



VictoriaMetrics Docker Cluster



Cluster version.pdf
PDF

參考此pdf，以繁體中文詳細說明用docker image 而不使用 k8s，建立cluster VictoriaMetrics



要在不使用 Kubernetes 的情況下，利用 Docker 建立 VictoriaMetrics Cluster，請依照以下步驟操作：

1. 確認系統環境

- 安裝 Docker 並確保其服務正常運作。
- 為每個 VictoriaMetrics 服務分配適當的主機或虛擬機（`vminsert`、`vmselect` 和 `vmstorage`），以實現服務解耦和水平擴展。

2. 拉取 Docker 映像檔

拉取 VictoriaMetrics Cluster 所需的 Docker 映像檔：

```
docker pull victoriametrics/vminsert:<version>
docker pull victoriametrics/vmselect:<version>
docker pull victoriametrics/vmstorage:<version>
```

請將 `<version>` 替換為所需版本。





3. 部署 `vmstorage`

啟動用於儲存資料的 `vmstorage` 節點：

```
docker run -d \  
  --name vmstorage \  
  -p 8482:8482 \  
  -v /path/to/storage:/storage \  
  victoriametrics/vmstorage:<version> \  
  -retentionPeriod=<days> \  
  -storageDataPath=/storage
```

- `-retentionPeriod``：資料保留天數。
- `/path/to/storage``：本地資料目錄。

4. 部署 `vminsert`

啟動用於接收數據並分散到 `vmstorage` 節點的 `vminsert` 節點：

```
docker run -d \  
  --name vminsert \  
  -p 8480:8480 \  
  victoriametrics/vminsert:<version> \  
  -storageNode=<vmstorage_host>:8482
```

- `-storageNode``：指向 `vmstorage` 節點的地址。

5. 部署 `vmselect`

啟動用於查詢的 `vmselect` 節點：



```
docker run -d \  
  --name vmselect \  
  -p 8481:8481 \  
  victoriametrics/vmselect:<version> \  
  -storageNode=<vmstorage_host>:8482
```

- `-storageNode`：指向 `vmstorage` 節點的地址。

6. 配置負載均衡（可選）

如果部署多個 `vminsert` 或 `vmselect` 節點，可使用 Nginx 作為負載均衡器：

Nginx 配置範例：

```
http {  
    upstream vminsert {  
        server <vminsert1>:8480;  
        server <vminsert2>:8480;  
    }  
    upstream vmselect {  
        server <vmselect1>:8481;  
        server <vmselect2>:8481;  
    }  
  
    server {  
        location /insert/ {  
            proxy_pass http://vminsert;  
        }  
        location /select/ {  
            proxy_pass http://vmselect;  
        }  
    }  
}
```



7. 測試服務

- 驗證數據寫入：

```
curl -X POST "http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write" \
  -d '<data_payload>'
```

- 驗證查詢：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

8. 持續監控與擴展

- 定期監控服務的性能與資源使用情況。
- 當負載增加時，可以通過增加節點數量來水平擴展 `vminsert`、`vmselect` 和 `vmstorage`。

如果需要更詳細的技術支持或設定，建議參考 [VictoriaMetrics 官方文檔](#)。



參考"Cluster version.pdf"，以繁體中文詳細說明怎麼跟 vmagent 測試



以下是參考《Cluster version.pdf》，以繁體中文詳細說明如何使用 **vmagent** 測試 VictoriaMetrics Cluster 的方法。



1. 安裝與配置 vmagent

vmagent 是一個高效的數據收集代理，用於從多個來源收集監控數據並將其發送到 VictoriaMetrics。

(1) 拉取 vmagent Docker 映像檔

```
docker pull victoriametrics/vmagent:<version>
```

將 `**<version>**` 替換為所需的版本。

(2) 配置 vmagent

建立 **vmagent** 的配置文件，例如 `**vmagent.yml**`，定義監控目標 (targets)：

```
global:
  scrape_interval: 15s
  scrape_timeout: 10s

scrape_configs:
  - job_name: 'example_job'
    static_configs:
      - targets:
        - '<target_host>:<port>'
```

- `**scrape_interval**`：數據收集的間隔時間，預設為 15 秒。
- `**targets**`：目標服務的地址，例如 `**localhost:9100**`。

(3) 啟動 vmagent

使用以下命令啟動 **vmagent**，並配置將數據發送到 VictoriaMetrics Cluster 的 `**vminsert**` 節點：

```
docker run -d \
  --name vmagent \
  -p 8429:8429 \
  -v /path/to/vmagent.yml:/etc/vmagent.yml \
  victoriametrics/vmagent:<version> \
```



```
-remoteWrite.url=http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write
\
-promscrape.config=/etc/vmagent.yml
```

- `-remoteWrite.url``：設置數據寫入的目標 URL，指向 ``vminsert`` 節點。
- `-promscrape.config``：設置 Prometheus 格式的配置文件。

2. 測試數據寫入

(1) 模擬數據寫入

vmagent 會自動從配置的目標 (targets) 中抓取數據，並將其寫入 VictoriaMetrics Cluster 的 ``vminsert`` 節點。

可以在目標服務上安裝 [Node Exporter](#) 或其他兼容的數據來源，然後在配置中添加目標地址：

```
static_configs:
- targets:
  - 'localhost:9100'
```

(2) 驗證數據是否成功寫入

執行以下查詢，驗證數據是否已寫入 ``vmselect`` 節點：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

3. 監控 vmagent 本身的運行狀態

vmagent 提供 ``/metrics`` 頁面，用於暴露其自身的運行指標。使用以下 URL 獲取 **vmagent** 的監控數據：



```
curl http://<vmagent_host>:8429/metrics
```

這些數據可用於監控 **vmagent** 的效能、資源使用及數據抓取的狀態。

4. 啟用多目標數據寫入

如果需要將數據寫入多個 VictoriaMetrics Cluster 節點，可以在啟動參數中添加多個 `-remoteWrite.url``：

```
-remoteWrite.url=http://<vminsert1>:8480/insert/0/prometheus/api/v1/write \
-remoteWrite.url=http://<vminsert2>:8480/insert/0/prometheus/api/v1/write
```

5. 常見測試與排錯方法

(1) 檢查 vmagent 配置

確保 `vmagent.yml`` 中的目標地址正確，且所有目標服務均正常運行。

(2) 驗證數據抓取狀態

訪問以下 URL 檢查 **vmagent** 的當前抓取目標：

```
curl http://<vmagent_host>:8429/targets
```

該頁面會顯示所有目標的抓取狀態及錯誤信息。

(3) 驗證數據傳輸

查看 **vminsert** 的 `/metrics`` 頁面，確認數據是否成功傳輸：

```
curl http://<vminsert_host>:8480/metrics
```





(4) 測試查詢數據

使用 `vmselect` 執行以下查詢以檢查數據：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query_range?
query=up&start=<start_time>&end=<end_time>&step=15s"
```

總結

利用 **vmagent** 測試 VictoriaMetrics Cluster 時，需確認以下幾個環節：

1. **vmagent** 配置正確，並成功抓取目標服務的數據。
2. **vminsert** 節點能正確接收並存儲數據。
3. **vmselect** 節點能正確查詢數據。

通過這些步驟，可以確保 **vmagent** 與 VictoriaMetrics Cluster 的整合和測試順利完成。



參考"Cluster version.pdf"，以繁體中文詳細說明怎麼跟 grafana測試



以下是參考《Cluster version.pdf》所提供的資訊，詳細說明如何將 VictoriaMetrics Cluster 與 **Grafana** 整合並進行測試。

1. 安裝與啟動 Grafana

(1) 使用 Docker 安裝 Grafana

執行以下命令拉取並啟動 Grafana：



```
docker pull grafana/grafana:<version>
docker run -d \
  --name=grafana \
  -p 3000:3000 \
  grafana/grafana:<version>
```

將 `**<version>**` 替換為需要的 Grafana 版本。

(2) 登入 Grafana

- 預設 Grafana 網頁界面地址為 `**http://<grafana_host>:3000`。**
- 登入預設帳號：
 - **使用者名稱：**`admin`
 - **密碼：**`admin`
- 第一次登入時系統會要求修改密碼。

2. 配置 VictoriaMetrics Cluster 資料來源

(1) 在 Grafana 中新增資料來源

1. 進入 Grafana 網頁界面。
2. 點擊左側邊欄的齒輪圖標 (Settings)。
3. 選擇 **Data Sources**，然後點擊 **Add data source**。
4. 在列表中選擇 **Prometheus** 作為資料來源類型。

(2) 配置資料來源

填寫以下字段：

- **URL：**
 - 格式為 `**http://<vmselect_host>:8481/select/0/prometheus/`**
 - 其中 `**<vmselect_host>**` 為 `vmselect` 節點的主機地址。



- **Access**：選擇 `Server (default)`。
 - 點擊 **Save & Test**，確認資料來源配置正確。
-

3. 匯入官方 Dashboard

VictoriaMetrics 提供了官方的 Grafana Dashboard，可用於監控 Cluster 的運行狀態。

(1) 下載 Dashboard JSON

前往 [Grafana 官方 Dashboard](#) 或 VictoriaMetrics 提供的資源頁面下載 JSON 檔案（如 Dashboard ID `11176`）。

(2) 在 Grafana 中匯入 Dashboard

1. 點擊左側邊欄的 **+** (Create)。
 2. 選擇 **Import**。
 3. 選擇以下方式之一匯入：
 - **匯入 ID**：輸入 `11176`，然後點擊 **Load**。
 - **匯入 JSON**：將下載的 JSON 文件內容貼入文本框。
 4. 選擇剛剛配置的 VictoriaMetrics 資料來源，然後點擊 **Import**。
-

4. 測試數據查詢

(1) 添加圖表進行測試

1. 點擊左側邊欄的 **+** (Create)，選擇 **Dashboard**。
2. 點擊 **Add new panel**。
3. 在查詢區域選擇資料來源為剛配置的 VictoriaMetrics。
4. 輸入以下 PromQL 查詢進行測試：



