

# gpt4.5-PromQL 查詢語言介紹



以繁體中文詳細說明:PromQL



以下以繁體中文詳細介紹 **PromQL** (Prometheus Query Language)：

## 一、什麼是 PromQL？

**PromQL** (Prometheus Query Language) 是一種專門為 Prometheus 時序資料庫設計的查詢語言，用於從 Prometheus 中獲取、篩選、運算、以及呈現時間序列 (time series) 資料。

PromQL 的設計目標是：

- 易於學習與理解。
- 能夠有效地查詢與計算大量的時序資料。
- 提供即時 (Instant) 和區間 (Range) 查詢。

## 二、PromQL 的資料類型 (Data Types)

PromQL 主要處理以下三種資料類型：

- **Instant Vector (即時向量)**  
代表一個特定時間點的指標集合。

範例：

```
node_cpu_seconds_total
```

- **Range Vector（區間向量）**

代表某一時間區間內的資料點集合。

範例：

```
node_cpu_seconds_total[5m] # 最近 5 分鐘內的資料
```

- **Scalar（純量）**

單一的數值結果，例如計算的平均值、總和。

範例：

```
count(up)
```

---

## 三、基本查詢（Basic Queries）

基本查詢用於查詢一組指標，通常以 metric 的名稱開頭：

```
http_requests_total
```

也可加上標籤（labels）來篩選資料：

```
http_requests_total{method="POST", status="200"}
```

---

## 四、函數（Functions）

PromQL 提供多種函數來對資料進行運算與轉換，主要分以下類型：

## 時間相關函數：

- **rate()**：計算區間內的指標「每秒」平均變化率，常用於 counter 類型。

```
rate(http_requests_total[5m])
```

- **irate()**：計算最近兩個點之間的瞬時變化率（即時反應）。

```
irate(http_requests_total[5m])
```

- **increase()**：計算指定時間區間內的總增量，常用於 counter。

```
increase(http_requests_total[1h])
```

## 聚合函數：

- **sum()**：總和

```
sum(rate(http_requests_total[5m]))
```

- **avg()**：平均

```
avg(node_memory_usage_bytes)
```

- **min()**：最小值

```
min(node_filesystem_free_bytes)
```

- **max()**：最大值

```
max(node_filesystem_free_bytes)
```

- **count()**：計算符合條件的 series 個數

```
count(up{job="node_exporter"})
```

- **stddev()**：標準差

```
stddev(rate(http_requests_total[5m]))
```

---

## 五、運算子 (Operators)

PromQL 提供數學與邏輯運算子來處理向量資料：

- 算術運算子（`+`，`-`，`\*`，`/`，`%`）

```
node_memory_MemTotal_bytes - node_memory_MemFree_bytes
```

- 比較運算子（`==`，`!=`，`>`，`<`，`>=`，`<=`）

```
up == 0
```

- 邏輯運算子（`and`，`or`，`unless`）

```
node_load1 > 0.5 and up{job="node_exporter"} == 1
```

---

## 六、Instant Query 與 Range Query 的區別

## Instant Query (即時查詢)

Instant Query 查詢特定單一時間點的資料，例如：

```
node_cpu_seconds_total
```

## Range Query (區間查詢)

Range Query 查詢指定時間範圍內的資料點集合，必須搭配 `[interval]` 語法：

```
rate(node_cpu_seconds_total[5m])
```

- `[5m]` 表示「過去 5 分鐘內」的時間區間資料集合，適合搭配 `rate()` 等函數使用。
- Prometheus 介面上的 `time` 參數指的是資料的取樣點，與 `[5m]` 的區間語法是不同概念：
  - `time` 是查詢時間點 (Instant)
  - `[5m]` 是該時間點之前 5 分鐘的範圍內的資料集合 (Range)

## 七、實用查詢範例

以下是一些常用的查詢範例：

- 查詢最近 5 分鐘內每秒 HTTP 請求次數的平均變化率：

```
rate(http_requests_total[5m])
```

- 查詢過去 1 小時內 HTTP 請求數總增量：

```
increase(http_requests_total[1h])
```

- 查詢集群內所有節點的 CPU 使用率平均值：

```
avg(rate(node_cpu_seconds_total{mode!="idle"}[5m]))
```

