



Prometheus Remote Write vs Federation



prometheus remote write、 federation



Prometheus 的 **Remote Write** 和 **Federation** 是两种用于分布式监控和数据共享的机制。以下是它们的功能和区别：

1. Prometheus Remote Write

功能：

Remote Write 是 Prometheus 用于将时间序列数据发送到其他存储后端的功能，适用于以下场景：

- 将监控数据转发到集中式存储。
- 长期存储和分析（例如对接 Grafana Mimir、VictoriaMetrics、Thanos、Cortex 等系统）。
- 实现多集群数据汇聚和共享。

原理：

Prometheus 使用 HTTP 协议将样本数据以指定格式推送到配置的远程存储端点。

配置示例：

```
remote_write:
- url: "http://remote-storage-endpoint/api/v1/write"
  write_relabel_configs:
  - source_labels: [__name__]
    regex: "node_.*"
    action: keep
```

优点：

- 支持高效数据传输。
- 适合数据持久化和集中化处理。

缺点：

- 不支持查询远程存储中的数据，Prometheus 本地存储中无数据时，查询会失败。
- 数据实时性取决于网络 and 远程存储的处理速度。



2. Prometheus Federation

功能：

Federation 是 Prometheus 的数据共享机制，用于从其他 Prometheus 实例中抓取（scrape）数据，适用于以下场景：

- 汇聚多集群的监控数据。
- 共享关键指标到上层集群。
- 创建层次化监控架构。

原理：

一个 Prometheus 实例（上层）通过标准的抓取机制从其他 Prometheus 实例（下层）获取指标数据。

配置示例：

```
scrape_configs:
  - job_name: 'federation'
    scrape_interval: 15s
    metrics_path: '/federate'
    params:
      'match[]':
        - '{job="node_exporter"}'
    static_configs:
      - targets:
        - 'prometheus-instance-1:9090'
        - 'prometheus-instance-2:9090'
```

优点：

- 无需额外存储组件。
- 支持对抓取的数据进行聚合和过滤。
- 可以动态调整抓取范围。

缺点：

- 对下游 Prometheus 实例有一定负载要求，尤其是实例较多时。
- 不适合长时间存储或超大规模的数据转发。

对比总结

功能	Remote Write	Federation
目的	数据转发到其他存储系统	共享数据至上层 Prometheus
数据流动方向	推送	拉取
使用场景	长期存储、跨区域数据汇聚	层级化监控、指标共享



功能	Remote Write	Federation
实现复杂性	依赖外部存储系统	原生支持，配置简单
实时性	取决于网络 and 存储延迟	通常较高
查询方式	无法查询远程存储数据	支持对抓取数据的查询

选择 **Remote Write** 或 **Federation**，取决于你的使用需求：

- **需要长期存储？** 使用 Remote Write。
- **需要跨实例共享关键指标？** 使用 Federation。



詳細說明:remote write、 federation



以下是更詳細的 **Prometheus Remote Write** 和 **Federation** 的介紹，從概念、用途、優點、缺點到具體配置與實例說明，幫助你深入理解。

1. Prometheus Remote Write

概念

Prometheus Remote Write 是一種數據推送機制，允許 Prometheus 將收集到的時間序列數據發送到遠端存儲系統，例如：

- 長期存儲解決方案（如 **Grafana Mimir**、**Thanos**、**VictoriaMetrics**、**Cortex**）。
- 集中式數據分析或多集群數據匯聚。

這種方式使得 Prometheus 不僅僅用於短期數據存儲，還能與專門的時序數據庫結合，解決長期數據存儲和檢索問題。

用途

1. **長期存儲：**
Prometheus 本地存儲數據有限，使用 Remote Write 可以將數據持久化到外部系統，便於後續分析和查詢。
2. **多集群數據匯聚：**
在多個 Prometheus 實例的架構中，將所有數據發送到一個集中式存儲，實現統一管理和分析。