```
up
```

此查詢會返回所有服務的健康狀態。

5. 配置圖表樣式後點擊 **Apply** 保存。

(2) 測試查詢範例

- 查看系統的監控指標，例如：

```
vm_http_requests_total
```

- 檢查 VM 集群中每秒處理的請求量：

```
rate(vm_http_requests_total[1m])
```

5. 常見問題排查

(1) 資料來源無法連線

- 確認 Grafana 與 `vmselect` 節點在同一網段或允許跨網段連接。
- 檢查 `vmselect` 的 8481 埠是否開放。

(2) 查詢無法返回數據

- 確認 `vminsert` 是否已成功寫入數據至 `vmstorage`。
- 確保 PromQL 查詢語法正確。

(3) 性能問題

- 使用 Grafana 官方 Dashboard 檢查集群資源使用情況，例如 CPU、記憶體或磁碟的佔用率。

總結



通過上述步驟，您可以將 VictoriaMetrics Cluster 與 Grafana 整合，並利用 Grafana 監控數據指標與系統性能。這樣的設置有助於實時了解集群運行狀況，並快速診斷問題。



參考"Cluster version.pdf"，以繁體中文詳細說明若vmagent失效



根據《Cluster version.pdf》內容，以下是針對 **vmagent** 失效時的詳細說明及處理方式：

1. vmagent 失效的可能影響

vmagent 是負責從多個監控目標收集數據並將其寫入 VictoriaMetrics 的代理工具。如果 **vmagent** 發生失效，可能會造成以下影響：

- **數據收集中斷**：來自 Prometheus、Node Exporter 或其他監控目標的數據無法傳輸至 VictoriaMetrics。
- **數據缺失**：期間產生的監控數據無法存儲，導致查詢和可視化時顯示不完整。
- **寫入壓力轉移**：如果有多個 **vmagent** 節點，其餘節點可能會面臨更大的寫入壓力。

2. 檢測 vmagent 失效的症狀

以下是一些常見的症狀，可以幫助判斷 **vmagent** 是否失效：

1. 無法抓取目標數據：

- 在 **vmagent** 的 `/targets` 頁面無法查看到任何目標數據。

```
curl http://<vmagent_host>:8429/targets
```



2. 無法寫入數據至 **vminsert**：

- 查看 **vminsert** 節點的 `/metrics`，檢查是否有數據接收減少或中斷。

```
curl http://<vminsert_host>:8480/metrics
```

3. Grafana 中無法查詢到最新數據：

- 在 Grafana 上執行 PromQL 查詢時，數據顯示為空或僅有歷史數據。

3. vmagent 失效的可能原因

1. 配置錯誤：

- 配置檔案 `vmagent.yml` 中的目標地址 (targets) 或寫入地址 (remoteWrite.url) 設定錯誤。

2. 資源不足：

- CPU、記憶體或磁碟空間不足，導致 **vmagent** 無法正常運行。

3. 網路問題：

- vmagent** 無法連接到目標或 `vminsert` 節點，可能是因網路配置錯誤。

4. 程序崩潰：

- vmagent** 進程意外終止。

4. 處理 vmagent 失效的步驟

(1) 檢查 vmagent 服務狀態

使用以下命令檢查 **vmagent** 的容器或進程是否運行：

```
docker ps | grep vmagent
```



若容器未運行，可以使用以下命令重新啟動：

```
docker restart vmagent
```

(2) 查看 vmagent 日誌

檢查 **vmagent** 的運行日誌以診斷問題：

```
docker logs vmagent
```

關鍵日誌信息包括：

- **target scraping errors**（目標抓取錯誤）。
- **remoteWrite errors**（數據寫入錯誤）。

(3) 檢查配置檔案

驗證 `vmagent.yml` 配置是否正確，特別是以下部分：

- **targets**：目標地址是否可訪問。
- **remoteWrite.url**：是否正確指向 `vminsert` 節點。

示例：

```
scrape_configs:
  - job_name: 'example_job'
    static_configs:
      - targets:
        - '<valid_target_host>:<port>'
```

(4) 確認資源是否充足

檢查主機的資源狀態（CPU、記憶體、磁碟空間）：

```
top # 查看 CPU 和記憶體使用率
df -h # 查看磁碟空間
```



如果資源不足，可以通過升級硬體或分散負載來解決。

(5) 測試連線

確保 **vmagent** 能連接到目標和 ``vminsert`` 節點：

```
curl http://<target_host>:<port>/metrics # 測試目標連線
curl http://<vminsert_host>:8480/metrics # 測試 vminsert 連線
```

(6) 啟用監控與告警

為 **vmagent** 設置監控和告警，以便及時發現問題。例如：

- 在 Grafana 中監控 **vmagent** 的指標（``vmagent_scrape_samples_scraped``）。
- 設置告警條件，如數據抓取速率低於預期。

5. 高可用性配置建議

(1) 部署多個 vmagent 節點

為提高可用性，建議部署多個 **vmagent** 節點，並在負載均衡器（如 Nginx）中配置輪詢。

(2) 配置多目標寫入

在 **vmagent** 中配置多個 ``-remoteWrite.url``，以確保數據可同時寫入多個 ``vminsert`` 節點：

```
-remoteWrite.url=http://<vminsert1>:8480/insert/0/prometheus/api/v1/write \
-remoteWrite.url=http://<vminsert2>:8480/insert/0/prometheus/api/v1/write
```

(3) 啟用自動重啟

為 **vmagent** 配置 Docker 的自動重啟策略：

```
docker update --restart always vmagent
```



總結

若 **vmagent** 發生失效，需快速檢查其運行狀態、配置檔案和系統資源，並採取修復措施。同時，透過高可用性配置（多節點部署、多目標寫入），可有效降低失效的影響並提升系統穩定性。



參考"Cluster version.pdf"，以繁體中文詳細說明若grafana失效



當 **Grafana** 發生失效時，會影響監控數據的可視化與查詢。以下是根據《Cluster version.pdf》的資料，詳細說明 **Grafana** 失效時的影響、原因分析及處理方法。

1. Grafana 失效的可能影響

1. 無法可視化數據：
 - 使用者無法透過圖形界面檢視 VictoriaMetrics Cluster 的監控數據。
2. 監控與告警中斷：
 - 設置在 Grafana 中的告警規則無法觸發。
3. 查詢受阻：
 - 通過 Grafana 執行的 PromQL 查詢將無法運作。

2. Grafana 失效的可能原因





1. Grafana 容器或服務停止：

- 容器未啟動或服務崩潰。

2. 資源不足：

- 主機的 CPU、記憶體或磁碟空間不足，導致 Grafana 無法正常運行。

3. 網路問題：

- Grafana 與 VictoriaMetrics 的 `vmselect` 節點無法連線。

4. 配置錯誤：

- 資料來源 (Data Source) 配置不正確或丟失。

5. 插件問題：

- Grafana 中的插件更新或錯誤導致功能無法正常使用。

3. 檢測 Grafana 失效的症狀

(1) Grafana 網頁界面無法訪問

在瀏覽器中打開 `http://<grafana_host>:3000`，如果界面無法加載，可能是 Grafana 容器未運行或網路問題。

(2) 資料來源無法連線

- 在 Grafana 的 **Data Sources** 頁面中測試資料來源，顯示連線失敗。

(3) 查詢報錯或無法返回數據

- 在面板中執行 PromQL 查詢時出現錯誤，或查詢結果為空。

(4) 無法啟動告警

- 設置的告警規則未觸發或報錯。



4. Grafana 失效的處理步驟

(1) 確認 Grafana 服務狀態

檢查 Docker 容器或系統服務是否正常運行：

```
docker ps | grep grafana
```

若容器未啟動，使用以下命令重啟：

```
docker restart grafana
```

若使用系統服務管理 Grafana：

```
sudo systemctl status grafana-server  
sudo systemctl restart grafana-server
```

(2) 查看 Grafana 日誌

檢查 Grafana 的日誌以診斷問題：

```
docker logs grafana
```

或：

```
sudo journalctl -u grafana-server
```

關鍵日誌信息包括：

- **資料來源錯誤**（例如：無法連接到 `vmselect`）。
- **資源限制**（如記憶體不足導致崩潰）。
- **插件錯誤**（如插件載入失敗）。

(3) 測試資料來源連線



1. 在 Grafana 的 **Data Sources** 頁面中，選擇已配置的資料來源並點擊 **Save & Test**。
2. 如果連線失敗：

- 確認 `vmselect` 節點是否可用：

```
curl http://<vmselect_host>:8481/metrics
```

- 檢查資料來源 URL 是否正確，例如：

```
http://<vmselect_host>:8481/select/0/prometheus/
```

(4) 檢查系統資源

確認主機的資源狀態：

```
top          # 檢查 CPU 和記憶體  
df -h        # 檢查磁碟空間
```

若資源不足，考慮升級硬體或清理不必要的檔案。

(5) 修復配置問題

若資料來源或面板配置錯誤：

1. 重新配置資料來源，確保指向正確的 `vmselect` 節點。
2. 在面板編輯頁面中，檢查並修正 PromQL 查詢語法。

(6) 測試告警功能

在 **Alerting** 頁面中測試告警是否正常觸發：

1. 查看告警規則狀態。
2. 如果告警無法觸發，檢查 PromQL 查詢是否正確。

(7) 更新或修復插件

如果問題與插件相關：

1. 在 **Plugins** 頁面中檢查插件狀態。



2. 移除或更新出現問題的插件。

5. 預防與高可用性配置建議

(1) 啟用自動重啟

為 Grafana 容器啟用自動重啟策略：

