

Khối Công Nghệ

Hướng dẫn sử dụng Serilog cho dev C#

| | | | |
|---------------------------|--|----------------------|--|
| Người soạn thảo: | Phạm Thái Hòa | Email: | Hoapt@bagroup.vn |
| Ngày phát hành: | 10/04/2020 | Phiên bản: | 1.0 |
| Phê duyệt bởi: | Phạm Thái Hòa | Ngày áp dụng: | 10/04/2020 |
| Mô tả và loại trừ: | Tài liệu này hướng dẫn cho các lập trình viên C# dễ dàng kết nối và thao tác với Serilog | | |



CÔNG TY TNHH PTCN ĐIỆN TỬ BÌNH ANH

Văn phòng giao dịch: Lô 14, Phố Nguyễn Cảnh Dị, Phường Đại Kim, Hoàng Mai, Hà Nội.
Điện thoại: 1900 6464
Fax: 024 3201 2069
Email: info@binhanh.vn

www.bagps.vn

Lịch sử tài liệu

| Ngày sửa | Người sửa | A,M,D | Nội dung điều chỉnh | Ghi chú |
|------------|-----------|-------|---------------------|---------|
| 10/04/2020 | Hoapt | A | Bản đầu tiên | |

Mục lục

| | |
|---|----|
| Lịch sử tài liệu | 2 |
| Mục lục..... | 3 |
| Danh sách bảng biểu | 4 |
| Danh sách hình vẽ | 5 |
| 1. Giới thiệu Serilog..... | 6 |
| 1.1. Mở đầu..... | 6 |
| 1.2. Các khái niệm trong Serilog | 7 |
| 1.3. Dữ liệu có cấu trúc | 8 |
| 1.4. Tự debug chính Serilog | 9 |
| 2. Cấu hình | 10 |
| 2.1. Cấu hình đơn giản | 10 |
| 2.2. Một số trường hợp cấu hình phức tạp | 10 |
| 3. Một số các Sink | 11 |
| 3.1. Một số Sink thông dụng và nhận xét | 11 |
| 3.2. Demo một số sink hay dùng..... | 12 |
| 3.2.1. Hướng dẫn nhận log qua Microsoft Teams | 13 |
| 3.2.2. Hướng dẫn nhận log qua Telegram | 14 |
| 3.3. Cấu hình thực tế và nhận log qua ELK..... | 16 |
| 3.4. Tự viết Sink của riêng mình..... | 20 |



CÔNG TY TNHH PTCN ĐIỆN TỬ BÌNH ANH

Văn phòng giao dịch: Lô 14, Phố Nguyễn Cảnh Dị, Phường Đại Kim, Hoàng Mai, Hà Nội.
Điện thoại: 1900 6464
Fax: 024 3201 2069
Email: info@binhanh.vn

www.bagps.vn

Danh sách bảng biểu

| | |
|---|----|
| Bảng 1 - Mức độ log..... | 7 |
| Bảng 2 - Danh sách các Sink thông dụng..... | 12 |
| Bảng 3 - Cấu hình một số Sink hay dùng..... | 13 |



CÔNG TY TNHH PTCN ĐIỆN TỬ BÌNH ANH

Văn phòng giao dịch: Lô 14, Phố Nguyễn Cảnh Dị, Phường Đại Kim, Hoàng Mai, Hà Nội.
Điện thoại: 1900 6464
Fax: 024 3201 2069
Email: info@binhanh.vn

www.bagps.vn

Danh sách hình vẽ

| | |
|---|----|
| Hình 1 - Serilog là một trong những thư viện log phổ biến nhất..... | 6 |
| Hình 2 - Cấu hình Microsoft Teams..... | 13 |

1. Giới thiệu Serilog














1.1. Mở đầu

Log luôn là một bài toán vô cùng cơ bản khi ta lập trình bất cứ một thứ gì, không có nó thì không thể biết được nguyên nhân lỗi ở đâu, tham số như thế nào hay phần mềm hoạt động ra sao. Mức độ cần thiết của log thì không còn gì phải bàn.

Có vô vàn cách log khác nhau mà phần lớn trong KCN đều tự viết và áp dụng. Phần lớn là tự log và ghi ra file text, một số sử dụng log4net. Việc khai thác dữ liệu chủ yếu là thủ công, mở file text bấm Ctrl + F hoặc tìm theo phong cách tẩm cám.

Việc cho ra đời của rất nhiều công cụ quản trị dữ liệu, lưu trữ dữ liệu, quản trị dữ liệu từ xa giúp cho việc xem, quản lý log đơn giản và trực quan hơn rất nhiều. Một công cụ Log tốt sẽ đáp ứng việc ghi dữ liệu vào các nhóm tool quản lý dữ liệu trên. Khi sử dụng tool phân tích log giúp các bạn dev, test có thể hình dung rất nhanh về vấn đề lỗi ở đâu, cần làm gì.

Có rất nhiều công cụ log: Nlog, Log4net, Microsoft Logging Library, Serilog. Về cơ bản là cách thức hoạt động đều đơn giản, rất giống nhau. Hiện tại Serilog nổi lên là một thư viện được sử dụng rất nhiều, lý do là dữ liệu của nó có thể xuất ra rất nhiều tool khác nhau. Tham khảo: <https://github.com/serilog/serilog/wiki/Provided-Sinks>

| | | |
|---|--|---------|
|  | Newtonsoft.Json  by James Newton-King, 347M downloads Json.NET is a popular high-performance JSON framework for .NET | v12.0.3 |
|  | Microsoft.Extensions.DependencyInjection  by Microsoft, 173M downloads Default implementation of dependency injection for Microsoft.Extensions.DependencyInjection. | v3.1.3 |
|  | Microsoft.Extensions.Logging  by Microsoft, 157M downloads Logging infrastructure default implementation for Microsoft.Extensions.Logging. | v3.1.3 |
|  | Moq  by Daniel Cazzulino, kzu, 101M downloads Moq is the most popular and friendly mocking framework for .NET. | v4.13.1 |
|  | Serilog  by Serilog Contributors, 93.1M downloads Simple .NET logging with fully-structured events | v2.9.0 |
|  | Microsoft.EntityFrameworkCore  by Microsoft, 92.8M downloads Entity Framework Core is a lightweight and extensible version of the popular Entity Framework data access technology. | v3.1.3 |
|  | Castle.Core by Castle Project Contributors, 87.9M downloads Castle Core, including DynamicProxy, Logging Abstractions and DictionaryAdapter | v4.4.0 |

