

目錄

- 1. [概述](#)
- 2. [更新日誌](#)
- 3. [背景知識](#)
- 4. [ELK Stack 簡介](#)
- 5. [Grafana Loki Stack 簡介](#)
- 6. [簡易比較](#)
- 7. [安裝 ELK stack 以及 Loki \(based on docker\)](#)
 - 7.1. [如何管理以及讓各個container互相通訊？](#)
 - 7.2. [元件config設定](#)
 - 7.2.1. [promtail](#)
 - 7.2.2. [loki](#)
 - 7.2.3. [filebeat](#)
 - 7.2.4. [logstash](#)
 - 7.3. [Grafana 以及 Kibana](#)
 - 7.3.1. [Grafana](#)
- 8. [kibana](#)
 - 8.1. [比較](#)
 - 8.1.1. [filter & search](#)
 - 8.1.2. [performance](#)

Log System 比較

1. 概述

1. 本篇將比較 ELK 與 Grafana Loki 兩套Log框架，並決定最後是否使用ELK
2. 本篇包含安裝以及相關資訊的整理以及網路上別人整理的比較

2. 更新日誌

2023/09/22 第一版 by Bing

3. 背景知識

由於東購2.0會架構在RedHat OpenShift(OPC)的平台上，OPC預計未來將不會支援ELK(EFK) Stack，因此目前須評估是否要額外加裝ELK的stack。評估的點為以下幾點

1. 是否有需求是只能透過ELK完成
2. 效能的比較
3. 技術以外的考量

4. ELK Stack 簡介

1. ELK Stack:

- Elasticsearch: 是一個實時分佈式搜索和分析引擎。它主要用於索引、搜尋和分析大量的數據快速地。在這個堆疊中，它主要用作日誌數據的存儲和搜索後端。
- Logstash: 是一個靈活的日誌收集、處理和轉發的工具。它可以接收從不同的源來的數據，加工這些數據，然後將其發送到像Elasticsearch這樣的存儲後端。
- Kibana: 是一個與Elasticsearch集成的視覺化和探索工具。它允許用戶建立儀表板來展示和分析在Elasticsearch中存儲的數據。

基於以上的特性因此在log的儲存分析上常使用ELK來建構log系統，但ELK能做的事情並非僅有log分析，ELK可應用在其他更多的資料分析上。

5. Grafana Loki Stack 簡介

- Prometheus: 是一個開源的監控和警報工具套件，主要用於度量和警報，而不是日誌管理。
- Loki: 一個由Grafana Labs開發的日誌聚合系統，與Grafana深度集成，設計上與Prometheus有許多相似之處。
- Grafana: 是一個用於時序數據的視覺化工具，它原生支持Prometheus和Loki，以及其他數據源，如Elasticsearch。

相較於ELK，Loki則主要是針對log所設計的儲存分析系統，主要訴求為輕量以及較不佔空間，搭配Grafana的儀表板有極好的表現。

6. 簡易比較

ELK因透過ElasticSearch進行儲存以及搜尋，擁有像是全文檢索的功能以及效率，缺點的話為很吃記憶體以及硬碟容量。

Loki為針對log所設計的儲存分析系統，雖然並無強調全文檢索的能力，但透過適當的Label方式並搭配regular expression也可有效的達到全文檢索的效果。

實測的結果顯示ELK擅長的為需要做Aggregation的Query，Loki因無對log content做index 因此這類型的query效能較差。但

7. 安裝 ELK stack 以及 Loki (based on docker)

首先先安裝docker完後可分別下載下列Image

- elastic/filebeat (file monitor)
- logstash (log pipeline)
- elasticsearch (storage/search)
- kibana (dashboard)
- grafana/promtail (file monitor)
- grafana/loki (storage/search)
- grafana/grafana (dashboard)

Images

[Give feedback](#)

Local

Hub

Artifactory

EARLY ACCESS

3.59 GB / 3.59 GB in use

7 images

Last refresh: 21 hours a

Q Search

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	elastic/filebeat dea1f52579ff	8.10.1	In use	6 days ago	318.69 MB	▶ ⋮
<input type="checkbox"/>	grafana/grafana 823b1c07a161	latest	In use	6 days ago	391.25 MB	▶ ⋮
<input type="checkbox"/>	logstash 13e133ae1d4d	8.9.2	In use	21 days ago	762.44 MB	▶ ⋮
<input type="checkbox"/>	elasticsearch 66c29cde15ce	7.16.2	In use	2 years ago	646.34 MB	▶ ⋮
<input type="checkbox"/>	kibana 8c46ec23123e	7.16.2	In use	2 years ago	1.29 GB	▶ ⋮
<input type="checkbox"/>	grafana/promtail 3635cc7fbc8d	latest	In use	5 days ago	195.3 MB	▶ ⋮
<input type="checkbox"/>	grafana/loki a63be545e811	latest	In use	5 days ago	67.45 MB	▶ ⋮
<input type="checkbox"/>	elastic/filebeat dea1f52579ff	8.10.1	In use	6 days ago	318.69 MB	▶ ⋮

7.1. 如何管理以及讓各個container互相通訊？

由於各個container之間彼此是互相獨立的，需要互相通訊時我們可以透過 `docker-compose` 的指令

首先建立一個File叫做 `docker-compose.yml`

名稱	修改日期	類型	大小
 <code>docker-compose.yml</code>	2023/9/19 下午 05:40	YML 檔案	1 KB

接著輸入以下內容

```
services:
  loki:
    image: grafana/loki
    ports:
      - 3100:3100
  promtail:
    image: grafana/promtail
    volumes:
      - 'C:/Users/b2badmin/AppData/Local/Docker/log/vm:/data/db' # <- extra
```

```
space here
grafana:
  image: grafana/grafana
  ports:
    - 3000:3000
```