- 查詢集群內記憶體最少的節點：

```
min(node_memory_MemAvailable_bytes)
```

- 查詢最近 5 分鐘內 HTTP 請求次數標準差：

```
stddev(rate(http_requests_total[5m]))
```

---

## 八、Grafana 搭配 PromQL

PromQL 通常搭配 Grafana 視覺化資料，例如：

- 在 Grafana Dashboard 中使用 PromQL，能以圖形、表格呈現系統效能變化、資源使用量，並進行 Capacity Planning 等效能規劃。

---

## 九、VictoriaMetrics 與 PromQL 相容性

- VictoriaMetrics 與 Prometheus 查詢語言相容，支持 PromQL，因此所有上述 PromQL 範例均可直接套用至 VictoriaMetrics Cluster 中。
- VictoriaMetrics 提供更高效的儲存與查詢效能，適合更大規模的監控場景。

---

## 十、總結

- **PromQL** 是 Prometheus 資料庫的核心查詢語言，專為時序資料設計。
- 支援豐富函數與運算子，適合多樣化的監控與資料分析需求。
- 搭配 Grafana 使用，可輕鬆實現數據視覺化與即時監控。

透過以上介紹，相信你已能初步掌握 PromQL 的使用方法與概念。



我想要理解rate()、irate()、increase()、sum()、avg()、min()、max()、count()、stddev()用法，為了用Grafana做VictoriaMetric- Cluster version的Capacity planning實驗，你能教我嗎，或是你能給更簡單的範例：



以下我用 **繁體中文** 為你以簡單、清楚的方式，逐一解釋你提到的這些 PromQL 函數，並給出清楚易懂的範例，方便你在使用 Grafana 搭配 VictoriaMetrics 做 Capacity Planning 實驗時參考。

## 一、速率與增量函數

這三個函數經常用於觀察與計算 metrics 的變化：

### ① `rate()``

**用途：**計算區間內 counter 指標每秒的平均變化率（通常用於計數器，如 HTTP 請求數）。

**範例：**

```
rate(http_requests_total[5m])
```

- 意思：計算過去 5 分鐘內，每秒 HTTP 請求數的平均增加率。

- 適合場景：觀察系統每秒平均負載（requests/sec）。
- 

## ② `irate()``

**用途：**計算最近兩個時間點之間的「瞬時變化率」，反映當前即時狀態。

**範例：**

```
irate(http_requests_total[5m])
```

- 意思：只看最近兩次取樣間的每秒 HTTP 請求變化率（瞬時速率）。
  - 適合場景：即時監控突發流量變化，例如觀察短時間內的流量尖峰。
- 

## ③ `increase()``

**用途：**計算一段時間內指標總共增加了多少（總增量）。

**範例：**

```
increase(http_requests_total[1h])
```

- 意思：計算過去 1 小時內 HTTP 請求數總共增加了多少次。
  - 適合場景：分析系統長期內的總請求量、總錯誤次數等。
- 

## 二、聚合函數 (Aggregation)



聚合函數主要用來將多個 metrics 聚合成一個數值：

#### ④ ``sum()``

**用途：**計算指標總和。

**範例：**

```
sum(rate(http_requests_total[5m]))
```

- 意思：將所有實例每秒 HTTP 請求數的平均值加總起來。
  - 適合場景：計算整個 cluster 的總負載。
- 

#### ⑤ ``avg()``

**用途：**計算指標平均值。

**範例：**

```
avg(node_memory_MemAvailable_bytes)
```

- 意思：計算 cluster 所有節點平均可用記憶體。
  - 適合場景：平均資源使用狀態，觀察系統資源利用情形。
- 

#### ⑥ ``min()`` 和 ``max()``

**用途：**

- ``min()``：找出指標的最小值。

- ``max()``：找出指標的最大值。

範例：

```
min(node_filesystem_free_bytes) # 最小可用硬碟空間  
max(node_cpu_seconds_total)    # CPU使用時間最高的節點
```