3. 異地備份與災難恢復：

將數據同步到遠程存儲系統，作為異地備份的一部分。

如何工作

Prometheus 使用 HTTP 請求，將數據推送到配置的遠端存儲後端（支持 Prometheus Remote Write 協議的系統）。

配置示例

以下是配置 Remote Write 的方式：

```
remote_write:
- url: "http://remote-storage.example.com/api/v1/write"
  headers:
    Authorization: "Bearer <your-token>"
  write_relabel_configs:
    - source_labels: [__name__]
      regex: "node_.*"
      action: keep
```

- ``url``：指定遠端存儲的寫入端點。
- ``headers``：用於身份驗證，例如 JWT Token。
- ``write_relabel_configs``：用於過濾或修改需要傳輸的數據。

優點

- **長期存儲**：將短期存儲的局限性交由外部存儲系統解決。
- **集中式分析**：支持將多個集群的數據匯聚到同一存儲系統。
- **與大規模分析工具集成**：可配合外部工具（如 Grafana）進行更強大的分析。

缺點

- **缺乏即時性**：傳輸延遲可能導致數據的即時性下降。
- **依賴外部存儲**：需要安裝和配置額外的存儲系統，增加運維成本。
- **不可直接查詢**：Prometheus 無法直接查詢遠端存儲數據，查詢依賴於存儲後端的支持。

常見應用



- **Grafana + Prometheus + Thanos**：長期存儲與統一查詢架構。
- **多集群監控架構**：在多地部署 Prometheus，數據匯聚到一個集中存儲系統。

2. Prometheus Federation

概念

Prometheus Federation 是一種數據共享機制，允許一個 Prometheus 實例（上層實例）通過 HTTP 抓取（scrape）另一個 Prometheus 實例（下層實例）的數據。這種方式適用於層次化監控架構。

例如：

- 上層實例聚合來自多個下層實例的數據。
- 僅抓取關鍵指標，減少上層負載。

用途

1. 層次化監控架構：

適合大型分佈式系統，讓每個子系統部署自己的 Prometheus 實例，然後由上層實例進行匯總。

2. 跨集群監控：

將多個集群的監控數據共享到中心集群，進行全局統一分析。

3. 關鍵指標共享：

只將部分指標（如核心業務指標）共享到上層，減少抓取負擔。

如何工作

Federation 使用標準的 Prometheus 抓取機制，但抓取目標是另一個 Prometheus 的 `/federate` 接口。`

配置示例

以下是配置 Federation 的方式：

```
scrape_configs:
  - job_name: 'federation'
    scrape_interval: 30s
    metrics_path: '/federate'
    params:
      'match[]':
        - '{job="node_exporter"}'
```



```
- 'up'
static_configs:
  - targets:
    - 'prometheus-instance-1:9090'
    - 'prometheus-instance-2:9090'
```

- ``metrics_path``：指定抓取 ``/federate`` 接口。
- ``params``：指定匹配的指標。例如 ``match[]`` 可指定需要抓取的指標。
- ``targets``：指定下層 Prometheus 實例的地址。

優點

- **內建支持**：不需要額外的存儲系統或服務。
- **數據即時性高**：與 Prometheus 本地抓取數據的即時性相當。
- **數據過濾**：支持僅抓取部分關鍵指標，減少不必要的負載。

缺點

- **適用於短期數據**：不適合用於長期存儲。
- **負載影響**：如果上層實例需要抓取多個下層實例，可能會對下層造成性能壓力。
- **有限數據處理能力**：只適用於 Prometheus 的短期數據存儲窗口。

常見應用

- **多層次監控架構**：
 - 子集群中的 Prometheus 監控特定範圍。
 - 上層 Prometheus 匯聚來自多個子集群的數據。
- **跨環境監控**：在 Dev、QA、Prod 等不同環境部署 Prometheus，通過 Federation 匯聚核心指標進行全局分析。

Remote Write 與 Federation 的選擇

功能	Remote Write	Federation
用途	長期存儲和集中式數據分析	層次化監控和指標共享
數據傳輸方向	推送 (Push)	拉取 (Pull)



