

# Grafana & Prometheus

- [Grafana\\_install](#)
- [Actuator+Prometheus+Grafana監控視覺化簡介](#)
- [Grafana alerting](#)
- [Grafana 下拉選單\(prometheus\)](#)
- [Prometheus 相關資源](#)
- [Prometheus PromQL](#)
- [Grafana\\_Promethues docker-compose](#)

# Grafana\_install

<https://fanatical-dentist-b1d.notion.site/Grafana-b93a5178e6a64bad886b83fd6bdcf4ea>

## Enterprise

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/enterprise/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

## OSS

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

## install

```
sudo yum install grafana

# install zabbix plugin
grafana-cli plugins install alexanderzobnin-zabbix-app
```

# Actuator+Prometheus+Grafana監控視覺化簡介

<https://www.tpisoftware.com/tpu/articleDetails/2446>

pom.xml 需增加以下依賴：

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

```
#management.endpoint.health.show-details=ALWAYS
#management.endpoints.web.base-path=/monitor
management.endpoints.web.exposure.include=prometheus,metrics
```

- Prometheus 快速入门教程（六）：Spring Boot Actuator 实时监控  
[https://www.cnblogs.com/chanshuyi/p/06\\_prometheus\\_with\\_springboot\\_actuator.html](https://www.cnblogs.com/chanshuyi/p/06_prometheus_with_springboot_actuator.html)
- Spring Boot 微服務應用整合Prometheus + Grafana 實現監控告警  
<https://juejin.cn/post/6844904052417904653>
- spring-prometheus的指標意義  
<https://blog.csdn.net/ssehs/article/details/123961221>
- prometheus 使用netdata 資料  
<https://learn.netdata.cloud/docs/exporting-metrics/prometheus#configure-prometheus-to-scrape-netdata-metrics>

# Grafana alerting

<https://www.cnblogs.com/liugp/p/17003484.html>

# Grafana 下拉選單(prometheus)

The screenshot shows the Grafana interface for configuring a query. The left sidebar contains a 'Settings' menu with options: General, Annotations, Variables (highlighted), Links, Versions, Permissions, and JSON Model. The main panel is titled 'Query options' and is divided into several sections:

- Data source:** A dropdown menu showing 'prometheus'.
- Query:** A section highlighted with a red box containing:
  - Query type:** A dropdown menu showing 'Label values'.
  - Label \*:** A dropdown menu showing 'application'.
  - Metric:** A dropdown menu showing 'jvm\_info'.
- Label filters:** A row of dropdown menus showing 'Select label', '=', 'Select value', and a '+' button.
- Regex:** A text input field containing the regex pattern `/.*(-(<text>.*)-(<value>.*)-.*/`.
- Sort:** A dropdown menu showing 'Alphabetical (asc)'.
- Refresh:** Two buttons: 'On dashboard load' and 'On time range change'.
- Selection options:** Two checkboxes:
  - ☐ **Multi-value**: Enables multiple values to be selected at the same time.
  - ☐ **Include All option**: Enables an option to include all variables.
- Preview of values:** A section highlighted with a red box showing a single value: 'live-api'.
- Buttons:** At the bottom, there are three buttons: 'Delete' (red), 'Run query' (grey), and 'Apply' (blue).

prometheus

The screenshot shows the Prometheus query results in Grafana. The top bar includes the Prometheus logo and navigation links: Alerts, Graph, Status, and Help. Below the bar, there are several checkboxes: 'Use local time', 'Enable query history', 'Enable autocomplete', 'Enable highlighting', and 'Enable linker'. The search bar contains the query 'jvm\_info'. The results are displayed in a table view, showing a single row with the following data: 'jvm\_info(application="live-api", instance="10.20.43.185:8080", job="actuator", runtime="OpenJDK Runtime Environment", vendor="Amazon.com Inc.", version="17.0.8.1+8-LTS")'. The table has a 'Table' tab and a 'Graph' tab. The 'Table' tab is selected, and the results are shown in a table with columns for 'Evaluation time' and 'Value'. The 'Value' column shows the query result. The bottom of the screen has an 'Add Panel' button.

