



# Prometheus Exporter CSV 解析



## ### Prometheus Exporter 技術文件

### #### 1. 簡介

本 Prometheus Exporter 會從 CSV 讀取 log 數據，解析 host, job\_name, 並動態偵測可能的標籤 (service\_name, container\_name, module\_name)。

根據不同的標籤組合，Exporter 會建立 \*\*3 種不同的 metrics\*\*：

- log\_host\_job\_basic：僅包含 host 和 job\_name
- log\_host\_job\_service：包含 service\_name, container\_name
- log\_host\_job\_module：包含 module\_name

### #### 2. CSV 檔案格式

示例 (bak-data\_collect-1.csv)：

```
host_1,job_A, {service_name="aaa",  
container_name="bbbb"}  
host_1,job_A  
host_1,job_B  
host_2,job_A  
host_2,job_C  
host_3,job_B, {module_name="cbbb"}  
host_3,job_B  
host_3,job_B
```

### #### 3. 運作流程

1. \*\*讀取 CSV\*\*，解析 host 和 job\_name
2. \*\*動態偵測額外標籤 (service\_name, container\_name, module\_name)\*\*
3. \*\*計算 metrics 數值\*\*
4. \*\*將 metrics 更新到 Prometheus\*\*

### #### 4. Prometheus 指標計算





```
python
counts_basic = {}
counts_service = {}
counts_module = {}

for host, job, extra_labels in log_data:
    # 基本 `metrics`
    basic_key = (host, job)
    counts_basic[basic_key] =
counts_basic.get(basic_key, 0) + 1

    # 包含 `service_name, container_name` 的
    `metrics`
    if "service_name" in extra_labels or
"container_name" in extra_labels:
        service_key = (host, job,
frozenset({k: v for k, v in
extra_labels.items() if k in ["service_name",
"container_name"]}.items()))
        counts_service[service_key] =
counts_service.get(service_key, 0) + 1

    # 包含 `module_name` 的 `metrics`
    if "module_name" in extra_labels:
        module_key = (host, job, frozenset({k:
v for k, v in extra_labels.items() if k ==
"module_name"}.items()))
        counts_module[module_key] =
counts_module.get(module_key, 0) + 1

#### 5. Prometheus 指標格式
# HELP log_host_job_service Count of
occurrences with service-related labels
# TYPE log_host_job_service gauge
log_host_job_service{container_name="bbbb",hos
t="host_1",job_name="job_A",service_name="aaa"
} 1.0

# HELP log_host_job_module Count of
```



```
occurrences with module-related labels
# TYPE log_host_job_module gauge
log_host_job_module{host="host_3",job_name="job_B",module_name="cbbb"} 1.0
```

#### #### 6. 主要技術概念

- **\*\*set\*\***：用來存 labels，確保不會重複。
- **\*\*dict\*\***：存 metrics 計數，使用 `.get()` 來累加數值。
- **\*\*frozenset\*\***：用來當 dict 的 key，讓 labels 組合成唯一標識。
- **\*\*print debug\*\***：觀察 metrics 計算過程，確保計算正確。

#### #### 7. Exporter 啟動方式

```
bash
python3 exporter.py
```

#### #### 8. 訪問 Prometheus Exporter

打開瀏覽器，訪問：

<http://localhost:8080/metrics>



推理關於Prometheus Exporter設計花了 1 秒

這份文件詳細介紹了一個基於 CSV 數據來源的 Prometheus Exporter，其核心目的是從 CSV 中讀取 log 數據，解析基本的 ``host`` 與 ``job_name``，並根據 log 中可能出現的額外標籤 (``service_name``, ``container_name``, ``module_name``) 來動態建立三類不同的 metrics。下面整理出文件的主要內容與技術重點：

## 1. 簡介



- **主要功能：**從 CSV 檔案中讀取並解析 log 數據。
- **基本元素：**
  - **必備欄位：**`host` 和 `job\_name`。
  - **動態標籤：**根據數據中是否存在 `service\_name`、`container\_name` 或 `module\_name`，Exporter 會額外建立對應的 metrics。
- **建立的 metrics 類型：**
  1. `log\_host\_job\_basic`：只包含 `host` 和 `job\_name` 的基本統計。
  2. `log\_host\_job\_service`：針對包含 `service\_name` 與 `container\_name` 的 log 數據進行統計。
  3. `log\_host\_job\_module`：針對包含 `module\_name` 的 log 數據進行統計。

---

## 2. CSV 檔案格式

文件中提供了一個示例 CSV 文件（例如 `bak-data\_collect-1.csv`），其中數據格式大致如下：

```
host_1,job_A, {service_name="aaa", container_name="bbbb"}
host_1,job_A
host_1,job_B
host_2,job_A
host_2,job_C
host_3,job_B, {module_name="cbbb"}
host_3,job_B
host_3,job_B
```