功能	Remote Write	Federation
即時性	視網路和存儲延遲	高即時性
依賴性	依賴外部存儲系統	完全內建，無需外部依賴
數據篩選	通過 relabel 配置進行篩選	通過 `match[]` 配置抓取所需指標
適合場景	長期存儲、大規模數據匯聚	分層監控、即時數據共享

建議：

- 如果需要長期存儲，選擇 **Remote Write**。
- 如果需要分層次共享數據或聚合指標，選擇 **Federation**。



victoriametrics



VictoriaMetrics 是一種高性能、開源的時序數據數據庫，專門設計用於處理大規模的指標存儲和查詢需求。它完全兼容 Prometheus 的數據格式和協議，並提供一些額外的優勢，如更高效的存儲壓縮、查詢性能以及多租戶支持。

以下是 VictoriaMetrics 的詳細介紹，包括其功能、架構、優勢、配置和常見應用。

1. VictoriaMetrics 的主要特點

高性能

- **壓縮存儲**：VictoriaMetrics 使用高效的存儲引擎，支持比其他存儲解決方案更好的壓縮比，降低存儲成本。
- **快速查詢**：針對多維數據進行了優化，支持快速執行查詢，適合大型查詢場景。
- **高吞吐**：每秒可處理數百萬個數據點，支持高寫入和讀取性能。

兼容性

- **Prometheus 協議**：完全支持 Prometheus 的 Remote Write 和 Remote Read 協議。
- **PromQL 查詢**：原生支持 Prometheus 查詢語言（PromQL）。
- **Grafana 集成**：可直接用作 Grafana 的數據源。

多租戶支持

- 提供多租戶模式（可選），適合在共享環境中隔離不同的用戶數據，對於服務商或多部門運營場景尤為重要。



高可靠性

- 無需外部數據庫或緩存支持，簡化了運維工作。
- 提供集群模式，可實現數據高可用和橫向擴展。

2. 架構模式

單機模式

VictoriaMetrics 可作為單機應用部署，適合小規模數據存儲場景。單機模式中，它能有效管理所有數據，並提供高性能。

集群模式

對於需要分佈式高可用的場景，可以使用 VictoriaMetrics 集群模式。它包括以下組件：

1. **vminsert**：負責接收數據並將其寫入存儲。
2. **vmstorage**：負責存儲數據，支持橫向擴展。
3. **vmselect**：負責處理查詢請求，從存儲中檢索數據。

架構如下：

```
Prometheus/Remote Write -> vminsert -> vmstorage -> vmselect -> Grafana
```

3. 優勢

特點	描述
高效壓縮	提供比 Prometheus 原生存儲更高的壓縮比（通常可以節省 10 倍存儲空間）。
可伸縮性	支持集群模式，能夠隨著需求增長進行水平擴展。
簡單運維	單機模式和集群模式都不需要依賴外部系統，降低運維負擔。
低延遲查詢	優化了指標數據的查詢性能，尤其是對大規模數據和高並發查詢場景。
多租戶支持	支持多租戶模式，通過租戶標籤（`__tenant_id`）隔離數據。
多種數據格式支持	除 Prometheus 外，還支持 Graphite、InfluxDB 等格式數據的導入和查詢。
高可用性與容錯性	集群模式支持自動故障切換和分佈式存儲，保障數據的高可靠性。

4. 部署與配置



單機模式部署

Docker 部署

VictoriaMetrics 可以通過 Docker 快速部署：

```
docker run -d --name victoriametrics -p 8428:8428 victoriametrics/victoria-metrics
```

- **Web 界面**：可以通過 `http://<IP>:8428` 訪問。
- **Prometheus Remote Write 配置**：在 Prometheus 中，添加以下配置：

```
remote_write:  
- url: "http://<victoriametrics-host>:8428/api/v1/write"
```

二進制部署

直接下載執行文件並運行：

```
wget https://github.com/VictoriaMetrics/VictoriaMetrics/releases/latest/download/victoria-metrics-linux-amd64  
chmod +x victoria-metrics-linux-amd64  
./victoria-metrics-linux-amd64
```