services: 代表要啟用一個由下方images所組成的service

loki: service名稱，image為要啟用哪個image

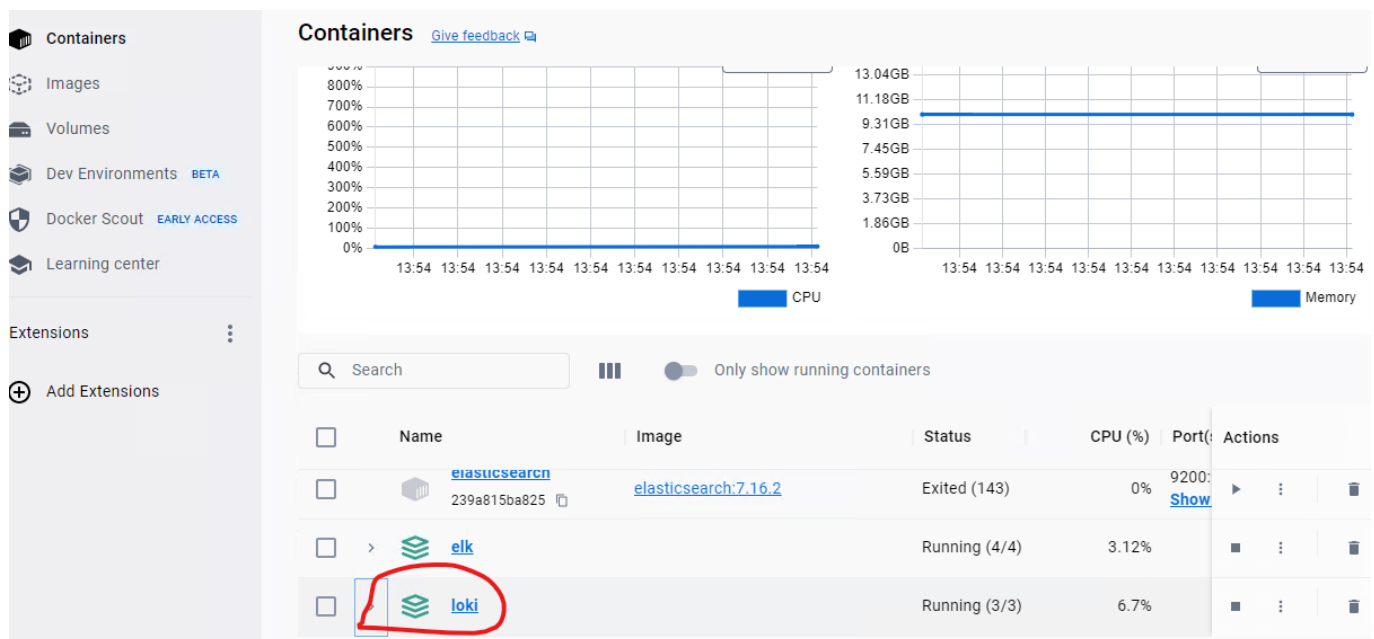
ports為要export的port

volumes為將硬碟資料夾對應到docker裡面的資料夾，在此用來監控本機log

儲存完後接著開啟 **cmd** 進入該資料夾後輸入 **docker-compose up**

```
C:\Users\b2badmin\Desktop\ELK>
C:\Users\b2badmin\Desktop\ELK>docker-compose up
```

執行後應該可以看到docker-desktop看到container的地方有個名為**loki**的container的群組



用同樣方式建立ELK的群組

以下為ELK的**docker-compose**檔案內容


```
version: "3.8"
services:
  elasticsearch:
    image: elasticsearch:7.16.2
    environment:
      - discovery.type=single-node
    ports:
      - 9200:9200
```

```
- 9300:9300
filebeat:
  image: elastic/filebeat:8.10.1
  volumes:
    - 'C:/Users/b2badmin/Documents:/data/db' # <- extra space here
logstash:
  image: logstash:8.9.2
  ports:
    - 5044:5044
    - 9600:9600
kibana:
  image: kibana:7.16.2
  ports:
    - 5601:5601
```

7.2. 元件config設定

7.2.1. promtail

loki-promtail-1

 [grafana/promtail](#)

f5d90950d427

STATUS
Running (2 days ago)

Logs







Inspect

Bind mounts

Terminal

Files

Stats

Name ↑	Note	Size	Last r
 passwd-		922 Bytes	18 day
 profile		769 Bytes	2 year
>  profile.d			6 mon
▼  promtail	MODIFIED		2 days
 config.yml	MODIFIED	299 Bytes	2 days
>  rc0.d			18 day

/etc/promtail/config.yml

YAML

```
1 server:
2   http_listen_port: 9080
3   grpc_listen_port: 0
4
5 positions:
6   filename: /tmp/positions.yaml
7
```

promtail的config檔案預設路徑為 `/etc/promtail/config.yam`

以下為config內容

```
server:
  http_listen_port: 9080
```

```
grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://loki:3100/loki/api/v1/push

scrape_configs:
  - job_name: system
    static_configs:
      - targets:
          - localhost
        labels:
          job: varlogs
          __path__: /data/db/*.log
```

promtail需要指名要monitor資料夾，這邊監測/data/db/下所有.log的檔案

值得注意的是<http://loki:3100/loki/api/v1/push>這行 由於在docker裡面的IP是浮動的且每次重啟都會不一樣，因此可以直接寫container的名稱當作該container的DNS

7.2.2. loki

以下為loki config檔的內容，預設路徑為/etc/loki/loki-local.yml

auth_enabled 為是否要進行驗證，可先行關閉 server 為設定server模式相關的config

```
auth_enabled: false

server:
  http_listen_port: 3100

common:
  path_prefix: /loki
  storage:
    filesystem:
      chunks_directory: /loki/chunks
      rules_directory: /loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory

schema_config:
  configs:
    - from: 2020-10-24
      store: boltdb-shipper
      object_store: filesystem
      schema: v11
      index:
```

```
    prefix: index_  
    period: 24h  
  
ruler:  
  alertmanager_url: http://localhost:9093
```

7.2.3. filebeat

以下為filebeat config檔的內容，預設路徑為/etc/filebeat/filebeat.yml

output.logstash代表輸出至logstash，可依照需求修改輸出至其他地方

```
filebeat.config:  
  modules:  
    path: ${path.config}/modules.d/*.yaml  
    reload.enabled: false  
  
filebeat.inputs:  
- type: log  
  enabled: true  
  paths:  
    - /data/db/*.log  
  
processors:  
- add_cloud_metadata: ~  
- add_docker_metadata: ~  
  
output.logstash:  
  hosts: "logstash:5044"
```

