

# Prometheus 加密設定



以繁體中文詳細說明prometheus scrape job要加上密碼



在 Prometheus 中，若要對某個 scrape job 加上密碼（通常是 Basic Authentication），可以透過 ``basic_auth`` 設定來達成。這種方式常見於你的目標服務（exporter）需要身份驗證的情境。

以下是詳細說明：



## 一、Basic Auth 是什麼？

Basic Authentication（基本身份驗證）是 HTTP 協議的一部分，會將使用者名稱與密碼透過 Base64 編碼後，附加在每次的 HTTP 請求中。

格式為：

```
Authorization: Basic base64(username:password)
```



## 二、Prometheus 如何設定 Basic Auth？

在 Prometheus 的 ``prometheus.yml`` 中，``scrape_configs`` 的每個 job 可以使用 ``basic_auth`` 欄位來加入使用者名稱與密碼。



範例設定：

```
scrape_configs:
  - job_name: 'my_secure_exporter'
    static_configs:
```



```
- targets: ['localhost:9100']
basic_auth:
  username: 'myuser'
  password: 'mypassword'
```

這段設定會讓 Prometheus 在向 `localhost:9100` 發送 scrape 請求時，自動加上 Basic Auth 頭部。

### ✅ 三、如果密碼比較敏感，如何避免寫死在設定檔中？

Prometheus 支援使用檔案讀取方式設定密碼，這樣可以避免直接把密碼寫在 `prometheus.yml` 裡。

🔒 使用 `password\_file`：

```
scrape_configs:
- job_name: 'my_secure_exporter'
  static_configs:
    - targets: ['localhost:9100']
  basic_auth:
    username: 'myuser'
    password_file: '/etc/prometheus/secrets/my_password.txt'
```

這裡的 `/etc/prometheus/secrets/my\_password.txt` 是一個純文字檔案，內容只包含一行密碼。

“✅ 適用於密碼存放在 K8s Secret、CI/CD 管理或檔案掛載的情境。”

### 🔒 四、Exporter 端也要支援 Basic Auth

要注意的是，你的目標 exporter 也要能處理 Basic Auth 驗證。舉例來說：

- 若 exporter 是自己寫的 Python server，就要從 `Authorization` header 解析 Basic Auth。
- 若使用 nginx 當反向代理，也可以在 nginx 中加上 Basic Auth。

---

## 五、進階建議

若需要更高的安全性，也可以考慮：

- 使用 HTTPS + Basic Auth。
- 使用 bearer token (`authorization` header)。
- 將 Prometheus 部署在內部網路，限制對外存取。

---

需要我提供 nginx 設定 Basic Auth 的範例或 K8s Secret 整合方式嗎？可以一起補上。



**Prometheus Remote Secrets.p...**  
PDF

參考"Prometheus Remote Secrets.pdf"，以繁體中文詳細說明



根據你提供的《**Prometheus Remote Secrets**》文件，這份提案是為了解讓 **Prometheus** 能夠從遠端 **Secret Providers**（例如 **Kubernetes Secrets**）讀取密碼或其他機密資訊，來取代傳統將密碼寫死在 `prometheus.yml` 或放在本機檔案系統的做法。以下是完整的說明與重點整理，並附上實作範例：

---

## 為什麼 Prometheus Scrape Job 要用 Remote Secrets 加上密碼？

### 傳統方法的問題：

目前 Prometheus 支援以下兩種方式設定密碼：

```
password: <明文密碼>           # 安全性差
password_file: <檔案路徑>      # 需要掛載 Secret 檔案
```

缺點包括：

- 密碼需要硬編碼或掛載到本機檔案 → 不安全。
- 在 Kubernetes 中會增加部署與 Secret 掛載的複雜度。
- 若 Prometheus 被部署於多節點上，其他 Pod 可能存取這些檔案（Node-level 權限風險）。

---

## Remote Secrets 解法設計（未來功能，正在提案中）

Prometheus 引入新的 `password_ref` 機制，搭配 `secret_providers` 區塊來實現機密讀取的動態化與集中管理。

### 設定結構總覽：

#### 設定 `secret_providers`

這部分定義從哪裡取得密碼：


```
secret_providers:
  - kubernetes_sp_config:
      config:
```

```
api-server: https://my-k8s-api
secrets:
  - name: xyz
    config:
      secret:
        name: my-k8s-secret
        namespace: my-namespace
        key: my-key
```