```
docker update --restart always grafana
```

(2) 設置多節點 Grafana

部署多個 Grafana 節點以實現高可用性，並使用負載均衡器（如 Nginx）進行流量分配。

(3) 定期備份 Grafana 配置

使用 Grafana 的匯出功能，定期備份以下內容：

- 資料來源配置。
- Dashboard 配置。
- 告警規則。

(4) 配置資源監控

在監控系統中加入 Grafana 的資源指標，定期檢查 Grafana 的 CPU、記憶體和磁碟使用情況。

6. 排錯範例：實際操作步驟

假設 Grafana 的面板查詢報錯，執行以下步驟：

1. 確認 **vmselect** 是否可用：

```
curl http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up
```



2. 測試 Grafana 資料來源：

- 在 **Data Sources** 中選擇資料來源，點擊 **Save & Test**。

3. 修改 PromQL 查詢：

- 確保查詢格式正確，例如：

```
rate(vm_http_requests_total[1m])
```

總結

當 Grafana 失效時，需快速檢查服務狀態、資料來源配置及系統資源，並採取對應的修復措施。通過高可用性配置和定期備份，可有效降低 Grafana 失效對監控系統的影響。



參考"Cluster version.pdf"，以繁體中文詳細說明若vminsert失效



當 **vminsert** 節點失效時，VictoriaMetrics Cluster 的數據寫入功能將受到影響。以下根據《Cluster version.pdf》的內容，詳細說明 **vminsert** 失效時的影響、原因分析及處理方法。

1. vminsert 失效的可能影響

vminsert 是負責接收數據並將其分發到 **vmstorage** 節點的服務。若 **vminsert** 失效，可能會導致：

1. 數據寫入中斷：

- 無法接收來自數據來源（如 Prometheus、vmagent）的數據。



2. 數據丟失：

- 在失效期間，無法保存數據。

3. 集群壓力轉移：

- 如果存在多個 **vminsert** 節點，其他節點的負載可能會增加。
-

2. vminsert 失效的可能原因

1. 資源不足：

- CPU、記憶體或磁碟空間不足，導致進程崩潰。

2. 網路問題：

- 無法與數據來源（如 **vmagent**）或 **vmstorage** 節點建立連線。

3. 配置錯誤：

- **vminsert** 的啟動參數錯誤，例如指向錯誤的 **vmstorage** 地址。

4. 服務崩潰：

- 程序因未處理的異常而停止運行。

5. 磁碟空間不足：

- 用於緩存的磁碟空間耗盡。
-

3. 檢測 vminsert 失效的症狀

(1) 無法接收數據

- 檢查數據來源（如 **vmagent**）的日誌中是否出現連線錯誤。

(2) 無法存儲數據

- 查看 **vmstorage** 節點的 `/metrics` 頁面，檢查是否未收到新的數據。



```
curl http://<vmstorage_host>:8482/metrics
```

(3) 查詢無法返回最新數據

- 在 **vmselect** 執行查詢時，僅能獲取歷史數據。

(4) vminsert 的健康檢查失敗

- 嘗試訪問 **vminsert** 節點的 `/health` 頁面確認狀態：

```
curl http://<vminsert_host>:8480/health
```

4. 處理 vminsert 失效的步驟

(1) 檢查服務狀態

確認 **vminsert** 的容器或進程是否正在運行：

```
docker ps | grep vminsert
```

若容器未運行，使用以下命令啟動：

```
docker restart vminsert
```

若使用系統服務管理：

```
sudo systemctl status vminsert  
sudo systemctl restart vminsert
```

(2) 查看日誌以診斷問題

檢查 **vminsert** 的運行日誌：



```
docker logs vminsert
```

或：

```
sudo journalctl -u vminsert
```

關注以下錯誤信息：

- 與 **vmstorage** 連接錯誤。
- 數據來源連接失敗。
- 資源不足錯誤（如 OOM 或磁碟空間不足）。

(3) 驗證配置

檢查 **vminsert** 的啟動參數是否正確：

- **-storageNode** 是否正確指向 **vmstorage** 節點。
- 例如：

```
docker run -d \  
  --name vminsert \  
  -p 8480:8480 \  
  victoriametrics/vminsert:<version> \  
  -storageNode=<vmstorage_host>:8482
```

(4) 測試數據寫入功能

從數據來源模擬寫入請求：

```
curl -X POST "http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write" \  
  -d '<data_payload>'
```

- 若返回錯誤，需進一步檢查配置與網路。

(5) 確保資源充足



檢查主機資源：

```
top # 檢查 CPU 和記憶體使用率
df -h # 檢查磁碟空間
```

若資源不足，考慮升級硬體或分散負載。

(6) 測試 vmstorage 節點

確認 **vmstorage** 是否正常運行，並能接收來自 **vminsert** 的數據：

```
curl http://<vmstorage_host>:8482/metrics
```

5. 預防與高可用性配置建議

(1) 部署多個 vminsert 節點

為提高可用性，部署多個 **vminsert** 節點，並透過負載均衡器（如 Nginx）進行流量分配。

(2) 配置多目標寫入

在數據來源（如 **vmagent**）中配置多個寫入地址：

```
-remoteWrite.url=http://<vminsert1>:8480/insert/0/prometheus/api/v1/write \
-remoteWrite.url=http://<vminsert2>:8480/insert/0/prometheus/api/v1/write
```

(3) 啟用自動重啟

為 **vminsert** 容器設置自動重啟策略：

```
docker update --restart always vminsert
```

(4) 監控 vminsert

使用 Prometheus 或 Grafana 監控 **vminsert** 的健康狀態與性能指標，例如：



- **vm_http_requests_total**：請求總數。
- **vm_insert_requests_total**：數據寫入請求總數。

6. 排錯範例：實際操作步驟

假設 **vminsert** 無法接收數據，執行以下步驟：

1. 確認 **vminsert** 是否運行：

```
docker ps | grep vminsert
```

2. 測試與 **vmstorage** 節點的連線：

```
curl http://<vmstorage_host>:8482/metrics
```

3. 查看 **vminsert** 日誌以診斷問題：

```
docker logs vminsert
```

4. 驗證數據來源是否正確指向 **vminsert**：

- 檢查數據來源的配置檔案。

總結

當 **vminsert** 失效時，需快速檢查服務狀態、配置及資源，並透過高可用性配置減少影響。此外，定期監控 **vminsert** 的性能和健康狀況，能有效預防問題的發生。



參考"Cluster version.pdf"，以繁體中文詳細說明若2個 **vminsert** 中的一個失效



當 **VictoriaMetrics Cluster** 中的兩個 **vminsert** 節點之一失效時，集群的數據寫入仍能繼續，但可能導致部分性能下降或負載轉移。以下根據《Cluster version.pdf》的內容，詳細說明 **vminsert** 節點失效的影響、診斷方法及處理方式。

1. 部分 vminsert 節點失效的影響

1. 數據寫入性能下降：

- 剩餘的 **vminsert** 節點需處理所有數據寫入請求，可能導致負載增加。

2. 高可用性受限：

- 如果剩餘節點也失效，整個數據寫入功能將中斷。

3. 數據重新路由：

- 數據來源（如 **vmagent** 或 Prometheus）會將請求重新路由到可用的 **vminsert** 節點。

2. 診斷部分 vminsert 節點失效的方法

(1) 檢查剩餘節點的負載

- 查看可用 **vminsert** 節點的指標數據，例如：

```
curl http://<active_vminsert_host>:8480/metrics
```

關注以下指標：

- **vm_http_requests_total**：請求總數。
- **vm_http_request_duration_seconds_sum**：請求的處理時間。

(2) 測試失效的 vminsert 節點

- 嘗試訪問失效節點的健康檢查端點：



```
curl http://<failed_vminsert_host>:8480/health
```

如果返回非 `200 OK`，說明該節點確實失效。

(3) 檢查數據來源的行為

- 查看 **vmagent** 或其他數據來源的日誌，確認是否存在重試或連線錯誤。

(4) 查看失效節點的日誌

- 登入失效節點的伺服器，檢查日誌信息以診斷問題：

```
docker logs vminsert
```

關注以下錯誤：

- 與 **vmstorage** 連線失敗。
- 資源不足（如 OOM 或磁碟空間不足）。

3. 部分 vminsert 節點失效的處理步驟

(1) 檢查失效節點的狀態

確認容器或服務是否運行：

```
docker ps | grep vminsert
```

如果未運行，嘗試重啟：

```
docker restart vminsert
```

若服務未啟動，檢查系統服務：

```
sudo systemctl status vminsert  
sudo systemctl restart vminsert
```



(2) 確保系統資源充足

- 檢查伺服器的 CPU、記憶體和磁碟空間：

```
top          # 檢查 CPU 和記憶體  
df -h        # 檢查磁碟空間
```

- 如果資源不足，請釋放空間或升級硬體。

(3) 測試與 **vmstorage** 節點的連線

確認失效的 **vminsert** 能連接到 **vmstorage**：

```
curl http://<vmstorage_host>:8482/metrics
```

若無法連接，檢查網路連線和 **vmstorage** 節點的運行狀態。

(4) 修復或替換失效節點

若節點無法恢復，考慮替換節點：

- 停止失效的節點。
- 部署一個新的 **vminsert** 節點，指向相同的 **vmstorage** 節點：

