



Prometheus vs VictoriaMetrics



詳細說明怎麼比較promethe、victoriametric-cluster



Prometheus 和 VictoriaMetrics Cluster 都是用來監控和存儲時序數據（TSDB，Time-Series Database）的解決方案，兩者在架構、效能、擴展性、功能支援上有所不同。以下是詳細的比較方式：

1. 架構對比

特色	Prometheus	VictoriaMetrics Cluster
架構	單節點存儲，可透過聯邦集群（Federation）或 Thanos / Cortex 進行擴展	分布式集群架構，具備原生水平擴展能力
存儲引擎	基於 TSDB（Chunk-based Storage）	自家設計的存儲引擎，使用 LSM-Tree
擴展性	需借助 Thanos / Cortex 來支持高可用和水平擴展	原生支持多節點水平擴展
高可用性	透過多個 Prometheus 節點+Thanos Query	內建 HA（High Availability）機制

2. 效能比較



特色	Prometheus	VictoriaMetrics Cluster
寫入吞吐量	單節點處理數百萬時間序列，但受限於本機存儲	可支持更高的寫入吞吐量，透過 sharding 進行負載均衡
查詢性能	受限於單節點，隨著數據增長查詢時間會增加	基於分布式查詢，能更快處理大量數據
壓縮率	約 1-2 byte per sample	0.3 byte per sample，壓縮率更高

VictoriaMetrics 通常比 Prometheus 更省資源，因為它有更優化的壓縮演算法，並支援更大規模的時序數據存儲。

3. 部署與維運

特色	Prometheus	VictoriaMetrics Cluster
安裝難度	簡單，單一二進制文件運行	較複雜，需要配置 multiple components (vmstorage, vminsert, vmselect)
擴展方式	透過聯邦集群（Federation）或 Thanos / Cortex 來擴展	可直接通過增加節點來擴展
維護成本	單節點架構易維護，但需要定期刪除過期數據	分布式系統需要專業維運，但存儲管理較佳

Prometheus 適合小型部署，而 VictoriaMetrics Cluster 更適合大規模分布式環境。

4. 功能支援

特色	Prometheus	VictoriaMetrics Cluster
原生 PromQL 支援	✅ 完整支援	✅ 完整支援，並提供額外擴展功能



特色	Prometheus	VictoriaMetrics Cluster
歷史數據導入	需透過 remote_write 或 TSDB API	可使用 CSV、JSON、OpenMetrics 等格式直接導入
遠端存儲	透過 remote_write / remote_read	內建高效遠端存儲支持
內存占用	較高，需手動優化配置	相對較低，存儲壓縮更佳

VictoriaMetrics Cluster 在數據存儲和遠端存儲方面較有優勢，而 Prometheus 則擁有更完整的生態系統。

5. 適用場景

- **Prometheus**
 - 適合 小型/中型監控環境
 - 適合 簡單部署
 - 需要 與 **Kubernetes** 結合（如 Prometheus Operator）
- **VictoriaMetrics Cluster**
 - 適合 大規模時序數據存儲
 - 適合 高吞吐量場景
 - 適合 需要高可用 HA 架構

6. 總結

如果你的需求是：

- 監控少量應用、單機架構 → **Prometheus**



- 需要擴展性好、查詢更快、節省存儲空間 → **VictoriaMetrics Cluster**

如果已有 Prometheus，可以考慮 **Prometheus + VictoriaMetrics (remote_write)**，利用 VictoriaMetrics Cluster 作為長期存儲解決方案。

這樣的比較方式應該能幫助你決定適合的監控系統！🚀



以繁體中文詳細說明:我用docker-compose 建立了 vmstorage, vminsert, vmselect, vmagent, vmaalert 各一個node，請問怎麼知道使用了多少cpu、mem



要監控您使用 Docker Compose 部署的 VictoriaMetrics 組件（如 vmstorage、vminsert、vmselect、vmagent、vmaalert）各自的 CPU 和記憶體使用情況，您可以採取以下方法：