集群模式部署

vminsert、vmselect 和 vmstorage 的分開部署

使用 Docker Compose 示例：

```
version: '3.7'  
services:  
  vminsert:  
    image: victoriametrics/vminsert  
    ports:  
      - 8480:8480  
  vmselect:  
    image: victoriametrics/vmselect  
    ports:  
      - 8481:8481  
  vmstorage:  
    image: victoriametrics/vmstorage  
    ports:  
      - 8400:8400
```

Prometheus 配置

在 Prometheus 中配置 Remote Write，指向 `vminsert`：

```
remote_write:  
- url: "http://<vminsert-host>:8480/insert/0/prometheus/"
```



5. 常見查詢與應用場景

查詢數據

使用 Grafana 配置 VictoriaMetrics 作為數據源後，可以使用 PromQL 語法進行查詢：

- 獲取某個指標的最近數據點：

```
up{job="node_exporter"}[5m]
```

- 計算 CPU 使用率平均值：

```
avg(rate(node_cpu_seconds_total[5m])) by (instance)
```

應用場景

- 大規模監控：** 適合企業級多集群監控場景，尤其是在數據量大且需要長期存儲的情況下。
- 多租戶監控：** 提供租戶隔離功能，支持多租戶共用同一數據庫，保證數據的安全性。
- 異構數據源整合：** 支持導入來自 Prometheus、Graphite 和 InfluxDB 的數據，實現統一管理。
- 長期存儲與分析：** 配合 Grafana 可對長期存儲數據進行分析，適用於 SLA 監控和歷史趨勢分析。

6. VictoriaMetrics 與其他存儲解決方案對比

特點	VictoriaMetrics	Prometheus + Thanos	InfluxDB
存儲效率	高壓縮率，存儲高效	儲存效率依賴 Thanos 配置	壓縮率相對較低
查詢性能	查詢速度優化好	性能較高，但查詢延遲較大	查詢性能好，但寫入壓力大
運維簡單性	單機/集群模式都簡單	集群模式較複雜	集群模式運維相對困難
多租戶支持	支持，內建多租戶功能	需要外部解決方案	不內建，需額外設計

VictoriaMetrics 是一個非常適合用於大規模、高效存儲的解決方案，特別是在與 Prometheus 集成的場景下，其表現尤為突出。



vu; vu6gji au/6



看起來您的輸入可能有些錯誤或不完整。如果您有具體的問題或需要說明的內容，請再提供更多細節或清楚描述，我很樂意協助您！ 😊



詳細說明victoriametrics、詳細說明怎麼開始使用



以下是 **VictoriaMetrics** 的詳細介紹，包括其功能、應用場景及如何開始使用的步驟。

1. VictoriaMetrics 的功能與特點

功能

1. 高效存儲

- 使用自適應壓縮技術，比傳統存儲系統（如 Prometheus 原生存儲）更高效。
- 支持大量數據的寫入和查詢。

2. 高性能

- 每秒可寫入數百萬條指標數據，支持多維數據的快速查詢。
- 在單機模式下也能處理高並發數據負載。

3. 完全兼容 Prometheus

- 支持 Prometheus 的 **Remote Write** 和 **Remote Read** 協議。
- 支持 Prometheus 查詢語言（PromQL）。