以下是logstash config，指定輸出至本機的elasticsearch

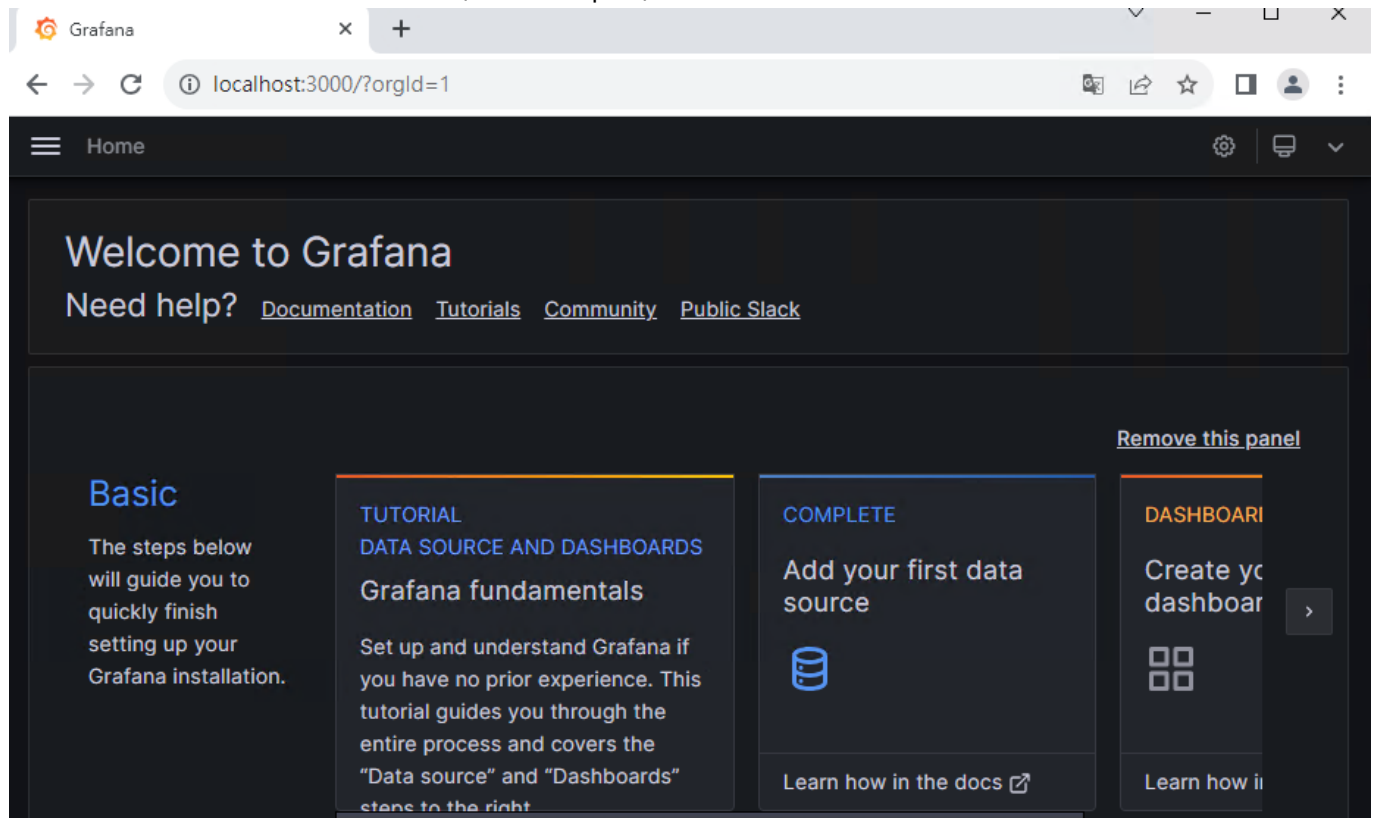
7.2.4. logstash

```
input {  
  beats {  
    port => 5044  
  }  
}  
  
output {  
  elasticsearch {  
    hosts => ["http://elasticsearch:9200"]  
    index => "%{[@metadata][beat]}-%{[@metadata][version]}"  
    action => "create"  
  }  
}
```