## 2 Scrape job 使用 `password\_ref`

然後你就可以在 `scrape\_configs` 中使用：

```
scrape_configs:
  - job_name: 'secure_job'
    static_configs:
      - targets: ['my-exporter:9100']
    basic_auth:
      username: 'myuser'
      password_ref: xyz
```

 Prometheus 在每次 scrape 時，會根據 `password\_ref` 查詢 secret provider 中對應名稱的密碼。

## ✨ 好處與特點

特點	說明
✅ 安全性提升	密碼不再存在於本機硬碟、ConfigMap 或 YAML 內。
✅ 易於管理	同一個密碼可重複使用於多個 job，不需要修改多份設定。
✅ 支援動態更新	Secret providers 可以實作快取與更新機制（例如用 Kubernetes Watch）。
✅ 多平台支援	可以擴充支援 AWS Secrets Manager、Vault、K8s Secrets...等

---

## 補充技術細節

- 每個 ``secret_provider`` 會實作 ``Fetch(ctx)`` 方法，於每次 scrape 時擷取密碼內容。
- 若是頻繁請求的平台（如 K8s），建議使用快取來減少 API server 壓力。
- Prometheus 會支援 Metrics 來監控 Secret 載入狀態（例如失敗次數、目前載入數量等）。

## 與未來方向

此提案目前仍在設計中，將來可能會：

-  支援更多 Secret Provider（如 AWS Secrets Manager、HashiCorp Vault）。
-  將原本的 ``password`` / ``password_file`` 轉換成新的 ``inline_sp_config``、``filesystem_sp_config``。
-  加入 Secret Federation 機制（集中存取與快取），避免多個 Prometheus 個體重複讀取 Secret。

## 小結與建議

若你目前：

建議

使用本機 ``password_file``

可繼續使用，但安全性有限

在 Kubernetes 環境中

建議規劃導入 Remote Secrets 機制，搭配 ``password_ref``

密碼管理由 Vault / AWS Secret 控制

建議持續關注此提案，將來可整合自動取得密碼的能力

如果你想要我幫你撰寫一份實際範例（包含 Kubernetes Secret + Prometheus 設定 YAML），也可以提出，我可以幫你搭好基本架構。需要嗎？



把Prometheus Remote Secrets.pdf翻成繁體中文



Prometheus Remote Secrets



Prometheus Remote Secrets

Prometheus 遠端機密 (Remote Secrets)

作者：Daniel Hrabovcak 文件狀態：審閱中 建立時間：2023 年 10 月 26 日 上次重大更新：2023 年 11 月 3 日

目標 允許 Prometheus 從遠端機密提供者 (Secret Providers) 讀取機密資料。

提案 引入類似服務發現 (Service Discovery) 的機密發現機制，允許不同的機密提供者撰寫相應的程式碼，以從其 API 中讀取機密。

我已將《Prometheus Remote Secrets.pdf》完整翻譯為繁體中文，並整理為文件「**Prometheus Remote Secrets**」，你可以在右側畫布中查看與編輯。

如果你希望我幫你進一步補充 Kubernetes Secret 實作範例、或整合進 GitLab CI/CD 或 Helm 等工具，也可以告訴我，我可以繼續協助。需要嗎？



以繁體中文詳細說明，我的Prometheus運行在虛擬機，目前是讓user自己修改gitlab pipeline來產生scrape job的yml，gitlab上不能有密碼要怎麼處理



你提到 Prometheus 是部署在 **虛擬機 (VM)** 上運行，目前的做法是使用者透過修改 **GitLab CI pipeline** 來產生 ``scrape_configs`` 的 YAML 檔案。  
但由於 **安全性考量**，**GitLab** 上不能出現明碼密碼 (**Plaintext Password**)，這是非常合理的限制。

下面我會詳細說明如何處理這個問題，讓密碼不出現在 GitLab 上，同時又能讓 Prometheus 正確 scrape 需要密碼驗證的 exporter：

## ✅ 目標與限制

- ✅ **GitLab CI/CD** 上不能放密碼
- ✅ 產生的 **Prometheus** 設定 **YAML** 檔必須包含 ``scrape_configs``
- ✅ **Prometheus** 運行在 **VM** 上 (不是 **Kubernetes**)
- ✅ 密碼需要保密，不能寫在 **Git** 倉庫中