1. 使用 `docker stats` 指令

Docker 提供了 `docker stats` 指令，可實時顯示所有運行中容器的資源使用情況，包括 CPU、記憶體、網路 I/O 等。

查看所有運行中容器的資源使用情況：

```
docker stats
```

此指令會持續刷新，顯示每個容器的以下資訊：

- **CONTAINER ID**：容器 ID
- **NAME**：容器名稱
- **CPU %**：CPU 使用率



- **MEM USAGE / LIMIT**：記憶體使用量和限制
- **MEM %**：記憶體使用率
- **NET I/O**：網路輸入/輸出
- **BLOCK I/O**：磁碟輸入/輸出
- **PIDS**：進程數

如果只想查看特定容器的資源使用情況，可以在指令後指定容器的名稱或 ID：

```
docker stats <container_name_or_id>
```

例如，查看名為 `vmstorage` 的容器：

```
docker stats vmstorage
```

要僅查看當前的資源使用狀態而不持續刷新，使用 `--no-stream` 選項：

```
docker stats --no-stream
```

這將顯示一次性結果，然後退出。

2. 使用 `docker inspect` 指令

`docker inspect` 指令可提供容器的詳細資訊，包括資源配置和使用情況。

查看特定容器的詳細資訊：

```
docker inspect <container_name_or_id>
```



此指令會輸出容器的 JSON 格式詳細資訊。您可以使用 `grep` 或其他工具過濾與記憶體相關的資訊：

```
docker inspect <container_name_or_id> | grep -i Memory
```

這將顯示與記憶體相關的配置和使用資訊。

3. 使用 `/sys/fs/cgroup` 檔案系統

Docker 容器的資源使用資訊也可以通過宿主機的 cgroup 檔案系統查看。

查看特定容器的記憶體使用情況：

1. 首先，找到容器的完整 ID：

```
docker ps --no-trunc
```

2. 然後，查看該容器的記憶體使用：

```
cat /sys/fs/cgroup/memory/docker/<container_id>/memory.usage_in_bytes
```

這將以位元組為單位顯示容器的當前記憶體使用量。

查看容器的 CPU 使用情況：

```
cat /sys/fs/cgroup/cpu/docker/<container_id>/cpuacct.usage
```

這將顯示容器的累積 CPU 使用時間（以奈秒為單位）。



4. 使用第三方監控工具

為了更直觀和全面地監控容器的資源使用情況，您可以考慮使用以下第三方工具：

- **cAdvisor**：專門用於監控容器資源使用，包括 CPU、記憶體、磁碟和網路。
- **Prometheus + Grafana**：結合使用，提供強大的監控和可視化功能。您可以配置 Prometheus 來抓取 Docker 容器的指標，並使用 Grafana 進行可視化。
- **Portainer**：提供 Docker 容器管理和資源使用的圖形介面。

這些工具可以提供更詳細的歷史數據和圖表，方便您分析和監控容器的性能。

通過上述方法，您可以有效地監控 Docker 容器的 CPU 和記憶體使用情況，確保您的 VictoriaMetrics 組件運行在最佳狀態。

資料來源



以繁體中文詳細說明:使用第三方監控工具Prometheus + Grafana



使用 Prometheus + Grafana 監控 Docker 容器 (VictoriaMetrics 組件)

Prometheus 和 Grafana 是最常用的監控和可視化工具，特別適用於監控 Docker 容器的 CPU、記憶體、網絡流量等資源使用狀況。以下是詳細的安裝與設定步驟：



1. 安裝與設定 Prometheus

Prometheus 是一個時序數據庫 (TSDB)，可以從 Docker 容器獲取監控數據。

1.1 建立 `docker-compose.yml`

在您的專案目錄下，建立一個 `docker-compose.yml` 檔案，並加入以下內容：

```
version: '3.7'
services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    restart: always
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    networks:
      - monitoring

  node-exporter:
    image: prom/node-exporter
    container_name: node-exporter
    restart: always
    ports:
      - "9100:9100"
    networks:
      - monitoring

  cadvisor:
    image: gcr.io/cadvisor/cadvisor
    container_name: cadvisor
    restart: always
    privileged: true
    ports:
      - "8080:8080"
    volumes:
      - "/:/rootfs:ro"
```





```
- "/var/run:/var/run:rw"
- "/sys:/sys:ro"
- "/var/lib/docker:/var/lib/docker:ro"
networks:
  - monitoring

grafana:
  image: grafana/grafana
  container_name: grafana
  restart: always
  ports:
    - "3000:3000"
  networks:
    - monitoring
  volumes:
    - grafana-data:/var/lib/grafana
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=admin

networks:
  monitoring:

volumes:
  grafana-data:
```