- 每一行至少包含 `host` 與 `job\_name`，而有些行會包含一段用大括號括起來的額外標籤字串，這些標籤需要進一步解析。



### 3. 運作流程

Exporter 的主要運作流程包括：

1. **讀取 CSV**：解析每一行中的 `host` 與 `job\_name` 資訊。
2. **動態偵測額外標籤**：檢查 log 中是否含有 `service\_name`、`container\_name` 或 `module\_name`，並根據實際存在的標籤決定進一步處理。
3. **計算 metrics 數值**：對不同的標籤組合進行計數統計。
4. **更新到 Prometheus**：最終將計算結果以 Prometheus 格式暴露出來。

### 4. Prometheus 指標計算

文件中提供了一段 Python 代碼作為指標計算的範例，流程大致如下：

```
counts_basic = {}
counts_service = {}
counts_module = {}

for host, job, extra_labels in log_data:
    # 基本 metrics
    basic_key = (host, job)
    counts_basic[basic_key] = counts_basic.get(basic_key, 0) + 1

    # 包含 service_name, container_name 的 metrics
    if "service_name" in extra_labels or "container_name" in extra_labels:
        service_key = (host, job, frozenset({k: v for k, v in
        extra_labels.items() if k in ["service_name", "container_name"]}.items()))
        counts_service[service_key] = counts_service.get(service_key, 0) + 1

    # 包含 module_name 的 metrics
    if "module_name" in extra_labels:
        module_key = (host, job, frozenset({k: v for k, v in
```



```
extra_labels.items() if k == "module_name"}.items()))
    counts_module[module_key] = counts_module.get(module_key, 0) + 1
```

- **基本計數 (`counts\_basic`)**：依據 (host, job) 組合累加出現次數。
- **服務相關計數 (`counts\_service`)**：當 log 包含 `service\_name` 或 `container\_name` 時，則使用一個 `frozenset` 將這些標籤組合起來，作為 dict 的 key，進行統計。
- **模組相關計數 (`counts\_module`)**：當 log 包含 `module\_name` 時，僅對該標籤進行統計。

使用 `frozenset` 來作為 key 的好處在於它能確保同樣的標籤組合只計算一次，避免重複。

## 5. Prometheus 指標格式

Exporter 最終會以 Prometheus 指標格式輸出數據，文件中提供了兩個示例：

```
# HELP log_host_job_service Count of occurrences with service-related labels
# TYPE log_host_job_service gauge
log_host_job_service{container_name="bbbb",host="host_1",job_name="job_A",service_name="aaa"} 1.0

# HELP log_host_job_module Count of occurrences with module-related labels
# TYPE log_host_job_module gauge
log_host_job_module{host="host_3",job_name="job_B",module_name="cbbb"} 1.0
```

- 每個指標均有 `HELP` 與 `TYPE` 說明。
- 指標名稱依據不同的標籤組合（例如 `service` 或 `module`）區分，並包含相關標籤在內。

## 6. 主要技術概念





- **set**：用於存放標籤，確保不會出現重複的情況。
- **dict**：用於儲存各指標的計數，利用 `.get()` 方法來累加計數值。
- **frozenset**：用作 dict 的 key，將標籤的組合轉化為不可變集合，保證 key 的唯一性與 hashability。
- **print debug**：在開發階段透過印出調試信息，方便確認 metrics 計算過程是否正確。

---

## 7. Exporter 啟動方式

啟動 Exporter 的命令十分簡單，透過以下命令執行：

```
python3 exporter.py
```

---

## 8. 訪問 Prometheus Exporter

Exporter 啟動後，Prometheus 指標會暴露在 HTTP 端點上，可以透過瀏覽器訪問以下 URL 來查看：

```
http://localhost:8080/metrics
```

這樣用戶或 Prometheus Server 就能夠獲取最新的 metrics 數據。

---

總結來說，這份技術文件詳細描述了如何根據 CSV 數據動態構建 Prometheus 指標，並涵蓋了從數據讀取、標籤解析、計數統計到最終數據暴露的整個流程。這種設計方式在處理動態且結構可能變化的 log 數據時，非常靈活且高效。