## 🔑 解決方案總覽

解法	說明	優點	缺點
1. 使用 <code>`password_file`</code>	密碼放在 VM 本機的檔案中	GitLab 無需存密碼	需要管理 VM 上的機密檔案



解法	說明	優點	缺點
2. 使用 Prometheus Remote Secrets 提案	支援從遠端 Secret Provider 取密碼	將來彈性高	尚未正式支援，需自行擴充
3. 使用 GitLab CI/CD Secret + Template 合併	GitLab 將密碼以 CI Secret 方式存放，不輸出到 YAML	CI 保密密碼	需特別處理 YAML 合併
4. 將密碼填入 YAML 後直接部署，不 commit	生成 `scrape.yml` 時從 Secret 填入	符合保密需求	無法追蹤生成內容、易錯誤

## ✅ 建議實作方式：GitLab Secret + VM password\_file

這是最穩定且簡單的做法，步驟如下：



### Step 1：在 GitLab 設定 Secret 變數

前往 GitLab 專案的：

***“Settings → CI / CD → Variables”***

新增：

- `SCRAPE\_PASSWORD="my\_secure\_password"`（可遮蔽）
- `SCRAPE\_USERNAME="prom\_user"`（非必要，但也可保密）

### 🔧 Step 2：CI pipeline 自動產生 YAML，不含密碼

CI Job 不直接將密碼寫進 YAML，而是將 username/password 分離，例如這樣產生：

```
# scrape_config.tpl.yaml (不包含密碼)
scrape_configs:
  - job_name: "secure_target"
    static_configs:
      - targets: ["10.0.0.123:8080"]
    basic_auth:
      username: ${SCRAPE_USERNAME}
      password_file: "/etc/prometheus/secrets/my_password.txt"
```

在 GitLab CI 階段執行時：

- 這份 `scrape\_config.tpl.yaml` 會被模板引擎處理（如 `envsubst` 或 `jinja2`）。
- 密碼則寫入目標 VM 的檔案 `/etc/prometheus/secrets/my\_password.txt`，這檔案不經過 GitLab。

---

## Step 3：在 Prometheus VM 上維護密碼檔案

於 VM 上創建：

```
sudo mkdir -p /etc/prometheus/secrets
echo "$SCRAPE_PASSWORD" > /etc/prometheus/secrets/my_password.txt
chmod 600 /etc/prometheus/secrets/my_password.txt
```

可手動建立，也可以：

- 使用 Ansible / Puppet / Shell script 自動部署。
- 或在 CI/CD 中 SCP 上傳密碼檔案（不推薦長期用，較難控管變更）。

## ⚙️ Prometheus 設定示範：

```
scrape_configs:
  - job_name: 'secure_target'
    static_configs:
      - targets: ['10.0.0.123:8080']
    basic_auth:
      username: prom_user
      password_file: /etc/prometheus/secrets/my_password.txt
```

## 🔒 額外安全強化建議

1. VM 上的密碼檔權限設為 600，僅允許 `prometheus` 用戶讀取。
2. 若密碼變更頻率高，可設計 script 定期從 GitLab Secret 抓取並更新。
3. 使用 CI/CD secret masking，確保密碼不會輸出在 job log 中。

## ✅ 延伸備案（進階）

若你願意實作 Remote Secrets 架構（如你提到的 PDF 提案），也可以考慮：

- 建立一個簡單的 **Secrets API Server**，讓 Prometheus 在 scrape 前透過 `password\_ref` 查詢密碼。
- Prometheus 在 VM 上部署時改為定期查詢這個 Secrets server。
- 此方法可擴展到多個 job、多組密碼共用管理。

但這做法較複雜，建議在未來導入 Kubernetes 時再推進。

## 總結

你目前使用 GitLab CI pipeline 產生 Prometheus scrape job，但不能寫入密碼的情況下，最穩定且推薦的做法是：

 使用 `password_file` 搭配 **GitLab Secret** 與 **VM 本機儲存密碼檔案**。

這樣可以保持 GitLab pipeline 無密碼、部署安全又符合 Prometheus 原生支援方式。

---

如果你願意，我可以幫你寫一份 GitLab CI Job、template YAML 範本，還有 VM 安裝腳本範例，要嗎？