



使用vmagent替代Prometheus



使用vmagent 代替Prometheus ...
PDF

參考"使用vmagent 代替Prometheus 採集監控指標- 技術顏良- 博客園.pdf"，以繁體中文詳細說明



使用 vmagent 代替 Prometheus 採集監控指標 詳細說明

vmagent 是一款能夠替代 **Prometheus** 進行監控指標採集的工具，它能夠從多種來源收集指標，並將這些指標存儲到 **VictoriaMetrics (VM)** 或任何支援 **Prometheus Remote Write** 協議的存儲系統中。

1. vmagent 介紹與特性

相比於 **Prometheus**，vmagent 具有更靈活的監控指標採集方式，例如：

1. 支援 Pull（拉取）與 Push（推送）

- **Prometheus** 主要採用 **pull 模式**（主動抓取指標）。
- **vmagent** 除了可以 **pull** 也可以 **push**，適用於動態環境。

2. 替代 Prometheus 直接作為 scraping target

- vmagent 可以作為一個 **scrape target**，負責收集來自不同來源的數據，再統一發送到遠端存儲。

3. 支援 Kafka 讀寫數據

- 可以從 **Kafka** 讀取監控數據，並將數據寫入 **Kafka**。





4. 支援 Prometheus relabeling

- 可以在數據發送到遠端存儲前，**修改、刪除或新增 labels**，提高數據處理靈活性。

5. 支援多種數據協議

- **InfluxDB line**
- **Graphite**
- **OpenTSDB**
- **Prometheus Remote Write**
- **JSON Lines**
- **CSV**

6. 多種存儲系統支援

- 支援數據存儲到 **多個遠端存儲系統**，提高數據冗餘與可靠性。

7. 緩衝機制

- 若遠端存儲不可用，vmagent 會將指標緩存至 `-remoteWrite.tmpDataPath``，當遠端存儲恢復後，會自動將數據發送過去。

8. 降低資源消耗

- 相較於 **Prometheus**，vmagent 使用更少的 **CPU、記憶體、磁碟 I/O 和網路頻寬**，特別適合大規模監控場景。

9. 支援高並發與分佈式架構

- vmagent 可以部署多個實例，將監控目標分配到不同的 **vmagent** 實例，以減少單點壓力。

2. 在 Kubernetes 環境中部署 vmagent





2.1. 設定 RBAC 權限

vmagent 需要從 Kubernetes API Server 獲取資源資訊，因此需要設定 **RBAC (Role-Based Access Control)** 權限。

建立 **RBAC 資源清單 (vmagent-rbac.yaml)**：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vmagent
  namespace: kube-vm
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: vmagent
rules:
  - apiGroups: ["", "networking.k8s.io", "extensions"]
    resources:
      - nodes
      - nodes/metrics
      - services
      - endpoints
      - endpointslices
      - pods
      - app
      - ingresses
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources:
      - namespaces
      - configmaps
    verbs: ["get"]
  - nonResourceURLs: ["/metrics", "/metrics/resources"]
    verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
```



```
name: vmagent
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: vmagent
subjects:
- kind: ServiceAccount
  name: vmagent
  namespace: kube-vm
```

2.2. 設定 vmagent 配置

使用 **ConfigMap** 來管理 **scrape** 設定，如下：

建立 **vmagent 配置 (vmagent-config.yaml)**：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: vmagent-config
  namespace: kube-vm
data:
  scrape.yml: |
    global:
      scrape_interval: 15s
      scrape_timeout: 15s
    scrape_configs:
      - job_name: nodes
        kubernetes_sd_configs:
          - role: node
        relabel_configs:
          - source_labels: [__address__]
            regex: "(.*):10250"
            replacement: "${1}:9111"
            target_label: __address__
            action: replace
```



```
- action: labelmap
  regex: __meta_kubernetes_node_label_(.+)
```

這裡的 **kubernetes_sd_configs** 設定讓 **vmagent** 可以透過 Kubernetes API 自動發現 **node** 角色的目標。

