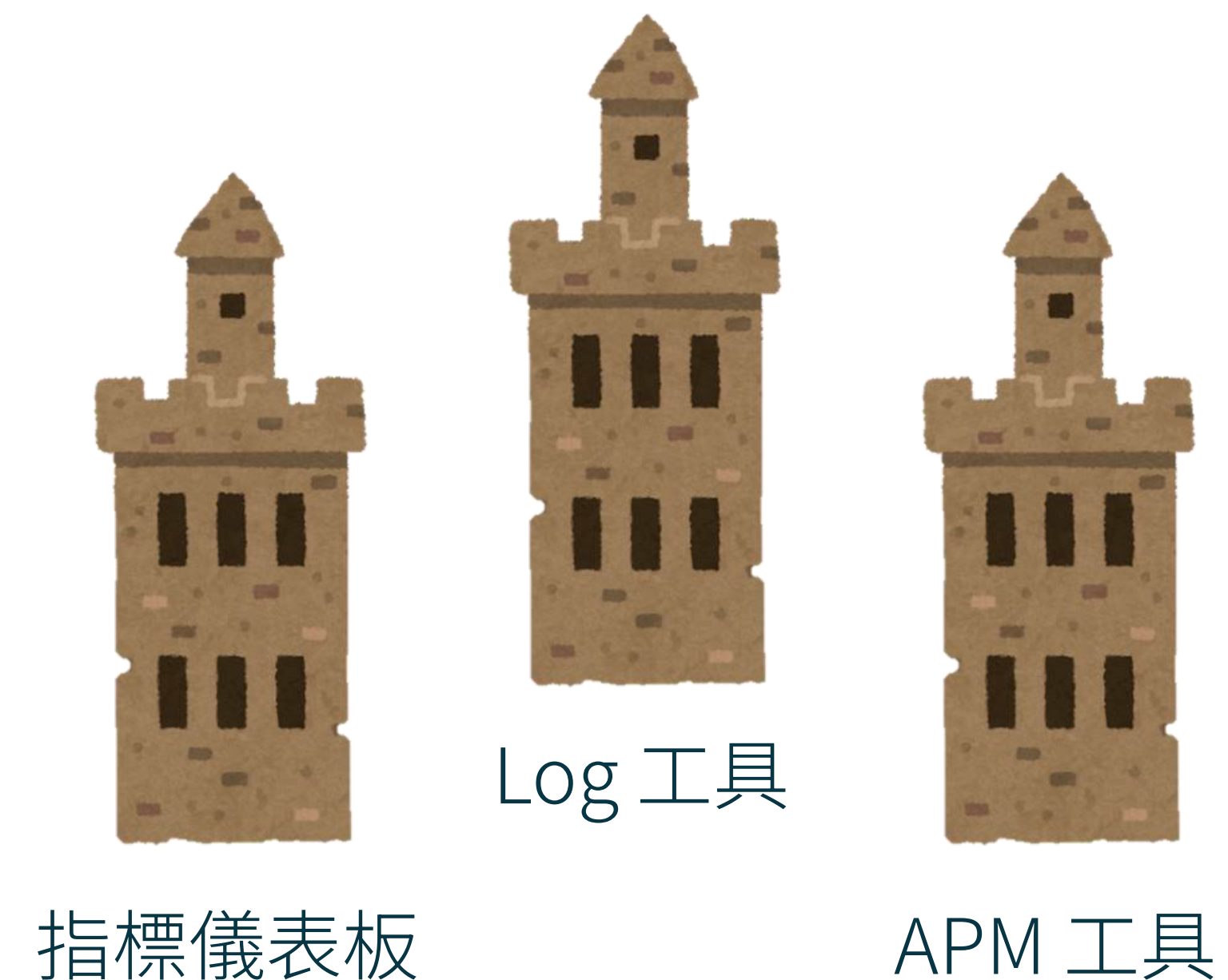
Observability Intro

透過各種資訊，清楚了解系統狀態

有足夠的資訊嗎？
他們都有被好好保存嗎？



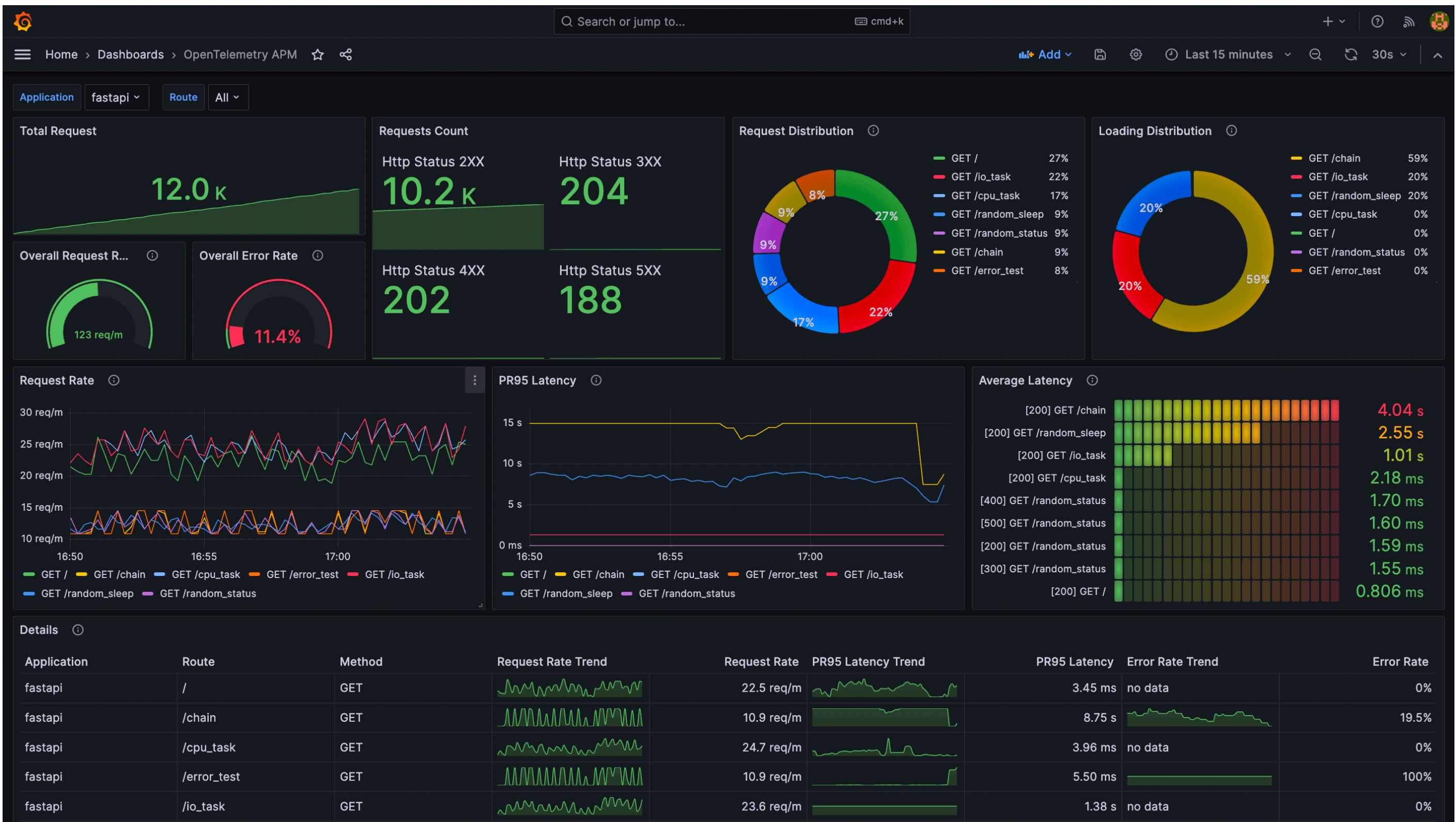
有好好利用這些資訊嗎？
是否散落個處，形成 Data Silo？



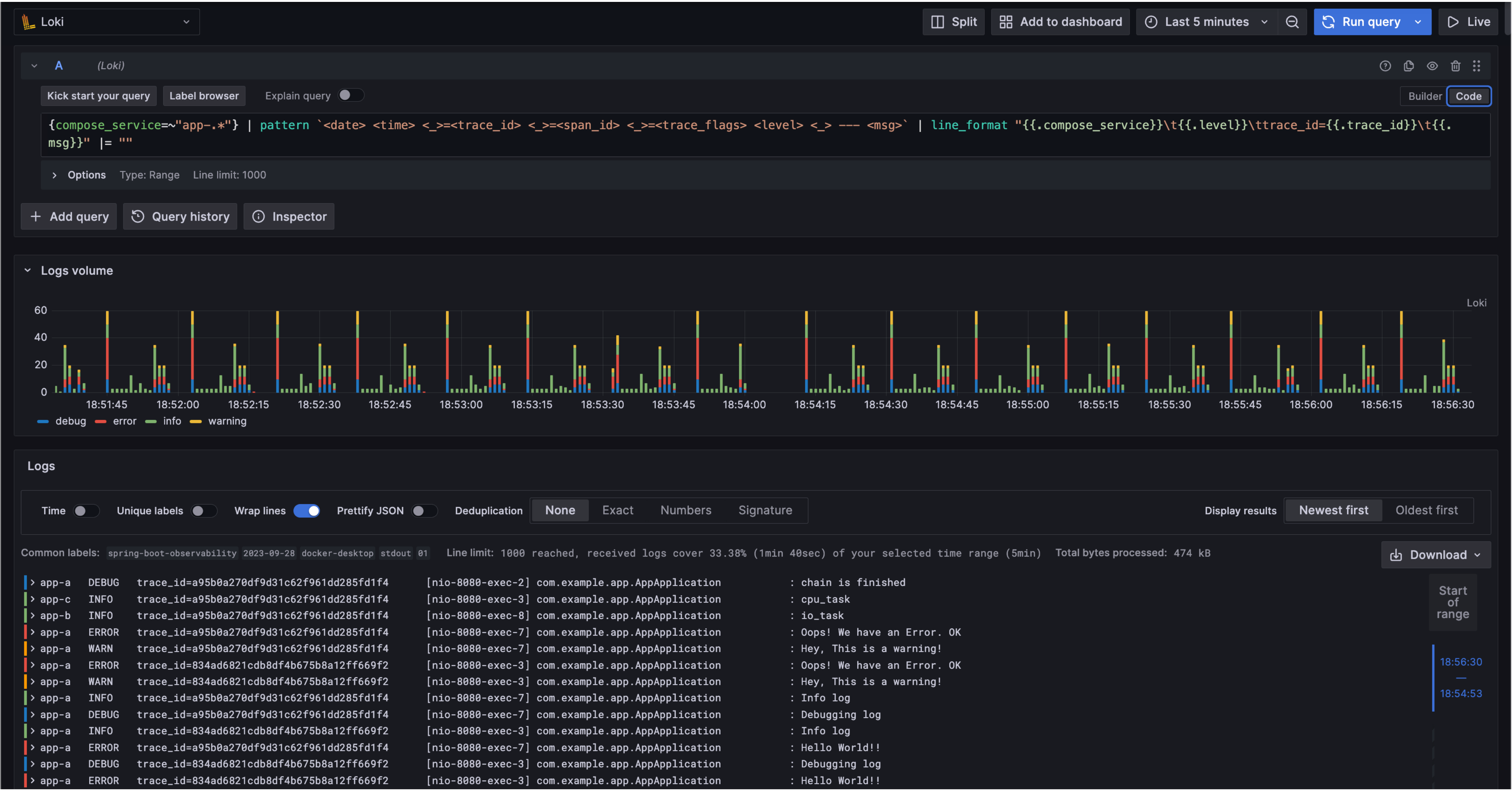
Observability Intro - Grafana



- Grafana Labs 開源的**視覺化**與儀表板工具，並提供告警功能

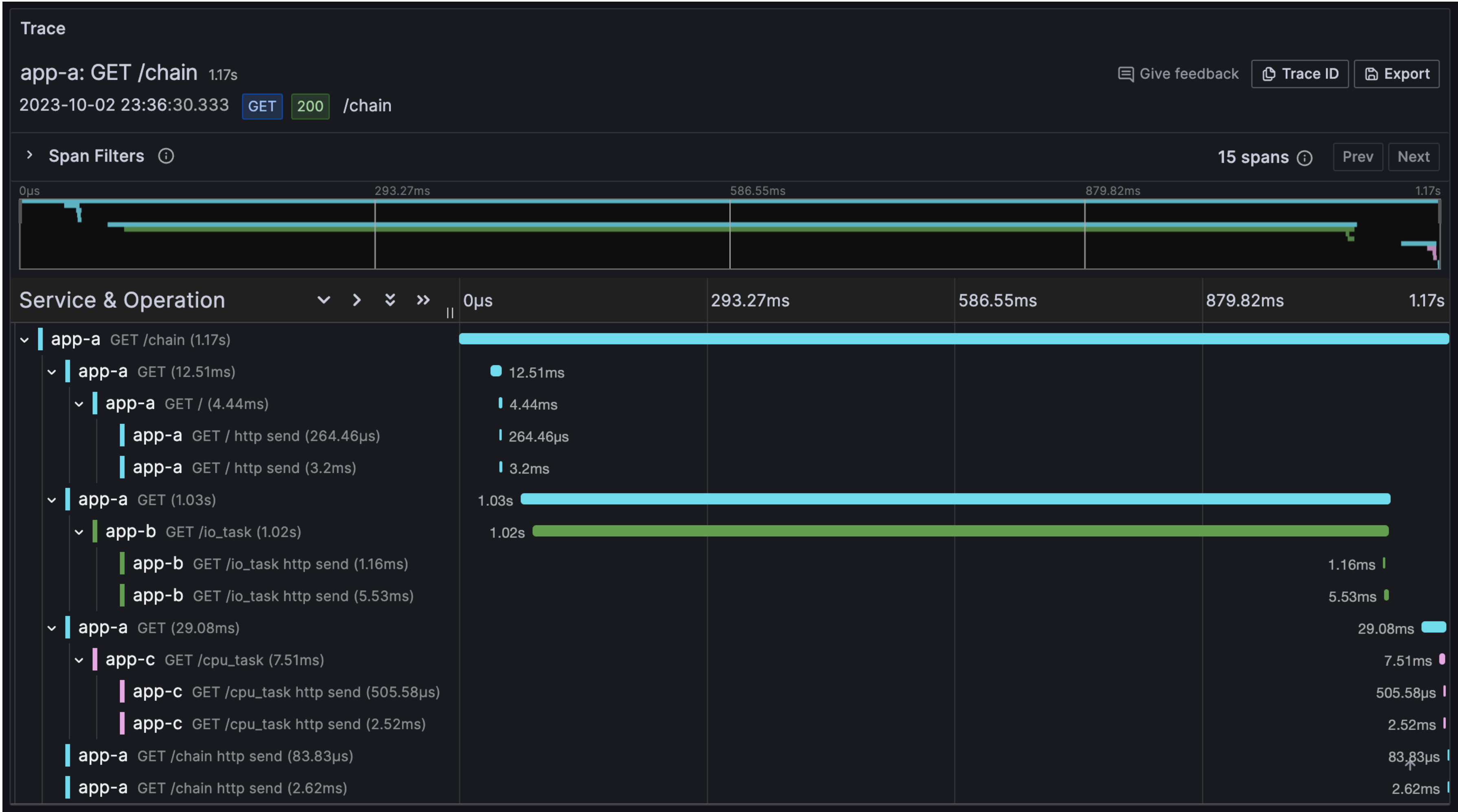


Observability Intro - Grafana



Grafana 檢視 Log

Observability Intro - Grafana



