

Tutorial ELK-Beat

Felipe Luís Pinheiro Marcelo Antonio

30 de Março de 2020

Resumo

Este tutorial foi desenvolvido para ajudar o desenvolvedores do Computretra a implementar aplicações em microserviços o sistema da Elastic, ELK Stack junto com o Filebeat, para realizar o log do sistema.

Usamos como exemplo uma aplicação .Net com o Serilog, porém esse sistema pode ser usado por qualquer tipo de aplicação com qualquer linguagem de programação.

O código fonte desse tutorial pode ser acessado no [github](https://github.com/flpinheiro/Tutorial-ELK-Beat.git) pelo link <https://github.com/flpinheiro/Tutorial-ELK-Beat.git>, sintam-se livre para usa-lo e modifica-lo, se possível mantenham referência para a fonte original.

1 Introdução

Nesta seção faremos uma rápida introdução das ferramentas e tecnologias utilizadas para o desenvolvimento desse projeto.

1.1 ELK Stack + Filebeat

O ELK Stack e o filebeat são mantidos e desenvolvidos pela [Elastic](https://www.elastic.co/)¹ sendo open source e de utilização gratuita e tendo a versão 7.6 como a mais recente, também possui uma versão paga que é chamada de [Elastic Cloud](https://www.elastic.co/cloud)² e é composto de:

- [Elasticsearch](https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html)³: um banco de dados nosql, que armazena o dados de forma indexada em formato sql de modo a ser rápido e leve e que possui uma api RESTfull.

- [Logstash](https://www.elastic.co/guide/en/logstash/current/index.html)⁴: um ingestor de dados que pode receber dados de várias fontes diferentes trata-los e reencaminha-los para os destinos apropriados com um formato mais adequado para o destino.

- [Kibana](https://www.elastic.co/guide/en/kibana/current/index.html)⁵ é um motor gráfico que serve para gerenciar os dados do Elasticsearch de forma fácil com o uso da api RESTfull.

¹<http://www.sharelatex.com>

²<https://www.elastic.co/cloud/>

³<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

⁴<https://www.elastic.co/guide/en/logstash/current/index.html>

⁵<https://www.elastic.co/guide/en/kibana/current/index.html>

- [Filebeat](#)⁶: é um despachador de logs, desempenha seu papel ao lado da aplicação onde lê os logs gerado pela aplicação e despacha para o destino apropriado de modo automatico e autonomo, existem outras versões de beat da elastic, tal como o metricbeat que serve para despachar metricas de uso e o Auditbeat que serve para auditar a atividade dos usuários e dos processos, para mais informações acesse a [Beats plataforma](#)⁷. Para mais informações consulte o site da mantedora.

1.2 .Net Framework

.Net Framework é um framework open source, gratuito, mantido e desenvolvido pela [Microsoft Corporation](#)⁸ junto com a [.Net Foundation](#)⁹ tendo como principal linguagem o [C#](#)¹⁰, estando atualmente na versão [.Net Core 3.1](#)¹¹.

1.3 Serilog

[Serilog](#)¹² é uma biblioteca para .Net que prove suporte para logs de diagnostico de diversas formas diferentes, sendo open source e gratuita.

Neste tutorial estamos usando o serilog para arquivo com a formatação de log em formato elasticsearch e definimos toda a configuração no appsettings.json da aplicação de modo que temos uma configuração mais dinâmica e livre de codificação, pois ao mudarmos o arquivo appsettings.json somos capazes de mudar o comportamento de todo sistema de log da aplicação sem fazer nenhuma alteração no código.

1.4 Docker

[Docker](#)¹³ é uma tecnologia de virtualização de aplicações a qual permite que as aplicações sejam executadas dentro de um container com todo o ambiente necessário para seu correto funcionamento. O sistema docker é mantido e desenvolvido pela Docker inc sendo open source e gratuito.

2 Desenvolvimento

O desenvolvimento desse sistema está dividido em duas partes que são:

- O ELK Stack que é executado em um servidor dedicado para essa solução.
- Aplicação + Serilog + Filebeat que são executados nos servidores de aplicação.