- 適合場景：
  - 透過 ``min()`` 發現系統資源最少的節點，避免資源耗盡。
  - 透過 ``max()`` 發現負載最高的節點，確認是否需要額外擴充。

---

## ⑦ ``count()``

用途：計算符合條件的 metrics 的數量。

範例：

```
count(up{job="node_exporter"})
```

- 意思：計算狀態為 up 的 node\_exporter 實例數量。
- 適合場景：確認目前活躍節點數量、服務運行中的實例數量。

---

## ⑧ ``stddev()``

用途：計算標準差，觀察指標的波動程度（變化情形）。

範例：

```
stddev(rate(http_requests_total[5m]))
```

- 意思：計算過去 5 分鐘內 HTTP 每秒請求數的標準差，標準差越高表示波動越劇烈。
- 適合場景：確認系統負載是否穩定或存在不規則突發流量。

### 三、實際 Capacity Planning 範例（簡易版）

以下為簡單的 Grafana + VictoriaMetrics 的 Capacity Planning 使用情境：

	說明
<code>total{mode!="idle"}[5m])) * 100</code>	觀察整體 CPU 使用情形（百分比）
<code>total[1m]))`</code>	估算短期內服務流量
<code>_bytes)`</code>	找到最容易耗盡記憶體 的節點
<code>es)`</code>	提前規劃節點的儲存擴 充
<code>total[5m]))`</code>	偵測異常流量尖峰或波 動
<code>} == 1)`</code>	觀察目前運行中的節點 數量

透過上述的範例，就能簡單、清楚地在 Grafana 上視覺化 VictoriaMetrics 的 metrics，從而進行更有效的 Capacity Planning。

## 四、VictoriaMetrics 的相容性提醒

以上 PromQL 語法，**完全適用於 VictoriaMetrics**（Cluster 版本）。VictoriaMetrics 具備極佳的 PromQL 相容性，支援完整語法，效能更優異。

你只需將上述查詢語句直接放入 Grafana 搭配 VictoriaMetrics 的 Data Source 即可。

## 五、總結與建議

- **rate()** 與 **irate()** 關注指標變化速率，前者為平均變化，後者為瞬時變化。
- **increase()** 適合看累積增量，長期趨勢。
- 聚合函數（sum、avg、min、max、count、stddev）適合整合多個節點或 metrics 的狀態，以整體系統層面來觀察與規劃。
- Capacity Planning 實驗建議從資源使用率（CPU、Memory、Disk）與服務負載（HTTP請求數）開始，逐步調整監控範圍與時間區間，確保資料的可靠性與規劃的有效性。

希望這些說明與範例能協助你順利完成 VictoriaMetrics + Grafana 的 Capacity Planning 實驗！



我想要理解by (job)、by (instance)用法，為了用Grafana做 VictoriaMetric- Cluster version的Capacity planning實驗，你能教我嗎，或是你能給更簡單的範例：



在 **PromQL** 中，`by()` 是一種「分組」運算方式。它允許你針對特定的標籤（例如 `job` 或 `instance`）進行聚合，這樣可以分別看到不同標籤的統計結果。以下我以簡單的例子說明 `by (job)` 和 `by (instance)` 的用法，並結合 Grafana 進行 Capacity Planning 的情境。

---

## 一、什麼是 `by()`` 分組？

通常在做聚合操作時，例如使用 `sum()`、`avg()` 等函數，我們希望聚合的數值能以某些標籤為基準進行分類，而不是直接將所有數據合併在一起。

- `by (job)`：按 `job` 標籤進行分組，結果會以不同的 `job` 值分開。
- `by (instance)`：按 `instance` 標籤進行分組，結果會以不同的 `instance` 值分開。

---

## 二、`by (job)` 範例

### 1. 基本範例：

假設我們想要知道每個 `job` 的 HTTP 請求數（每秒增長率）：

```
sum(rate(http_requests_total[5m])) by (job)
```