Grafana 檢視 Trace

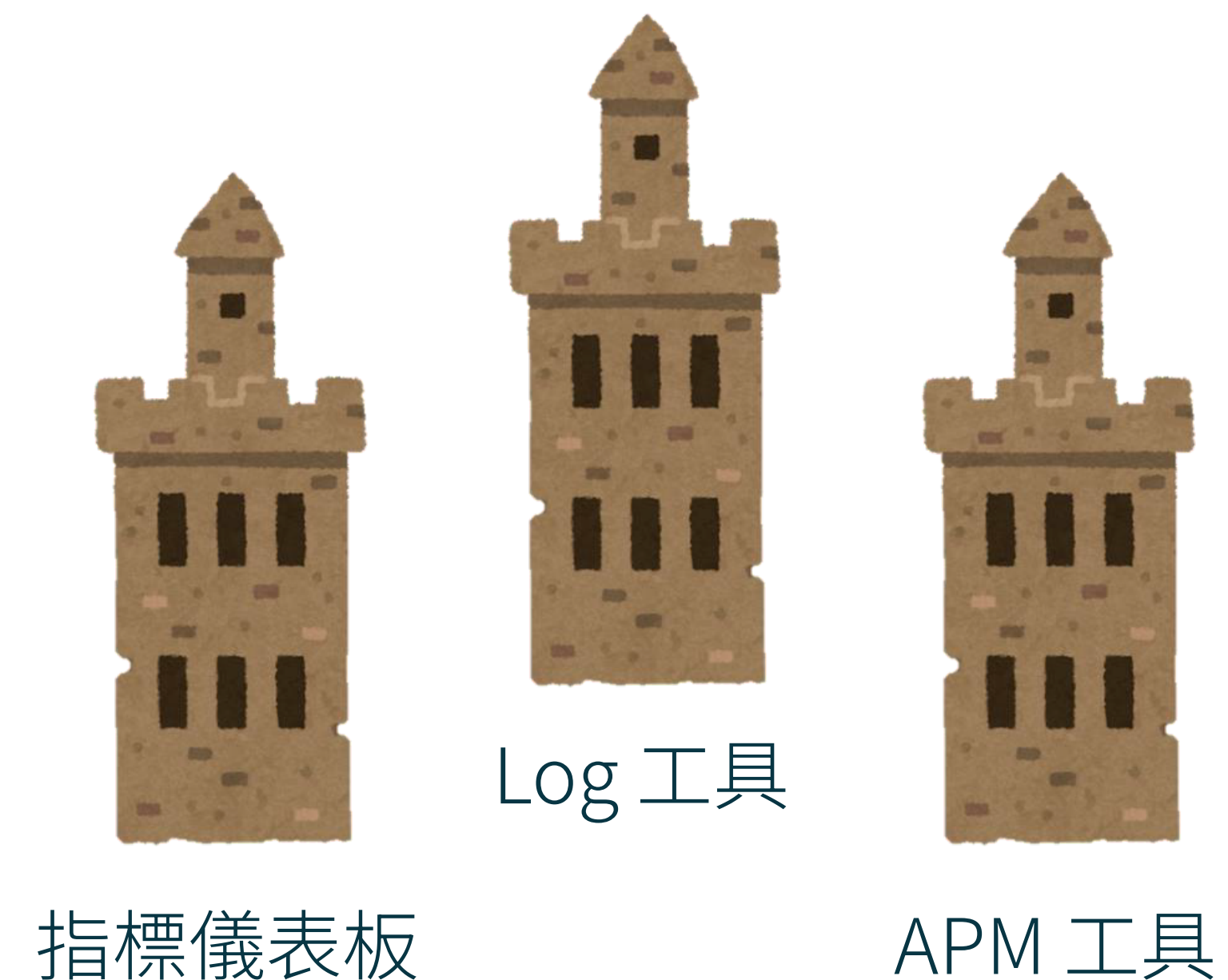
Observability Intro

透過各種資訊，清楚了解系統狀態

有足夠的資訊嗎？
他們都有被好好保存嗎？



有好好利用這些資訊嗎？
是否散落個處，形成 Data Silo？



Observability Signals Data Flow

Observability Signals Data Flow

資訊處理四步驟

生成



收集



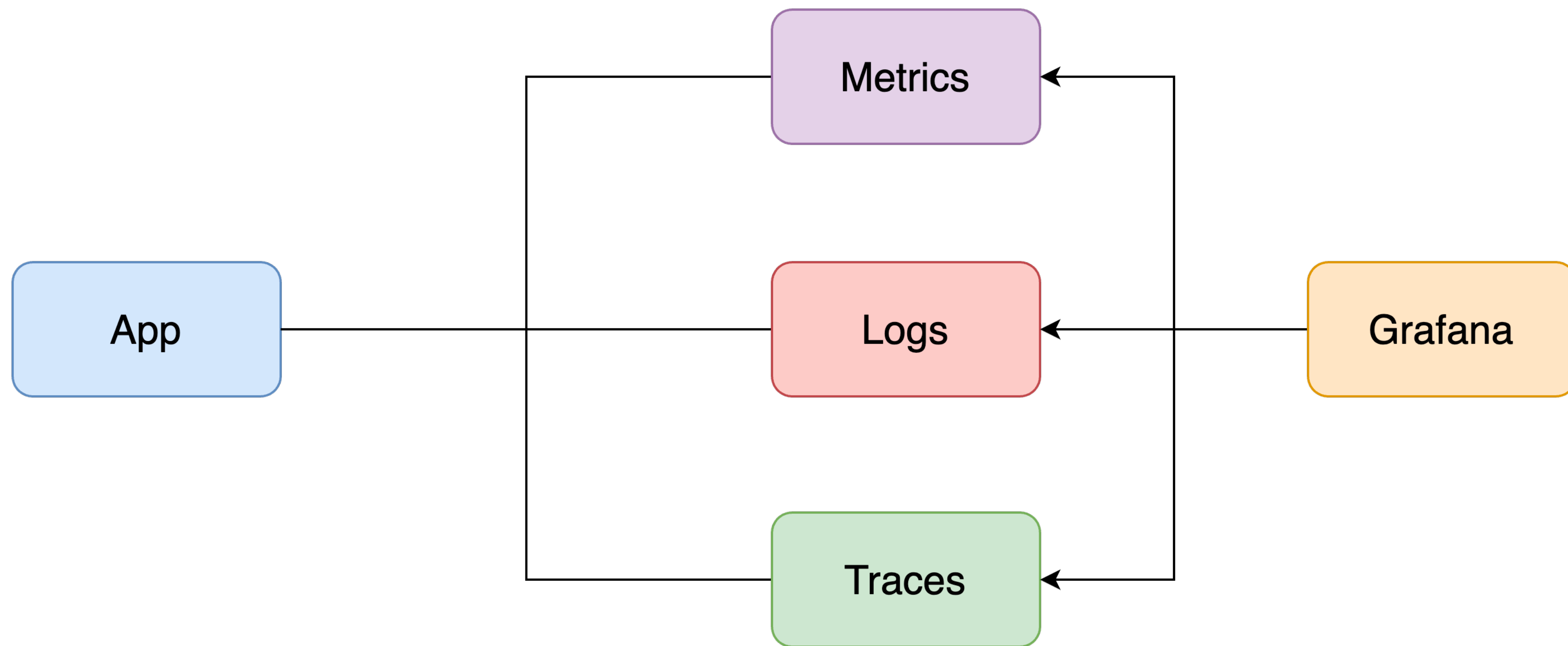
儲存



使用



Observability Signals Data Flow



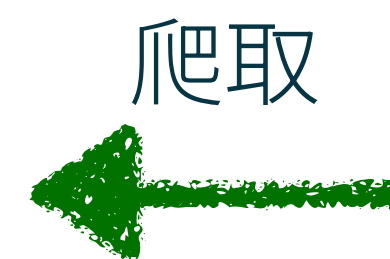
Observability Signals Data Flow - Metrics