prometheus 設定檔，新增自訂label

```
prometheus > prometheus > ! prometheus.yml
```

```
29 # The job name is added as a label `job=<job_name>` to any timeseries s
30 # 任務名稱
31
32 - job_name: 'actuator'
33   # Override the global default and scrape targets from this job every
34   scrape_interval: 15s
35   metrics_path: '/actuator/prometheus'
36   static_configs:
37     - targets: ['10.20.43.185:19999']
38       labels:
39         application: live-api
40
41 - job_name: 'netdata'
42
43   scrape_interval: 15s
44   metrics_path: '/api/v1/allmetrics'
45   params:
46     format: [ prometheus ]
47   # http://10.20.43.185:19999/api/v1/allmetrics?format=prometheus
48   honor_labels: true
49
50   static_configs:
51     - targets: ['10.20.43.185:19999']
52       labels:
53         application: netdata
```

# Prometheus 相關資源

- [官方範例設定檔](#)
- [prometheus-book](#)
- [prometheus實戰](#)

# Prometheus PromQL

## PROMQL BASIC QUERY

- Starts with a metric name. Like `ping_average_response_ms`
- Filter with labels, label filters support four operators
  - `=` equal
  - `!=` not equal
  - `=~` matches regex
  - `!~` doesn't match regex

☐ Enable query history

Try experimental React UI

ping\_average\_response\_ms(url=~"^amazon.\*")

Load time: 26ms  
Resolution: 14s  
Total time series: 4

Execute - insert metric at cursor -

Remove Graph

Graph Console

◀ Moment ▶

Element	Value
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.cn")	214.026
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.com")	111.021
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.de")	37.991
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.jp")	182.279

Add Graph

☐ Enable query history

Try experimental React UI

ping\_average\_response\_ms(url=~"^amazon.\*", url!= "amazon.cn")

Load time: 20ms  
Resolution: 14s  
Total time series: 3

Execute - insert metric at cursor -

Remove Graph

Graph Console

◀ Moment ▶

Element	Value
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.com")	110.362
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.de")	38.659
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.jp")	185.114

Add Graph

☐ Enable query history

Try experimental React UI

ping\_average\_response\_ms(url=~"^amazon.\*", url!= "amazon.cn")<100

Load time: 20ms  
Resolution: 14s  
Total time series: 1

Execute - insert metric at cursor -

Remove Graph

Graph Console

◀ Moment ▶

Element	Value
ping_average_response_ms(environment="testing",host="telegraf",instance="telegraf:9273",job="telegraf",service_name="amazon",url="amazon.de")	35.107

Add Graph



# RANGE VECTOR & INSTANT VECTOR

- Range vector selector: `http_requests_total{job="prometheus"}[5m]`
- Instant vector selector: `http_requests_total{job="prometheus",group="canary"}`

☐ Enable query history

[Try experimental React UI](#)

net\_bytes\_recv[1m]

Execute

- insert metric at cursor -

Load time: 28ms  
Resolution: 14s  
Total time series: 2

[Remove Graph](#)

Graph

Console

◀ Moment ▶

Element	Value
net_bytes_recv(environment="testing",host="telegraf-1",instance="telegraf-1:9273",interface="eth0",job="telegraf")	674896 @1604446690 678146 @1604446700 681462 @1604446710 684670 @1604446720 687986 @1604446730
net_bytes_recv(environment="testing",host="telegraf-2",instance="telegraf-2:9274",interface="eth0",job="telegraf")	668230 @1604446690 671504 @1604446700 674754 @1604446710 678070 @1604446720 681386 @1604446730 684636 @1604446740

☐ Enable query history

[Try experimental React UI](#)

rate(net\_bytes\_recv[1m])\*8

Execute

- insert metric at cursor -

Load time: 38ms  
Resolution: 14s  
Total time series: 2

[Remove Graph](#)