```
docker run -d \  
  --name vminsert-new \  
  -p 8480:8480 \  
  victoriametrics/vminsert:<version> \  
  -storageNode=<vmstorage_host>:8482
```

4. 優化配置以減少影響

(1) 配置多目標寫入

在數據來源中配置多個 **vminsert** 節點，實現自動故障轉移。例如，在 **vmagent** 中：



```
remoteWrite:
```

- url: http://<vminsert1_host>:8480/insert/0/prometheus/api/v1/write
- url: http://<vminsert2_host>:8480/insert/0/prometheus/api/v1/write

(2) 使用負載均衡器

在 **vminsert** 節點前部署負載均衡器（如 Nginx 或 HAProxy），實現自動流量分配。範例 Nginx 配置：

```
upstream vminsert {  
    server <vminsert1_host>:8480;  
    server <vminsert2_host>:8480;  
}  
  
server {  
    listen 8480;  
    location / {  
        proxy_pass http://vminsert;  
    }  
}
```

(3) 啟用自動重啟策略

為所有 **vminsert** 節點配置 Docker 的自動重啟策略：

```
docker update --restart always vminsert
```

5. 排錯範例：實際操作步驟

假設有兩個 **vminsert** 節點，其中一個失效：

1. 確認失效節點的狀態：

```
docker ps | grep vminsert
```



2. 測試失效節點的健康檢查：

```
curl http://<failed_vminsert_host>:8480/health
```

3. 重啟失效節點：

```
docker restart vminsert
```

4. 如果重啟無效，檢查日誌以診斷問題：

```
docker logs vminsert
```

5. 如果問題無法修復，部署新節點替代：

```
docker run -d \  
  --name vminsert-new \  
  -p 8480:8480 \  
  victoriametrics/vminsert:<version> \  
  -storageNode=<vmstorage_host>:8482
```

6. 結論

當兩個 **vminsert** 節點之一失效時，VictoriaMetrics Cluster 仍能部分運行，但性能和穩定性可能受到影響。通過以下措施可減少影響：

- 部署多個 **vminsert** 節點。
- 配置負載均衡和多目標寫入。
- 定期監控節點健康狀況。

這樣的配置能確保集群在部分節點失效時仍能穩定運行，並快速恢復功能。



參考"Cluster version.pdf"，以繁體中文詳細說明若 vmstorage 失效





當 **vmstorage** 節點失效時，會對 **VictoriaMetrics Cluster** 的數據存取產生影響，以下根據《Cluster version.pdf》的內容詳細說明 **vmstorage** 節點失效的影響、診斷方法及處理步驟。

1. vmstorage 節點失效的可能影響

1. 數據寫入受影響：

- **vminsert** 無法將數據寫入失效的 **vmstorage** 節點。
- 數據會重新路由到其他健康的 **vmstorage** 節點，導致負載增加。

2. 數據查詢不完整：

- **vmselect** 在查詢數據時可能缺失存儲於失效節點的部分數據。
- 查詢結果可能顯示為 **partial response**。

3. 集群性能下降：

- 健康節點需要承擔更高的負載，可能導致性能下降。

2. 診斷 vmstorage 節點失效的方法

(1) 驗證健康檢查端點

測試失效的 **vmstorage** 節點的健康狀態：

```
curl http://<vmstorage_host>:8482/health
```

- 返回 `200 OK` 表示正常。
- 返回其他狀態或無響應則可能失效。

(2) 檢查 vminsert 日誌



查看 **vminsert** 是否報告與失效的 **vmstorage** 節點連線錯誤：

```
docker logs vminsert
```

關鍵錯誤信息：

- 無法連接 **vmstorage**。
- **storageNode unreachable**。

(3) 檢查 **vmselect** 查詢行為

在 **vmselect** 中執行查詢，檢查是否有 **partial response**：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

- 如果返回部分數據且包含 `"isPartial": true`，表示部分 **vmstorage** 節點無法提供數據。

(4) 查看失效節點的日誌

登入失效的 **vmstorage** 節點伺服器，查看運行日誌：

```
docker logs vmstorage
```

關注以下錯誤：

- 磁碟空間不足。
- **OOM**（記憶體不足）。
- 與其他節點的網路連接失敗。

3. **vmstorage** 節點失效的處理步驟

(1) 檢查服務狀態



確認 **vmstorage** 的容器或服務是否正在運行：

```
docker ps | grep vmstorage
```

若未運行，嘗試啟動：

```
docker restart vmstorage
```

若服務未啟動，檢查系統服務：

```
sudo systemctl status vmstorage  
sudo systemctl restart vmstorage
```

(2) 檢查資源使用情況

- 確認伺服器的 CPU、記憶體和磁碟空間是否充足：

```
top          # 檢查 CPU 和記憶體  
df -h        # 檢查磁碟空間
```

- 如果磁碟空間不足，可以刪除無用檔案或擴展存儲。

(3) 測試與其他節點的連線

確認失效節點是否能與其他節點正常通信：

```
ping <other_node_host>  
curl http://<other_vmstorage_host>:8482/metrics
```

(4) 重啟或替換失效節點

如果無法修復失效節點，可以部署新的 **vmstorage** 節點：

1. 停止失效的節點。
2. 部署新的 **vmstorage** 節點：



```
docker run -d \  
  --name vmstorage-new \  
  -p 8482:8482 \  
  -v /path/to/storage:/storage \  
  victoriametrics/vmstorage:<version> \  
  -retentionPeriod=<days> \  
  -storageDataPath=/storage
```

3. 更新 **vminsert** 和 **vmselect** 的配置以包含新節點：

- **vminsert** :

```
docker restart vminsert --storageNode=<new_vmstorage_host>:8482
```

- **vmselect** :

```
docker restart vmselect --storageNode=<new_vmstorage_host>:8482
```

(5) 測試集群功能

- 測試數據寫入：

```
curl -X POST "http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write" \  
 \  
  -d '<data_payload>'
```

- 測試數據查詢：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query? \  
query=up"
```

4. 優化與高可用性配置建議

(1) 部署多個 **vmstorage** 節點

增加 **vmstorage** 節點數量，分散數據存儲壓力，提高集群穩定性。



(2) 啟用數據複製

為了避免單節點故障導致數據丟失，可以啟用數據複製功能：

- 配置 **vminsert** 的 `-replicationFactor=N``，設定複製數量。
- 複製數據將存儲於多個 **vmstorage** 節點。

(3) 使用自動發現功能

使用文件或 DNS SRV 的方式，自動發現並更新 **vmstorage** 節點：

- 文件方式：

```
-storageNode=file:/path/to/vmstorage-list
```

- DNS SRV：

```
-storageNode=srv+vmstorage-autodiscovery
```

(4) 定期監控與告警

設置監控和告警規則，監控以下指標：

- **vm_rpc_vmstorage_is_reachable**：監控節點是否可達。
- **vm_storage_is_read_only**：監控節點是否進入只讀模式。

5. 排錯範例：實際操作步驟

假設有三個 **vmstorage** 節點，其中一個失效：

1. 確認失效節點的狀態：

```
docker ps | grep vmstorage
```

2. 測試健康檢查端點：

```
curl http://<failed_vmstorage_host>:8482/health
```



3. 查看日誌：

```
docker logs vmstorage
```

4. 若無法修復，部署新節點並更新配置：

```
docker run -d \  
  --name vmstorage-new \  
  -p 8482:8482 \  
  -v /path/to/storage:/storage \  
  victoriametrics/vmstorage:<version> \  
  -retentionPeriod=<days> \  
  -storageDataPath=/storage
```

5. 測試數據寫入和查詢。

6. 結論

當 **vmstorage** 節點失效時，集群的高可用性機制可減少影響，但仍需快速處理失效節點以恢復集群功能。建議透過以下措施提高穩定性：

- 增加節點數量。
- 啟用數據複製。
- 使用自動發現功能。
- 設置有效的監控與告警規則。

這樣可確保集群在故障發生時仍能穩定運行並快速恢復。



參考"Cluster version.pdf"，以繁體中文詳細說明若2個vmstorage中的一個失效





當 **VictoriaMetrics Cluster** 中的兩個 **vmstorage** 節點之一失效時，集群的數據存取功能會部分受影響，但由於集群架構的高可用性設計，集群仍能繼續運行。以下根據《Cluster version.pdf》的內容詳細說明 **vmstorage** 節點部分失效的影響、診斷方法以及處理步驟。

1. 部分 vmstorage 節點失效的影響

1. 數據寫入受限：

- **vminsert** 無法將數據寫入失效的 **vmstorage** 節點，可能導致部分數據存取失敗。
- 其他健康的 **vmstorage** 節點需要承擔更多負載。

2. 數據查詢不完整：

- 如果失效節點存有特定數據，**vmselect** 在查詢時可能返回 **partial response**，查詢結果不完整。

3. 性能下降：

- 健康的 **vmstorage** 節點負載增加，可能導致整體性能下降。

4. 數據複製（如果啟用）可能提供冗餘：

- 若啟用了數據複製功能（``-replicationFactor=N``），數據可能已分散在多個節點，減少數據丟失風險。

2. 診斷部分 vmstorage 節點失效的方法

(1) 驗證健康檢查端點

測試失效的 **vmstorage** 節點的健康狀態：

```
curl http://<failed_vmstorage_host>:8482/health
```

- 返回 ``200 OK`` 表示節點健康。



- 返回其他狀態或無響應則可能失效。

(2) 檢查 vminsert 的日誌

查看 **vminsert** 的日誌，檢查是否有寫入失敗的錯誤：

```
docker logs vminsert
```

關鍵錯誤：

- **storageNode unreachable。**
- **failed to send data to vmstorage。**

(3) 檢查 vmselect 查詢行為

在 **vmselect** 中執行查詢，檢查是否返回 **partial response**：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