Metrics - Prometheus



- Prometheus
 - 監控 (Monitoring) 與告警 (Alerting) 工具
 - **收集**不同時間點的量化指標，**儲存**後能以 PromQL 語法查詢

```
# HELP fastapi_requests Total count of requests by method and path.  
# TYPE fastapi_requests counter  
fastapi_requests_total{method="GET",path="/"} 6.0  
fastapi_requests_total{method="GET",path="/metrics"} 647.0  
fastapi_requests_total{method="GET",path="/io_task"} 3.0
```



Application 揭露的 Prometheus Metrics

Metrics - 生成：Prometheus Metrics

- Metrics **生成**
 - 通用格式：Prometheus Metrics/OpenMetrics
- 系統指標
 - **機器相關**的資訊，如：CPU、Memory、Network、磁碟空間
 - 使用 Exporter 採集，如：Node Exporter、cAdvisor
- 業務指標
 - **應用、業務相關**的資訊，如：Request 頻率、API 回應時間、Error Rate
 - 使用 Prometheus Client Library 生成，或包裝過的 Package 如：prometheus-fastapi-instrumentator

Metrics - 生成 : Prometheus Metrics

container_memory_usage_bytes{name="garafna"} [value]

Metrics Name

Label Pair

```
# HELP container_memory_usage_bytes Current memory usage in bytes
# TYPE container_memory_usage_bytes gauge
container_memory_usage_bytes{image="prom/prometheus:v2.54.0",name="prometheus"} 6.4237568e+07
container_memory_usage_bytes{image="grafana/grafana:11.2.0",name="grafana"} 1.08425216e+08
container_memory_usage_bytes{image="gcr.io/cadvisor/cadvisor:v0.49.1",name="cadvisor"} 8.8117248e+07

# HELP fastapi_requests Total count of requests by method and path.
# TYPE fastapi_requests counter
fastapi_requests_total{method="GET",path="/"} 10.0
fastapi_requests_total{method="GET",path="/cpu_task"} 10.0
fastapi_requests_total{method="GET",path="/io_task"} 10.0

# HELP fastapi_responses Total count of responses by method, path and status codes.
# TYPE fastapi_responses counter
fastapi_responses_total{method="GET",path="/random_status",status_code="200"} 5.0
fastapi_responses_total{method="GET",path="/random_status",status_code="300"} 2.0
```

Metrics - 生成：Prometheus Metrics

統計數值分佈

可增減的數值

只增不減的累計值

```
basic.py

from fastapi import FastAPI, Request
from prometheus_client import Counter, Gauge, Histogram, make_asgi_app

REQUEST_TIME = Histogram("request_processing_seconds", "Time spent processing request")

REQUEST_IN_PROCESSING = Gauge("request_in_processing", "Requests in progress")

REQUEST_COUNT = Counter("requests_count", "Total count of requests")

@app.get("/")
async def root(request: Request):
    REQUEST_IN_PROCESSING.inc()
    REQUEST_COUNT.inc()
    with REQUEST_TIME.time():
        await random_sleep()
    REQUEST_IN_PROCESSING.dec()
    return {"Hello": "World"}
```

Metrics - 收集與儲存：Prometheus

<http://app-a:8000/metrics>



Prometheus

```
# HELP request_processing_seconds Time spent processing request
# TYPE request_processing_seconds histogram
request_processing_seconds_bucket{le="0.005"} 1.0
request_processing_seconds_bucket{le="0.05"} 1.0
request_processing_seconds_bucket{le="0.5"} 1.0
request_processing_seconds_bucket{le="2.5"} 6.0
request_processing_seconds_bucket{le="7.5"} 17.0
request_processing_seconds_bucket{le="+Inf"} 17.0
request_processing_seconds_count 17.0
request_processing_seconds_sum 52.04826683411375
# HELP request_in_processing Requests in progress
# TYPE request_in_processing gauge
request_in_processing 0.0
# HELP requests_count_total Total count of requests
# TYPE requests_count_total counter
requests_count_total 17.0
```

```
prometheus.yml

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: 'cadvisor'
    static_configs:
      - targets: ['cadvisor:8080']

  - job_name: 'app-a'
    scrape_interval: 5s
    metrics_path: '/metrics'
    static_configs:
      - targets: ['app-a:8000']
```

Scrape

Metrics - 使用：Grafana Dashboard



Metrics - Data Flow



Observability Signals Data Flow - Logs

Logs - 生成

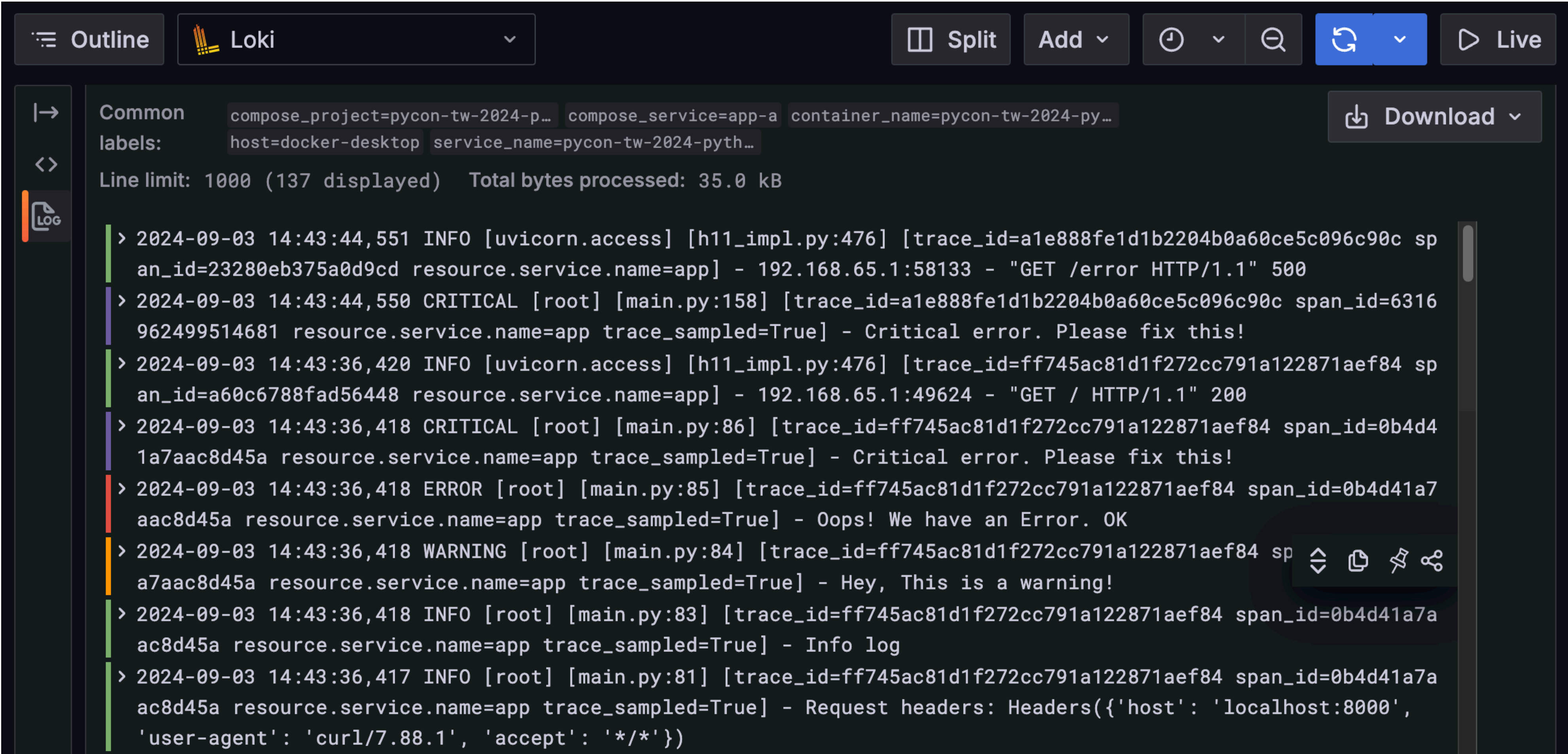
- 內容
 - 時間、Level、其他補充資訊
- 格式
 - 非結構化
 - 一般搭配 Log Pattern，例如 `%d{yyyy-MM-dd HH:mm:ss} [%-5p] [%c] [%t] %m%n`
 - 結構化
 - JSON
 - Logfmt
 - `time="2021-09-19 15:00:00" level="INFO" logger="main" message="Hello World!"`
- 輸出方式
 - Console：通過 STDOUT 或 STDERR 輸出至 Console
 - File：記錄在檔案中，可能搭配 Rotate 進行檔案管理
 - Stream：透過網路傳輸，將 Log 即時輸出至儲存 Log 的服務

Logs - 收集與儲存：Loki & Docker Driver



- Loki
 - Grafana Labs 開源的 Log **儲存** 工具
 - 借鑑 Prometheus 與 PromQL，透過 LogQL 查詢 Log
 - 簡單易用，就像是使用 Ctrl + F / CMD + F 一樣
- Loki Docker Driver
 - 使用 Docker 的 Logging Driver 機制，從 Container **收集** Log 至 Loki

Logs - 使用：Grafana Explore Log



The screenshot displays the Grafana Explore Log interface. At the top, there's a navigation bar with 'Outline', 'Loki', and various action buttons like 'Split', 'Add', 'Refresh', and 'Live'. Below this, a search bar contains filters: 'compose_project=pycon-tw-2024-p...', 'compose_service=app-a', and 'container_name=pycon-tw-2024-py...'. A 'Download' button is also present. The main area shows log entries with a 'Line limit: 1000 (137 displayed)' and 'Total bytes processed: 35.0 kB'. The logs are color-coded by severity: INFO (green), CRITICAL (red), ERROR (red), and WARNING (orange). Each entry includes a timestamp, log level, source, and message. For example, one entry shows a 'GET /error HTTP/1.1' 500 response, followed by a 'Critical error. Please fix this!' message. Another entry shows a 'GET / HTTP/1.1' 200 response, followed by another 'Critical error. Please fix this!' message. A third entry shows an 'Oops! We have an Error. OK' message. A fourth entry shows a 'Hey, This is a warning!' message. The last two entries are 'Info log' and 'Request headers: Headers({'host': 'localhost:8000', 'user-agent': 'curl/7.88.1', 'accept': '*//*'})'.

```
> 2024-09-03 14:43:44,551 INFO [uvicorn.access] [h11_impl.py:476] [trace_id=a1e888fe1d1b2204b0a60ce5c096c90c span_id=23280eb375a0d9cd resource.service.name=app] - 192.168.65.1:58133 - "GET /error HTTP/1.1" 500
> 2024-09-03 14:43:44,550 CRITICAL [root] [main.py:158] [trace_id=a1e888fe1d1b2204b0a60ce5c096c90c span_id=6316962499514681 resource.service.name=app trace_sampled=True] - Critical error. Please fix this!
> 2024-09-03 14:43:36,420 INFO [uvicorn.access] [h11_impl.py:476] [trace_id=ff745ac81d1f272cc791a122871aef84 span_id=a60c6788fad56448 resource.service.name=app] - 192.168.65.1:49624 - "GET / HTTP/1.1" 200
> 2024-09-03 14:43:36,418 CRITICAL [root] [main.py:86] [trace_id=ff745ac81d1f272cc791a122871aef84 span_id=0b4d41a7aac8d45a resource.service.name=app trace_sampled=True] - Critical error. Please fix this!
> 2024-09-03 14:43:36,418 ERROR [root] [main.py:85] [trace_id=ff745ac81d1f272cc791a122871aef84 span_id=0b4d41a7aac8d45a resource.service.name=app trace_sampled=True] - Oops! We have an Error. OK
> 2024-09-03 14:43:36,418 WARNING [root] [main.py:84] [trace_id=ff745ac81d1f272cc791a122871aef84 span_id=0b4d41a7aac8d45a resource.service.name=app trace_sampled=True] - Hey, This is a warning!
> 2024-09-03 14:43:36,418 INFO [root] [main.py:83] [trace_id=ff745ac81d1f272cc791a122871aef84 span_id=0b4d41a7aac8d45a resource.service.name=app trace_sampled=True] - Info log
> 2024-09-03 14:43:36,417 INFO [root] [main.py:81] [trace_id=ff745ac81d1f272cc791a122871aef84 span_id=0b4d41a7aac8d45a resource.service.name=app trace_sampled=True] - Request headers: Headers({'host': 'localhost:8000', 'user-agent': 'curl/7.88.1', 'accept': '*//*'})
```