結果解釋：

- 系統會將所有 `http\_requests\_total` 的數據按 `job` 標籤分組，計算每個 `job` 的每秒增長率。
- 最後的輸出會像這樣：

```
{job="frontend"}: 125.3  
{job="backend"}: 98.7  
{job="database"}: 45.1
```

這樣你就可以看出不同 `job` 的負載差異。

## 2. Grafana 應用：

在 Grafana 中：

1. 設定查詢為：

```
sum(rate(http_requests_total[5m])) by (job)
```

2. 選擇一個條形圖（Bar Chart）或分組的時間序列（Grouped Time Series）視覺化，這樣可以清楚看到每個 `job` 的負載分佈。
3. 配合 Capacity Planning 的目標，你可以比較不同 `job` 的負載，決定是否需要為某些 `job` 增加資源或進行優化。

---

## 三、`by (instance)` 範例

### 1. 基本範例：

假設我們想要觀察每個實例（`instance`）的記憶體使用狀況：

```
avg(node_memory_MemAvailable_bytes) by (instance)
```

結果解釋：

- 系統會將每個實例（例如 `instance="192.168.1.1:9100"`）的可用記憶體平均值分開。
- 最後的輸出可能是：

```
{instance="192.168.1.1:9100"}: 1.2e+09
{instance="192.168.1.2:9100"}: 3.5e+09
{instance="192.168.1.3:9100"}: 2.8e+09
```

這樣你可以看到哪個實例的記憶體較少，可能需要調整資源配置。

---

## 2. Grafana 應用：

在 Grafana 中：

1. 設定查詢為：

```
avg(node_memory_MemAvailable_bytes) by (instance)
```

2. 選擇一個折線圖 (Line Graph)，每條線代表一個實例，這樣可以直觀地比較不同實例的記憶體使用狀況。
3. 進一步的 Capacity Planning 可以基於這些資訊來決定是否需要替特定實例擴充硬體或平衡負載。

---

## 四、`by()` 的應用場景

- **`by (job)`**：  
用於區分不同工作類型的負載（例如前端 vs. 後端 vs. 資料庫），幫助你了解哪類工作最需要擴展資源。
- **`by (instance)`**：  
用於區分不同伺服器或容器的負載，幫助你找到可能的瓶頸實例，進行資源優化或替換。

---

## 五、總結

1. **`by (job)` 和 `by (instance)`** 是用來按標籤分組的關鍵工具，幫助你從整體數據中提取出有意義的子群組結果。

2. 在 Capacity Planning 中，透過 `by (job)` 可以觀察哪個工作類型負載最高；透過 `by (instance)` 可以找到最吃資源的實例。
3. 搭配 Grafana，可以用視覺化的方式更容易判斷系統瓶頸、資源分配是否合理。

希望這些範例和說明對你的 Capacity Planning 實驗有所幫助！



我想要理解折線圖、長條圖、stat圖用法，為了用Grafana做 VictoriaMetric- Cluster version的Capacity planning實驗，你能教我嗎，或是你能給更簡單的範例：



推理花了 4 秒 >

以下提供使用 Grafana 視覺化工具，結合 VictoriaMetrics Cluster 版做 Capacity Planning 實驗時，如何運用折線圖、長條圖和 Stat 圖的解說與範例說明：

## 一、折線圖（Line Chart）

### 1. 功能與用途

折線圖主要用來顯示時間序列資料的變化趨勢。對於 Capacity Planning 而言，折線圖可以：

- **追蹤** 系統在一段時間內各項指標（例如 CPU 使用率、記憶體使用量、網路流量）的波動情形。
- **預測** 未來趨勢，藉由觀察歷史數據的模式和變化率來辨識資源瓶頸。

### 2. Grafana 中應用範例

假設要觀察集群中 CPU 使用率的趨勢，可使用以下 PromQL 語法：



```
avg(rate(node_cpu_seconds_total{mode!="idle"}[5m])) by (instance)
```