Hình 1 - Serilog là một trong những thư viện log phổ biến nhất

Trong tài liệu này sẽ giới thiệu nhanh cách thức, khai niệm của Serilog để các dev có thể sử dụng nhanh nhất vào trong project của mình.

Phần mở rộng sẽ hướng dẫn đẩy dữ liệu vào Elasticsearch. Sử dụng Docker và Container để triển khai.

1.2. Các khái niệm trong Serilog

a. Sinks

Trong Serilog định nghĩa. **Sinks là một loại đầu ra ghi dữ liệu**. Khi bạn muốn log dữ liệu thì các loại đầu ra là các loại sinks khác nhau. Có rất nhiều loại Sink: console, ghi ra file, ghi ra DB, ghi qua tin nhắn...

Mỗi một loại Sink thì ứng với một DLL mà bạn phải get qua Nuget. Có thể ghi đồng thời gian qua nhiều sinks khác nhau. Bạn có thể vừa lưu vào file text, vừa nhắn qua Telegram, vừa đẩy vào ElasticSearch... đồng thời. Mỗi loại đầu ra sẽ có cấu hình riêng.

b. Định dạng đầu ra

Để ghi ra log thì tất nhiên bạn muốn định dạng xem cái dòng nó ghi ra như thế nào: có thời gian, có loại lỗi, thuộc tính khác nhau. Trong Serilog thì phần này được định nghĩa là **Output Templates**. Ví dụ: `outputTemplate: "{Timestamp:yyyy-MM-dd HH:mm:ss.fff zzz} [{Level:u3}] {Message:lj}{NewLine}{Exception}"`

c. Mức độ log

Thông tin mà bạn muốn log thì vô cùng đa dạng, có những log chỉ là thông tin, có những log gỡ lỗi, log thông tin, cảnh báo, lỗi... Không phải lúc nào mình cũng muốn log toàn bộ thông tin này. Lúc chạy test thì muốn log hết, lúc chạy thực tế thì chỉ log những lỗi lớn. Để quyết định tùy chọn log những lỗi nào thì Serilog sử dụng khái niệm **Minimum Level**. Bảng các level được phân theo thứ tự từ thấp đến cao như sau:

| Mức độ | Giải thích |
|-------------|--|
| Verbose | Là kiểu thông tin bất kì, thường rất nhiều, ví dụ như các message chung chung. Mức độ này thường rất ít khi áp dụng khi phát hành ứng dụng. |
| Debug | Debug là mức thông tin nội bộ sử dụng để xem chuyện gì đang xảy ra (mô tả quá trình). |
| Information | Cao hơn debug một chút, thông tin mô tả nhưng hoạt động chung của ứng dụng. Những tính năng tương ứng có hoạt động tốt hay không. Ví dụ DB có Ok hay không, đã insert bao nhiêu bản ghi... |
| Warning | Cảnh báo, ghi lại những phần xảy ra không như kì vọng thiết kế ban đầu. Ví dụ giá trị nằm ngoài khoảng, thời gian thực thi lâu hơn dự kiến... Đây là mức log hay sử dụng nhất. |
| Error | Lỗi. Một dịch vụ nào đó không hoạt động, hoặc có một điều kiện không được tính đến: ví dụ DB ngừng hoạt động, SQL timed-out, tràn bộ nhớ... |
| Fatal | Mức độ nguy hiểm nhất, phải xử lý ngay lập tức. |

Bảng 1 - Mức độ log

Với mức độ trên thì bạn có thể hình dung được trong từng hoàn cảnh sử dụng của nó như thế nào. Ta có thể code để lưu lại toàn bộ thông tin từ Verbose đến Fatal. Trong lúc code thì dùng hết, nhưng trong giai đoạn production thì chỉ log đến mức warning. Cấu hình mức này rất dễ:

```
Log.Logger = new LoggerConfiguration()  
    .MinimumLevel.Debug()  
    .WriteTo.Console()  
    .CreateLogger();
```

Ngoài ra bạn có thể cấu hình mỗi một sink thì có mức Minimum khác nhau. Ví dụ file text thì muốn ghi đầy đủ, mà SMS chỉ muốn nhận mức độ Error thì có thể cấu hình override như sau:

```
Log.Logger = new LoggerConfiguration()  
    .MinimumLevel.Debug()  
    .WriteTo.File("log.txt")  
    .WriteTo.Console(restrictedToMinimumLevel: LogEventLevel.Information)  
    .CreateLogger();
```

d. Enrichers

Thông tin ghi ra log cơ bản chỉ gồm thông tin như thời gian, thông tin lỗi... nhưng có rất nhiều thông tin khác mà dev muốn thêm giúp cho quá trình tìm kiếm đơn giản hơn ví dụ như ThreadID, MachineID, UserName.... Serilog chỉ cung cấp một số thông tin tối thiểu. Muốn ghi log thêm cái gì thì tùy dev bổ sung thêm. Các phần bổ sung này gọi là **Enrichers**. Có rất nhiều Enricher. Xem thêm ví dụ trong project đính kèm.

