Prometheus Deep Dive - Vietnamese

Created	@May 21, 2025 3:40 PM
Tags	Course
URL Course	https://learn.acloud.guru/course/0eaae074-9914-47d1-9239-3d6f267d302b/dashboard
URL Documentation	https://prometheus.io/docs

Giới thiệu

Tổng quan

Prometheus là gì?

`Prometheus` **- Một công cụ giám sát và cảnh báo mã nguồn mở. Nó thu thập dữ liệu về các ứng dụng và hệ thống và cho phép trực quan hóa dữ liệu và phát hành cảnh báo dựa trên dữ liệu.**

`Prometheus` là một bộ công cụ giám sát và cảnh báo hệ thống nguồn mở ban đầu được xây dựng tại `SoundCloud` **.** Kể từ khi thành lập vào năm 2012, nhiều công ty và tổ chức đã áp dụng Prometheus và dự án có một `cộng đồng` người dùng và nhà phát triển rất tích cực . Hiện tại, đây là một dự án nguồn mở độc lập và được duy trì độc lập với bất kỳ công ty nào. Để nhấn mạnh điều này và làm rõ cấu trúc quản trị của dự án, Prometheus đã tham gia `Cloud Native Computing Foundation` vào năm 2016 với tư cách là dự án được lưu trữ thứ hai, sau `Kubernetes` .

Prometheus thu thập và lưu trữ số liệu của mình dưới dạng dữ liệu chuỗi thời gian, nghĩa là thông tin số liệu được lưu trữ theo dấu thời gian khi nó được ghi lại, cùng với các cặp khóa-giá trị tùy chọn được gọi là nhãn.

Thuộc tính/Đặc trưng

Các thuộc tính chính của Prometheus là:

- `một mô hình dữ liệu` đa chiều với dữ liệu chuỗi thời gian được xác định bằng tên số liệu và cặp khóa/giá trị
- PromQL, một `ngôn ngữ truy vấn linh hoạt` để tận dụng tính đa chiều này
- Không phụ thuộc vào lưu trữ phân tán; các nút máy chủ đơn lẻ là tự chủ
- Việc thu thập chuỗi thời gian diễn ra thông qua mô hình kéo qua HTTP
- `Đ ẩy chuỗi thời gian` được hỗ trợ thông qua một cổng trung gian
- Mục tiêu được phát hiện thông qua khám phá dịch vụ hoặc cấu hình tĩnh
- Nhiều chế độ hỗ trợ biểu đồ và bảng điều khiển

Các thành phần của Prometheus là gì?

Hai thành phần cơ bản nhất của một hệ thống Prometheus là:

- `Máy chủ Prometheus`: Một máy chủ trung tâm thu thập và cung cấp các số liệu.
- `Các công cụ xuất dữ liệu`: Các tác nhân xuất dữ liệu về hệ thống và ứng dụng để máy chủ Prometheus thu thập.

`Mô hình pull` - Máy chủ Prometheus kéo dữ liệu số liệu từ các công cụ xuất dữ liệu. Các tác nhân không đẩy dữ liệu đến máy chủ Prometheus.

Các thành phần của Prometheus:

- `Máy chủ Prometheus:` Thu thập dữ liệu số liệu.
- `Các công cụ xuất dữ liệu:` Cung cấp dữ liệu số liệu cho Prometheus xử lý.
- `Thư viện client:` Dễ dàng biến ứng dụng tùy chỉnh của bạn thành một công cụ xuất dữ liệu cung cấp số liệu ở định dạng mà Prometheus có thể xử lý.
- `Prometheus Pushgateway`: Cho phép đẩy số liệu đến Prometheus cho một số trường hợp sử dụng cụ thể.
- `Alertmanager`: Gửi thông báo khi có số liệu kích hoạt.
- `Các công cụ trực quan hóa`**:** Cung cấp các cách hữu ích để xem dữ liệu số liệu. Những điều này không nhất thiết phải là một phần của Prometheus.

Prometheus Background - Use Case - Strengths and Limitations

Background:

- **Language:** Prometheus is written primarily in Go, with some components implemented in Java, Python, and Ruby.
- **License:** Prometheus uses the open-source Apache 2.0 license.
- **History:** Matt T. Proud and Julius Volz started Prometheus development with initial sponsorship from SoundCloud. Today, it is a fully open-source project maintained by many individuals and organizations.
- **Website:** More information and full documentation can be found at `prometheus.io`

Strengths (Good Use Cases):

- **Metric collection:** Collect important metrics about your systems and applications in one place.
- **Visualization:** Build dashboards that provide an overview of the health of your systems.