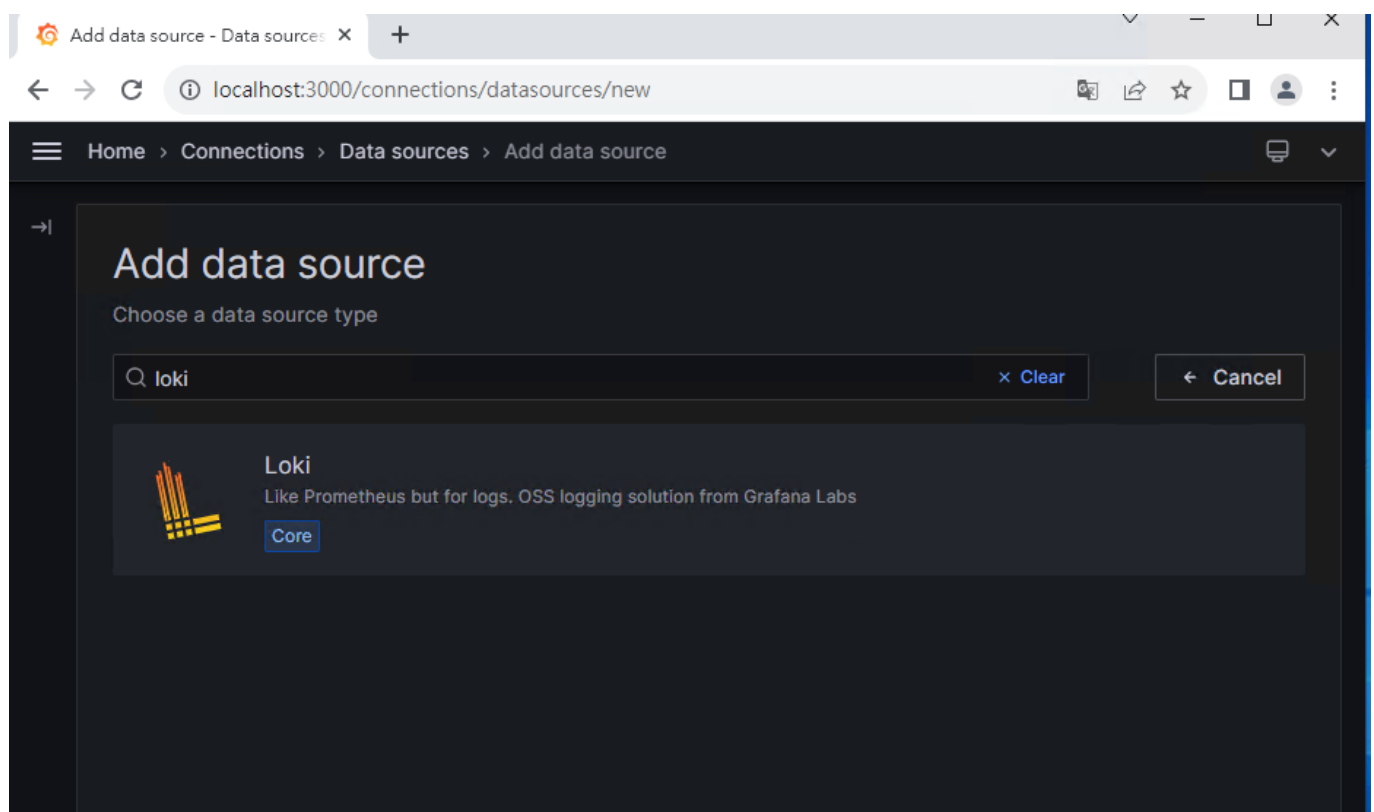
7.3. Grafana 以及 Kibana

7.3.1. Grafana

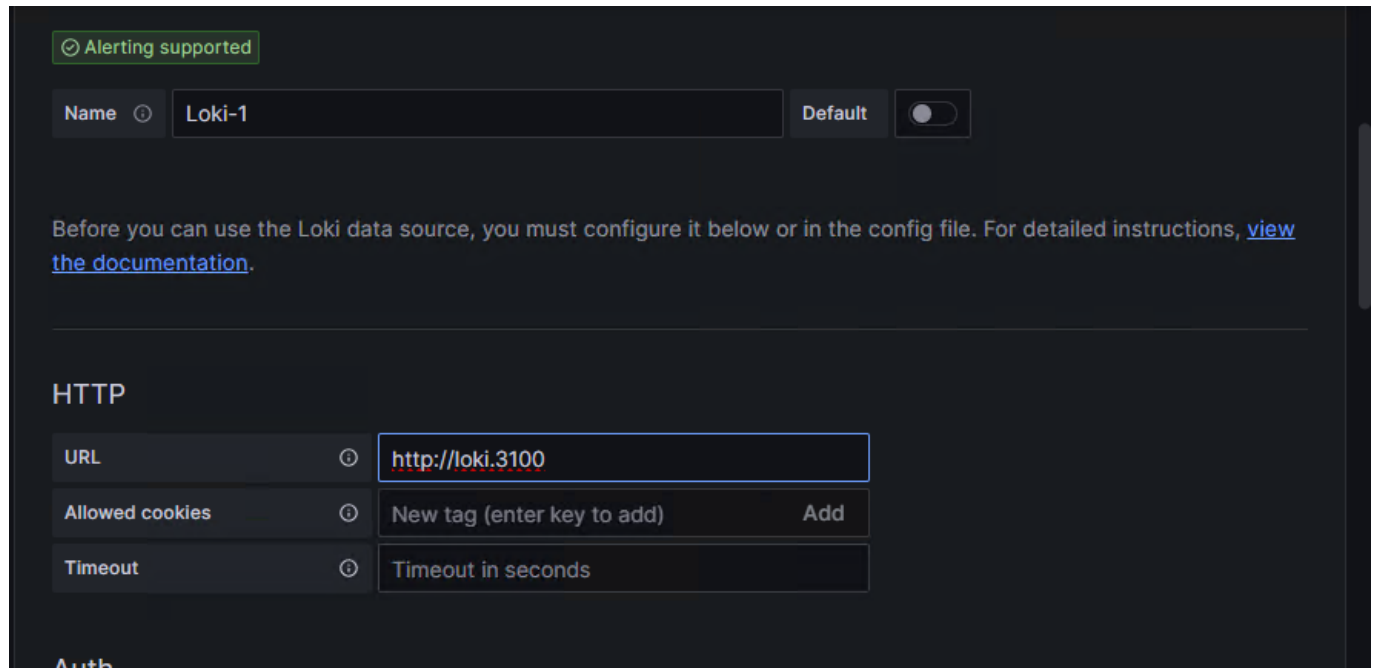
安裝完Grafana後打開localhost:3000 (預設3000 port)