e. Filters

Phần này y hệt như Debug with Condition trong Visual Studio. Chỉ ghi log với điều kiện nào đó. Tham khảo ví dụ dễ hiểu:

```
Log.Logger = new LoggerConfiguration()  
    .WriteTo.Console()  
    .Filter.ByExcluding(Matching.WithProperty<int>("Count", p => p < 10))  
    .CreateLogger();
```

f. Sub-loggers

Trong trường hợp bạn muốn ghi log theo kiểu nhiều nơi, nhiều lớp thì có chế Sub-Logger. Tham khảo:

```
Log.Logger = new LoggerConfiguration()  
    .WriteTo.Console()  
    .WriteTo.Logger(lc => lc  
        .Filter.ByIncludingOnly(...)  
        .WriteTo.File("log.txt"))  
    .CreateLogger();
```

Nếu trường hợp trên vẫn không đáp ứng được thì ta dùng nhiều đối tượng log. Viết kế thừa ILogger.

1.3. Dữ liệu có cấu trúc

Khi log lại dữ liệu, rất nhiều trường hợp bạn muốn log lại các tham số, các data, các dữ liệu trả về xem nó ra sao mà gây ra lỗi. Nếu công cụ log mà hỗ trợ thì việc log các object, log các value rất thuận lợi. Sau đây ta xét đến việc Serilog sẽ log các object ra sao.

a. Các đối tượng cơ bản

Xét trường hợp log như sau:

```
var count = 456;  
Log.Information("Retrieved {Count} records", count);
```

Giá trị count sẽ được ghi ra là 456, tương tự với các trường dữ liệu cơ bản:

- *Booleans* - `bool`
- *Numerics* - `byte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double`, `decimal`
- *Strings* - `string`, `byte[]`
- *Temporals* - `DateTime`, `DateTimeOffset`, `TimeSpan`
- *Others* - `Guid`, `Uri`
- *Nullables* - nullable versions of any of the types above

Với các đối tượng cơ bản trong nhóm trên thì Serilog sẽ ghi ra values của nó vào log. Đối với các collections có thể `IEnumerable` thì sẽ ghi ra dạng JSON.

```
var fruit = new[] { "Apple", "Pear", "Orange" };  
Log.Information("In my bowl I have {Fruit}", fruit);
```

Kết quả đầu ra:

```
{ "Fruit": { "Apple": 1, "Pear": 5 } }
```

Hoàn toàn tương tự với Dictionary, List...

b. Các object

Rất khó để parsing một object ra thế nào để ghi log cho đúng. Nếu Serilog không phát hiện ra cấu trúc thì nó sẽ sử dụng `ToString()` để ghi. Có thể ép nó ghi theo dữ liệu có cấu trúc bằng cách sau:

```
var sensorInput = new { Latitude = 25, Longitude = 134 };  
Log.Information("Processing {@SensorInput}", sensorInput);
```

c. Biến đổi tùy chọn

Có thể custom một số trường muốn ghi, tham khảo đoạn code sau:

```
Log.Logger = new LoggerConfiguration()  
    .Destructure.ByTransforming<HttpRequest>(  
        r => new { RawUrl = r.RawUrl, Method = r.Method } )  
    .WriteTo...
```

1.4. Tự debug chính Serilog

Câu hỏi đặt ra là chính chương trình log cũng hoạt động không đúng, vậy làm sao để log ra lỗi. Serilog cũng cấp một hàm tự ghi ra lỗi của nó như sau:

```
var file = File.CreateText(...);  
Serilog.Debugging.SelfLog.Enable(TextWriter.Synchronized(file));
```

Hoặc: `Serilog.Debugging.SelfLog.Enable(msg => Debug.WriteLine(msg));`

2. Cấu hình

2.1. Cấu hình đơn giản

Cấu hình Serilog thì vô cùng đơn giản. Chỉ cần vài dòng lệnh. Cấu hình Serilog có tác dụng Global trong project. Bạn chỉ cần cấu hình 1 nơi là dùng mọi chỗ trong project.

Đầu tiên là lấy Serilog từ Nuget, sau đó chọn loại Sink mà bạn muốn lưu. Ví dụ là file text. Chỉ cần thêm dòng code như sau:

```
using System;
using Serilog;

namespace SerilogExample
{
    class Program
    {
        static void Main()
        {
            Log.Logger = new LoggerConfiguration()
                .MinimumLevel.Debug()
                .WriteTo.Console()
                .WriteTo.File("logs\\myapp.txt", rollingInterval: RollingInterval.Day)
                .CreateLogger();

            Log.Information("Hello, world!");

            int a = 10, b = 0;
            try
            {
                Log.Debug("Dividing {A} by {B}", a, b);
                Console.WriteLine(a / b);
            }
            catch (Exception ex)
            {
                Log.Error(ex, "Something went wrong");
            }

            Log.CloseAndFlush();
            Console.ReadKey();
        }
    }
}
```

Chưa bao giờ đơn giản hơn thế. Chưa có file thì Serilog tự tạo, chưa có folder cũng tự tạo, tự tách file theo thời gian, tự limit dung lượng file ... Còn rất nhiều tùy chọn khác.

2.2. Một số trường hợp cấu hình phức tạp

Xét trường hợp bạn có một project phức tạp, cần cấu hình nhiều loại log và nhiều nơi log khác nhau.