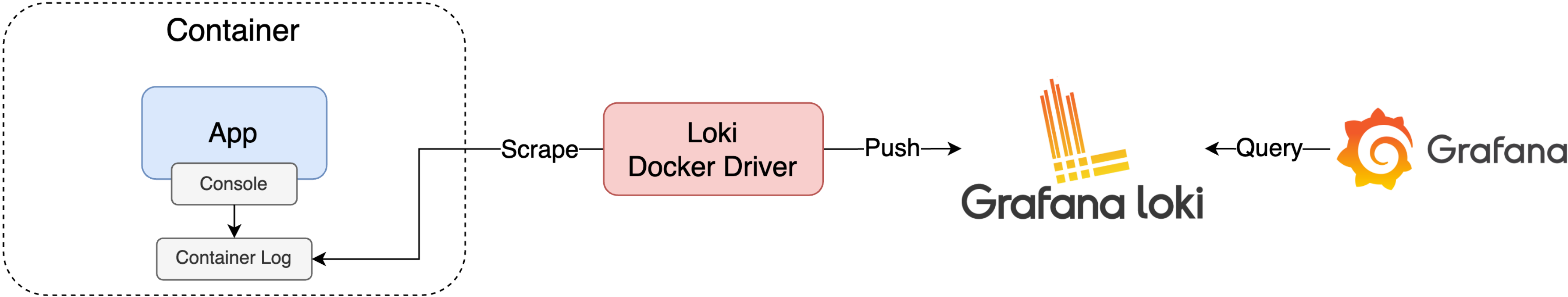
Logs - Data Flow

生成

收集

儲存

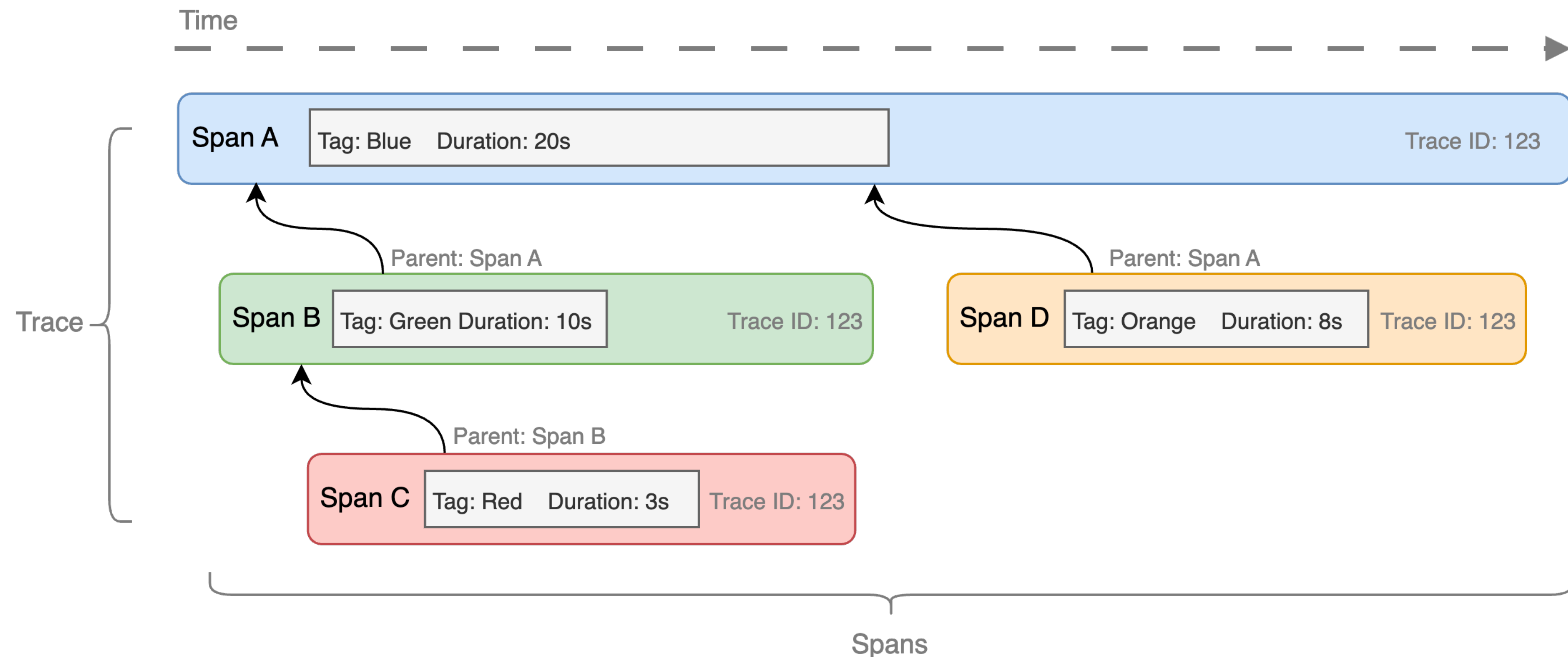
使用



Observability Signals Data Flow - Traces

Traces(Distributed Tracing)

- 用於監控**跨服務請求**，透過統一的 **Trace ID** 串聯，紀錄一個行為在不同服務間的**歷程與資訊**，例如：執行時間、錯誤訊息、請求來源 IP、SQL 語法等



Traces - OpenTelemetry



- OpenTelemetry
 - 處理 Telemetry Data (Metrics、Logs、Traces) 的 **API**、**SDK**、**Tool** 組合包
 - CNCF Project，目前主流的 Trace 規範，已經有超過 60 個組織 支援
 - 2019 年由 OpenTracing 與 OpenCensus 合併而成

API

資料格式(OTLP)
傳輸方式(HTTP、gRPC)

SDK

開發工具
Instrumentation

Collector

Telemetry Data 收集器

Traces - OpenTelemetry



- OpenTelemetry
 - 處理 Telemetry Data (Metrics、Logs、Traces) 的 **API**、**SDK**、**Tool** 組合包
 - CNCF Project，目前主流的 Trace 規範，已經有超過 60 個組織 支援
 - 2019 年由 OpenTracing 與 OpenCensus 合併而成

API

資料格式(OTLP)
傳輸方式(HTTP、gRPC)

SDK

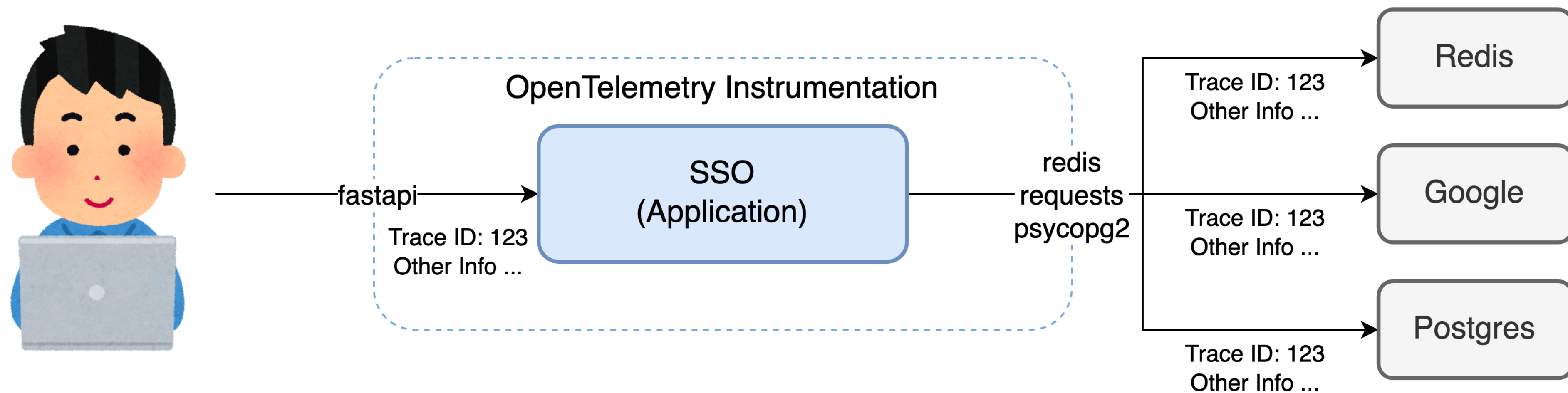
開發工具
Instrumentation

Collector

Telemetry Data 收集器

Traces - 生成：OpenTelemetry Instrumentation

- 使用 OpenTelemetry 的 Instrumentation Package 採集資料作為 Trace 資訊
- 支援的 Package 清單：[opentelemetry-python-contrib](#)



Traces - 生成：OpenTelemetry Instrumentation

The screenshot displays a trace for a GET request to the root path. The trace is titled "basic GET / (5.02s)". The method is GET, the service is basic, and the status is unset. The duration is 5.02s, the start time is 0µs (22:07:51.411), the child count is 2, and the kind is server. The library name is opentelemetry.instrumentation.fastapi and the library version is 0.48b0. The span attributes include http.flavor, http.host, http.method, http.route, http.scheme, http.server_name, http.status_code, http.target, http.url, http.user_agent, net.host.port, net.peer.ip, and net.peer.port. The resource attributes include service.name, telemetry.auto.version, and telemetry.sdk.language.