- 如果返回部分數據並包含 `"isPartial": true`，則表明某些 **vmstorage** 節點無法提供數據。

(4) 查看失效節點的日誌

登入失效節點的伺服器，檢查運行日誌：

```
docker logs vmstorage
```

關注以下錯誤：

- 磁碟空間不足。
- 記憶體不足（OOM）。
- 網路連接失敗。



3. 部分 vmstorage 節點失效的處理步驟

(1) 確認服務狀態

檢查失效的 **vmstorage** 容器或進程是否正在運行：

```
docker ps | grep vmstorage
```

若未運行，嘗試啟動：

```
docker restart vmstorage
```

若服務未啟動，檢查系統服務：

```
sudo systemctl status vmstorage  
sudo systemctl restart vmstorage
```

(2) 檢查資源狀態

- 確認伺服器的 CPU、記憶體和磁碟空間是否充足：

```
top          # 檢查 CPU 和記憶體使用  
df -h        # 檢查磁碟空間
```

- 若磁碟空間不足，清理無用檔案或擴展存儲。

(3) 測試網路連接

確認失效節點是否能與其他節點正常通信：

```
ping <other_vmstorage_host>  
curl http://<other_vmstorage_host>:8482/metrics
```

(4) 重啟或替換失效節點

如果無法修復失效節點，可考慮替換節點：

1. 停止失效的節點。



2. 部署新節點：

```
docker run -d \  
  --name vmstorage-new \  
  -p 8482:8482 \  
  -v /path/to/storage:/storage \  
  victoriametrics/vmstorage:<version> \  
  -retentionPeriod=<days> \  
  -storageDataPath=/storage
```

3. 更新 **vminsert** 和 **vmselect** 的配置以包含新節點：

- **vminsert**：

```
docker restart vminsert --storageNode=<new_vmstorage_host>:8482
```

- **vmselect**：

```
docker restart vmselect --storageNode=<new_vmstorage_host>:8482
```

(5) 測試集群功能

- 測試數據寫入：

```
curl -X POST "http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write" \  
  -d '<data_payload>'
```

- 測試數據查詢：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

4. 優化與高可用性配置建議

(1) 啟用數據複製

在 **vminsert** 的啟動參數中啟用數據複製：



```
-replicationFactor=2
```

此配置會將數據複製到多個 **vmstorage** 節點，減少單節點故障對數據的影響。

(2) 增加 vmstorage 節點

擴展 **vmstorage** 節點的數量，分散負載，提高集群的穩定性。

(3) 使用自動發現功能

配置自動發現，確保節點列表始終更新：

- 文件方式：

```
-storageNode=file:/path/to/vmstorage-list
```

- DNS SRV 方式：

```
-storageNode=srv+vmstorage-autodiscovery
```

(4) 設置監控與告警

監控 **vmstorage** 的指標，設置告警規則，及時檢測問題：

- **vm_rpc_vmstorage_is_reachable**：監控節點是否可達。
- **vm_storage_is_read_only**：檢測是否進入只讀模式。

5. 排錯範例：實際操作步驟

假設兩個 **vmstorage** 節點之一失效：

1. 確認失效節點的狀態：

```
docker ps | grep vmstorage
```

2. 測試失效節點的健康檢查：



```
curl http://<failed_vmstorage_host>:8482/health
```

3. 查看失效節點的日誌：

```
docker logs vmstorage
```

4. 若無法修復，部署新節點並更新配置：

```
docker run -d \  
  --name vmstorage-new \  
  -p 8482:8482 \  
  -v /path/to/storage:/storage \  
  victoriametrics/vmstorage:<version> \  
  -retentionPeriod=<days> \  
  -storageDataPath=/storage
```

6. 結論

當兩個 **vmstorage** 節點之一失效時，集群的高可用性機制能確保部分功能正常運行，但需要迅速處理失效節點以恢復完全的數據存取能力。透過以下措施可減少影響：

- 增加節點數量並啟用數據複製。
- 使用自動發現功能管理節點。
- 定期監控節點健康狀況。

這些措施可提高集群穩定性，確保在故障發生時仍能維持正常運作並快速恢復。



參考"Cluster version.pdf"，以繁體中文詳細說明若vmselect失效





當 **vmselect** 節點失效時，會對 **VictoriaMetrics Cluster** 的數據查詢功能產生影響，但數據的寫入和存儲通常不會受到影響。以下根據《Cluster version.pdf》的內容，詳細說明 **vmselect** 節點失效的影響、診斷方法以及處理步驟。

1. vmselect 節點失效的可能影響

1. 查詢功能中斷：

- **vmselect** 是負責從 **vmstorage** 中檢索數據的節點。如果所有 **vmselect** 節點失效，用戶將無法查詢數據。

2. 部分查詢受限：

- 如果集群中有多個 **vmselect** 節點且僅部分失效，負載會重新分配到健康的節點，可能導致性能下降。

3. Grafana 顯示異常：

- Grafana 可能無法正確顯示監控數據，因為其主要依賴 **vmselect** 進行數據查詢。

2. 診斷 vmselect 節點失效的方法

(1) 測試健康檢查端點

訪問失效節點的 `/health` 接口確認其狀態：

```
curl http://<vmselect_host>:8481/health
```

- 返回 `200 OK` 表示健康。
- 無響應或返回錯誤狀態則表示節點失效。

(2) 檢查 Grafana 的行為

- 在 Grafana 中測試查詢：



- 如果返回錯誤，如 `Bad Gateway` 或 `timeout`，可能表示 **vmselect** 節點不可用。

(3) 查看 **vmselect** 的日誌

檢查 **vmselect** 節點的運行日誌，診斷失效原因：

```
docker logs vmselect
```

關注以下錯誤：

- 與 **vmstorage** 連接失敗。
- 資源不足（如記憶體不足或 CPU 過載）。
- 請求超時。

(4) 測試與 **vmstorage** 的連線

確認失效的 **vmselect** 是否能與 **vmstorage** 節點正常通信：

```
curl http://<vmstorage_host>:8482/metrics
```

如果無法連接，問題可能出在網路配置。

3. **vmselect** 節點失效的處理步驟

(1) 確認服務狀態

檢查 **vmselect** 容器或服務是否正在運行：

```
docker ps | grep vmselect
```

若未運行，嘗試啟動：

```
docker restart vmselect
```



若服務未啟動，檢查系統服務：

```
sudo systemctl status vmselect
sudo systemctl restart vmselect
```

(2) 檢查資源使用情況

- 確認伺服器的 CPU 和記憶體使用是否過載：

```
top          # 查看 CPU 和記憶體
```

- 如果資源不足，考慮優化配置或升級硬體。

(3) 檢查配置參數

確保 **vmselect** 的啟動參數正確配置，尤其是 `-storageNode` 是否正確指向 **vmstorage** 節點：

```
docker run -d \
  --name vmselect \
  -p 8481:8481 \
  victoriametrics/vmselect:<version> \
  -storageNode=<vmstorage_host>:8482
```

(4) 測試查詢功能

執行查詢測試，確認修復效果：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

如果查詢返回正確結果，則問題已解決。

(5) 替換失效節點

如果無法修復失效的 **vmselect** 節點，可以部署新節點替代：

1. 停止失效的節點。
2. 部署新的 **vmselect** 節點：



```
docker run -d \  
  --name vmselect-new \  
  -p 8481:8481 \  
  victoriametrics/vmselect:<version> \  
  -storageNode=<vmstorage_host>:8482
```

4. 優化與高可用性配置建議

(1) 部署多個 vmselect 節點

為實現高可用性，部署多個 **vmselect** 節點，並使用負載均衡器（如 Nginx 或 HAProxy）進行流量分配。

(2) 配置負載均衡器

使用 Nginx 作為負載均衡器，確保查詢流量能自動切換到健康的節點：

```
upstream vmselect {  
    server <vmselect1_host>:8481;  
    server <vmselect2_host>:8481;  
}  
  
server {  
    listen 8481;  
    location / {  
        proxy_pass http://vmselect;  
    }  
}
```

(3) 監控與告警

設置監控和告警規則，及時檢測 **vmselect** 節點問題：

- **vm_rpc_vmselect_is_reachable**：檢測節點是否可達。
- **vm_http_request_duration_seconds_sum**：監控請求的處理時間。



(4) 啟用自動重啟策略

為 **vmselect** 容器啟用自動重啟策略：

```
docker update --restart always vmselect
```

5. 排錯範例：實際操作步驟

假設有兩個 **vmselect** 節點，其中一個失效：

1. 確認失效節點的狀態：

```
docker ps | grep vmselect
```

2. 測試失效節點的健康檢查：

```
curl http://<failed_vmselect_host>:8481/health
```

3. 查看失效節點的日誌：

```
docker logs vmselect
```

4. 若無法修復，部署新節點替代：

```
docker run -d \  
  --name vmselect-new \  
  -p 8481:8481 \  
  victoriametrics/vmselect:<version> \  
  -storageNode=<vmstorage_host>:8482
```

6. 結論



當兩個 **vmselect** 節點之一失效時，查詢功能可能部分中斷，但可透過其他健康節點繼續提供服務。為減少影響，建議：

- 部署多個 **vmselect** 節點。
- 使用負載均衡器進行流量分配。
- 定期監控節點健康狀態並設置告警。

這些措施可提高集群的查詢高可用性，確保在單點故障發生時仍能維持穩定的數據查詢能力。



參考"Cluster version.pdf"，以繁體中文詳細說明若2個 **vmselect** 中的一個失效



當 **VictoriaMetrics Cluster** 中的兩個 **vmselect** 節點之一失效時，集群的數據查詢功能可能部分受到影響，但由於 **vmselect** 的高可用性架構，健康的節點仍可承擔查詢工作。以下根據《Cluster version.pdf》的內容詳細說明 **vmselect** 節點部分失效的影響、診斷方法以及處理步驟。