接著首先需要新增Data Source 點選首頁的Add your first data source之後在type選擇Loki

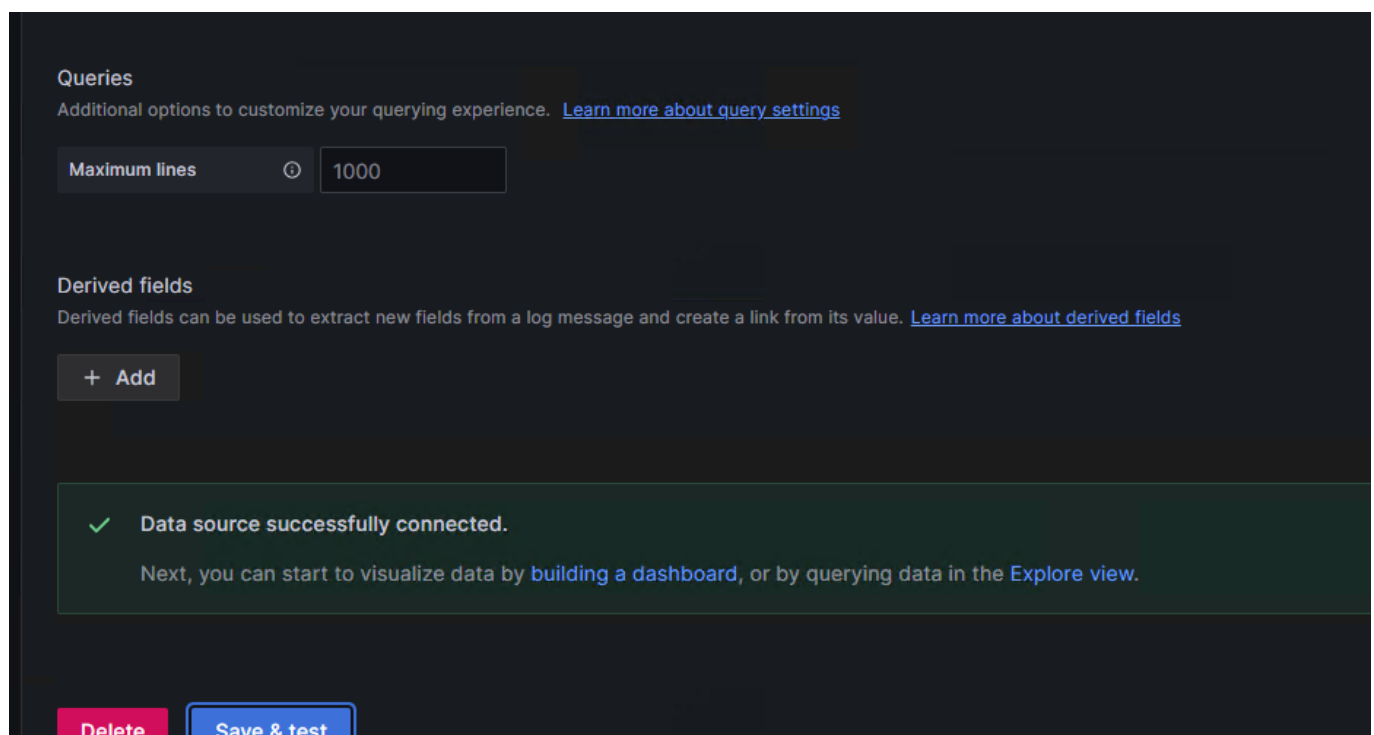


選完後再下方的URL那邊輸入 `http://loki:3100/` 可直接使用 `loki` 做為DNS name



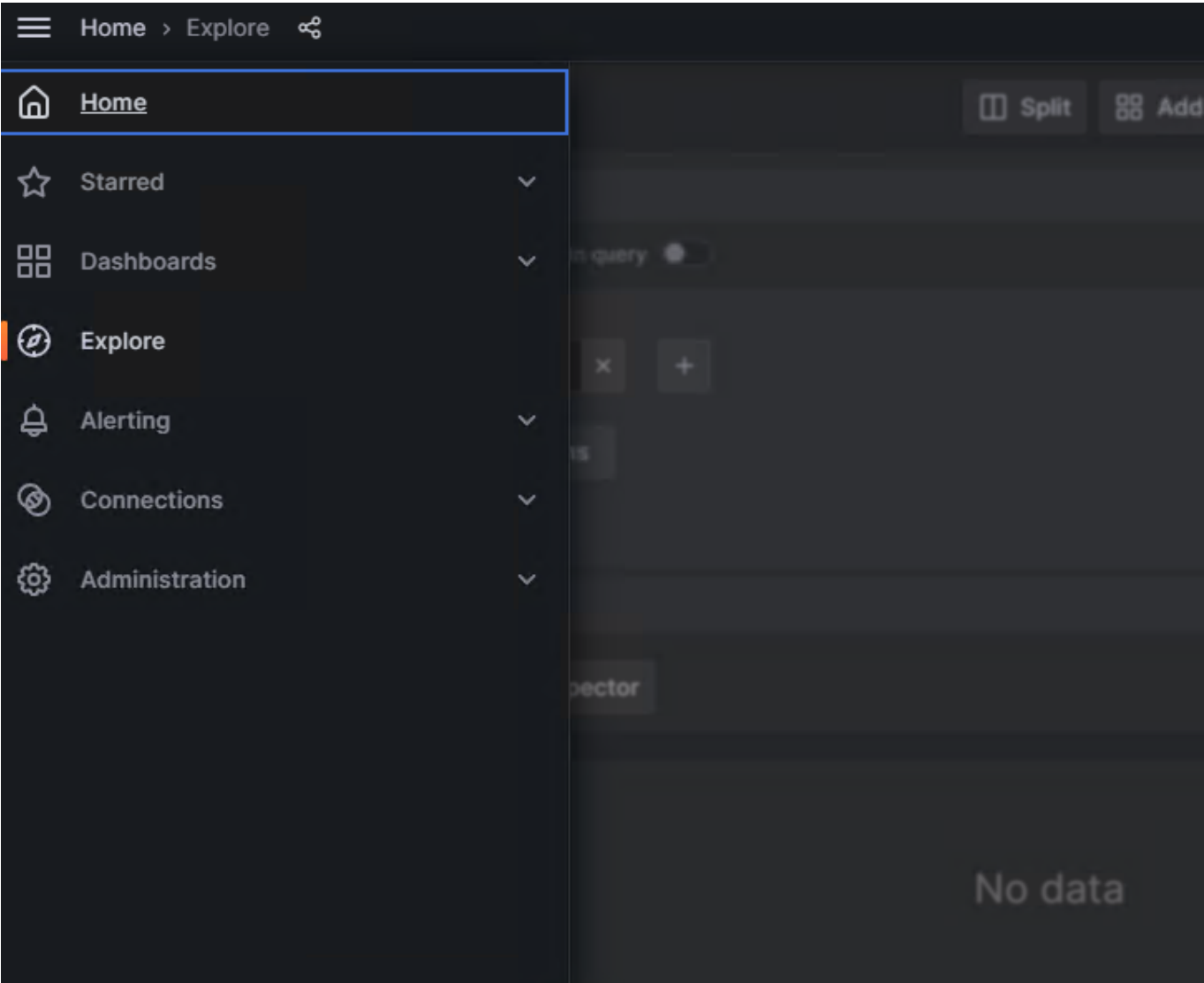
The screenshot shows the configuration page for a new Loki data source. At the top, there's a green status indicator that says "Alerting supported". Below it, the "Name" field is set to "Loki-1" and the "Default" toggle is turned off. A message states: "Before you can use the Loki data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#)." Under the "HTTP" section, the "URL" field contains "http://loki.3100". The "Allowed cookies" field has a placeholder "New tag (enter key to add)" and an "Add" button. The "Timeout" field has a placeholder "Timeout in seconds". The "Auth" section is partially visible at the bottom.