可以自依不同label組合產生不同metric嗎4:

```
import csv
import os
import time
import logging
from prometheus_client import Gauge,
start_http_server

# === [1] 設定 log 紀錄格式，方便我們觀察 metrics 的更
新狀況 ===
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(
message)s",
)

# === [2] 設定要讀取的 CSV 檔案 ===
CSV_FILE = "bak-data_collect.csv"

# === [3] 初始化 Prometheus 的 Gauge（我們稍後會根據
labels 來建立） ===
log_host_job_count = None # 先留空，等 labels 決定好再
建立

def parse_csv():
    """解析 CSV，動態收集所有出現過的 labels，並統計每組
labels 的出現次數"""
    counts = {} # key = frozenset(labels), value = 次數
    dynamic_labels = {"host", "job_name"} # 初始固定兩
個 labels

    if not os.path.exists(CSV_FILE):
        logging.error(f"CSV 檔案 {CSV_FILE} 不存在！")
        return counts, sorted(list(dynamic_labels))

    with open(CSV_FILE, 'r', encoding='utf-8') as f:
        reader = csv.reader(f)
```





```
for row in reader:
    if len(row) < 2:
        continue # 至少要有 host 和 job_name

    host = row[0].strip()
    job_name = row[1].strip()
    extra_labels = {} # 額外的 labels

    # === 處理第三欄之後的欄位：包含額外 labels
    ===
    for col in row[2:]:
        col = col.strip()

        if col.startswith("{") and col.endswith("}"):
            col = col[1:-1].strip() # 拿掉大括號

            # 拆解 key=value
            key_value_pairs = col.split(",")
            for pair in key_value_pairs:
                pair = pair.strip()
                if "=" in pair:
                    key, value = map(str.strip, pair.split("=",
1))
                    key = key.replace("{", "").replace("}",
"".strip()
                    value = value.replace("'",
"".replace("'", "").strip("\\")

                    if key and value:
                        extra_labels[key] = value # 收集有效
label

    # 更新動態 labels (只加入 key, 不重複)
    dynamic_labels.update(extra_labels.keys())

    # 組合完整的 labels
    full_label_dict = {**extra_labels, "host": host,
"job_name": job_name}
    key = frozenset(full_label_dict.items())
    counts[key] = counts.get(key, 0) + 1
```



```
logging.info(f"[parse_csv] 最終 labels:
{sorted(dynamic_labels)}")
return counts, sorted(dynamic_labels)

def update_metrics():
    """每次更新 metrics (Prometheus 會來抓)，我們就重建
    一次"""
    global log_host_job_count
    counts, labels_list = parse_csv()

    # 初始化或重新初始化 Gauge
    log_host_job_count = Gauge("log_host_job_count",
    "Count of host/job_name with optional labels",
    labels_list)

    log_host_job_count._metrics.clear() # 清掉舊資料
    logging.info("[update_metrics] 開始設定 metrics...")

    for key, count in counts.items():
        labels_dict = dict(key)
        complete_labels = {k: labels_dict.get(k, "") for k in
        labels_list} # 確保所有 label key 都存在
        logging.info(f"[metric] {complete_labels} =>
        {count}")

    log_host_job_count.labels(**complete_labels).set(count
    )

if __name__ == "__main__":
    start_http_server(8080)
    logging.info("Exporter 正在
    http://localhost:8080/metrics 執行...")

    while True:
        update_metrics()
        time.sleep(10) # 每 10 秒更新一次
```



你現在的程式邏輯是「所有資料的 label 組合都混在一個 metric `log_host_job_count` 裡」，即使 `label` 組合不同，也都放進同一個 metric，只是 label 欄位變多變少而已。

這在 Prometheus 的世界裡是不建議的，因為：

### ❗ 問題點：

#### 1. 同一個 metric 名稱應該有固定 label 組合 (label keys)

Prometheus 會假設一個 metric 的 label 組合是固定的，否則會造成查詢、顯示與效率問題。

#### 2. 目前程式會每次重新註冊 Gauge，這樣等於是重建 metric，這也不符合 prometheus 的常態運作方式。

### ✅ 想要的效果：

你應該根據「不同的 label 組合」產生「不同的 metric 名稱」，像是你原本文件中提到的：

- `log_host_job_basic`
- `log_host_job_service`
- `log_host_job_module`

這才是 Prometheus 建議的作法。

### ✅ 改寫建議 (關鍵邏輯)

你可以根據 label 組合來動態建立對應的 metric 物件，例如：



```
log_host_job_basic = Gauge("log_host_job_basic", "Count with host & job_name",  
["host", "job_name"])  
log_host_job_service = Gauge("log_host_job_service", "Count with service info",  
["host", "job_name", "service_name", "container_name"])  
log_host_job_module = Gauge("log_host_job_module", "Count with module info",  
["host", "job_name", "module_name"])
```

然後根據你解析出來的 `extra\_labels` 來決定放進哪一個 metric：

```
if "service_name" in extra_labels or "container_name" in extra_labels:  
    labels = {  
        "host": host,  
        "job_name": job,  
        "service_name": extra_labels.get("service_name", ""),  
        "container_name": extra_labels.get("container_name", ""),  
    }  
    log_host_job_service.labels(**labels).inc()  
  
elif "module_name" in extra_labels:  
    labels = {  
        "host": host,  
        "job_name": job,  
        "module_name": extra_labels.get("module_name", "")  
    }  
    log_host_job_module.labels(**labels).inc()  
  
else:  
    labels = {  
        "host": host,  
        "job_name": job  
    }  
    log_host_job_basic.labels(**labels).inc()
```

## ✓ 小結



## 你的需求

## 建議實作方式

根據不同 label 組合產生不同 metric

✅ 是正確且推薦的做法

同一個 metric 包含不固定的 labels

❌ 會造成 Prometheus 無法正確處理和查詢指標

🔧 **Bonus**：如果你希望我幫你重寫你的程式碼來配合這種多 metric 的設計，我也可以幫你調整！只要回我說「幫我改成多 metric 版本」，我會幫你拆成 `basic / service / module` 三種 metric 實作 💡