1. 部分 **vmselect** 節點失效的可能影響

1. 部分查詢重定向：

- 如果使用了負載均衡器，查詢流量會自動重定向到健康的 **vmselect** 節點，但負載可能增加。

2. 性能下降：

- 健康的節點需承擔全部查詢請求，可能導致響應時間增加。

3. 查詢中斷：

- 若數據來源或用戶直接連接到失效的節點，則該查詢可能會中斷或返回錯誤。





2. 診斷部分 vmselect 節點失效的方法

(1) 測試健康檢查端點

訪問失效節點的 `/health` 接口確認其狀態：

```
curl http://<vmselect_host>:8481/health
```

- 返回 `200 OK` 表示健康。
- 返回其他狀態或無響應則表明該節點失效。

(2) 測試查詢功能

對失效節點執行查詢測試：

```
curl "http://<failed_vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

- 若返回錯誤或無法連接，說明該節點已失效。

(3) 檢查負載均衡器行為

如果使用了負載均衡器（如 Nginx），檢查其日誌以確認流量是否被正確分配到健康的 **vmselect** 節點。

(4) 查看失效節點的日誌

檢查失效節點的運行日誌以診斷問題：

```
docker logs vmselect
```

關注以下錯誤：

- 與 **vmstorage** 節點連線失敗。
- 資源不足（記憶體不足、CPU 過載等）。



- 請求處理超時。

3. 部分 vmselect 節點失效的處理步驟

(1) 確認服務狀態

檢查失效的 **vmselect** 容器或服務是否正在運行：

```
docker ps | grep vmselect
```

若未運行，嘗試啟動：

```
docker restart vmselect
```

若服務未啟動，檢查系統服務：

```
sudo systemctl status vmselect  
sudo systemctl restart vmselect
```

(2) 檢查系統資源

- 確認伺服器的 CPU 和記憶體是否過載：

```
top          # 查看 CPU 和記憶體使用情況
```

- 如果資源不足，考慮升級伺服器硬體或優化查詢負載。

(3) 驗證配置

檢查 **vmselect** 的啟動參數是否正確配置：

- **-storageNode** 是否正確指向所有 **vmstorage** 節點。例如：

```
docker run -d \  
  --name vmselect \  
  --storageNode vmstorage1,vmstorage2,vmstorage3
```



```
-p 8481:8481 \  
victoriametrics/vmselect:<version> \  
-storageNode=<vmstorage1_host>:8482 \  
-storageNode=<vmstorage2_host>:8482
```

(4) 測試修復結果

重啟後測試查詢功能是否恢復：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

如果查詢成功，則問題已解決。

(5) 部署新節點（若無法修復）

如果無法修復失效的節點，可考慮部署一個新的 **vmselect** 節點替代：

1. 停止失效的節點。
2. 部署新節點：

```
docker run -d \  
  --name vmselect-new \  
  -p 8481:8481 \  
  victoriametrics/vmselect:<version> \  
  -storageNode=<vmstorage1_host>:8482 \  
  -storageNode=<vmstorage2_host>:8482
```

4. 優化與高可用性配置建議

(1) 使用負載均衡器

在所有 **vmselect** 節點前部署負載均衡器（如 Nginx 或 HAProxy），確保流量自動分配到健康節點。範例 Nginx 配置：



```
upstream vmselect {  
    server <vmselect1_host>:8481;  
    server <vmselect2_host>:8481;  
}  
  
server {  
    listen 8481;  
    location / {  
        proxy_pass http://vmselect;  
    }  
}
```

(2) 增加 vmselect 節點

增加 **vmselect** 節點數量以分散查詢流量，提高查詢性能和高可用性。

(3) 啟用自動重啟策略

為所有 **vmselect** 容器設置自動重啟：

```
docker update --restart always vmselect
```

(4) 定期監控與告警

設置監控指標和告警規則，提前檢測潛在問題：

- **vm_rpc_vmselect_is_reachable**：監控節點是否可達。
- **vm_http_request_duration_seconds_sum**：監控查詢處理時間。

5. 排錯範例：實際操作步驟

假設兩個 **vmselect** 節點之一失效：

1. 測試健康檢查：



```
curl http://<failed_vmselect_host>:8481/health
```

2. 查看失效節點日誌：

```
docker logs vmselect
```

3. 嘗試重啟失效節點：

```
docker restart vmselect
```

4. 如果無法修復，部署新節點：

```
docker run -d \  
  --name vmselect-new \  
  -p 8481:8481 \  
  victoriametrics/vmselect:<version> \  
  -storageNode=<vmstorage1_host>:8482 \  
  -storageNode=<vmstorage2_host>:8482
```

6. 結論

當兩個 **vmselect** 節點之一失效時，查詢功能會部分受影響，但負載均衡和高可用性設計能減少影響。建議：

1. 部署多個 **vmselect** 節點並使用負載均衡器。
2. 定期監控節點性能和健康狀態。
3. 啟用自動重啟策略以快速恢復失效節點。

透過上述措施，能有效降低單節點故障對集群的影響，確保數據查詢的穩定性和可靠性。



參考"Cluster version.pdf"，以繁體中文詳細說明怎麼debug VictoriaMetrics Cluster





以下是根據《Cluster version.pdf》內容整理的 **VictoriaMetrics Cluster** debug 方法，以繁體中文詳細說明如何排查和解決常見問題，包括 **vminsert**、**vmselect** 和 **vmstorage** 的診斷步驟。

1. 基礎準備

在開始 debug 之前，需確認以下條件：

1. **確保訪問權限**：能夠訪問所有集群節點的運行環境（Docker 或裸機）。
2. **了解系統配置**：熟悉 VictoriaMetrics 的部署架構，包括 ``vminsert``、``vmselect`` 和 ``vmstorage`` 的主機和端口配置。
3. **工具準備**：
 - 使用 ``curl`` 測試健康檢查和指標接口。
 - 日誌檢視工具（如 ``docker logs`` 或 ``journalctl``）。

2. 常見問題與排查步驟

(1) 數據寫入問題

問題描述：

- **vmagent** 或數據來源無法將數據寫入集群。

排查步驟：

1. **檢查 vminsert 健康狀態** 測試 ``vminsert`` 的健康檢查端點：

```
curl http://<vminsert_host>:8480/health
```





- 返回 `200 OK` 表示正常。
- 如果無響應或返回錯誤，需檢查 `vminsert` 是否正在運行：

```
docker ps | grep vminsert
```

- 若未運行，嘗試重啟：

```
docker restart vminsert
```

2. 檢查數據來源配置 確認數據來源的 `remoteWrite.url` 是否正確指向 **vminsert**：

```
http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write
```

3. 檢查 **vminsert** 日誌 查看 `vminsert` 是否報告與 `vmstorage` 的連線問題：

```
docker logs vminsert
```

4. 測試數據寫入 使用 `curl` 模擬數據寫入：

```
curl -X POST "http://<vminsert_host>:8480/insert/0/prometheus/api/v1/write" \
  -d '<data_payload>'
```

- 如果返回錯誤，檢查 `vmstorage` 是否健康（參見下一步）。

(2) 數據存儲問題

問題描述：

- **vminsert** 無法將數據存儲到 **vmstorage**，或 **vmstorage** 無法正常運行。

排查步驟：

1. 檢查 **vmstorage** 健康狀態 測試 `vmstorage` 的健康檢查端點：



```
curl http://<vmstorage_host>:8482/health
```

- 返回 `200 OK` 表示健康。
- 無響應則需檢查 `vmstorage` 是否運行：

```
docker ps | grep vmstorage
```

2. 檢查資源使用情況

- 確認磁碟空間是否不足：

```
df -h
```

- 檢查 CPU 和記憶體使用：

```
top
```

3. 檢查 **vmstorage** 日誌 查看 `vmstorage` 的運行日誌：

```
docker logs vmstorage
```

- 若出現磁碟空間錯誤，需清理或擴展存儲。
- 若報告連線失敗，檢查網路配置。

4. 測試與 **vminsert** 的連線 確認 `vminsert` 能連接到所有 `vmstorage` 節點：

```
curl http://<vmstorage_host>:8482/metrics
```

(3) 數據查詢問題

問題描述：

- 查詢數據返回錯誤或結果不完整，可能涉及 **vmselect** 節點。



排查步驟：

1. 檢查 **vmselect** 健康狀態

測試 `vmselect` 的健康檢查端點：

```
curl http://<vmselect_host>:8481/health
```

- 返回 `200 OK` 表示健康。
- 無響應則需檢查 `vmselect` 是否運行：

```
docker ps | grep vmselect
```

2. 檢查 **vmselect** 與 **vmstorage** 的連線

確認 `vmselect` 能正確訪問所有 `vmstorage` 節點：

```
curl http://<vmstorage_host>:8482/metrics
```

3. 檢查 **vmselect** 日誌

查看 `vmselect` 的運行日誌，查找查詢超時或連線錯誤信息：