3. Một số các Sink

3.1. Một số Sink thông dụng và nhận xét

Danh sách các Sink: <https://github.com/serilog/serilog/wiki/Provided-Sinks>

Trên này có rất nhiều Sink khác nhau, ta chỉ xem xét các sink có > 1M lượt tải. Tính đến thời điểm ngày 01/04/2020.

| Sink | Nhóm | Nhận xét |
|-----------------------------|-------------------------|--|
| Amazon CloudWatch | Cloud Managment | Các nền tảng cloud đều cung cấp cổng nhận log và phân tích log đồng thời giám sát toàn bộ hoạt động của hệ thống. Ta chỉ cần một nơi để giám sát toàn hệ thống, từ những hàm bé tí đến toàn bộ một datacenter. |
| Application Insights | Cloud Managment | Nguyên tắc tính giá thì tính theo dung lượng và thời gian lưu trữ. Tính ra giá cũng không rẻ. Cả Amazon, Google, Microsoft đều có. Tham khảo dễ hiểu nhất: https://azure.microsoft.com/en-us/services/monitor/#features |
| Async Wrapper | File | Ghi ra file thì tất nhiên rồi, đơn giản nhất. Khác với File thông thường là phần này chạy kiểu Async. Đối với file thì nếu bạn log liên tục sẽ dẫn đến đĩa cứng bị trễ -> mọi hàm khi log đều phải chờ đoạn ghi xuống đĩa cứng -> bottle necks. Để tránh việc này xảy ra thì Async sẽ đưa dữ liệu vào một queue và ghi xuống từ từ. Bạn có thể cấu hình max queue, chu kì ghi file. |
| Console | Giao diện | Ghi debug ra giao diện, phù hợp với các ứng dụng dạng console |
| Colored Console | Giao diện | Giống như trên nhưng hỗ trợ màu sắc. |
| Debug | | Ghi ra cửa sổ debug của VS |
| Elasticsearch | Database & Tools | ElasticSearch và nhiều công cụ của Elastic là một phần tuyệt vời để tổ chức và tìm kiếm dữ liệu. Từ nó có thể làm tương tự những công cụ trên Cloud. Trong tài liệu này sẽ giới thiệu sâu hơn về Elastic. |
| File | File | Ghi ra file |
| HTTP | Truyền đi ứng dụng khác | Truyền log qua giao thức http. Nhiều công cụ log và phân tích log khác nhận dữ liệu qua Http. Thư viện này là trung gian |
| Literate Console | | Cũng là ghi log ra console nhưng kiểu trình bày dữ liệu đa dạng hơn |
| Period Batching | File | Tương tự như Async và RollingFile |
| Rolling File | File | Ghi ra file, khi đến một giới hạn về kích thước hoặc thời gian thì tự động khi ra file mới. |

| | | |
|-------------------|------------------|---|
| Seq | Database & Tools | Seq cũng là một công cụ thu thập và thể hiện dữ liệu. Ưu điểm có là rất nhẹ, cài trực tiếp trên máy và chạy khá ok. Tham khảo: https://datalust.co/seq |
| Splunk | Database & Tools | Tương tự như ElasticSearch |
| SQL Server | DB | Log vào SQL Server |
| Trace | | Ghi thông tin log chi tiết hơn. |

Bảng 2 - Danh sách các Sink thông dụng

3.2. Demo một số sink hay dùng

Các Sink sau được lấy trong ví dụ bằng C# đính kèm theo tài liệu này. Ngoài ra bạn có thể tham khảo tại từng Sink riêng biệt (<https://github.com/serilog/serilog/wiki/Provided-Sinks>), rất đơn giản. Một số ví dụ dưới đây chỉ mang tính chất ví dụ cho các bạn làm quen nhanh nhất với việc sử dụng Serilog trong thực tế.

| Sink | Mô tả |
|-----------|--|
| File | <p>Sink này dùng để ghi log ra file, là dạng đơn giản nhất. Chú ý là nó chạy ở chế độ đồng bộ, không khuyến dùng khi viết multithread hay dùng trong các services.</p> <pre>Log.Logger = new LoggerConfiguration() .WriteTo.File("Logs\\DemoLog_.txt", restrictedToMinimumLevel: Serilog.Events.LogEventLevel.Verbose, outputTemplate: "[{Timestamp:HH:mm:ss} {Level:u3}] {Message:l}]{NewLine}{Exception}", rollingInterval: RollingInterval.Day, flushToDiskInterval: new TimeSpan(0,0,30)) .CreateLogger();</pre> <p>Dòng cấu hình tương đối đơn giản:</p> <ul style="list-style-type: none"> Tên file: bạn không cần quan tâm đến thư mục làm vì, nếu không có thì Serilog sẽ tự tạo. restrictedToMinimumLevel: tham khảo mức độ log. outputTemplate: định dạng đầu ra khi log. rollingInterval: chia file theo ngày. Định kỳ ghi xuống đĩa cứng, không block file. |
| Console | <p>Là dạng đơn giản nhất, dùng trong ứng dụng chỉ hiển thị phần console lên.</p> <pre>public static void Test() { Log.Logger = new LoggerConfiguration() .MinimumLevel.Debug() .WriteTo.Console() .CreateLogger(); WriteLog.WriteAllTest(); }</pre> |
| AsyncFile | <p>Phần này rất giống với File nhưng chạy ở chế độ Multithread, nó có buffer lưu dữ liệu lại. Định kỳ thì nó ghi ra đĩa cứng, nếu đến max thì nó block lại, hoặc đến interval thì nó ghi ra.</p> |

| | |
|-----------|--|
| Enrichers | Là những thành phần ghi thêm ra log. Có rất nhiều Enrichers, có thể tìm trên Nuget |
|-----------|--|