2.3. 部署 vmagent

建立 **vmagent** 部署資源 (**vmagent-deploy.yaml**)：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vmagent
  namespace: kube-vm
  labels:
    app: vmagent
spec:
  selector:
    matchLabels:
      app: vmagent
  template:
    metadata:
      labels:
        app: vmagent
    spec:
      serviceAccountName: vmagent
      containers:
        - name: agent
          image: "victoriametrics/vmagent:v1.77.0"
          imagePullPolicy: IfNotPresent
          args:
            - -promscrape.config=/config/scrape.yml
            - -remoteWrite.tmpDataPath=/tmpData
            - -remoteWrite.url=http://vminsert:8480/insert/0/prometheus
            - -envflag.enable=true
```



```
- -envflag.prefix=VM_
- -loggerFormat=json
ports:
  - name: http
    containerPort: 8429
volumeMounts:
  - name: tmpdata
    mountPath: /tmpData
  - name: config
    mountPath: /config
volumes:
  - name: tmpdata
    emptyDir: {}
  - name: config
    configMap:
      name: vmagent-config
```

3. 使用 vmagent 集群模式

當需要監控大量目標時，可以部署多個 **vmagent** 實例，每個實例負責部分監控目標。

3.1. 集群配置

```
args:
  - -promscrape.cluster.membersCount=3
  - -promscrape.cluster.memberNum=0
  - -remoteWrite.url=http://vminsert:8480/insert/0/prometheus
```

- ``-promscrape.cluster.membersCount=3``：定義 **vmagent** 集群共有 3 個實例。
- ``-promscrape.cluster.memberNum=0``：當前 **vmagent** 實例的 ID 為 0。



4. 刷新 vmagent 配置

在 Kubernetes 環境下，有兩種方式可以刷新 **vmagent** 配置：

1. 發送 SIGHUP 訊號

```
kill -SIGHUP <vmagent-pid>
```

2. 發送 HTTP 請求

```
curl -X POST http://vmagent:8429/-/reload
```

5. 監控 vmagent 自身

vmagent 提供自身的監控指標：

- `http://vmagent:8429/metrics`：可透過 Prometheus 或 Grafana 監控 **vmagent** 狀態。

總結

- vmagent 可替代 **Prometheus**，減少資源消耗。
- 支援 **Pull + Push**，並可與 Kafka、InfluxDB 等協議整合。
- 在 **Kubernetes** 環境，可搭配 **ConfigMap + StatefulSet** 部署。
- 透過 **集群模式** 提高擴展能力。

這樣的架構特別適合 **大規模監控**，同時降低系統資源消耗！🚀



我想要from vmagent的image寫自己的dockerfile，promscrape.cluster.memberNum怎麼使用變數



如果你想要自訂 `vmagent` 的 Dockerfile，並且讓 `-promscrape.cluster.memberNum` 參數可以使用環境變數，這可以透過 **entrypoint script** 或 **Docker ARG/ENV** 來實現。以下是完整的解法：

1. 自訂 Dockerfile

在 Dockerfile 中，你可以這樣設定：

```
FROM victoriametrics/vmagent:v1.77.0

# 設定工作目錄
WORKDIR /vmagent

# 複製自訂的配置檔案（可選）
COPY scrape.yml /vmagent/scrape.yml

# 設定環境變數（如果沒指定，則預設為0）
ENV VMAGENT_MEMBER_NUM 0
ENV VMAGENT_CLUSTER_SIZE 3

# 透過 entrypoint script 來處理變數
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]
```




2. 編寫 `entrypoint.sh` 來動態設定 `memberNum`

在 `entrypoint.sh` 中，使用 **環境變數** 來指定 `promscrape.cluster.memberNum`：

