Tutorial ELK-Beat

Felipe Luís Pinheiro Marcelo Antonio

2 de Abril de 2020

Resumo

Este tutorial foi desenvolvido para ajudar o desenvolvedores da Compuletra 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 programaçàо.

O código fonte desse tutorial pode ser acessado no github 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[1] sendo open source e de utilização gratuita e tendo a versão 7.6 como a mais recente, também possuí uma versão paga que é chamada de Elastic Cloud[2], sendo que são compostas de:

- Elasticsearch[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[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[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 plataform[7].
Para mais informações consute 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 appsetings.json da aplicação de modo que temos uma configuração mais dinâmica e livre de codificação, pois ao mudarmos o arquivo appsettings.json somos capazesde mudar o comportamento de todo sistema de logo 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/

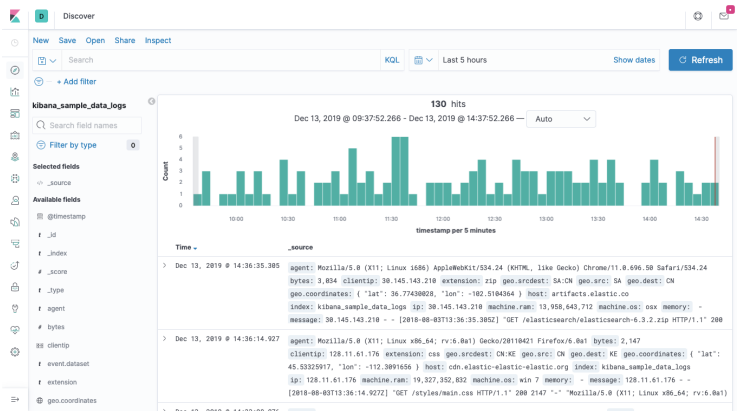

Figura 1: Tela Dicovery do Kibana

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 códogo completo dessa parte pode ser encontrado em https://github.com/flpinheiro/docker-elk.git que é uma versão modificada de repositorio https://github.com/deviantony/docker-elk.git proposto pelo site Logz.io[14], sendo que no readme do repositorio 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/

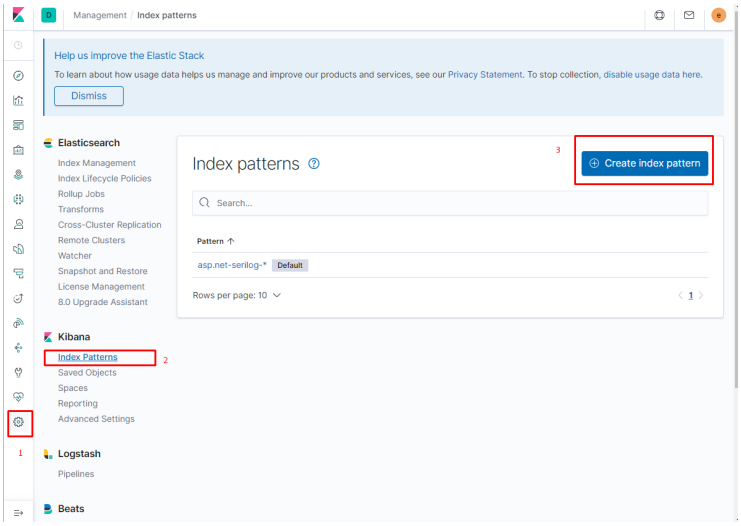

Figura 2: Tela Management do Kibana

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 Serilog

Nesta Seção implementaremos em uma aplicação Web tipo mvc o sistema log da Serilog para ser usado com o Filebeat.

Crie uma aplicação nova com o comando

```
dotnet new mvc -o <<nome_projeto>>
```

onde `<<nome_projeto>>` é o nome do projeto escolhido, ou use uma aplicação já existênte, por exemplo a desenvolvida na seção anterior, recomendado.