4. 多租戶支持

- 內建多租戶功能，通過租戶標籤（`__tenant_id__`）來隔離不同用戶或部門的數據。

5. 支持多種數據格式

- 可直接導入和查詢 Prometheus、Graphite、InfluxDB 等格式的數據。

6. 簡單運維

- 單一可執行文件可運行完整功能，無需依賴外部數據庫或緩存系統。
- 集群模式支持高可用性和水平擴展。



2. 典型應用場景

1. 長期數據存儲與分析

- 解決 Prometheus 本地存儲限制（默認僅保留數據 15 天）。
- 適合長期數據存儲和 SLA 趨勢分析。

2. 高負載監控

- 每秒處理數百萬條指標數據，適合大規模監控場景（如 IoT、金融數據分析等）。

3. 多租戶環境

- 支持多租戶，適合多團隊、多部門共用同一監控平台。

4. 異構數據源整合

- 接收來自 Prometheus、Graphite、InfluxDB 等不同格式的數據，實現統一管理。

3. 開始使用 VictoriaMetrics

3.1 部署方式選擇

VictoriaMetrics 提供兩種主要部署模式：

1. 單機模式

- 適合小規模環境或測試場景。
- 所有功能集中在一個實例中運行。

2. 集群模式

- 適合大規模場景，支持高可用性和水平擴展。
- 分為三個組件：`vminsert`、`vmselect` 和 `vmstorage`。

3.2 單機模式快速開始

使用 Docker 部署

1. 啟動 VictoriaMetrics 容器：

```
docker run -d --name victoriametrics -p 8428:8428 victoriametrics/victoriametrics
```

2. 確認服務運行：

- 訪問 Web 界面：`http://<主機IP>:8428`
- Prometheus 寫入 URL：`http://<主機IP>:8428/api/v1/write`



使用二進制文件部署

1. 下載執行文件：

```
wget https://github.com/VictoriaMetrics/VictoriaMetrics/releases/latest/download/victoria-metrics-linux-amd64
chmod +x victoria-metrics-linux-amd64
```

2. 啟動服務：

```
./victoria-metrics-linux-amd64
```

3. 確認服務：

- 預設監聽在 `8428` 端口。

配置 Prometheus

在 Prometheus 配置中，添加以下內容：

```
remote_write:
- url: "http://<victoriametrics-host>:8428/api/v1/write"
remote_read:
- url: "http://<victoriametrics-host>:8428/api/v1/read"
```

3.3 集群模式部署

核心組件

1. **vminsert**

- 接收數據寫入請求，將其轉發給 `vmstorage`。
- 默認端口：`8480`

2. **vmselect**

- 負責處理查詢請求，從 `vmstorage` 中檢索數據。
- 默認端口：`8481`

3. **vmstorage**

- 實際存儲數據，支持分佈式存儲。
- 默認端口：`8490`

部署方式

可以通過 Docker Compose 部署：

```
version: '3.7'
services:
  vminsert:
    image: victoriametrics/vminsert
    ports:
```



```
- 8480:8480

vmselect:
  image: victoriametrics/vmselect
  ports:
    - 8481:8481

vmstorage:
  image: victoriametrics/vmstorage
  ports:
    - 8400:8400
```

3.4 查詢數據

使用 PromQL

VictoriaMetrics 完全支持 Prometheus 的 PromQL 語法，可直接用於 Grafana：

1. 添加數據源：

- **類型**：Prometheus。
- **URL**：`http://<victoriametrics-host>:8428`。

2. 查詢示例：

- 查詢最近 5 分鐘的指標數據：

```
up[5m]
```

- 計算某指標的平均值：

```
avg(rate(node_cpu_seconds_total[1m])) by (instance)
```

數據導入

除了 Prometheus，VictoriaMetrics 還支持其他數據格式的導入：

- **Graphite 格式**：

```
curl -d 'metric.name 123 1609459200' http://<victoriametrics-host>:2003
```

- **InfluxDB 格式**：

```
curl -X POST -d 'measurement,tag1=value1 value=123 1609459200000' http://<victoriametrics-host>:8428/write
```



3.5 運維與監控

1. 系統指標

- VictoriaMetrics 提供自監控指標，可通過 `/metrics` 查詢。

2. 備份與恢復

- 支持增量備份：

```
./victoria-metrics-linux-amd64 -snapshot.create
```

- 恢復數據：

```
./victoria-metrics-linux-amd64 -storageDataPath=/path/to/snapshot
```

4. 結論與建議

VictoriaMetrics 是一個高效且靈活的時序數據存儲解決方案，適用於以下場景：

- **大規模監控**：每秒處理數百萬條指標數據。
- **長期存儲**：超高壓縮比降低存儲成本。
- **多租戶環境**：內建數據隔離功能，適合多用戶場景。

無論是測試環境還是生產環境，你都可以根據需求選擇單機或集群模式，並輕鬆與 Prometheus 和 Grafana 集成，實現高效的監控和分析。