2. 設定 Prometheus

在 `docker-compose.yml` 所在的目錄下，建立 `prometheus.yml` 配置文件，內容如下：

```
global:
  scrape_interval: 5s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```



```
- job_name: 'node-exporter'
  static_configs:
    - targets: ['node-exporter:9100']

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

3. 啟動監控系統

在終端機執行以下指令：

```
docker-compose up -d
```

- `-d` 參數表示以背景模式運行。
- 運行後，可透過 `http://localhost:9090` 存取 Prometheus。

4. 設定 Grafana

4.1 進入 Grafana

在瀏覽器打開 `http://localhost:3000`，預設帳號密碼如下：

- 帳號：`admin`
- 密碼：`admin`（第一次登入會要求更改密碼）

4.2 新增 Prometheus 資料來源

1. 點擊左側「齒輪」圖標 (⚙️) → `Data Sources`。



2. 點擊「Add data source」。
3. 選擇「Prometheus」。
4. 在「URL」欄位填入：

```
http://prometheus:9090
```

5. 點擊「Save & Test」確認連接成功。

5. 建立監控儀表板

5.1 匯入現成的 Dashboard

1. 點擊左側「四方形圖標」→「Dashboards」。
2. 點擊「Import」。
3. 在「Import via Grafana.com」輸入以下 ID：
 - **Docker 監控儀表板 ID**：「893」（Docker & system metrics）
 - **Node Exporter 監控 ID**：「1860」（Linux 監控）
4. 點擊「Load」，選擇 Prometheus 作為數據來源，然後點擊「Import」。

5.2 建立自訂 Dashboard

1. 點擊「+ Create」→「Dashboard」。
2. 點擊「Add new panel」。
3. 選擇「Prometheus」作為數據來源。
4. 在「Query」欄位輸入：

```
container_memory_usage_bytes{container_label_com_docker_compose_service="vmstorage"}
```

這條查詢會顯示「vmstorage」容器的記憶體使用量。

5. 點擊「Save」，命名儀表板。



6. 監控 VictoriaMetrics

VictoriaMetrics 組件 (vmstorage、vminsert、vmselect、vmagent、vmaalert) 本身也能暴露 Prometheus 指標，可在 `prometheus.yml` 中新增：

```
- job_name: 'victoriametrics'
  static_configs:
    - targets: ['vmstorage:8428', 'vminsert:8480', 'vmselect:8481',
               'vmagent:8429', 'vmaalert:8880']
```

然後重啟 Prometheus：

```
docker-compose restart prometheus
```

7. 查看監控結果

- Prometheus 查詢界面：`http://localhost:9090`
- Grafana 儀表板：`http://localhost:3000`
- cAdvisor (容器監控)：`http://localhost:8080`

8. 查詢 VictoriaMetrics 監控數據

在 Prometheus UI 中，可使用以下 PromQL 查詢 VictoriaMetrics 相關的指標：

8.1 監控 CPU 使用率





```
rate(process_cpu_seconds_total{job="victoriametrics"}[5m])
```

8.2 監控記憶體使用

```
container_memory_usage_bytes{job="victoriametrics"}
```

8.3 監控磁碟 IO

```
rate(container_fs_io_time_seconds_total{job="victoriametrics"}[5m])
```

8.4 監控網路流量

```
rate(container_network_receive_bytes_total{job="victoriametrics"}[5m])
```

9. 總結

1. 安裝 **Prometheus**、**Grafana**、**cAdvisor**、**Node Exporter**
2. 配置 `prometheus.yml` 來收集 **Docker** 和 **VictoriaMetrics** 的監控數據
3. 啟動 `docker-compose up -d`
4. 進入 **Grafana**，設定 **Prometheus** 作為數據來源
5. 匯入或建立監控 **Dashboard**
6. 查詢 **CPU**、**記憶體**、**磁碟**和**網絡**使用情況
7. 使用 **PromQL** 查詢 **VictoriaMetrics** 組件的指標

這樣就能成功透過 **Prometheus + Grafana** 監控 **VictoriaMetrics** (**vmstorage**、**vminsert**、**vmselect**、**vmagent**、**vmalert**) 及其他 Docker 容器的資源使用情況！