Bảng 3 - Cấu hình một số Sink hay dùng

3.2.1. Hướng dẫn nhận log qua Microsoft Teams

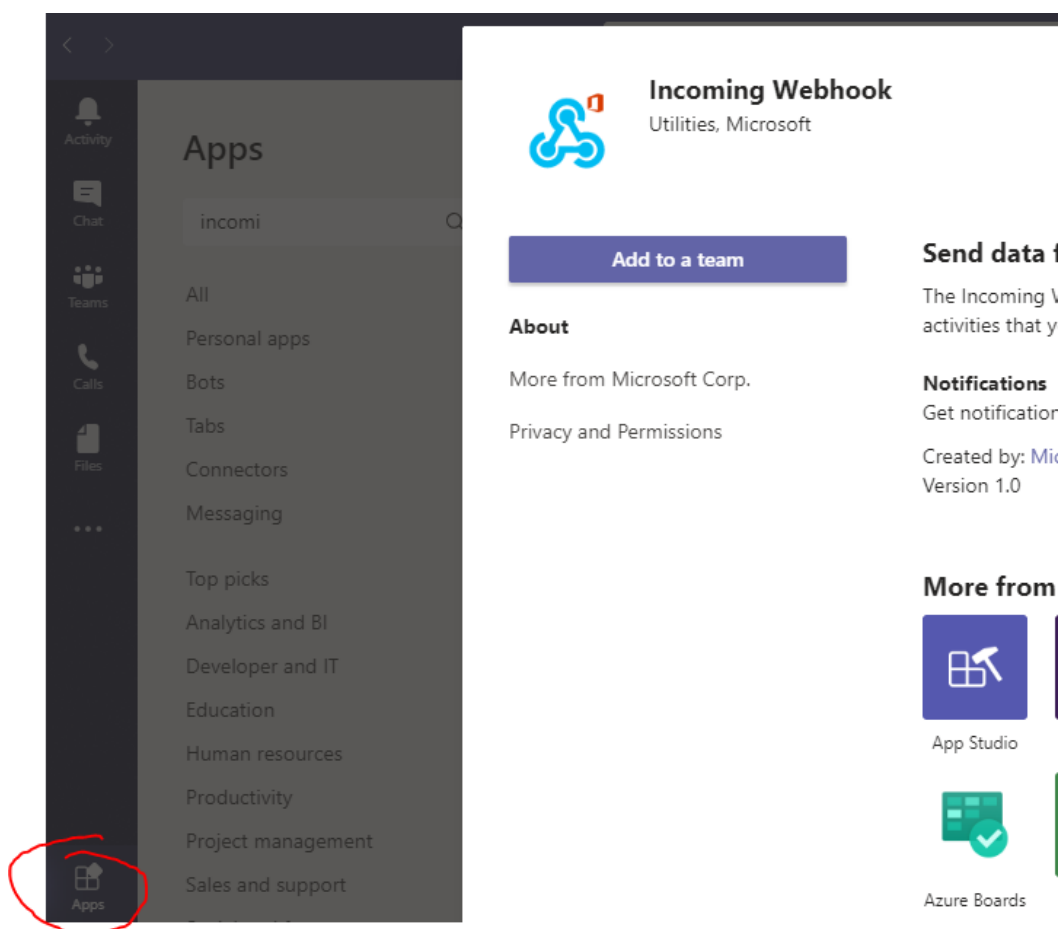
Rất nhiều hệ thống của công ty vận hành liên tục, điều quan trọng nhất là ngay lập tức nhận được các log qua tin nhắn nhất là các Error hay Fatal. Phần này sẽ hướng dẫn các bạn cấu hình qua Microsoft Teams, với các chương trình nhận tin khác vui lòng xem thêm phần tự viết Sink.

Về cấu hình thì tương đối đơn giản, bạn get phần Microsoft Teams về, cấu hình như sau:

```
public static void Test()
{
    //Chú ý: có rất nhiều tùy chọn của File
    Log.Logger = new LoggerConfiguration()
        .WriteTo.MicrosoftTeams(webHookUri, title: title)
        .CreateLogger();

    WriteLog.WriteAllTest();
}
```

Phần quan trọng là webHookUri lấy ở đâu.



Hình 2 - Cấu hình Microsoft Teams

Mở Teams, chọn Apps -> tìm Incoming Webhook. Add nó vào Teams của bạn, chờ vài phút sẽ có Uri để nhấn. Bạn paste vào log kia là ok.

Phần này khá hay ở đoạn một teams có thể nhận được dữ liệu. Bạn có thể kết nối phần log của mình đến nhiều nhóm trong công ty. QA một nhóm riêng, Dev một nhóm riêng, IT một nhóm riêng.

3.2.2. Hướng dẫn nhận log qua Telegram

Cũng giống như Teams, Telegram là chương trình nhắn tin nhưng nó mở hơn, bất cứ ai cũng có thể nhận được. Teams thì thường gắn với một tên miền của tổ chức, còn Telegram sẽ gắn với một nhóm bất kì. Telegram cũng bảo mật rất tốt.

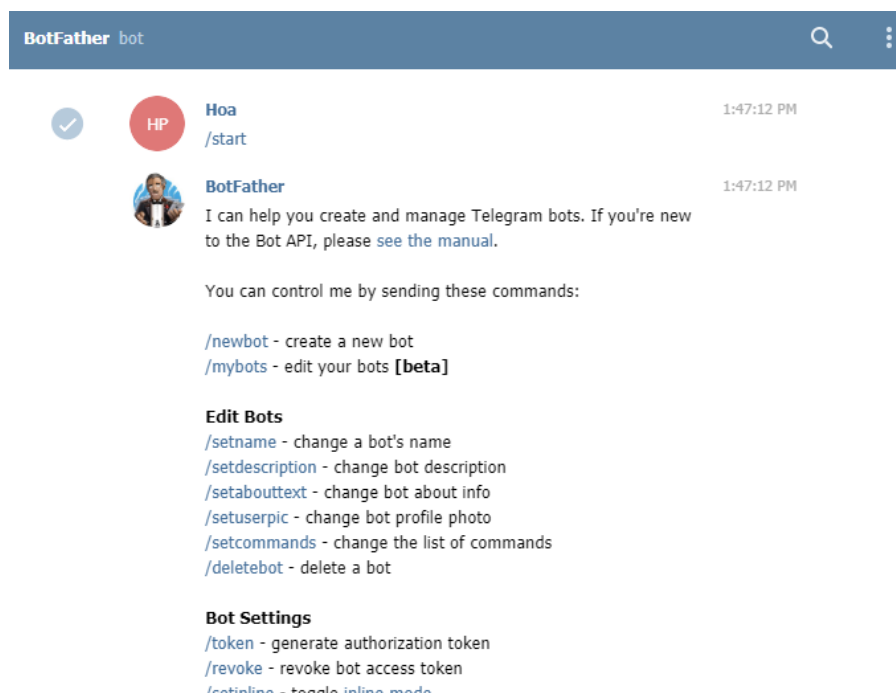
Bạn add Sink sau: <https://github.com/carlozamagni/serilog-sinks-telegram>