⁶<https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html>

⁷<https://www.elastic.co/guide/en/beats/libbeat/7.6/index.html>

⁸<https://www.microsoft.com/>

⁹<https://dotnetfoundation.org/>

¹⁰<https://docs.microsoft.com/pt-br/dotnet/csharp/>

¹¹<https://dotnet.microsoft.com/>

¹²<https://serilog.net/>

¹³<https://www.docker.com/>

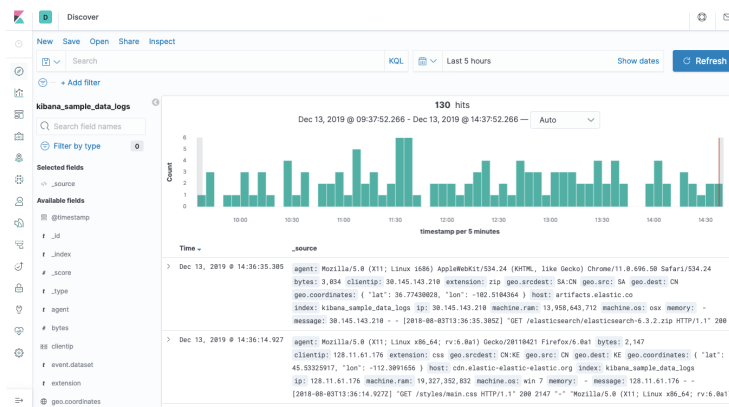


Figura 1: Tela Discovery do Kibana

Sendo que todos os sistemas estão configurados para serem rodados em container docker, micro serviços.

Abaixo estaremos discutindo a implementação completa de cada parte do sistema a ser implementada.

2.1 ELK Stack

O código completo dessa parte pode ser encontrado em <https://github.com/flpinheiro/docker-elk.git> que é uma versão modificada de repositório <https://github.com/deviantony/docker-elk.git> proposto pelo site [Logz.io](https://logz.io)¹⁴, sendo que no readme do repositório contam todas as informações necessárias para que o seja executado o projeto, porém aqui faremos um pequeno resumo.

Primeiramente é necessário fazer o clone do projeto desejado, para tanto utilize o comando:

```
git clone https://github.com/flpinheiro/docker-elk.git
```

Para rodar o Stack basta executar

```
cd /docker-elk
docker-compose up -d
```

O sistema do docker automaticamente fará o download das imagens necessárias criará os volumes e as networks fará o sistema começar a funcionar, depois de alguns minutos se tudo tiver dado certo basta acessar localhost:5601 para poder ter acesso ao kibana, usando o login padrão: elastic e a senha padrão: changeme, visualizando a figura 1

Se for a primeira vez que se acessa o kibana no seu sistema será necessário clicar em Management, veja figura 2, depois em Index Patterns e por fim em Create Index Pattern para poder criar um index para o Kibana, siga a numeração

¹⁴<https://logz.io/blog/elk-stack-on-docker/>

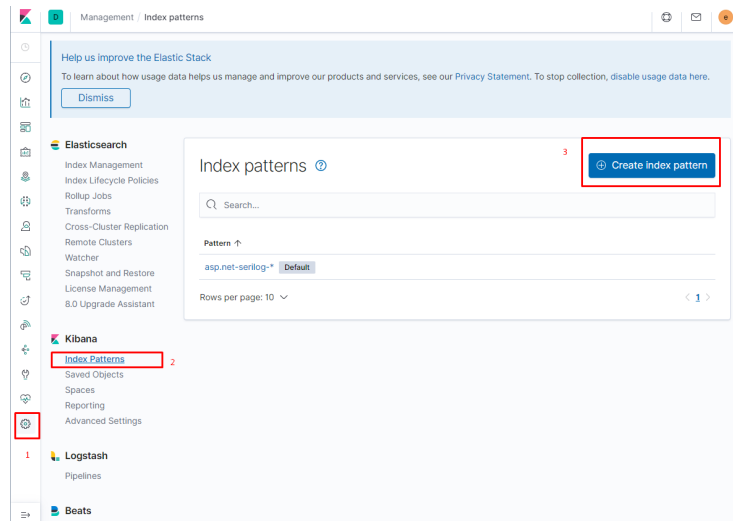


Figura 2: Tela Management do Kibana

dos quadrados vermelhos, após criar o index volte para a tela do Discovery [1](#) e os logs do sistema apareceram como uma lista em ordem crescente de antiguidade.