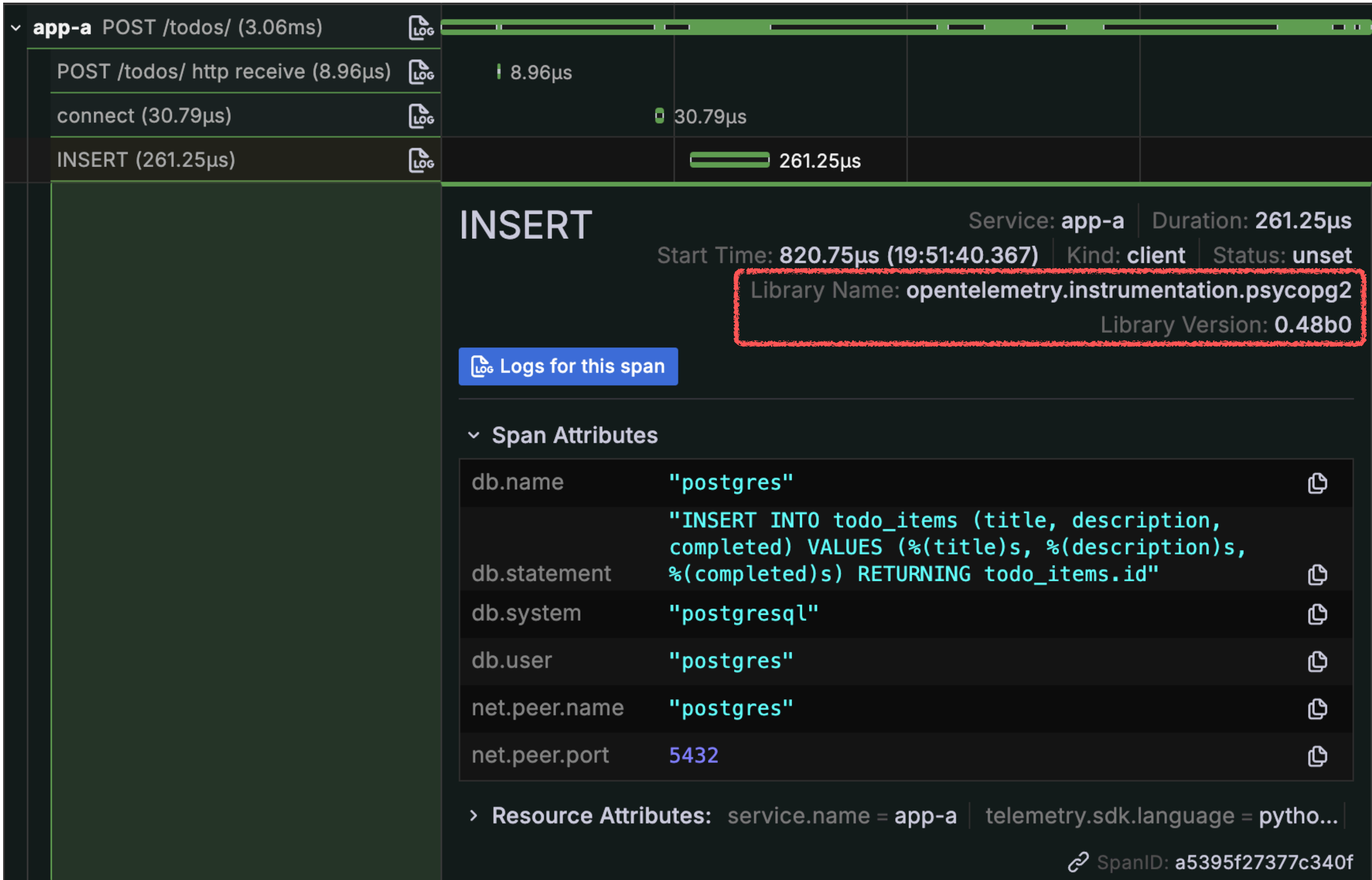
Attribute	Value
http.flavor	"1.1"
http.host	"172.18.0.7:8080"
http.method	"GET"
http.route	"/"
http.scheme	"http"
http.server_name	"localhost:8080"
http.status_code	200
http.target	"/"
http.url	"http://172.18.0.7:8080/"
http.user_agent	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
net.host.port	8080
net.peer.ip	"192.168.65.1"
net.peer.port	36449

Resource Attributes: service.name = basic | telemetry.auto.version = 0.48b0 | telemetry.sdk.language = py...

SpanID: 181c58ac9eccd796

FastAPI Instrumentation，紀錄 URL、IP、Status Code 等資訊

Traces - 生成：OpenTelemetry Instrumentation



psycopg2 Instrumentation，紀錄 DB、SQL 相關資訊

Traces - 生成：Instrumentation 使用方式

- Instrumentation 的兩種使用方式
 - Code-based Instrumentation：需調整程式，自行啟用 Intrumentator
 - Zero-code Instrumentation：無須調整程式碼，利用 monkey patching 動態載入
 - 安裝 Package opentelemetry-distro、opentelemetry-exporter-otlp
 - 掃描可用的 Instrumentation Package 並安裝

```
# 根據已安裝的 Package 列出可用的 Instrumentation Package 清單
opentelemetry-bootstrap -a requirements
# 直接安裝可用的 Instrumentation Package 清單
opentelemetry-bootstrap -a install
```

- 使用 `opentelemetry-instrument` 啟動服務

```
opentelemetry-instrument python main.py
```

Traces - 儲存：Tempo



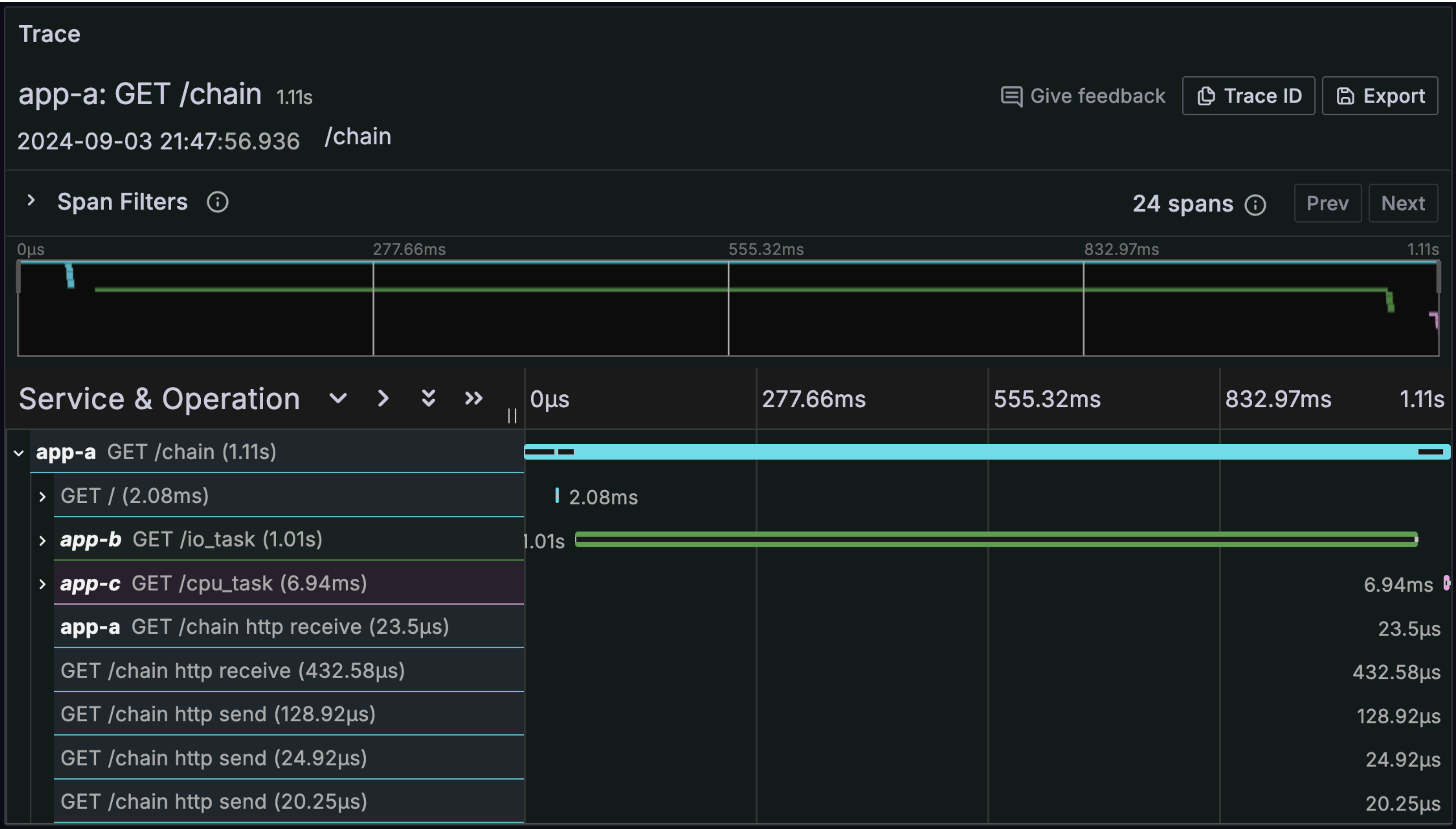
- Tempo
 - Grafana Labs 開源的 Distributed Tracing 儲存工具
 - 支援 OpenTelemetry、Jaeger、Zipkin

```
app:  
  image: fastapi-app  
  environment:  
    - OTEL_TRACES_EXPORTER=otlp  
    - OTEL_EXPORTER_OTLP_ENDPOINT=http://tempo:4317  
    - OTEL_SERVICE_NAME=basic  
  command: opentelemetry-instrument python main.py
```