```
docker logs vmselect
```

4. 測試查詢功能

使用 `curl` 測試查詢功能：

```
curl "http://<vmselect_host>:8481/select/0/prometheus/api/v1/query?query=up"
```

- 若查詢失敗，檢查 PromQL 語法和數據範圍。

(4) 整體性能問題

問題描述：

- 集群性能下降，如查詢延遲增加、寫入速度減慢等。



排查步驟：

1. **檢查指標數據** 在 `vmselect` 和 `vminsert` 上檢查性能相關指標：
 - **vm_http_request_duration_seconds_sum**：請求處理時間。
 - **vm_http_requests_total**：請求總數。
 - **vm_rpc_vmstorage_is_reachable**：檢測節點連線狀態。
 2. **監控資源使用**
 - 檢查所有節點的資源使用情況，包括 CPU、記憶體和磁碟。
 3. **優化 PromQL 查詢** 確認查詢是否過於複雜或消耗資源，優化 PromQL 語法。
-

3. 高效 Debug 技巧

1. **分層排查：**
 - 從數據來源（**vmagent**）到數據存儲（**vmstorage**）再到數據查詢（**vmselect**），逐步檢查。
 2. **使用負載均衡：**
 - 為多個 `vmselect` 和 `vminsert` 節點配置負載均衡器（如 Nginx），確保查詢和寫入的高可用性。
 3. **啟用指標和告警：**
 - 使用 Prometheus 和 Grafana 監控集群的健康狀況，及時發現問題。
-

4. 結論



Debug **VictoriaMetrics Cluster** 時，需先確定問題發生的位置（寫入、存儲或查詢），再根據節點類型逐步排查。通過健康檢查接口、日誌檢視及性能指標監控，可以快速找到並解決問題。同時，建議啟用高可用性配置（多節點部署和負載均衡）以減少故障影響。



參考"Cluster version.pdf"，以繁體中文詳細說明若改用 docker-compose 建立 cluster VictoriaMetrics



以下是參考《Cluster version.pdf》內容，詳細說明如何使用 **Docker Compose** 建立 **VictoriaMetrics Cluster** 的步驟。

1. 準備環境

(1) 安裝必要軟體

確保已安裝以下工具：

- **Docker**：提供容器化運行環境。
- **Docker Compose**：簡化多容器部署的工具。

(2) 建立專案目錄

創建一個專案目錄以存放 `docker-compose.yml` 配置文件及相關資料：

```
mkdir victoria-metrics-cluster  
cd victoria-metrics-cluster
```

2. 建立 docker-compose.yml 文件



在專案目錄內新建 `docker-compose.yml` 文件，定義 **vminsert**、**vmselect** 和 **vmstorage** 的服務配置。以下是一個基本的範例：

```
version: '3.7'

services:
  vmstorage:
    image: victoriametrics/vmstorage:v1.93.2
    container_name: vmstorage
    ports:
      - "8482:8482"
    volumes:
      - ./storage:/storage
    command:
      - "-retentionPeriod=30"
      - "-storageDataPath=/storage"
    restart: always

  vminsert:
    image: victoriametrics/vminsert:v1.93.2
    container_name: vminsert
    ports:
      - "8480:8480"
    depends_on:
      - vmstorage
    command:
      - "-storageNode=vmstorage:8482"
    restart: always

  vmselect:
    image: victoriametrics/vmselect:v1.93.2
    container_name: vmselect
    ports:
      - "8481:8481"
    depends_on:
      - vmstorage
    command:
      - "-storageNode=vmstorage:8482"
    restart: always
```



文件解釋：

1. ``vmstorage``：

- 負責數據的存儲。
- 使用 ``-retentionPeriod`` 設置數據保留天數（此處為 30 天）。
- 將本地目錄 `./storage`` 掛載到容器內的 ``/storage``。

2. ``vminsert``：

- 負責數據寫入，並將數據傳遞至 ``vmstorage``。
- 使用 ``-storageNode`` 指向 ``vmstorage`` 節點。

3. ``vmselect``：

- 負責數據查詢，並從 ``vmstorage`` 中檢索數據。
- 使用 ``-storageNode`` 指向 ``vmstorage`` 節點。

3. 啟動集群

在專案目錄中運行以下命令啟動集群：

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

驗證服務運行：

1. 檢查容器狀態：

```
docker-compose ps
```

確認所有服務的狀態為 ``up``。

2. 訪問健康檢查端點：



- **vminsert** : `http://localhost:8480/health`
- **vmselect** : `http://localhost:8481/health`
- **vmstorage** : `http://localhost:8482/health` 若返回 `200 OK` 表示服務正常。

4. 測試集群功能

(1) 測試數據寫入

使用 `curl` 模擬數據寫入至 **vminsert** :

```
curl -X POST "http://localhost:8480/insert/0/prometheus/api/v1/write" \  
-d '<data_payload>'
```

(2) 測試數據查詢

通過 **vmselect** 查詢數據 :

```
curl "http://localhost:8481/select/0/prometheus/api/v1/query?query=up"
```

5. 擴展配置 (多節點)

如果需要支持更高的負載，可以擴展 **vminsert** 和 **vmselect** 節點。

更新 **docker-compose.yml** 文件 :

新增多個 `vminsert` 和 `vmselect` 節點，並使用負載均衡器分配流量。

範例 :

```
vminsert1:  
  image: victoriametrics/vminsert:v1.93.2  
  container_name: vminsert1
```



```
ports:
  - "8483:8480"
depends_on:
  - vmstorage
command:
  - "-storageNode=vmstorage:8482"
restart: always

vmselect1:
  image: victoriametrics/vmselect:v1.93.2
  container_name: vmselect1
  ports:
    - "8484:8481"
  depends_on:
    - vmstorage
  command:
    - "-storageNode=vmstorage:8482"
  restart: always
```

6. 停止與清理

停止集群：

```
docker-compose down
```

清理數據（可選）：

刪除存儲目錄內的數據以重置環境：

```
rm -rf ./storage
```

7. 建議與最佳實踐



1. **資料持久化**：確保 `./storage`` 目錄存放於可靠的存儲設備上，以防數據丟失。
2. **監控和告警**：
 - 使用 **Prometheus** 監控各服務的健康狀況和性能。
 - 在 **Grafana** 中設置儀表板，視覺化數據指標。
3. **高可用性**：
 - 部署多個節點的 **vminsert** 和 **vmselect**，並使用負載均衡器（如 Nginx）分配流量。
4. **定期備份**：定期備份存儲目錄以防止數據損失。

以上步驟可協助使用 **Docker Compose** 快速部署和管理 **VictoriaMetrics Cluster**，滿足監控與數據存儲需求。



詳細說明:怎麼改成k8s上，透過statefulset部署vminsert



在 Kubernetes 上使用 StatefulSet 部署 vminsert (VictoriaMetrics Cluster)

1. 概述

在 **Kubernetes (K8s)** 中，``vminsert`` 負責接收監控數據並將其寫入 ``vmstorage``。為確保 ``vminsert`` 在 K8s 中穩定運行並具備持久性，我們將使用 **StatefulSet** 進行部署，以確保：

- ``vminsert`` 節點具備 **固定的 DNS 名稱**，便於 ``vmagent`` 發送數據。



- 多個 `vminsert` 副本 可水平擴展，提高吞吐量。
- 持久化 `vminsert` 的日誌和配置（儘管 `vminsert` 本身為無狀態應用）。

2. 建立 StatefulSet 配置

我們將使用 `StatefulSet` 部署 **vminsert**，並確保它能與 **vmstorage** 正確連接。

(1) 建立 ConfigMap

ConfigMap 用於存儲 `vminsert` 的啟動參數，如 `storageNode` 的 URL。

創建 `vminsert-configmap.yaml`：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: vminsert-config
  namespace: victoria-metrics
data:
  vminsert-flags: |
    -storageNode=vmstorage-0.vmstorage:8482
    -storageNode=vmstorage-1.vmstorage:8482
```

部署 ConfigMap：

```
kubectl apply -f vminsert-configmap.yaml
```

(2) 建立 StatefulSet

創建 `vminsert-statefulset.yaml`：



```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: vminsert
  namespace: victoria-metrics
spec:
  serviceName: "vminsert"
  replicas: 2
  selector:
    matchLabels:
      app: vminsert
  template:
    metadata:
      labels:
        app: vminsert
    spec:
      containers:
        - name: vminsert
          image: victoriametrics/vminsert:v1.93.2
          args:
            - "--storageNode=$(STORAGE_NODES)"
          env:
            - name: STORAGE_NODES
              valueFrom:
                configMapKeyRef:
                  name: vminsert-config
                  key: vminsert-flags
      ports:
        - containerPort: 8480
          name: http
      readinessProbe:
        httpGet:
          path: /health
          port: 8480
        initialDelaySeconds: 5
        periodSeconds: 10
      livenessProbe:
        httpGet:
          path: /health
          port: 8480
```



```
    initialDelaySeconds: 15
    periodSeconds: 20
  resources:
    requests:
      cpu: "500m"
      memory: "512Mi"
    limits:
      cpu: "1000m"
      memory: "1Gi"
  terminationGracePeriodSeconds: 10
```

此 `StatefulSet` 配置：

- `replicas: 2`：創建兩個 `vminsert` 副本。
- `serviceName: "vminsert"`：確保 `vminsert` 節點有固定的 DNS 名稱。
- 環境變數 `STORAGE_NODES`：從 `ConfigMap` 讀取 `vmstorage` 節點地址。
- 健康檢查 (`livenessProbe` & `readinessProbe`)：確保 `vminsert` 運行正常。

部署 StatefulSet：

```
kubectl apply -f vminsert-statefulset.yaml
```

(3) 建立 Headless Service

為 `vminsert` 創建一個無頭服務 (Headless Service)，以便 `vmagent` 能夠動態發現 `vminsert` 節點。

創建 `vminsert-service.yaml`：

```
apiVersion: v1
kind: Service
metadata:
  name: vminsert
  namespace: victoria-metrics
```



```
spec:
  ports:
    - port: 8480
      targetPort: 8480
      protocol: TCP
      name: http
  clusterIP: None
  selector:
    app: vminsert
```