- **Alerting**: Receive an email when something is broken.

Limitations (Not-So-Good Use Cases):

- 100% accuracy (e.g., per-request billing): Prometheus prioritizes system availability over perfect accuracy. It continues operating during failures and outages, which means some data points may be missing or delayed. For use cases requiring absolute precision—like per-request billing—Prometheus isn't the ideal choice.
- Non time-series data (e.g., log aggregation): Prometheus specializes in collecting numerical, time-series metrics. It's not designed for handling other types of data, such as system logs.

Architecture - Kiến trúc

Khái niệm

Số liệu/chỉ số là gì?

Số liệu là các phép đo số theo thuật ngữ của người bình thường. Thuật ngữ chuỗi thời gian đề cập đến việc ghi lại các thay đổi theo thời gian. Những gì người dùng muốn đo lường khác nhau tùy theo ứng dụng. Đối với máy chủ web, đó có thể là thời gian yêu cầu; đối với cơ sở dữ liệu, đó có thể là số lượng kết nối đang hoạt động hoặc truy vấn đang hoạt động, v.v.

Số liệu đóng vai trò quan trọng trong việc hiểu lý do tại sao ứng dụng của bạn hoạt động theo một cách nhất định. Giả sử bạn đang chạy một ứng dụng web và phát hiện ra rằng nó chậm. Để biết điều gì đang xảy ra với ứng dụng của bạn, bạn sẽ cần một số thông tin. Ví dụ, khi số lượng yêu cầu cao, ứng dụng có thể trở nên chậm. Nếu bạn có số liệu về số lượng yêu cầu, bạn có thể xác định nguyên nhân và tăng số lượng máy chủ để xử lý tải.

`Số liệu trong Prometheus` là một tập hợp các `chuỗi thời gian` được đặt tên, tùy chọn `có nhãn` mô tả các chiều bổ sung.

Introduction to Prometheus Data

Introduction to Prometheus Data Model

What is Time-Series Data? - Dữ liệu chuỗi thời gian là gì?

Prometheus xây dựng xung quanh việc lưu trữ `dữ liệu dạng chuỗi thời gian`.

`Dữ liệu chuỗi thời gian` bao gồm một chuỗi các giá trị liên kết với những điểm khác nhau trong cùng thời gian.

Tất cả dữ liệu Prometheus được lưu trữ dưới dạng dữ liệu chuỗi thời gian. Đó là nghĩ là Prometheus **không chỉ** theo dõi giá trị hiện tại của mỗi chỉ số/số liệu, **mà còn** theo dõi sự thay đổi của từng số liệu/chỉ số theo thời gian.

Metrics and Labels - Chỉ số và nhãn

Mỗi chuỗi thời gian được xác định duy nhất bằng tên số liệu và các cặp khóa-giá trị tùy chọn được gọi là nhãn.

- **Số liệu**

`Tên số liệu` - Mỗi số liệu trong Prometheus đều có tên số liệu. Tên số liệu đề cập đến tính năng chung của hệ thống hoặc ứng dụng đang được đo lường.

Ví dụ về tên số liệu: `node_cpu_seconds_total`. Tên này đo tổng số giây CPU.

- **Nhãn**

`Nhãn số liệu` - Prometheus sử dụng nhãn để cung cấp mô hình dữ liệu theo chiều. Điều này có nghĩa là chúng ta có thể sử dụng nhãn để chỉ định những thứ bổ sung, chẳng hạn như mức sử dụng CPU của nút nào đang được biểu diễn.

Sự kết hợp duy nhất giữa tên số liệu và một tập hợp nhãn xác định một tập hợp dữ liệu chuỗi thời gian cụ thể. Ví dụ này sử dụng nhãn có tên là CPUT để chỉ mức sử dụng của một CPU cụ thể: `node_cpu_seconds_total{cpu="0"}` **hoặc** `node_cpu_seconds_total{cpu="1",`

`instance="192.168.60.136:9100", job="Linux Server", mode="idle"}`

Metric Types - Kiểu chỉ số/số liệu

- **Counter**

`Counter` là một số duy nhất chỉ có thể tăng hoặc được đặt lại thành 0. Bộ đếm biểu thị các giá trị tích lũy.

Khi truy vấn `http_requests_total` **:**

Tổng số yêu cầu HTTP được phục vụ: 0 → 15 → 85 → 276 → 0

Ví dụ:

- Số lượng yêu cầu HTTP được phục vụ bởi một ứng dụng
- Số lượng bản ghi được xử lý
- Số lượng ứng dụng khởi động lại
- Số lượng lỗi

- **Gauge**

`Gauge` là một số duy nhất có thể tăng và giảm theo thời gian.