透過環境變數設定 Trace 資訊發送目標

Traces - 使用：Grafana Explore Traces



Grafana 視覺化顯示 Trace

Traces - Data Flow

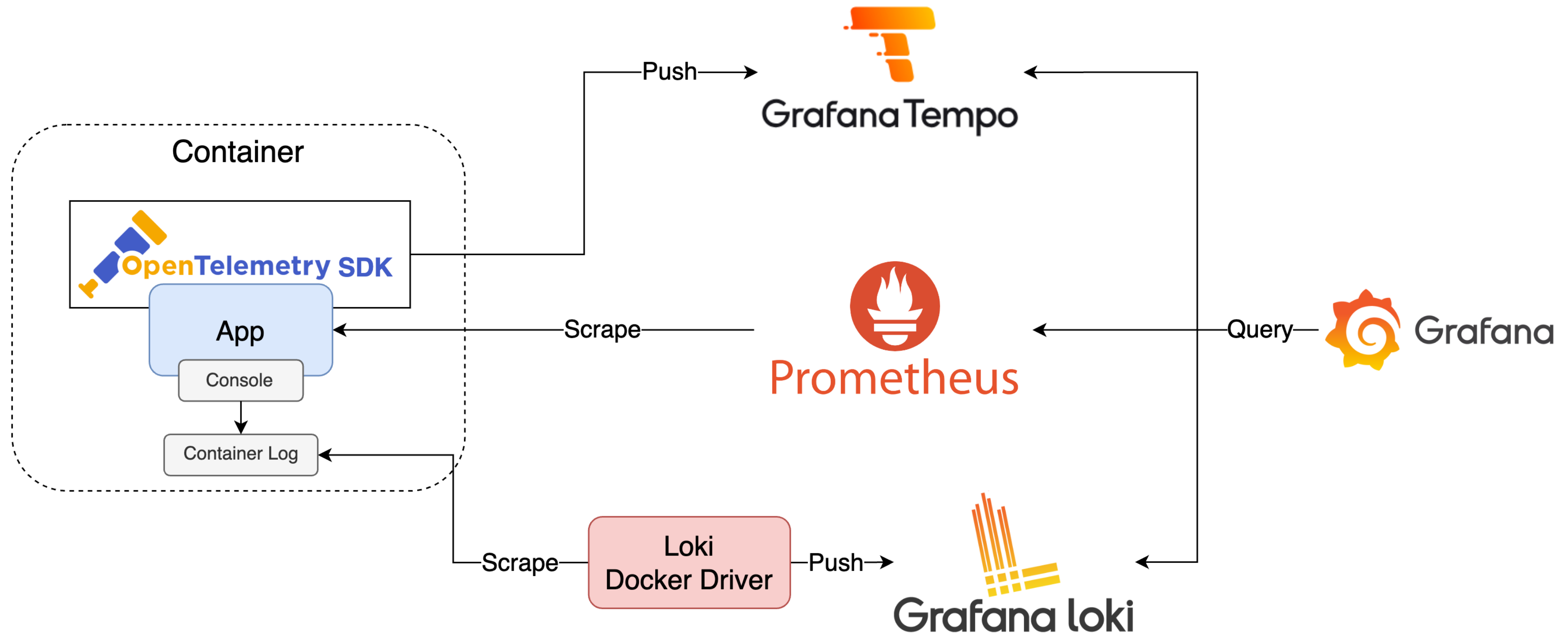
生成
推送

儲存

使用



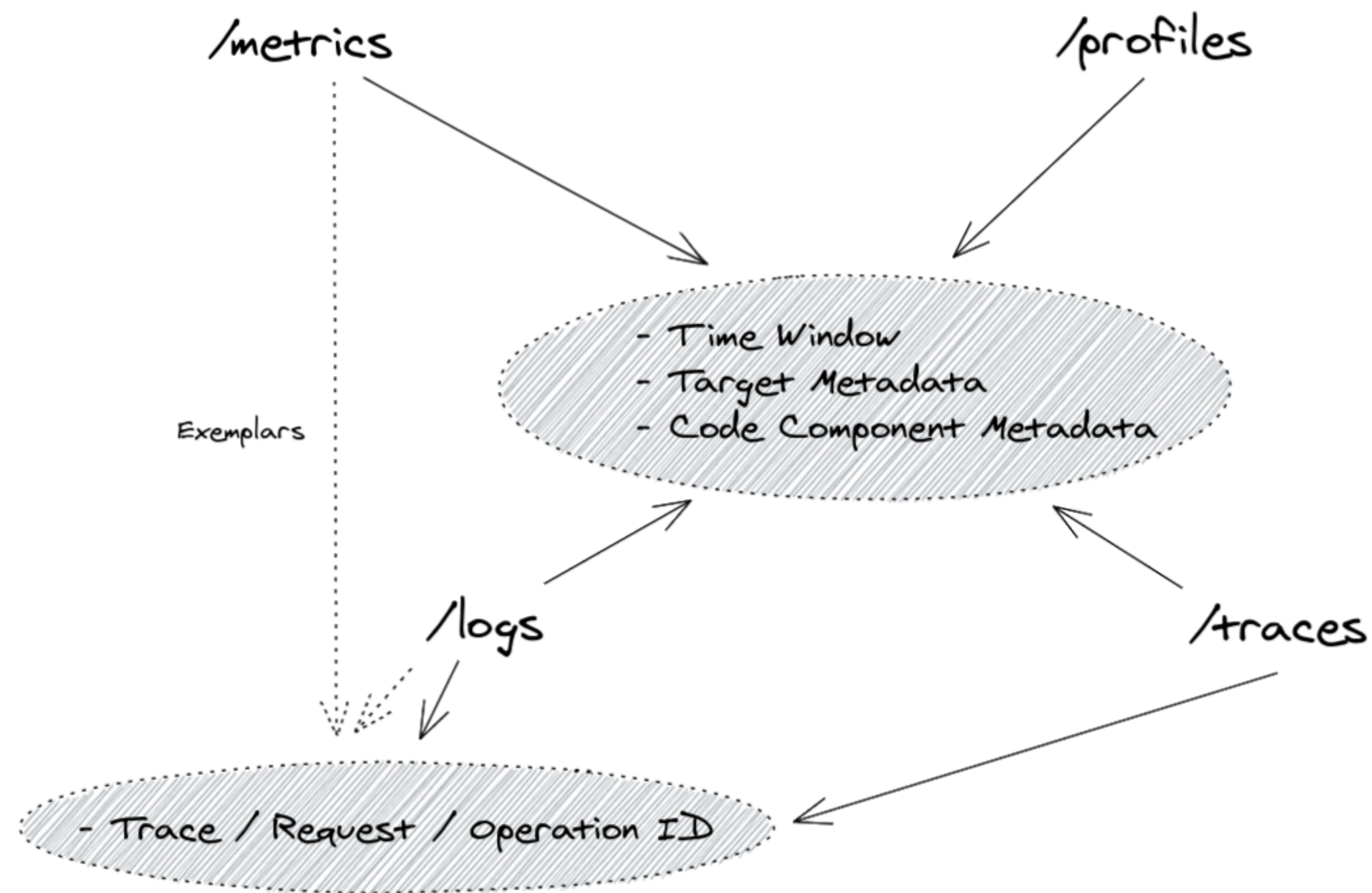
Observability Signals Data Flow



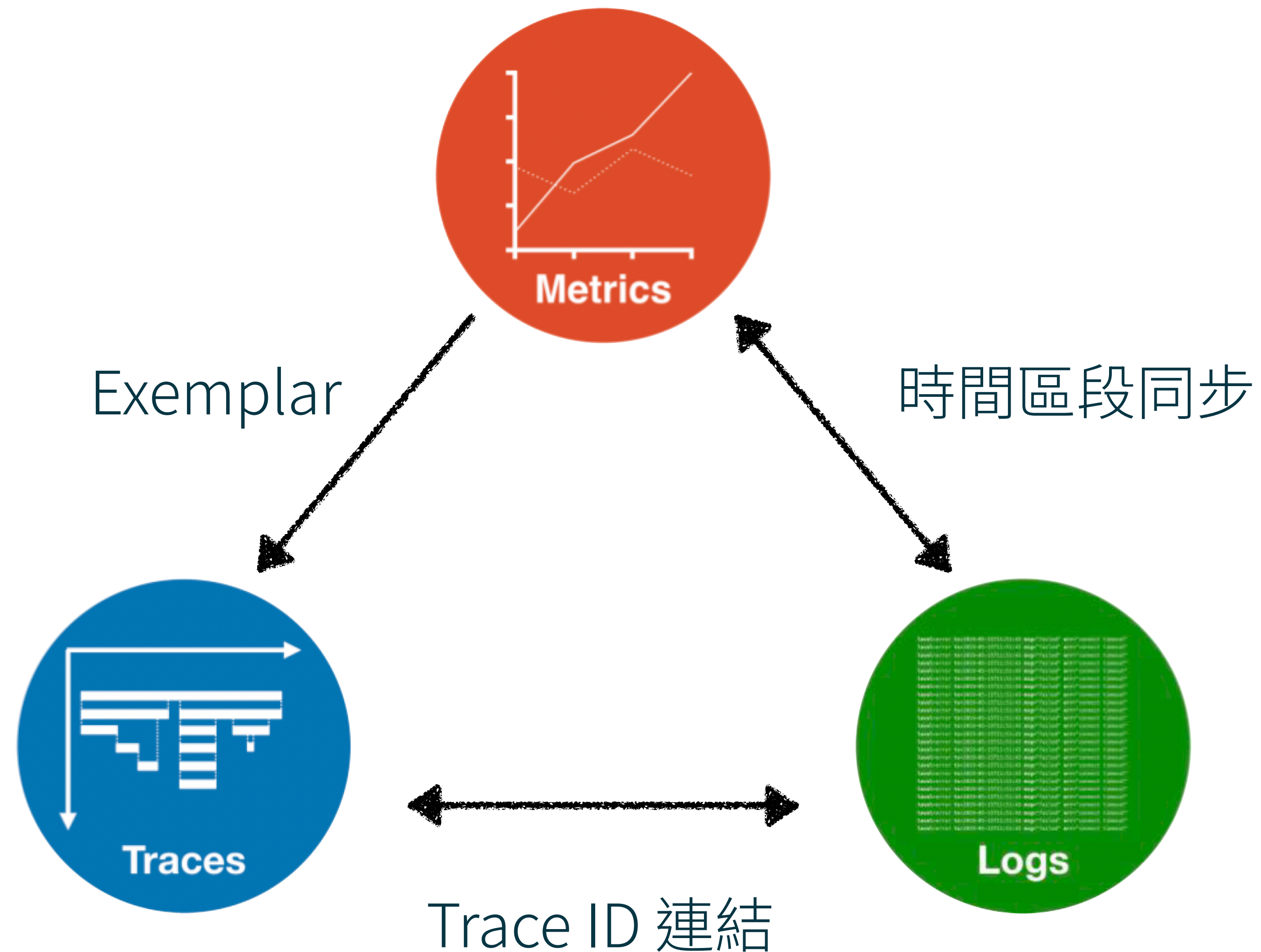
資訊交互應用

資訊交互應用

可觀測資訊彼此間是有關係的，例如時間區段、Trace ID



資訊交互應用 - Grafana



資訊交互應用 - Traces vs Logs

- OpenTelemetry 的 Logging Instrumentation，注入 Trace 資訊到 Log 中
 - 透過環境變數啟用或自行啟動 Logging Instrumentor

```
OTEL_PYTHON_LOG_CORRELATION_LOG_FORMAT=true opentelemetry-instrument python basic.py
```

```
from opentelemetry.instrumentation.logging import LoggingInstrumentor  
  
LoggingInstrumentor().instrument(set_logging_format=True)
```



```
> 2024-09-03 14:43:36,418 INFO [root] [main.py:83] [trace_id=ff745ac81d1f272cc791a122871aef84  
span_id=0b4d41a7aac8d45a resource.service.name=app trace_sampled=True] - Info log
```

帶有 Trace 與 Span 資訊的 Log

資訊交互應用 - Traces vs Metrics

- OpenMetrics 的 Exemplar 能夠在 Metrics 後註記特定一筆的 Trace ID 與值

```
from opentelemetry import trace
# retrieve trace id for exemplar
span = trace.get_current_span()
trace_id = trace.format_trace_id(span.get_span_context().trace_id)

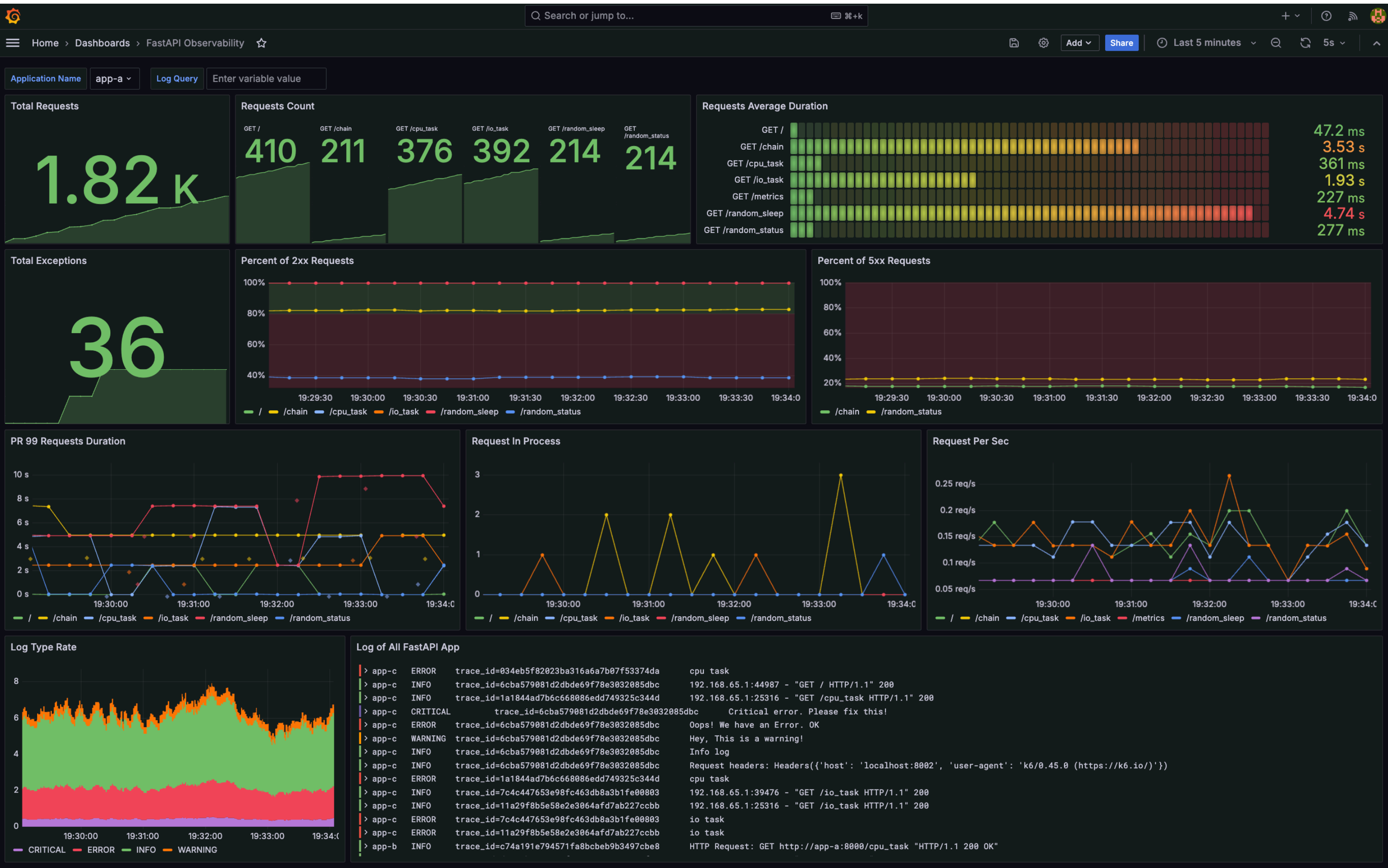
REQUESTS_PROCESSING_TIME.labels(
    method=method, path=path, app_name=self.app_name
).observe(after_time - before_time, exemplar={"TraceID": trace_id})
```



```
fastapi_requests_duration_seconds_bucket{app_name="app",le="7.5",method="GET",path="/random_sleep"} 2.0 #
{TraceID="c79d6a47cb1ca3782f8c87c1d16594bc"} 5.010398501995951 1725373200.8618953
fastapi_requests_duration_seconds_bucket{app_name="app",le="10.0",method="GET",path="/random_sleep"} 2.0
fastapi_requests_duration_seconds_bucket{app_name="app",le="+Inf",method="GET",path="/random_sleep"} 5.0 #
{TraceID="af8bfbbf7fc847e66fc2fa89dfd05b31"} 12.018873464083299 1725373200.8604093
```