點選下方的 `Save & test` 確認是否能新增成功，若有成功會看到下方的畫面

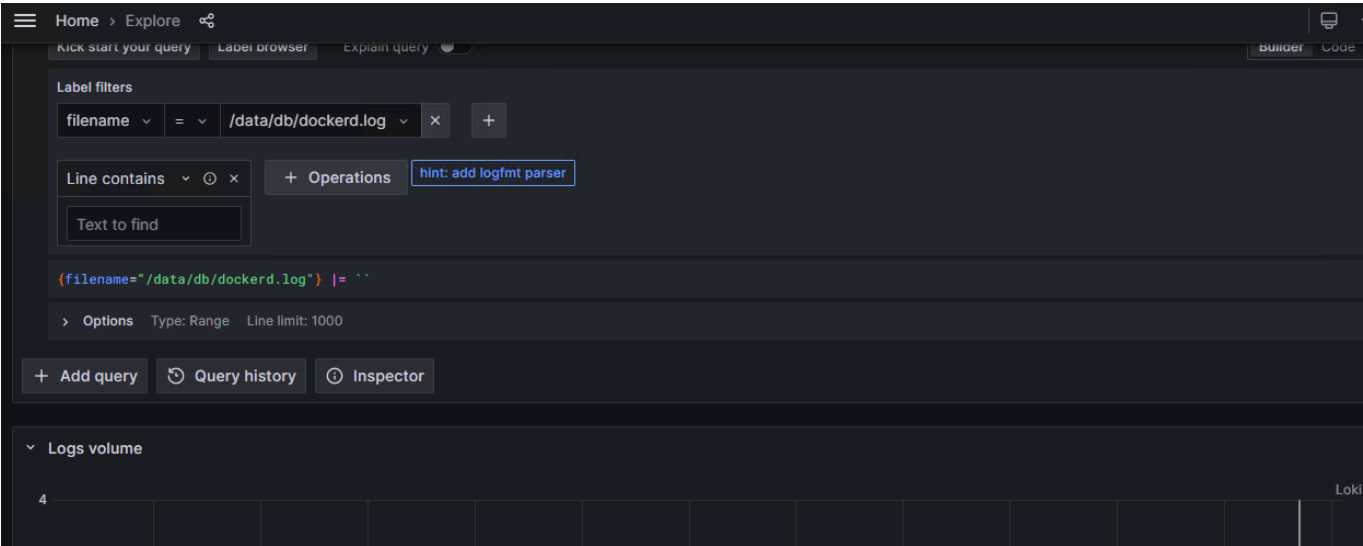


The screenshot shows the "Queries" section of the Grafana interface. It includes a "Maximum lines" setting set to 1000. Below that, the "Derived fields" section has a "+ Add" button. A large green success message states: "Data source successfully connected. Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#)." At the bottom, there are two buttons: "Delete" (in red) and "Save & test" (in blue).