Có rất nhiều Sink cho Telegram nhưng Sink trên chạy ok nhất.

```
new LoggerConfiguration()
    .MinimumLevel.Information()
    .WriteTo.TeleSink(
        telegramApiKey:"my-bot-api-key",
        telegramChatId:"the target chat id",
        minimumLevel:LogEventLevel.Warning)
    .CreateLogger();
```

Để gửi Log của bạn đến Telegram thì việc đầu tiên bạn phải làm là tạo một bot và lấy được tokenId của group chat muốn nhận.

Cài đặt ứng dụng Telegram và điện thoại, có thể dùng app hoặc truy cập web.telegram.org. Tìm và chat với @botfather.



Để tạo ra một bot mới thì gõ /newbot. Sau đó BotFather sẽ hỏi tên, bạn nhập tên phù hợp:

**BotFather**

1:53:12 PM

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.

**Hoa**

1:53:31 PM

hoapt_Serilog_bot

**BotFather**

1:53:34 PM

Done! Congratulations on your new bot. You will find it at t.me/hoapt_Serilog_bot. You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:

1016361104:AAGDh0avLSLQanuAD4s0kEUBXiBnQsrkIvc

Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page:

<https://core.telegram.org/bots/api>

Vậy ta đã có thông tin để sử dụng là t.me/hoapt_Serilog_bot và token (màu đỏ).

Bước tiếp theo ta tạo một Group chat, và add @RawDataBot (có thể nó sẽ hỏi việc tạo tên). Sau khi add nó vào Group (cùng với bot bạn tạo), gõ `/start` thì nó hiển thị ra như sau:

Hoa Pham invited Telegram Bot Raw

**Telegram Bot Raw**

2:12:54 PM


```
{
  "update_id": 754612556,
  "message": {
    "message_id": 393308,
    "from": {
      "id": 951460756,
      "is_bot": false,
      "first_name": "Hoa",
      "last_name": "Pham",
      "username": "Hoapt309",
      "language_code": "en"
    },
    "chat": {
      "id": -434024944,
      "title": "Serilog",
      "type": "group",
      "all_members_are_administrators": true
    },
    "date": 1587539575,
    "new_chat_participant": {
```

Chat id là số trên. Sau đó bạn add botid và chatid vào trong source thì nó như sau:

```
Log.Logger = new LoggerConfiguration()  
    .MinimumLevel.Verbose()  
    .WriteTo.TeleSink("1016361104:AAGDh0avLSLQanuAD4s0kEUBXibnQsrkIvc", "-434024944")  
    .CreateLogger();  
  
Log.Information("Demo: Logging Information");  
Log.Debug("Demo: Logging Debug");  
Log.Warning("Demo: Logging Warning");  
Log.Error("Demo: Logging Error");  
Log.Fatal("Demo: Logging Fatal");  
  
Log.CloseAndFlush();
```

Kết quả thực hiện thì trên web, app Telegram của bạn sẽ hiển thị thông tin sau:

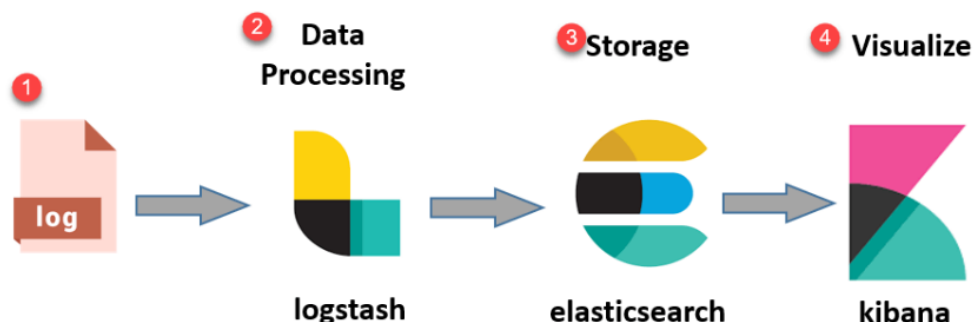
Hoa Pham invited **Telegram Bot Raw**

| Unread messages | | |
|---|--|------------|
|  | hoaptSerilogBot | 2:27:25 PM |
| | Demo: Logging Information | |
| | Demo: Logging Debug | 2:27:26 PM |
| | Demo: Logging Warning | 2:27:26 PM |
| | Demo: Logging Error | 2:27:27 PM |
| | Demo: Logging Fatal | 2:27:27 PM |
| | Demo: #tag | 2:27:31 PM |
| | Demo: Log object DemoObject { ValueA: 1, ValueB: 2, ListC: ["a", "b", "c"] } | 2:27:32 PM |
| | Dividing 10 by 0 | 2:27:32 PM |
| | Something went wrong | 2:27:33 PM |

3.3. Cấu hình thực tế và nhận log qua ELK

ELK là viết tắt của Elasticsearch, LogStash và Kibana trong đó:

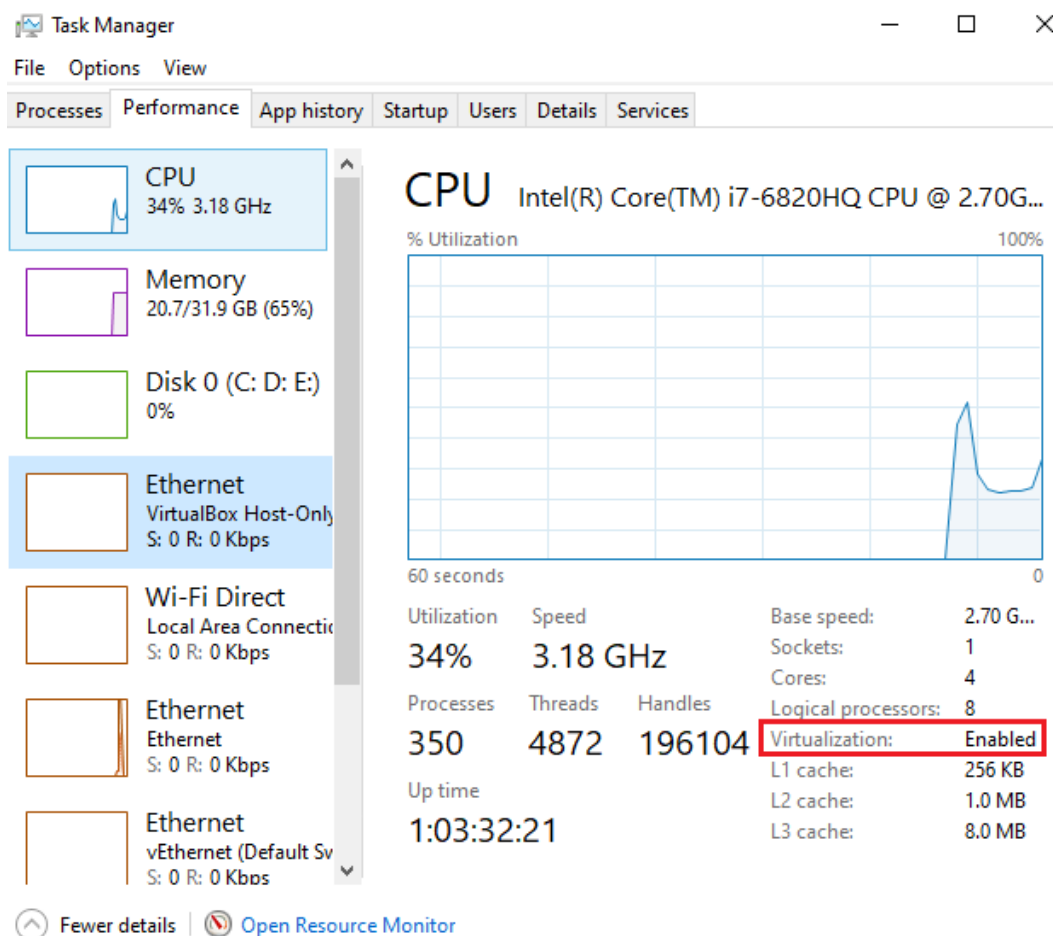
- Elasticsearch là công cụ storage, indexing
- LogStash là công cụ ghi log
- Kibana là cung cụ giúp hiển thị dữ liệu



Để nhận log, xử lý, index, lưu trữ, hiển thị thì bộ ba ELK thường đi kèm với nhau.

Cả bộ 3 trên đều hỗ trợ cài đặt trên Windows, trong phạm vi tài liệu này sẽ bắt đầu nó với Docker (tương lai các công cụ cũng sẽ sử dụng Docker).

Để bắt đầu với Docker cũng rất đơn giản, nhất là máy tính của bạn đã cài Windows 10. Trước khi cài docker thì bạn phải chắc chắn CPU của máy hỗ trợ ảo hóa, vào Task Manager, mục Performance, xem mục sau:



Đến thời điểm tháng 4/2020 thì bạn chú ý trong máy tính không nên cài một chương trình máy ảo như VirtualBox và VMWare chung với Docker for Windows. (Dự kiến đến tháng 6,7 / 2020 thì Microsoft phát hành một bản hỗ trợ song song). Việc sử dụng

Docker vô cùng đơn giản. Trước hết bạn truy cập địa chỉ sau để cài đặt:

<https://docs.docker.com/docker-for-windows/install/> Việc cài đặt rất đơn giản, nếu không vi phạm điều kiện nào thì sau khoảng vài phút là xong. Để kiểm tra việc cài đặt thành công, bạn mở Windows Power Shell (PS) gõ lệnh `docker --version` nó sẽ hiển thị ra version hiện tại.

Tiếp theo là add Image của ELK để thực hiện việc test. Bạn truy cập

<https://hub.docker.com/r/sebp/elk/>

Trên trang này đã có sẵn pull command để lấy Image về. Bạn mở Windows Power Shell hoặc CMDer (<https://cmdr.net/>). Khuyến khích sử dụng cmdr vì nó hỗ trợ cả tập lệnh Windows và Linux. Sau khi gõ các lệnh trên thì docker sẽ cài đặt image về máy:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\admin> docker --version
Docker version 19.03.8, build afacb8b
PS C:\Users\admin> docker pull sebp/elk
Using default tag: latest
latest: Pulling from sebp/elk
c64513b74145: Pull complete
01b8b12bad90: Pull complete
c5d85cf7a05f: Pull complete
b6b268720157: Pull complete
e12192999ff1: Pull complete
d39ece66b667: Pull complete
65599be66378: Pull complete
9ab7392d53a4: Downloading [=====] 71.56MB/131MB
6389c1d79bee: Downloading [=] 6.962MB/298.1MB
7123b4fed3ed: Waiting
036e4c8a19c7: Waiting
a121a102f7b7: Waiting
122bf958021c: Waiting
b7b1cc048c9c: Waiting
e22c806866e9: Waiting
2b2a7d5fcb00: Waiting
60298c011976: Waiting
989a68350c03: Waiting
```

Rất đơn giản, bạn chỉ việc chờ là xong. Để cấu hình ELK thì bạn truy cập <https://elk-docker.readthedocs.io/#usage> Sẽ có lệnh `docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -it --name elk sebp/elk`

Trong đó:

- -p là lệnh mapping cổng
- -it là chạy ở chế độ interactive
- --name là tên

Sau khi gõ xong thì image sẽ chạy như sau:

```
λ docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -it --name elk sebp/elk
* Starting periodic command scheduler cron [ OK ]
* Starting Elasticsearch Server [ OK ]
waiting for Elasticsearch to be up (1/30)
waiting for Elasticsearch to be up (2/30)
waiting for Elasticsearch to be up (3/30)
```

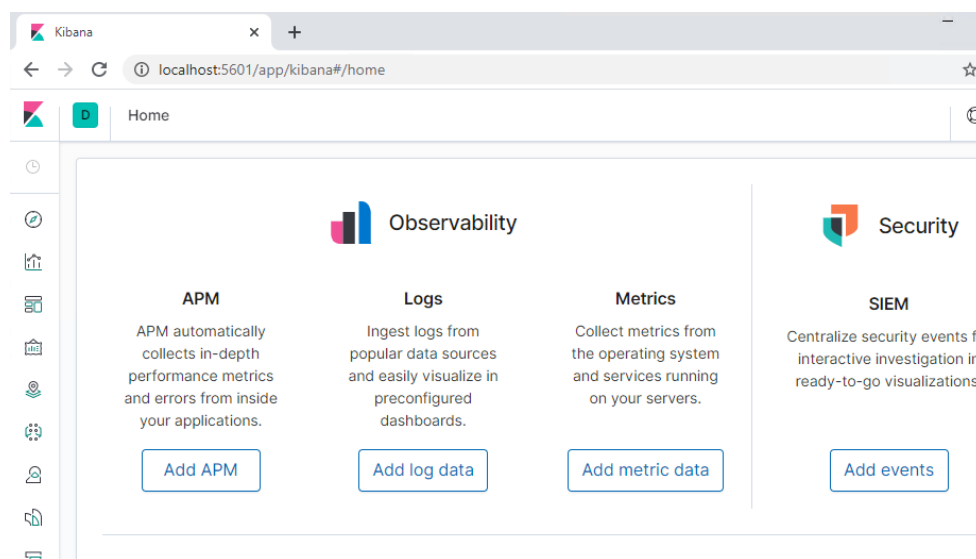
Bạn chờ 1-2p cho hệ thống boot xong. Truy cập <http://localhost:9200> bạn sẽ thấy Elasticsearch đã chạy:



← → ↻ ⓘ localhost:9200

```
{
  "name" : "elk",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "ScLNVRA7RimNePOcXJVfTA",
  "version" : {
    "number" : "7.6.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "aa751e09be0a5072e8570670309b1f12348f023b",
    "build_date" : "2020-02-29T00:15:25.529771Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Truy cập <http://localhost:5601> thì Kibana đã chạy (có thể chờ lâu một chút).



Đến bước này thì coi như đã cài đặt xong bộ ELK.

Quay về ví dụ trong source code thì bạn sẽ thấy cấu hình như sau:

```
Log.Logger = new LoggerConfiguration()
// .ReadFrom.Configuration(config)
// .Enrich.WithAspNetCoreHttpContext(provider, AddCustomContextDetails)
// .Enrich.FromLogContext()
// .Enrich.WithMachineName()
// .Enrich.WithProperty("Assembly", $"{name.Name}")
// .Enrich.WithProperty("Version", $"{name.Version}")
// .WriteTo.File(new CompactJsonFormatter(),
//     $"Logs\\Test.json")
// .WriteTo.Logger(lc => lc
//     .Filter.ByIncludingOnly(Matching.WithProperty("ElapsedMilliseconds"))
//     .WriteTo.MSSqlServer(
//         connectionString: @"Server=.\\sqlexpress;Database=Logging;Trusted_Connection=True;",
//         tableName: "PerfLogNew",
//         autoCreateSqlTable: true,
//         columnOptions: GetSqlColumnOptions()))
// .WriteTo.Logger(lc => lc
//     .Filter.ByIncludingOnly(Matching.WithProperty("UsageName"))
//     .WriteTo.Elasticsearch(new ElasticsearchSinkOptions(new Uri("http://localhost:9200"))
//     {
//         AutoRegisterTemplate = true,
//         AutoRegisterTemplateVersion = AutoRegisterTemplateVersion.ESv6,
//         IndexFormat = "usage-{0:yyyy.MM.dd}"
//     }
// ))
// .WriteTo.Logger(lc => lc
//     .Filter.ByExcluding(Matching.WithProperty("ElapsedMilliseconds"))
//     .Filter.ByExcluding(Matching.WithProperty("UsageName"))
//     .WriteTo.Elasticsearch(new ElasticsearchSinkOptions(new Uri("http://localhost:9200"))
//     {
//         AutoRegisterTemplate = true,
//         AutoRegisterTemplateVersion = AutoRegisterTemplateVersion.ESv6,
//         IndexFormat = "error-{0:yyyy.MM.dd}"
//     }
// ))
// .CreateLogger();
```

Thực tế thì cấu hình Log sẽ như trên. Có thể ghi dữ liệu vào rất nhiều Sink khác nhau. Bạn chú ý đến các thuộc tính Filter. Chỉ một số thông tin là đáng ghi log và đánh giá. Không nên ghi hết sẽ làm chương trình phân tích rất nặng.

Thử ghi vào log vào ElasticSearch thì bạn sẽ thấy ở Kibana như sau:

3.4. Tự viết Sink của riêng mình