- **說明：**  
計算每個實例在過去 5 分鐘內的平均 CPU 使用率（非 idle 的部分），並依照 `instance` 分組。
- **Grafana 設定：**
  - 選擇折線圖 Panel。
  - 配置數據來源為 VictoriaMetrics。
  - 設定查詢並調整時間區間，使圖表呈現動態趨勢。

## 二、長條圖（Bar Chart）

### 1. 功能與用途

長條圖主要用來比較不同類別（例如不同的 job 或 instance）在某一特定時刻或時間區間內的數值。

- **比較** 各組別的資源使用情況，找出最吃資源的項目。
- **視覺化** 聚合結果，清楚呈現個別分組間的差異。

### 2. Grafana 中應用範例

例如，若想比較集群中各個 job 的 HTTP 請求數的變化，可使用：

```
sum(rate(http_requests_total[5m])) by (job)
```

- **說明：**  
將近 5 分鐘內每秒 HTTP 請求數依照 `job` 分組後計算總和。

- **Grafana 設定：**

- 選擇長條圖 Panel。
- 在查詢中利用 `by (job)` 分組，讓長條圖顯示不同 job 的指標數值，便於直接比較各工作類型的負載狀況。
- 選擇合適的排序方式，例如由高到低排序，直觀呈現出負載較高的 job，以便針對性進行 Capacity Planning。

---

## 三、Stat 圖 (SingleStat 或 Stat Panel)

### 1. 功能與用途

Stat 圖主要用來呈現單一數值或關鍵指標 (KPI)，通常是聚合後的結果或當前最新狀態。適用於：

- **監控即時狀態：**例如當前總活躍節點數、整體 CPU 平均使用率、記憶體可用量等。
- **儀表板 KPI：**將關鍵數據直觀呈現，以便快速決策。

### 2. Grafana 中應用範例

假設你需要呈現目前整個集群的總可用記憶體，可以這樣查詢：

```
sum(node_memory_MemAvailable_bytes)
```

- **說明：**

使用 `sum()` 聚合所有實例的可用記憶體值，獲得集群層面的總可用記憶體。

- **Grafana 設定：**

- 選擇 Stat 面板或 SingleStat 面板。
- 數據來源設置好後，把查詢結果設定為主要顯示數值。
- 進一步可設定閾值 (Threshold)，例如當記憶體低於某個值時顯示警告色彩，方便即時監控資源使用狀態。

---

## 四、如何選擇視覺化圖表以達成 Capacity Planning 目標

- **折線圖：**

當你希望追蹤資源使用率或請求趨勢時，折線圖能夠給出時間上的動態變化，例如 CPU 或網路流量隨時間的演變，協助預測未來資源是否可能不足。

- **長條圖：**

當需要比較不同分組（例如不同 job 或 instance）的資源使用或負載情況時，長條圖提供直觀的對比效果，幫助找到最需要優化或擴容的部分。

- **Stat 圖：**

當需要快速知道當前關鍵數值（如活躍節點數、總記憶體可用量、總請求數）時，Stat 圖可即時展示這些關鍵指標，作為快速參考，便於即時監控和提醒管理者關注特定資源瓶頸。

---

## 五、總結

- **折線圖** 適用於追蹤時間序列數據，適合展示趨勢和變化模式，適合 Capacity Planning 中的長期觀察與預警。
- **長條圖** 強調不同類別間的比較，例如各 job 或各 instance 的資源使用狀況，有助於找到系統中負載不均衡的地方。
- **Stat 圖** 用於顯示單一、關鍵的數字，使你可以一目了然地了解當前狀態，方便做出即時決策。

這些視覺化工具能夠協同運作，幫助你在 Grafana 中構建一個全面的 Capacity Planning 儀表板。根據需求，你可以混合使用不同圖表來從多角度觀察系統表現，進而做出更合適的資源配置與擴容策略。

希望這些說明與範例能夠幫助你更好地利用 Grafana 配合 VictoriaMetrics 進行 Capacity Planning 的實驗！