接著可點選左方的 `Explore` 確認logstash有沒有順利接收到log



在label區選擇filename 並且選擇我們所監控的檔案名，按下右邊的 `Run Query`

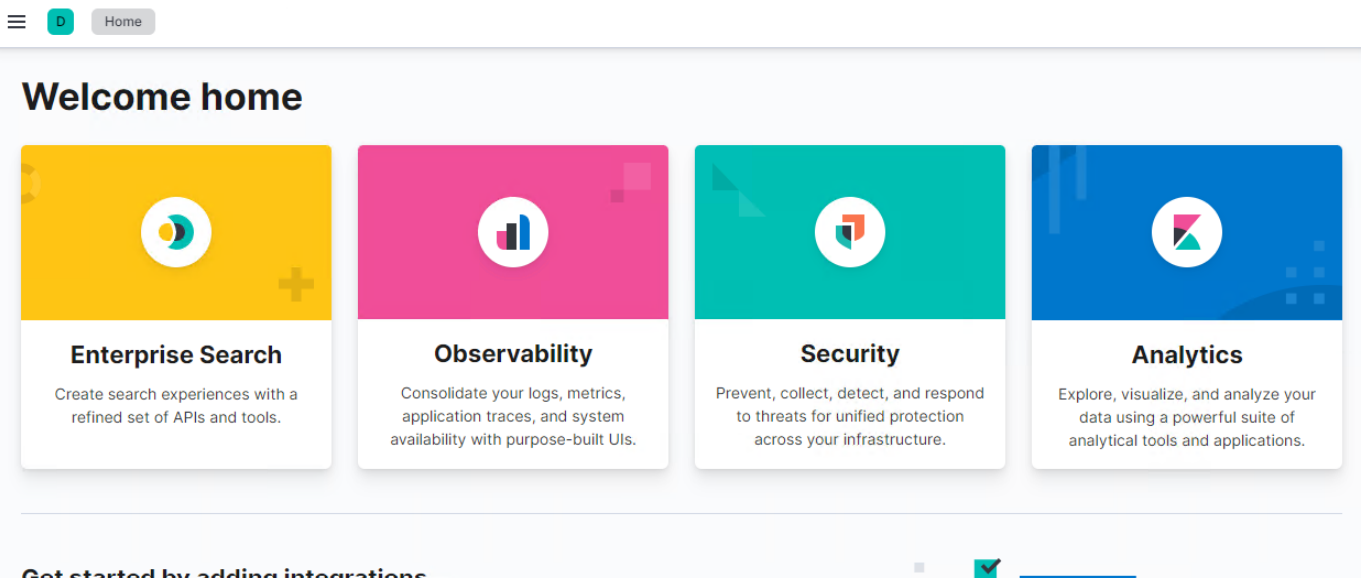


可成功看到下方有log出現

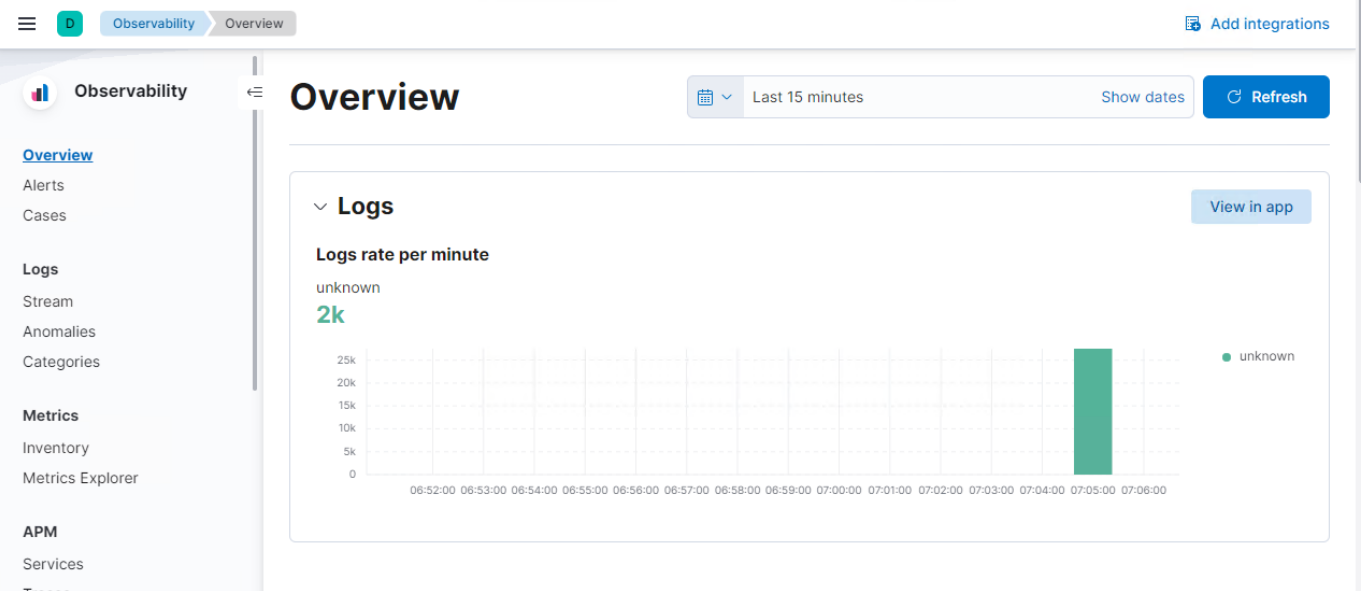


8. kibana

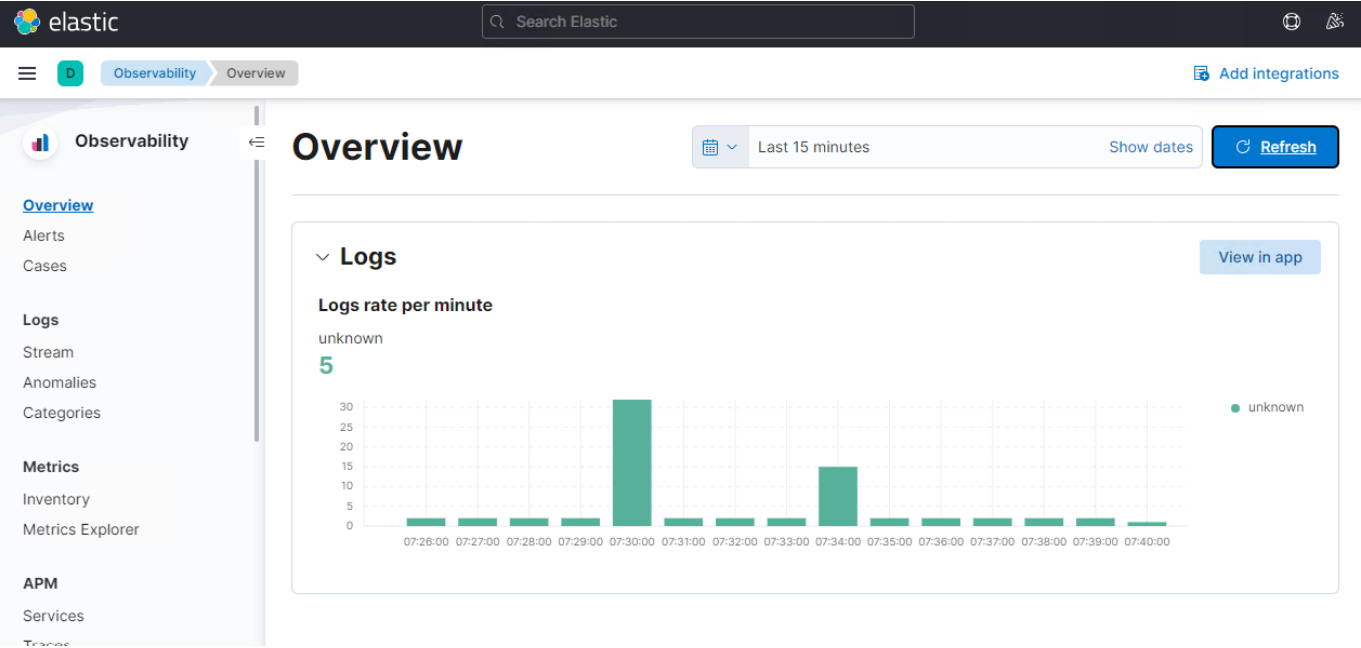
開啟kibana的畫面後可以選Observability



左方有個 log 可點選



按下去之後若有成功從 elasticsearch 顯示出 logs



The screenshot shows the Elastic Observability 'Stream' page. The left sidebar has 'Stream' selected. The main area contains a table of log entries with columns for date, dataset, and message. A search bar at the top allows filtering log entries. The table shows several log entries from 'event.dataset' on 'Sep 23, 2023'.

Sep 23, 2023	event.dataset	Message
		1100328Z][lifecycle-server.apiproxy][I] proxy << HEAD /_ping (400.198μs)
15:15:25.780		[2023-09-23T07:15:18.757685851Z][02-docker][I] [2023-09-23T07:15:18.757598452Z][lifecycle-server.apiproxy][I] Upgrading to raw stream
15:15:25.780		[2023-09-23T07:15:18.758752044Z][02-docker][I] [2023-09-23T07:15:18.758614845Z][lifecycle-server.apiproxy][I] proxy << POST /grpc (1.539189ms)
15:15:25.780		[2023-09-23T07:15:18.758771344Z][02-docker][I] [2023-09-23T07:15:18.758617545Z][lifecycle-server.apiproxy][I] proxy << POST /grpc (1.42709ms)
15:15:25.780		[2023-09-23T07:15:18.755860564Z][02-docker][I] [2023-09-23T07:15:18.755739065Z][lifecycle-server.apiproxy][I] proxy << HEAD /_ping (464.697μs)

可以下 filter 去過濾log

Stream

⚙ Customize 📌 Highlights
📅 Last 1 day Show dates
▶ Stream live

Sep 23, 2023	event.dataset	Message
Showing entries from Sep 23, 15:05:11		
15:05:11.169		[2023-09-23T07:03:58.389681606Z][02-docker][I] [2023-09-23T07:03:58.389577708Z][lifecycle-server.apiproxy][I] proxy << GET /v1.43/container s/231fc6e812c4316f80dfce3d76ccb3f163938eb7169c2f76008fd53cf1e40e/jso n (615.189µs)
15:05:11.169		[2023-09-23T07:03:57.149520867Z][02-docker][I] [2023-09-23T07:03:57.149302771Z][lifecycle-server.apiproxy][I] proxy >> GET /containers/231fc 6e812c4316f80dfce3d76ccb3f163938eb7169c2f76008fd53cf1e40e/changes
15:05:11.169		[2023-09-23T07:03:58.483979641Z][02-docker][I] [2023-09-23T07:03:58.48