Começaremos instalando os pacotes necessários para que a ferramenta funcione conforme queremos, portanto, instale:

1. Install-Package Serilog.Sinks.File -version 4.1.0
2. Install-Package serilog.formatting.elasticsearch -version 8.0.1
3. Install-Package Serilog.Extensions.Logging -version 3.0.1
4. Install-Package Serilog.Settings.Configuration -version 3.1.0
5. Install-Package Serilog.Sinks.Elasticsearch -version 8.0.1

ou

1. dotnet add package Serilog.Sinks.File --version 4.1.0
2. dotnet add package serilog.formatting.elasticsearch --version 8.0.1

3. dotnet add package Serilog.Extensions.Logging --version 3.0.1

4. dotnet add package Serilog.Settings.Configuration --version 3.1.0

5. dotnet add package Serilog.Sinks.Elasticsearch --version 8.0.1

qualquer duvida ou dificuldade procure no http://www.nuget.org, lá aparecerá os comandos exatos a serem inseridos, inclusive com a opção de versão, que está sendo omitida aqui o que fará com que o sistema instala a última versaão estável disponivel.

Inclua no appsettings.json a seguinte sequencia de código:

```
  "Serilog": {
    "Using": [ "Serilog.Formatting.Elasticsearch",
"Serilog.Sinks.Elasticsearch",
"Serilog.Sinks.File" ],
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "File",
        "Args": {
          "path": "/var/log/serilog/log.log",
          "rollingInterval": "Hour",
          "formatter":
"Serilog.Formatting.Elasticsearch.ElasticsearchJsonFormatter,
Serilog.Formatting.Elasticsearch",
          "shared": "true"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithMachineName", "WithThreadId" ],
    "Properties": {
      "Application": "<<nome_projeto>>",
      "Version":  "<<versao_projeto>>"
    }
  }
```

este são os parametros de configuração que serão usados pelo serilog para realizar os logs do sistema.

Caso esteja trabalhando em windows o path deverá ser escrito como, por exemplo,

```
c:\\temp\\log\\serilog\\log.log
```

Os parametros `<<nome_projeto>>` e `<<versao_projeto>>` serão utilizados pelo kibana e pelo elasticsearch para indexação de consulta e serão enviados juntos com os logs.

Na classe Startup.cs do seu projeto faça as seguinte modificações
- nos usings

```
using Microsoft.Extensions.Logging;
using Serilog;
```

- No Construtor adicione:

```
Log.Logger = new LoggerConfiguration()
.ReadFrom.Configuration(configuration)
.CreateLogger();
```

- acrescente a assinatura do metodo configure:

```
ILoggerFactory loggerFactory
```

e acrescente logo nas primeiras linhas do metodo

```
loggerFactory.AddSerilog();
```

Para realizar logs utilize o `ILogger` do controlador, sendo que é possível realizar logs de informações, de erros e de outros tipos.

2.2.1 Serilog + ELK Stack

Nesta seção vamos modificar o Serilog para que ele possa enviar os logs diretamente para o Logstash ou o Elasticsearch sem o intermediário do Filebeat, para tanto só precisamos fazer algumas pequenas alterâções no appsettings.json, que deverá ficar como se segue:

```
  "Serilog": {
    "Using": [ "Serilog.Formatting.Elasticsearch",
"Serilog.Sinks.Elasticsearch",
"Serilog.Sinks.File" ],
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "Elasticsearch",
        "Args": {
"nodeUris": "http://localhost:5000;http://remotehost:5000/",
          "indexFormat": "custom-index-{0:yyyy.MM}",
          "templateName": "myCustomTemplate",
          "formatter":
"Serilog.Formatting.Elasticsearch.ElasticsearchJsonFormatter,
Serilog.Formatting.Elasticsearch"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithMachineName", "WithThreadId" ],
    "Properties": {
      "Application": "<<nome_projeto>>",
      "Version":  "<<versao_projeto>>"
    }
  }
```

com essas modificações o serilog já estará enviando os logs diretamente para o Logstash, ou poderá enviar diretamente para o elasticsearch se for utilizado a porta 9200, que é a porta padrão do elasticsearch, mas nesse caso pode ser necessário passar a senha e o login de conexão do elasticsearch.