Nas pastas elasticsearch, kibana e logstash existe uma pasta config em cada com os arquivos elasticsearch.yml, kibana.yml, logstash.yml, respectivamente, estes são os arquivos de configuração de segurança dos serviços, onde se configura as senhas e o endereço ip de funcionamento do sistema, para testes locais não é necessário nenhuma modificação neles.

Na pasta logstash existe uma pasta pipeline a qual possui um arquivo logstash.conf, este arquivo define o funcionamento básico do logstash definindo seus input, filter e output como no exemplo a seguir, que é o que estamos usando nesse sistema.

```
input {
  tcp {
    port => 5000
  }
  beats {
    port => 5044
  }
}

filter {
  json {
    source => "message"
  }
}
```

```

output {
  elasticsearch {
    hosts => "elasticsearch:9200"
    user => "elastic"
    password => "changeme"
    index => "%{[@metadata][beat]}"
  }
}

```

Aqui podemos ver as três partes que compoem o arquivo logstash.conf:

input que são as entradas do logstash, no caso a porta 5000 para acesso direto via TCP e a porta 5044 que é a porta padrão de funcionamento do filebeat.


filter neste caso estamos apenas fazendo um parse da mensagem de string para json, pois na aplicação já estamos tratando a os logs de modo a chegarem corretamente no ELK Stack.

output apenas enviamos para o elasticsearch utilizando o index “%[@metadata][beat]” que está chegando pronto do filebeat e enviamos para o host definido no próprio elastisearch.

2.2 ASP.Net Core + Serilog + Filebeat + Docker

Nesta seção trabalhamos com as explicações para construção de uma aplicação .Net desenvolvida para trabalhar em orquestração de container docker via micro serviços e posteriormente implementaremos o serviço do serilog e o micro serviço do Filebeat.

2.3 .Net no Visual Studio IDE

Case esteja trabalhando em Windows com o Visual Studio fica relativamente simples de criar uma aplicação dotnet e defini-la como orquestração docker, simplesmente crie a aplicação e posteriormente clique com o botão direito no projeto e selecione Add  Container Orchestrator Support, por fim selecione o sistema operacional Linux ou Windws e pronto a orquestração docker está funcionando, execute normlamente o docker-compose e a aplicação estará funcionando.

2.4 .Net via linha de comando

Porém caso esteja trabalhando em linux usando o visual Studio code ou outro eidtor de codido que não seja o visual Studio IDE teremos que fazer manualmente, a seguir detalhamos o processo, que também pode ser encontrado no site do docs da microsoft, nos seguintes links:

- [Containerize um aplicativo .NET Core](#)
- [Imagens do Docker para o ASP.NET Core](#)

Neste estudo focaremos no segundo exemplo já que estamos interessados em Aplicações Web.

crie no diretório raiz do projeto um arquivo chamado “Dockerfile”, sem as aspas e acrescente o seguinte código:

```
# https://hub.docker.com/_/microsoft-dotnet-core
FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
WORKDIR /source

# copy csproj and restore as distinct layers
COPY *.sln .
COPY <<nome_projeto>>/*.csproj ./<<nome_projeto>>/
RUN dotnet restore

# copy everything else and build app
COPY <<nome_projeto>>/. ./<<nome_projeto>>/
WORKDIR /source/<<nome_projeto>>
RUN dotnet publish -c release -o /app --no-restore

# final stage/image
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1
WORKDIR /app
EXPOSE 80
EXPOSE 443
COPY --from=build /app ./
ENTRYPOINT ["dotnet", "<<nome_projeto>>.dll"]
```