```
#!/bin/sh

# 設定 cluster memberNum，使用環境變數，如果沒設定則預設為0
VMAGENT_MEMBER_NUM=${VMAGENT_MEMBER_NUM:-0}
VMAGENT_CLUSTER_SIZE=${VMAGENT_CLUSTER_SIZE:-3}

exec /vmagent-prod \
  -promscrape.config=/vmagent/scrape.yml \
  -remoteWrite.url=http://vminsert:8480/insert/0/prometheus \
  -promscrape.cluster.membersCount=${VMAGENT_CLUSTER_SIZE} \
  -promscrape.cluster.memberNum=${VMAGENT_MEMBER_NUM} \
  "$@"
```

- `VMAGENT_MEMBER_NUM` 預設為 `0`，可以透過 **環境變數** 來覆寫。
- `VMAGENT_CLUSTER_SIZE` 預設為 `3`，同樣可以調整。
- `exec` 用來確保 `vmagent` 正常運行，並允許外部參數。

3. 構建與執行 Docker 容器

3.1. 構建映像

```
docker build -t my-vmagent .
```

3.2. 使用環境變數啟動容器

```
docker run -d \
  -e VMAGENT_MEMBER_NUM=1 \
```



```
-e VMAGENT_CLUSTER_SIZE=5 \  
--name my-vmagent my-vmagent
```

這樣，`-promscrape.cluster.memberNum` 會動態設為 **1**，`-promscrape.cluster.membersCount` 設為 **5**。

4. 在 Kubernetes 中動態設定

如果你要在 **Kubernetes** 中使用 `StatefulSet` 部署，可以透過 **環境變數** 和 **Pod 名稱** 來自動分配 `memberNum`。

4.1. Kubernetes Deployment (StatefulSet)

```
apiVersion: apps/v1  
kind: StatefulSet  
metadata:  
  name: vmagent  
  namespace: kube-vm  
spec:  
  replicas: 3  
  serviceName: vmagent  
  selector:  
    matchLabels:  
      app: vmagent  
  template:  
    metadata:  
      labels:  
        app: vmagent  
    spec:  
      containers:  
        - name: vmagent  
          image: my-vmagent  
          env:  
            - name: VMAGENT_MEMBER_NUM  
              valueFrom:
```



```
fieldRef:
  fieldPath: metadata.name
- name: VMAGENT_CLUSTER_SIZE
  value: "3"
```

這裡使用 `metadata.name` 來動態設定 **VMAGENT_MEMBER_NUM**，例如：

- `vmagent-0` → `memberNum=0`
- `vmagent-1` → `memberNum=1`
- `vmagent-2` → `memberNum=2`

總結

1. 使用 **Dockerfile** 搭配 **entrypoint.sh** 來讓 `promsrape.cluster.memberNum` 支援環境變數。
2. 使用 `docker run -e` 來設定不同的 `memberNum`。
3. 在 **Kubernetes StatefulSet** 中動態設定 `memberNum` 透過 Pod 名稱來確保正確的 `memberNum` 分配。

這樣可以讓 `vmagent` 在 **動態環境** 中更加靈活 🚀！



詳細說明:entrypoint.sh，有3個pod，這樣會是membernum 多少呢



1. `entrypoint.sh` 詳細解析

在 `entrypoint.sh` 中，我們會根據環境變數 **動態設定** `promsrape.cluster.memberNum`，這樣每個 `vmagent` pod 啟動時都能獲得正確的 `memberNum`。



1.1. `entrypoint.sh` 內容

```
#!/bin/sh

# 設定 cluster memberNum, 使用環境變數, 如果沒設定則預設為 0
VMAGENT_MEMBER_NUM=${VMAGENT_MEMBER_NUM:-0}
VMAGENT_CLUSTER_SIZE=${VMAGENT_CLUSTER_SIZE:-3}

echo "Starting vmagent with cluster settings:"
echo "  Members Count: ${VMAGENT_CLUSTER_SIZE}"
echo "  Member Num: ${VMAGENT_MEMBER_NUM}"

exec /vmagent-prod \
  -promscrape.config=/vmagent/scrape.yml \
  -remoteWrite.url=http://vminsert:8480/insert/0/prometheus \
  -promscrape.cluster.membersCount=${VMAGENT_CLUSTER_SIZE} \
  -promscrape.cluster.memberNum=${VMAGENT_MEMBER_NUM} \
  "$@"
```

2. 當有 3 個 pod 時, `memberNum` 的分配

在 **Kubernetes StatefulSet** 中, 每個 `vmagent` pod 會有唯一的名稱, 例如:

```
vmagent-0
vmagent-1
vmagent-2
```

這些名稱可以用來動態設定 `VMAGENT_MEMBER_NUM`。

3. 如何讓 `entrypoint.sh` 自動設定 `memberNum`



在 `StatefulSet` 中，我們會設定：

```
env:
  - name: VMAGENT_MEMBER_NUM
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
```

這會讓 `VMAGENT_MEMBER_NUM` 取得 pod 的名稱，例如：

- `vmagent-0` 取得 `vmagent-0`
- `vmagent-1` 取得 `vmagent-1`
- `vmagent-2` 取得 `vmagent-2`

但這樣的變數值是 **字串**，我們需要 **取最後的數字** 來當作 `memberNum`。

4. 修改 `entrypoint.sh` 來解析 pod 名稱

```
#!/bin/sh

# 確保環境變數 VMAGENT_MEMBER_NUM 存在，否則預設為 vmagent-0
POD_NAME=${VMAGENT_MEMBER_NUM:-vmagent-0}

# 透過 shell 擷取 pod 名稱最後的數字 (memberNum)
VMAGENT_MEMBER_NUM=$(echo $POD_NAME | grep -oE '[0-9]+$')

# 設定 cluster 總數 (可用環境變數自訂，預設為 3)
VMAGENT_CLUSTER_SIZE=${VMAGENT_CLUSTER_SIZE:-3}

echo "Starting vmagent with cluster settings:"
echo "  Members Count: ${VMAGENT_CLUSTER_SIZE}"
echo "  Member Num: ${VMAGENT_MEMBER_NUM}"

exec /vmagent-prod \
```



```
-promscrape.config=/vmagent/scrape.yml \  
-remoteWrite.url=http://vminsert:8480/insert/0/prometheus \  
-promscrape.cluster.membersCount=${VMAGENT_CLUSTER_SIZE} \  
-promscrape.cluster.memberNum=${VMAGENT_MEMBER_NUM} \  
"$@"
```

5. pod 啟動時，`memberNum` 會是？

假設 `StatefulSet` 有 3 個 pod：

Pod 名稱	`memberNum` 設定
vmagent-0	0
vmagent-1	1
vmagent-2	2

這樣 `promscrape.cluster.memberNum` 會正確分配到 **0、1、2**，確保多個 `vmagent` 實例能夠協同運行。

6. Kubernetes `StatefulSet` 設定

在 `StatefulSet` 中，我們使用：

```
env:  
- name: VMAGENT_MEMBER_NUM  
  valueFrom:  
    fieldRef:  
      fieldPath: metadata.name  
- name: VMAGENT_CLUSTER_SIZE  
  value: "3"
```



這樣 **每個 pod** 會自動根據名稱取得 ``memberNum``，不用手動設定。

7. 如何驗證？

啟動 ``vmagent`` 後，執行：

```
kubectl logs -f vmagent-0 -n kube-vm
kubectl logs -f vmagent-1 -n kube-vm
kubectl logs -f vmagent-2 -n kube-vm
```

你應該會看到類似輸出：

```
Starting vmagent with cluster settings:
Members Count: 3
Member Num: 0
```

```
Starting vmagent with cluster settings:
Members Count: 3
Member Num: 1
```

```
Starting vmagent with cluster settings:
Members Count: 3
Member Num: 2
```

總結

- ✅ ``entrypoint.sh`` 會根據 **pod 名稱** 自動設定 ``memberNum``
- ✅ ``memberNum`` 分配為 **0, 1, 2**，確保分散工作負載
- ✅ ``StatefulSet`` 可自動管理 ``memberNum``，無需手動修改配置



這樣，**vmagent** 就能 自動化擴展，避免 ``promscrape.cluster.memberNum`` 需要手動配置的問題 🚀