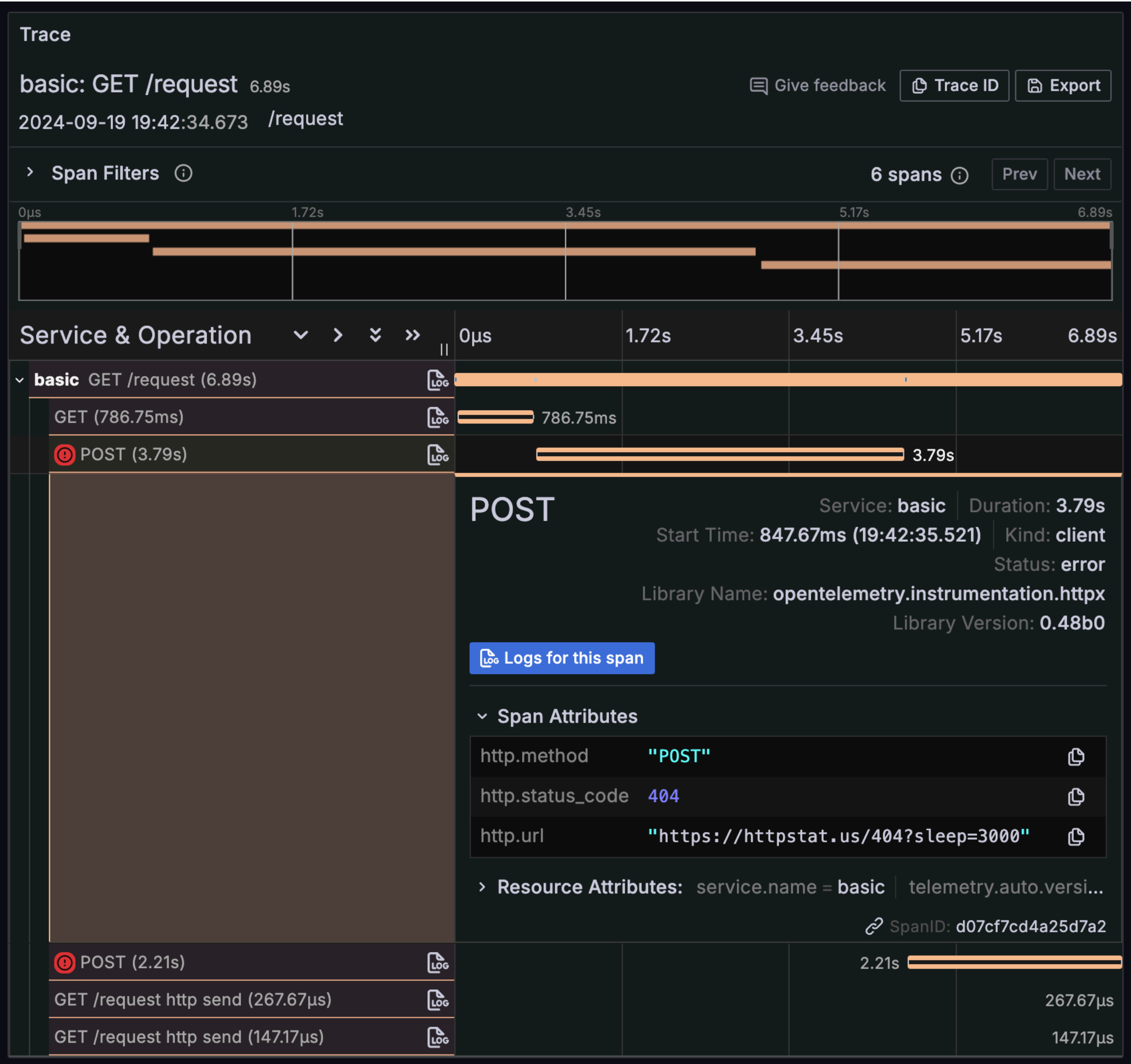
帶有 Trace ID 的 Metrics

Demo - OpenTelemetry Instrumentation 與資訊交互應用



<https://github.com/blueswen/pycon-tw-2024-python-observability>

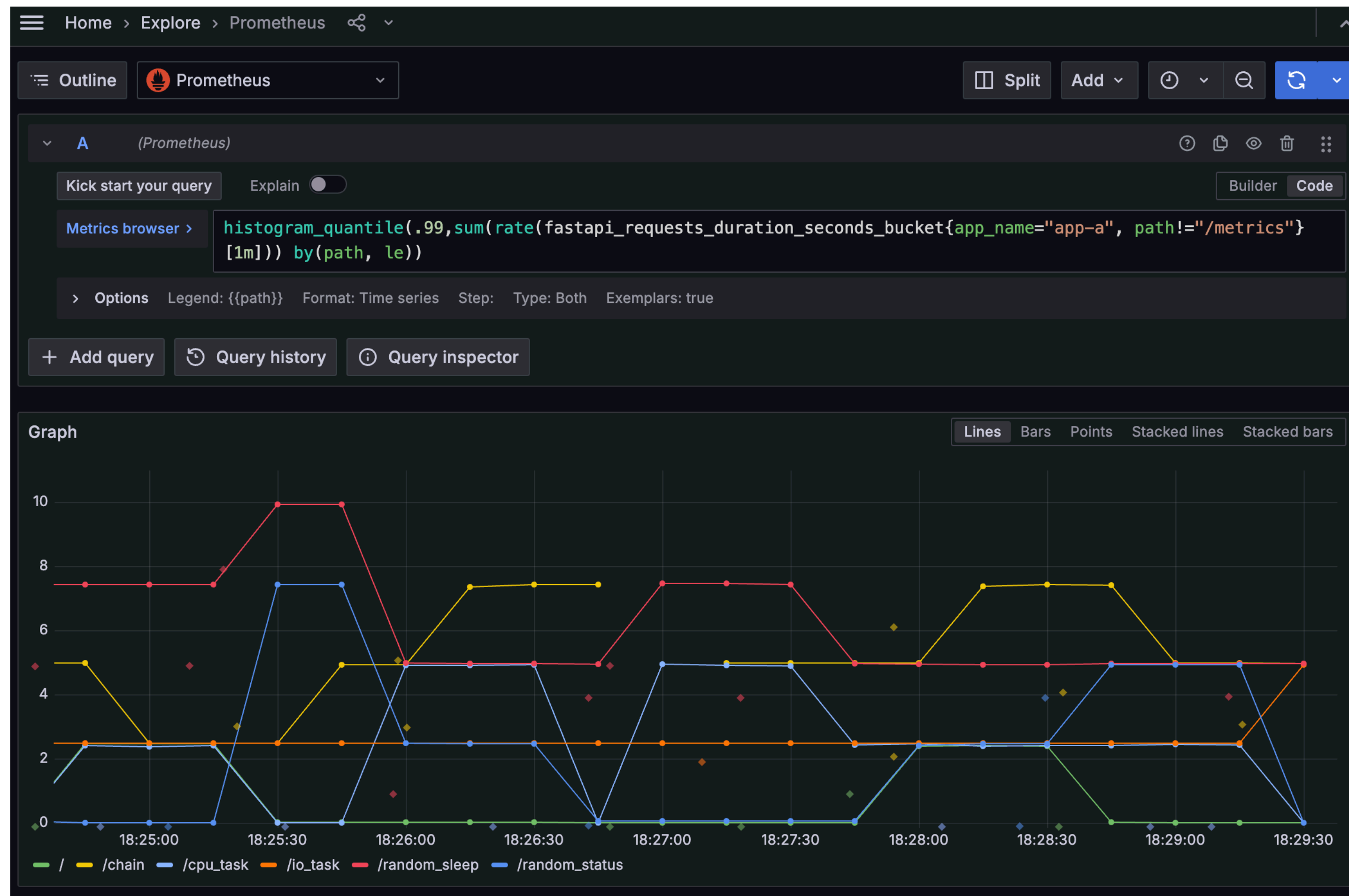
Demo - Zero-code Instrumentation



```
@app.get("/request")
async def request(request: Request):
    httpx.get("https://httpstat.us/200")
    httpx.post("https://httpstat.us/404?sleep=3000")
    httpx.post("https://httpstat.us/500?sleep=1500")
    return {"request": "completed"}
```