部署 Service :

```
kubectl apply -f vminsert-service.yaml
```

3. 測試與驗證

(1) 驗證 Pod 狀態

檢查 `vminsert` 是否成功部署：

```
kubectl get pods -n victoria-metrics
```

應該看到類似輸出的結果：

NAME	READY	STATUS	RESTARTS	AGE
vminsert-0	1/1	Running	0	10s
vminsert-1	1/1	Running	0	10s

(2) 測試 vminsert 健康狀態

```
kubectl port-forward svc/vminsert 8480 -n victoria-metrics
curl http://localhost:8480/health
```



返回 `200 OK` 則表示 `vminsert` 正常運行。

(3) 測試數據寫入

模擬 **vmagent** 或 Prometheus 進行數據寫入：

```
curl -X POST "http://vminsert-0.vminsert:8480/insert/0/prometheus/api/v1/write" \
  -d '<data_payload>'
```

4. 擴展與最佳實踐

(1) 水平擴展 vminsert

如果需要處理更大流量，可以調整 `replicas` 參數：

```
kubectl scale statefulset vminsert --replicas=3 -n victoria-metrics
```

(2) 使用 `vmagent` 自動發現 vminsert

在 `vmagent` 配置文件中，使用 `kubernetes_sd_config` 自動發現 `vminsert`：

```
scrape_configs:
  - job_name: 'vminsert'
    kubernetes_sd_configs:
      - role: endpoints
        namespaces:
          names:
            - victoria-metrics
```

(3) 監控與告警

可以使用 **Prometheus + Grafana** 監控 `vminsert`：

- 監控指標：



- `vm_http_requests_total` (HTTP 請求總數)
- `vm_rpc_vmstorage_is_reachable` (vmstorage 連線狀態)

5. 停止與刪除

如果需要刪除 `vminsert`：

```
kubectl delete statefulset vminsert -n victoria-metrics
kubectl delete service vminsert -n victoria-metrics
kubectl delete configmap vminsert-config -n victoria-metrics
```

6. 總結

組件	角色
StatefulSet	確保 `vminsert` 具備固定 DNS 名稱和擴展能力。
ConfigMap	設定 `storageNode`，確保 `vminsert` 連接 `vmstorage`。
Headless Service	確保 `vmagent` 能自動發現 `vminsert` 節點。
水平擴展	可根據需求增加 `vminsert` 節點，提高寫入能力。

透過這些步驟，即可在 **Kubernetes** 上透過 **StatefulSet** 部署 **vminsert**，並確保其高可用性與可擴展性！🚀



詳細說明:怎麼改成k8s上，透過statefulset部屬vmagent





在 Kubernetes 上使用 StatefulSet 部署 vmagent (VictoriaMetrics Cluster)

1. 概述

在 **Kubernetes (K8s)** 環境中，`vmagent` 負責從監控目標（如 **Prometheus**、**Node Exporter**、**cAdvisor**）收集數據，並將其發送至 `vminsert`，以寫入 **VictoriaMetrics Cluster**。

使用 **StatefulSet** 部署 `vmagent` 具有以下優勢：

- 具備固定的 **DNS** 名稱，便於 `vmagent` 在多副本運行時能被可靠定位。
- 多副本支持（可水平擴展），增加 `vmagent` 的抓取能力，避免單點故障。
- 持久化存儲 `vmagent` 緩存數據，以防止數據丟失。

2. 建立 Kubernetes 配置文件

我們將使用 **ConfigMap** 存儲 `vmagent` 的 **scrape** 配置，並使用 **StatefulSet** 部署 `vmagent`。

(1) 創建 ConfigMap

ConfigMap 用於存儲 `vmagent` 的抓取目標（如 `Node Exporter`、`cAdvisor`、`kubelet`）。

創建 `vmagent-configmap.yaml`：



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: vmagent-config
  namespace: victoria-metrics
data:
  vmagent.yml: |
    global:
      scrape_interval: 15s
      scrape_timeout: 10s
    scrape_configs:
      - job_name: 'kubernetes-nodes'
        kubernetes_sd_configs:
          - role: node
      - job_name: 'kubernetes-pods'
        kubernetes_sd_configs:
          - role: pod
      - job_name: 'vminsert'
        static_configs:
          - targets: ['vminsert.victoria-metrics.svc.cluster.local:8480']
```

部署 ConfigMap :

```
kubectl apply -f vmagent-configmap.yaml
```

(2) 創建 StatefulSet

創建 `vmagent-statefulset.yaml` :

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: vmagent
  namespace: victoria-metrics
```



```
spec:
  serviceName: "vmagent"
  replicas: 2
  selector:
    matchLabels:
      app: vmagent
  template:
    metadata:
      labels:
        app: vmagent
    spec:
      volumes:
        - name: config-volume
          configMap:
            name: vmagent-config
      containers:
        - name: vmagent
          image: victoriametrics/vmagent:v1.93.2
          args:
            - "-remoteWrite.url=http://vminsert.victoria-
metrics.svc.cluster.local:8480/insert/0/prometheus/api/v1/write"
            - "-promscrape.config=/etc/vmagent/vmagent.yml"
          volumeMounts:
            - name: config-volume
              mountPath: /etc/vmagent/
          ports:
            - containerPort: 8429
              name: http
          readinessProbe:
            httpGet:
              path: /targets
              port: 8429
              initialDelaySeconds: 5
              periodSeconds: 10
          livenessProbe:
            httpGet:
              path: /metrics
              port: 8429
              initialDelaySeconds: 15
              periodSeconds: 20
```



```
resources:
  requests:
    cpu: "500m"
    memory: "512Mi"
  limits:
    cpu: "1000m"
    memory: "1Gi"
terminationGracePeriodSeconds: 10
```

說明：

- `replicas: 2`：啟動兩個 `vmagent` 副本。
- `-remoteWrite.url`：指定 `vminsert` 節點作為數據存儲目標。
- `-promscrape.config=/etc/vmagent/vmagent.yml`：從 `ConfigMap` 加載 `vmagent` 配置。
- **健康檢查 (`readinessProbe` & `livenessProbe`)**：
 - `/targets` 用於檢查 `vmagent` 是否正確抓取數據。
 - `/metrics` 用於監控 `vmagent` 的健康狀態。

部署 StatefulSet：

```
kubectl apply -f vmagent-statefulset.yaml
```

(3) 創建 Headless Service

為 `vmagent` 創建 **無頭服務 (Headless Service)**，確保 `vmagent` 節點能夠被 `kubernetes_sd_configs` 自動發現。

創建 `vmagent-service.yaml`：

```
apiVersion: v1
kind: Service
```



```
metadata:
  name: vmagent
  namespace: victoria-metrics
spec:
  ports:
    - port: 8429
      targetPort: 8429
      protocol: TCP
      name: http
  clusterIP: None
  selector:
    app: vmagent
```

部署 Service :

```
kubectl apply -f vmagent-service.yaml
```

3. 測試與驗證

(1) 確認 vmagent 是否成功運行

```
kubectl get pods -n victoria-metrics
```

應該看到類似的輸出：

NAME	READY	STATUS	RESTARTS	AGE
vmagent-0	1/1	Running	0	10s
vmagent-1	1/1	Running	0	10s

(2) 測試 vmagent 健康狀態



```
kubectl port-forward svc/vmagent 8429 -n victoria-metrics  
curl http://localhost:8429/metrics
```

如果返回 Prometheus 指標，表示 `vmagent` 正常運行。

(3) 測試數據寫入

模擬數據寫入到 `vminsert`：

```
curl -X POST "http://vminsert.victoria-  
metrics.svc.cluster.local:8480/insert/0/prometheus/api/v1/write" \  
-d '<data_payload>'
```

4. 擴展與最佳實踐

(1) 水平擴展 vmagent

如需增加 `vmagent` 副本數量，提高數據抓取能力：

```
kubectl scale statefulset vmagent --replicas=3 -n victoria-metrics
```

(2) 監控 vmagent

可在 **Prometheus** 中監控以下 `vmagent` 指標：

- `vm_promscrape_targets`：監控抓取的目標數量。
- `vm_promscrape_scrapes_total`：監控成功的數據抓取次數。

(3) 使用 `kubernetes_sd_configs` 自動發現 vmagent

在 Prometheus 配置中，啟用 `kubernetes_sd_configs` 以自動發現 `vmagent`：



```
scrape_configs:
  - job_name: 'vmagent'
    kubernetes_sd_configs:
      - role: endpoints
        namespaces:
          names:
            - victoria-metrics
```

(4) 設置告警

可以設置 `vmagent` 失效的告警：

```
ALERT vmagent_down
  IF up{job="vmagent"} == 0
  FOR 5m
  LABELS {severity="critical"}
  ANNOTATIONS {
    summary = "vmagent is down",
    description = "No vmagent instance has been up for the last 5 minutes."
  }
```

5. 停止與刪除

如果需要刪除 `vmagent`：

```
kubectl delete statefulset vmagent -n victoria-metrics
kubectl delete service vmagent -n victoria-metrics
kubectl delete configmap vmagent-config -n victoria-metrics
```



6. 總結

組件	角色
StatefulSet	確保 `vmagent` 具備固定 DNS 名稱與擴展能力。
ConfigMap	存儲 `vmagent` 抓取目標的設定。
Headless Service	確保 Prometheus 能自動發現 `vmagent`。
水平擴展	增加 `vmagent` 副本，提高數據抓取能力。

透過這些步驟，即可在 **Kubernetes** 上透過 **StatefulSet** 部署 **vmagent**，並確保其高可用性與可擴展性！🚀