Khi truy vấn `active_connections` **:**

Yêu cầu HTTP hiện tại đang hoạt động: 76 → 82 → 24 → 56

Ví dụ:

- Số lượng yêu cầu HTTP đồng thời
- Mức sử dụng CPU

    - Mức sử dụng bộ nhớ
    - Luồng đang hoạt động hiện tại
- **Histogram**

`Histogram` đếm số lượng quan sát/sự kiện nằm trong một tập hợp các thùng có thể cấu hình, mỗi thùng có chuỗi thời gian riêng biệt. Biểu đồ sẽ sử dụng nhãn để phân biệt giữa các thùng.

- **Summary**

`Summary` là loại số liệu Prometheus lấy mẫu các quan sát (ví dụ: thời lượng yêu cầu, kích thước phản hồi) và cung cấp tổng số, tổng của tất cả các giá trị quan sát và các phân vị có thể cấu hình (như phần trăm thứ 95) được tính toán trên một cửa sổ thời gian trượt.

Introduction to Prometheus Querying

Querying là gì?

Truy vấn cho phép bạn lấy và xử lý dữ liệu metric.

Sử dụng ngôn ngữ **PromQL** để viết các truy vấn và hiển thị dữ liệu.

Có thể dùng PromQL qua:

- Web UI (Graph / Table)
- HTTP API
- Công cụ như Grafana

Query Basics

- **Selectors**: node_cpu_seconds_total hoặc kèm label: {cpu="0"}
- **Label Matching**: =, !=, =~, !~

Ví dụ:

node_cpu_seconds_total{cpu="0"}

node_cpu_seconds_total{cpu!="0"}

- **Range Vector Selectors**: chọn khoảng thời gian: [5m]
- **Offset Modifier**: dịch thời gian lùi về quá khứ: offset 5m

Operators

- Số học: +, -, *, /, %, ^
- So sánh: ==, !=, <, >, <=, >=
- Logic: and, or, unless

Aggregation

- sum, avg, min, max, count, stddev, topk, bottomk, quantile,...

Ví dụ:

avg(node_cpu_seconds_total{mode="idle"})

Functions

Một số hàm phổ biến:

- rate() – tính tốc độ tăng trung bình theo thời gian.

- abs(), clamp_min(), clamp_max()

HTTP API

Gửi truy vấn qua HTTP:

```
/api/v1/query?query=node_cpu_seconds_total{cpu="0"}
```

Introduction to Visualization

Visualization là gì?

Visualization giúp hiển thị metric dưới dạng biểu đồ, bảng hoặc dashboard.

Công cụ trực quan hóa:

- **Expression Browser** (/graph)
- **Console Templates** (Go templates)
- **Grafana** – công cụ phổ biến nhất

Introduction to Collecting Metrics

Exporters

Exporter là ứng dụng cung cấp dữ liệu metric cho Prometheus.

Cấu hình trong file prometheus.yml tại phần scrape_config.

Application Monitoring

Ngoài Node Exporter, có thể giám sát ứng dụng cụ thể như database, web, v.v.

Jobs và Instances

- **Instance**: endpoint cụ thể mà Prometheus truy cập.
- **Job**: nhóm các instance cùng vai trò.

Ví dụ:

- job: api-server

static_configs:

- targets: ["1.2.3.4:5670", "1.2.3.4:5671"]

Pushgateway

Dùng cho batch jobs hoặc khi không thể sử dụng pull.

- Ứng dụng đẩy dữ liệu vào Pushgateway.
- Prometheus pull từ Pushgateway.

Chỉ nên dùng khi thật sự cần thiết.

Recording Rules

Pre-compute truy vấn và lưu lại làm metric mới.

Giúp cải thiện hiệu suất và đơn giản hóa truy vấn.

Introduction to Alerting

Alertmanager là gì?

Alertmanager xử lý các alert do Prometheus gửi đến:

- Loại bỏ trùng lặp
- Gom nhóm
- Gửi cảnh báo qua email, Slack, PagerDuty...

Alerting Rules

Thiết lập điều kiện sinh cảnh báo trong rule_files.

Sử dụng PromQL để xác định điều kiện cảnh báo.

Advanced Concepts

(Phần mở rộng về federation, remote write/read, scaling, v.v. – bổ sung tùy nhu cầu)