# Prometheus

Là một bộ công cụ giám sát và cảnh báo hệ thống mã nguồn mở

Nó có khả năng thu thập thông số/số liệu (metric) từ các mục tiêu được cấu hình theo các khoảng thời gian nhất định và hiển thị kết quả và có thể kích hoạt cảnh báo

Prometheus giúp thu thập dữ liệu từ ứng dụng web để phân tích. Thư viện này chứa các lớp để thu thập và xuất bản các số liệu phù hợp.

# Cài đặt

Để cung cấp dữ liệu cho prometheus ta cài đặt 2 package sau

Install-Package prometheus-net

Install-Package prometheus-net.AspNetCore

# Metrics là gì?

Metrics là số liệu được đo tại một thời điểm của hệ thống.

Đơn vị đo có thể là giá trị, mốc thời gian và định danh về một đối tượng nào đó

Các dữ liệu được thu thập bất cứ khi nào có dữ liệu diện ra, nhưng số liệu thì chỉ được thu thập tại một khoảng thời gian nhất định.

Prometheus lưu trữ tất cả dữ liệu dưới dạng chuỗi thời gian được xác định bằng tên chỉ số và các cặp khóa / giá trị. Định dạng dữ liệu như sau:

<metric name>{<label name>=<label value>, …}

api\_http\_requests\_total{method=”POST”, handler=”/messages”}

# Metrics types

## Counters (bộ đếm)

Là một số liệu đại diện cho một bộ đếm có giá trị chỉ có thể tăng hoặc trở về 0 khi được khởi động lại. Không sử dụng để đếm các giá trị có thể giảm.

Dùng để đếm tổng số lượng request, số lần call action cụ thể hay số lỗi mà action đó trả ra

Code example:

Counter request\_total = Metrics.CreateCounter("request\_total", "tổng số request được yêu cầu", new CounterConfiguration {

LabelNames = new[] {"path", "method", "action", "controller", "status\_code"}

});

request\_total.WithLabels().Inc();

## Gauges

Là số liệu đại diện cho một giá trị số duy nhất, có thể tăng hoặc giảm.

Code example:

Gauge JobsInQueue = Metrics.CreateGauge("myapp\_jobs\_queued", "Number of jobs waiting for processing in the queue.");

## Histogram

Đếm các giá trị thường xảy ra theo các tiêu chí mà ta cấu hình, nó cung cấp một cơ chế giúp ta có thể tính tổng các giá trị của các tiêu chí đó.

Code example:

Histogram request\_total = Metrics.CreateHistogram("request\_total", "tổng số request được yêu cầu", new HistogramConfiguration {

Buckets = Histogram.ExponentialBuckets(0.01,2,10),

LabelNames = new[] {"path", "method", "action", "controller", "status\_code"}

});

**total sum:** tổng của tất cả các giá trị được quan sát, được hiển thị dưới dạng <basename> \_sum

**count:** số lượng sự kiện đã được quan sát, được hiển thị dưới dạng <basename> \_count (giống với <basename> \_bucket {le = "+ Inf"} ở trên)

## Summary

Code example:

Summary RequestSizeSummary = Metrics.CreateSummary("myapp\_request\_size\_bytes", "Summary of request sizes (in bytes) over last 10 minutes.");

Tương tự Histogram. Nhưng là lựa chọn tốt hơn cho việc tính toán chính xác hơn.

# Service

## ASP.NET Core HTTP request metrics

Thư viện hiển thị một số số liệu từ các ứng dụng ASP.NET Core

* Số lượng request đang được xử lý.
* Tổng số request HTTP đã nhận.
* Thời lượng của các request

Trong lớp startup.cs

public void Configure(IApplicationBuilder app, ...)

{

app.UseRouting();

app.UseHttpMetrics();

}

## IHttpClientFactory metrics

Thư viện này cho phép bạn hiển thị các số liệu về các phiên bản HttpClient được tạo bằng IHttpClientFactory. Các chỉ số được tiếp xúc bao gồm:

* Số lượng yêu cầu HTTP đang được xử lý.
* Tổng số yêu cầu HTTP đã bắt đầu.
* Khoảng thời gian của các yêu cầu máy khách HTTP (từ khi bắt đầu yêu cầu đến khi kết thúc đọc tiêu đề phản hồi).
* Khoảng thời gian phản hồi của ứng dụng khách HTTP (từ khi bắt đầu yêu cầu đến khi kết thúc phần đọc phản hồi).

public void ConfigureServices(IServiceCollection services)

{

services.AddHttpClient(Options.DefaultName)

.UseHttpClientMetrics();

}

# ASP.NET Core with basic authentication

Dùng để xác thực khi call vào link /metrics

https://johanbostrom.se/blog/adding-basic-auth-to-your-mvc-application-in-dotnet-core

app.Map("/metrics", metricsApp =>

{

metricsApp.UseMiddleware<BasicAuthMiddleware>("Contoso Corporation");

// We already specified URL prefix in .Map() above, no need to specify it again here.

metricsApp.UseMetricServer("");

});

# Kestrel stand-alone server

cấu hình cổng riêng cho /metrics

var metricServer = new KestrelMetricServer(port: 9095);

metricServer.Start();

# Publishing via standalone HTTP handler

cấu hình cổng riêng cho /metrics

var metricServer = new MetricServer(port: 1234);

metricServer.Start();