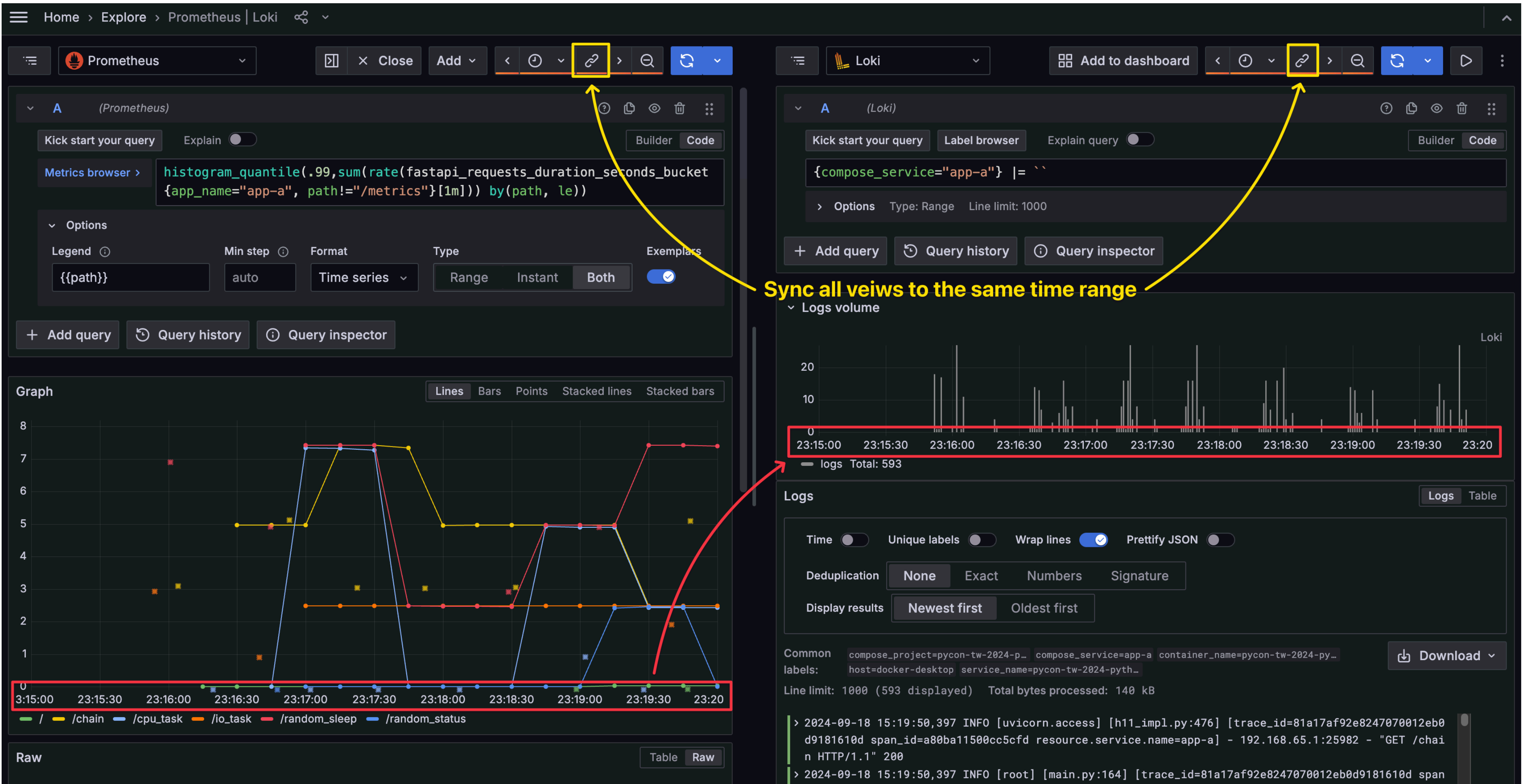
```
basic:
  image: ghcr.io/blueswen/pycon-tw-2024-python-observability/app:latest
  ports:
    - "8080:8000"
  environment:
    - OTEL_TRACES_EXPORTER=otlp
    - OTEL_EXPORTER_OTLP_ENDPOINT=http://tempo:4317
    - OTEL_METRICS_EXPORTER=none
    - OTEL_SERVICE_NAME=basic
  command: opentelemetry-instrument python basic.py
```

Demo - 資訊交互應用 - Metrics vs Logs



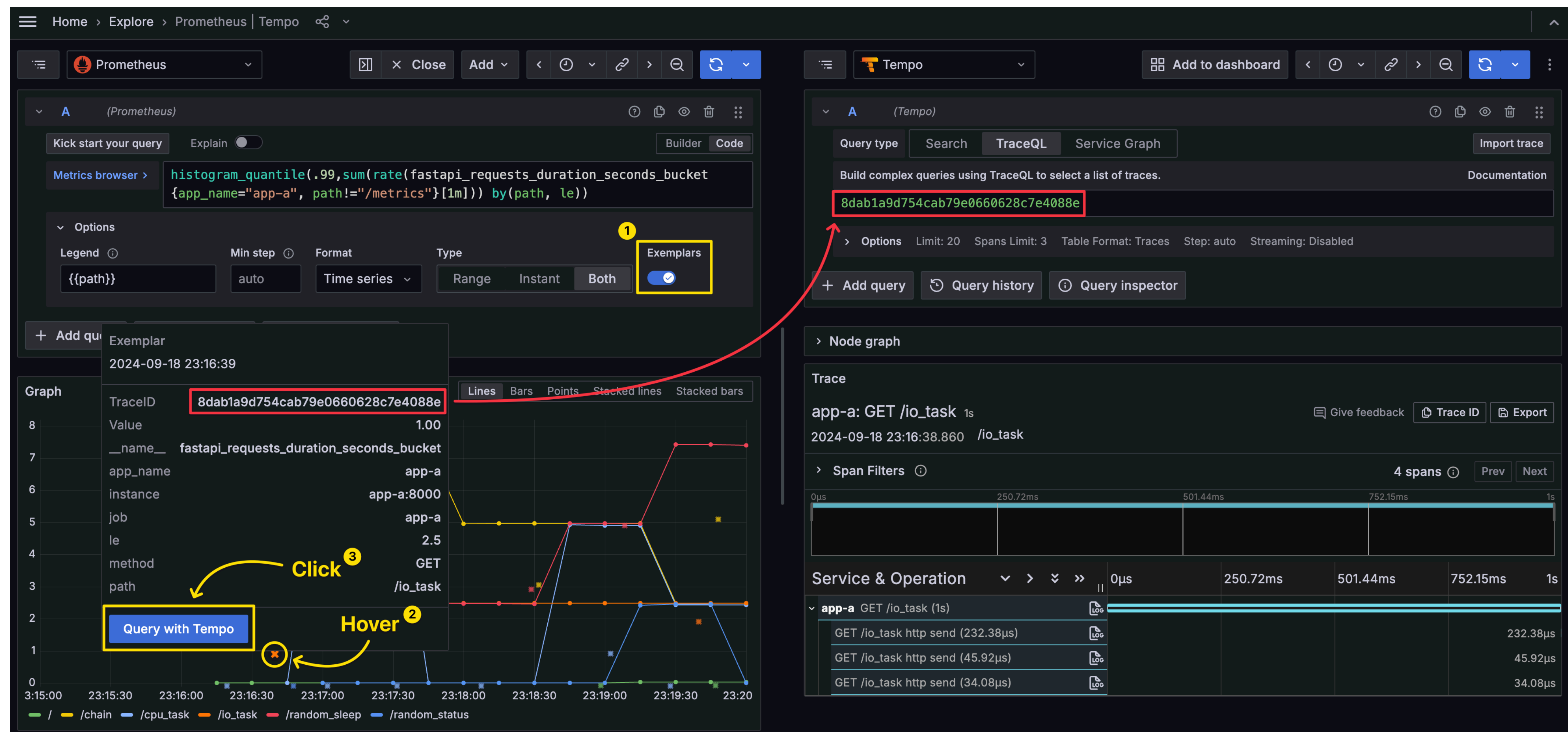
利用同步時間功能同時查詢 Metrics 與 Logs

Demo - 資訊交互應用 - Metrics vs Logs



利用同步時間功能同時查詢 Metrics 與 Logs

Demo - 資訊交互應用 - Exemplar: Metrics to Traces



搭配 Exemplar 在 Metrics 上紀錄 Trace ID，連結 Metrics 與 Traces

Demo - 資訊交互應用 - Traces to Logs

The image displays two side-by-side dashboard panels from the Grafana ecosystem, specifically the Tempo and Loki components.

Tempo Panel (Left):

- Query type:** Set to 'TraceQL'. A red box highlights the query `6753d6cd43be8ac9ffa5035bec5625aa`.
- Trace View:** Shows a trace for `app-a: GET /chain` with a duration of 1.1s. A timeline at the top shows the trace's progress. Below the timeline, a table lists spans. A yellow box highlights the 'Log' icon in the first span's row.
- Span Details:** A yellow arrow points from the 'Log' icon to a 'Logs for this span' button.

Loki Panel (Right):

- Query type:** Set to 'Label browser'. A red box highlights the query `{compose_service='app-a'} |="6753d6cd43be8ac9ffa5035bec5625aa"`.
- Logs volume:** Shows a list of logs. A yellow arrow points from the 'Log' icon in the Tempo panel to the first log entry in the Loki panel.

Annotations:

- A red arrow points from the highlighted query in the Tempo panel to the highlighted query in the Loki panel.
- A yellow arrow points from the 'Log' icon in the Tempo panel to the 'Logs for this span' button.

點擊 Span 右側的 Log ICON 可查詢對應 Log

Demo - 資訊交互應用 - Logs to Traces

Home > Explore > Loki | Tempo

Loki

Add to dashboard

< > 🔍 ↺

Logs

Time Unique labels Wrap lines Prettify JSON

Deduplication None Exact Numbers Signature

Display results Newest first Oldest first

Common labels: compose_project=pycon-tw-2024-p... compose_service=app-a container_name=pycon-tw-2024-py... host=docker-desktop service_name=pycon-tw-2024-pyth...

Download

2024-09-18 14:44:36,538 INFO [uvicorn.access] [h11_impl.py:476] [trace_id=8de000c7316385c21e38ca4a7fc7a span_id=817616b39feb6e0f resource.service.name=app-a] - 192.168.65.1:54016 - "GET /favicon.ico HTTP/1.1" 404

Fields

compose_project	pycon-tw-2024-python-observability
compose_service	app-a
container_name	pycon-tw-2024-python-observability-app-a-1
detected_level	info
filename	/var/log/docker/f0d8d3200011fd4c9c9250b42d9cd1abd54e3fea1cdf3b59847b68679a0cbfa5/json.log
host	docker-desktop
service_name	pycon-tw-2024-python-observability-app-a-1
source	stdout

Links

TraceID

8de000c7316385c21efe538ca4a7fc7a

Tempo

Click

Tempo

Add to dashboard

< > 🔍 ↺

A (Tempo)

Query type Search TraceQL Service Graph Import trace

Build complex queries using TraceQL to select a list of traces. Documentation

8de000c7316385c21efe538ca4a7fc7a

> Options Limit: 20 Spans Limit: 3 Table Format: Traces Step: auto Streaming: Disabled

+ Add query Query history Query inspector

Node graph

Trace

app-a: GET 3.95ms

2024-09-18 22:44:36.536 GET 404 /favicon.ico

Span Filters

4 spans

Prev Next

Service & Operation

app-a GET (3.95ms)

GET http send (416.04µs)

GET http send (309.17µs)

GET http send (75µs)

展開 Log 明細點擊連結可查看該筆 Trace

57

Recap

Recap

- 可觀測性
 - 系統可被觀測或測量的一種特性，反映我們對系統的掌握度
 - 評量標準：透過各種**資訊**，**清楚**了解系統狀態
- 擴充**可觀測性資訊**
 - Metrics、Logs、Traces
 - 生成、收集、儲存、使用
- 單一平台檢視，概覽系統現狀
 - 於 Grafana **統一檢視**，資訊建立關聯，交互應用、**產生綜效**

延伸閱讀

- [Observability 101：從零開始了解可觀測性：可觀測性基礎介紹](#)
- [從零開始打造可觀測性平台：可觀測性平台 Workshop 簡報與 Lab](#)
- iThome 鐵人賽
 - [時光之鏡：透視過去、現在與未來的 Observability：Observability 概念、工具與應用介紹](#)
 - [淺談DevOps與Observability：Observability 工具與概念介紹](#)
 - [Grafana Zero to Hero：Grafana 應用與管理介紹（連載中）](#)

延伸閱讀

- Demo Project
 - FastAPI Observability
 - FastAPI Tracing with Jaeger through OpenTelemetry
 - OpenTelemetry Application Performance Management
- Prometheus: The Documentary - Prometheus 紀錄片
- The Grafana documentary: The first 10 years - Grafana 紀錄片

工商時間

Berry AI

www.berry-ai.com

Berry AI

爆單成長中

歡迎各路好手毛遂自薦



徵同事！
歡迎過來聊聊



全面介紹 Grafana、Prometheus、OpenTelemetry！

可觀測性入門指南

Metrics、Logs、Traces 三大實戰應用，
用 24 個 Lab 鞏固你的 SRE 技能樹

劉義瑋 (Blueswen) 著



實現全面系統監控和分析！

從零開始，構建強大的可觀測性架構

概念導覽
深入淺出解析
可觀測性概念

完整範例
使用 Docker Compose
展示多種實戰範例

多處適用
Kubernetes 與
傳統服務皆適用

資訊整合
結合 Metrics、Logs
和 Traces 全面洞察



iThome
鐵人賽

天瓏網路書店



可觀測性概念、工具介紹