Graph

Console

- 1h + ◀ Until ▶ Res. (s) ☐ stacked



☐ Enable query history

[Try experimental React UI](#)

sum(ping\_packets\_received) by (service\_name)

Execute

- insert metric at cursor -

Load time: 28ms  
Resolution: 14s  
Total time series: 2

[Remove Graph](#)

Graph

Console

◀ Moment ▶

Element	Value
{service_name="github"}	6
{service_name="amazon"}	40

Add Graph

# Grafana\_Promethues docker-compose

```
version: '3.3'

volumes:
  prometheus_data: {}
  grafana_data: {}

networks:
  monitoring:
    driver: bridge

services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    restart: always
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./prometheus:/etc/prometheus/
      - prometheus_data:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/usr/share/prometheus/console_libraries'
      - '--web.console.templates=/usr/share/prometheus/consoles'
    networks:
      - monitoring
    # links:
    #   - alertmanager
    #   - node_exporter
    expose:
      - '9090'
    ports:
      - 9090:9090

# alertmanager:
#   image: prom/alertmanager
#   container_name: alertmanager
#   restart: always
#   volumes:
#     - /etc/localtime:/etc/localtime:ro
#     - ./alertmanager:/etc/alertmanager/
#   command:
#     - '--config.file=/etc/alertmanager/config.yml'
#     - '--storage.path=/alertmanager'
#   networks:
#     - monitoring
#   expose:
#     - '9093'
#   ports:
#     - 9099:9093

# node_exporter 為了能夠采集到主機的運行指標如CPU, 內存, 磁盤
# node_exporter:
#   image: prom/node-exporter:v0.18.0
#   container_name: node_exporter
#   restart: always
#   volumes:
#     - /etc/localtime:/etc/localtime:ro
#     - /proc:/host/proc:ro
#     - /sys:/host/sys:ro
#     - /:/rootfs:ro
```

```

# command:
#   - '--path.procfs=/host/proc'
#   - '--path.sysfs=/host/sys'
#   - '--collector.filesystem.ignored-mount-points'
#   -
"^(/([sys|proc|dev|host|etc|rootfs/var/lib/docker/containers|rootfs/var/lib/docker/overlay2|rootfs/run/docker/netns|rootfs/var/lib/docker/aufs
($$/))"
# networks:
#   - monitoring
# expose:
#   - '9100'

grafana:
  image: grafana/grafana
  user: "472"
  container_name: grafana
  restart: always
  environment:
    GF_SECURITY_ADMIN_PASSWORD: admin
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - ./grafana/grafana_data:/var/lib/grafana
    - ./grafana/provisioning:/etc/grafana/provisioning/
  # env_file:
  #   - ./grafana/config.monitoring
  networks:
    - monitoring
  links:
    - prometheus
  ports:
    - 3000:3000
  depends_on:
    - prometheus

```

```

version: "3.7"
services:
  db:
    image: postgres:13.2-alpine
    restart: always
    environment:
      POSTGRES_DB: postgres
      POSTGRES_USER: postgres #postgres
      POSTGRES_PASSWORD: 12345678 #1234
      PGDATA: /var/lib/postgresql/data
    volumes:
      - ./data/db:/var/lib/postgresql/data
    ports:
      - "5432:5432"
  # pgadmin:
  #   image: dpage/pgadmin4:latest
  #   restart: always
  #   environment:
  #     PGADMIN_DEFAULT_EMAIL: admin@fmt.com.tw #xxx@gmail
  #     PGADMIN_DEFAULT_PASSWORD: 12345678 #abcd
  #     PGADMIN_LISTEN_PORT: 80
  #   ports:
  #     - "5433:80"
  #   volumes:
  #     - ./data/pgadmin-data:/var/lib/pgadmin
  #     - ./data/upload:/tmp/upload
  #   links:
  #     - "db:pgsql-server"
volumes:
  db-data:
  pgadmin-data:

```