8.1. 比較

8.1.1. filter & search

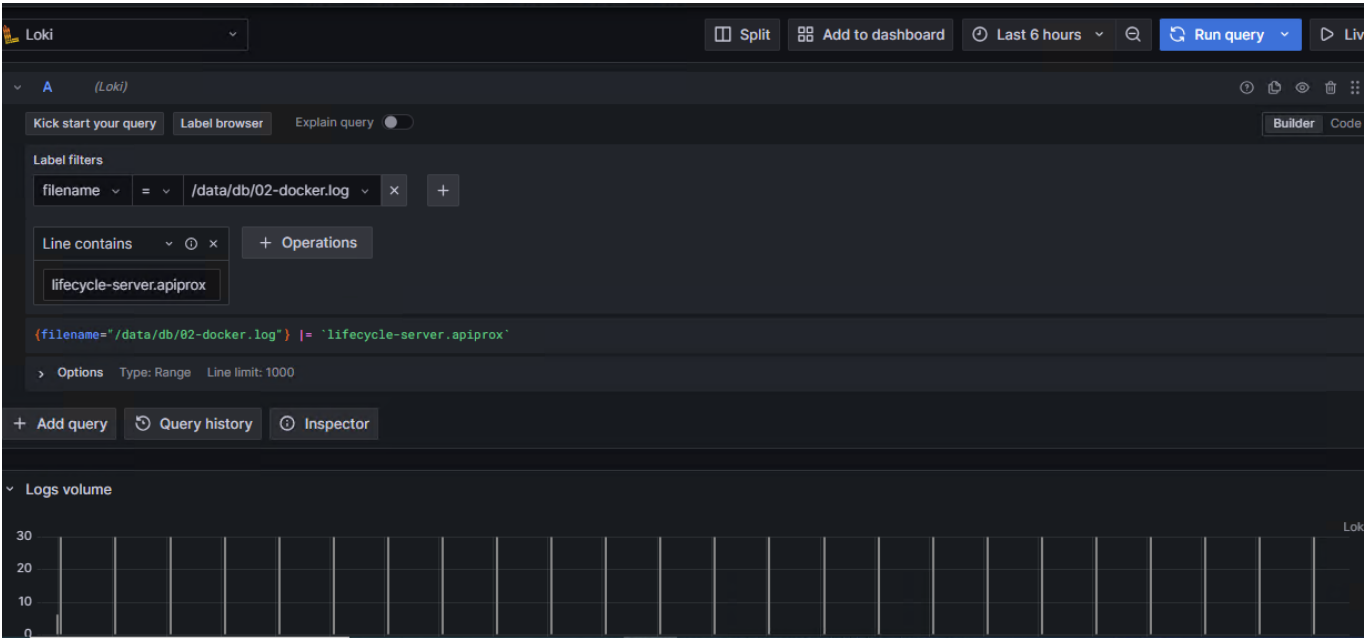
基本的搜尋測試，ELK可直接輸入所想要的關鍵字，並過濾出含有關鍵字的log

Customize
Highlights

▼
Last 1 day
Show dates
Stream live

Sep 25, 2023	event.dataset	Message
Showing entries from Sep 25, 08:30:25		
08:30:25.797		[2023-09-25T00:30:17.251458104Z][02-docker][I] [2023-09-25T00:30:17.251341405Z][lifecycle-server.apiproxy][I] proxy >> POST /grpc
08:30:25.797		[2023-09-25T00:30:17.274987155Z][02-docker][I] [2023-09-25T00:30:17.274869056Z][lifecycle-server.apiproxy][I] Upgrading to raw stream
08:30:25.797		[2023-09-25T00:30:17.251821900Z][02-docker][I] [2023-09-25T00:30:17.251731101Z][lifecycle-server.apiproxy][I] Upgrading to raw stream
08:30:25.797		[2023-09-25T00:30:17.252836989Z][02-docker][I] [2023-09-25T00:30:17.252744502Z][lifecycle-server.apiproxy][I] proxy >> POST /grpc

Loki則需要先選擇label，這邊選擇的是filename，可依需求改成service等方式達到類似全文檢索的功能。選擇完label後下方的operation可以選擇line contain代表要過濾所包含的關鍵字，此時輸入完關鍵字後即可過濾出包含關鍵字的log。



8.1.2. performance

以下資訊引用自 [log solution comparison](#)

此比較為輸入大量的log資訊後各log系統所使用的disk size 以及 query long-term 的資訊時的比較。
ingestion time elapsed 代表輸入資料時所耗費的時間

Consumed Index disk size 代表用來index的硬碟空間

Query 1 month (12/12h interval) query 一個月的時間並以12小時為Interval所花費的時間

Query 1 month (30s interval) query 一個月的時間並以30秒為Interval所花費的時間

Query 1 day (12/12 interval) query 一天的時間並以12秒為Interval所花費的時間

Query 1 day (30s interval) query 一個月的時間並以30秒為Interval所花費的時間

	ELK	Splunk	Loki
Ingestion time elapsed	~ 10hours	~ 2h20minutes	~6hours
Consumed indexed disk size	~ 107Gb	~ 40Gb	~9Gb
Query 1 month (12/12h interval)	4mins	1h30min	27mins
Query 1 month (30s interval)	✗	✗	27mins
Query 1 day (12/12interval)	few secs	4mins	1min
Query 1 day (30s interval)	✗	✗	1min

可以看出以硬碟的使用量來看ELK使用了107GB，而Loki使用了約9GB。以query時間來看ELK在Query 一個月時若interval為12小時，則時間為4分鐘，但loki需要27分鐘。

紅色打X的部分代表不支援此功能

若query為一天的話ELK只需要幾秒的時間，而loki則需要一分鐘。

由以上的數據可以得知，ELK擅長的是需要做aggregation的query，而Loki則不擅於此。但若單純的query filter兩者應該是不會差太多Loki支援的interval相對較為有彈性。

小結

1. 目前在使用上單純做Filter的話Loki並無明顯感受到效能上的問題，雖說ELK有提供全文檢索而Loki並無標榜此功能，但透過Loki的meta index以及加上regular expression的功能也可達到類似的功能，應能應付大部分的搜尋需求。
2. 若query時需要額外的aggregation或其他複雜的功能，則ELK的效能表現更為突出。
3. 考慮到東購2.0的架構，若需要額外加裝ELK，等於額外要申請VM並安裝ELK且在其他vm上還需將log導流到ELK vm上，也可能會多出技術以外的effort（與其他單位的negotiation）

回到開頭的問題

是否有需求是只能透過ELK完成

A:若需要針對log進行aggregation則ELK的表現較佳，單純的filter query Loki效能尚在可接受範圍。但ELK所占用的硬碟容量約為Loki的十倍。

技術以外的考量

A:會多出技術以外的effort (與其他單位的negotiation)

參考資料

[Loki](#)

[ELK](#)

[loki-vs-elasticsearch](#)

[log solution comparison](#)