Para mais parametros de configuração acesse https://github.com/serilog/serilog-sinks-elasticsearch.

2.3 ASP.Net Core + Docker

Nesta seção trabalhamos com as explicações para construção de uma aplicação .Net desenvolvida para trabalhar em orquestração de container docker via micro serviços.

2.3.1 .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 normalamente o docker-compose e a aplicação estará funcionando.

Para mais dúvidas veja os tutoriais da Microsoft nos links abaixo:

1. Container Tools in Visual Studio
2. Tutorial: Create a multi-container app with Docker Compose

2.3.2 .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.

Abra o terminal e navegue até a pasta onde deseja salvar o projeto utilizando o comando cd, então execute o seguinte comando:

```
dotnet new mvc -o <<nome_projeto>>
cd <<nome_projeto>>
code .
```

O primeiro comando cria o projeto com o template mvc como o nome `<<nome_projeto>>`, o segundo comando navega para dentro da pasta do projeto e por fim abrimos o Visual Studio Code com o ultimo comando.

Agora crie um arquivo chamado “Dockerfile”, sem as aspas e insira o seguinte conteúdo dentro dele.

```
# 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 *.csproj ./
RUN dotnet restore

# copy everything else and build app
COPY . ./aspnetapp/
WORKDIR /source/aspnetapp
RUN dotnet publish -c release -o /app

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

Execute

```
docker build -t <<nome_imagem>> .
```

Este comando irá criar a imagem para o seu projeto, verifique executando o comando

```
docker images
```

Caso deseje executar a imagem use

```
docker run -it --rm -p 5000:80 --name <<nome_container>> <<nome_imagem>>
```

Execute em outro console

```
docker ps
```

para ver seu container sendo executado, termine a execução com “ctrl + C”.

Agora crie um arquivo chamado “docker-compose.yml”, nele acrescente o seguinte código

```
version: '3.4'

services:
  docker_mvc:
    container_name: docker_mvc_container
```

```
    image: docker_mvc
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "5000:80"
```

agora execute

```
docker-compose up -d
```

Pronto se tudo funcionou como deveria você tem um Container sendo executado.

2.4 Filebeat

Para utilizarmos o filebeat é importante que tenha sido feita as duas etapas anteriores no mesmo projeto, ou seja, que ele esteja configurado para realizar log utilizando o serilog e esteja configurado para ser executado em container docker via docker-compose.

Crie no diretorio raiz do seu projeto, o mesmo onde se encontra o arquivo docker-compose.yml, o seguinte conjunto de pastas `etc/filebeat` e dentro inclua um arquivo filebeat.yml com o seguinte conteúdo

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/filebeat/*.log

filebeat.config.modules:
  reload.enabled: false

output.logstash:
  hosts: ["0.0.0.0:5044"]

processors:
  - add_host_metadata: ~
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~

logging.level: debug
logging.selectors: ["*"]
```

este é o arquivo de configuração básica do filebeat, existem diversas outras configurações que podem ser usadas, porém essas são as minimas para que o sistema funcione corretamente.

Dentro do arquivo “docker-compose.yml” iremos adicionar o serviço do filebeat, mais um volume compartilhado para os dois container, de modo que o filebeat tenha acesso aos arquivos de log gerados pelo sistema de modo que o arquivo “docker-compose.yml” vai ficar assim:

```
version: "3.4"

services:
  docker_mvc:
    container_name: docker_mvc_container
    image: docker_mvc
    build:
      context: .
      dockerfile: Dockerfile
    restart: unless-stopped
    ports:
      - "5000:80"
    volumes:
      - dotnetvolume:/var/log/serilog
    networks:
      - dotnet

  filebeat:
    container_name: beats
    image: docker.elastic.co/beats/filebeat:7.6.1
    user: root
    restart: unless-stopped
    depends_on:
    - docker_mvc
    volumes:
      - dotnetvolume:/var/log/filebeat/:ro
      - /var/run/docker.sock:/var/run/docker.sock
      - /var/lib/docker/containers:/var/lib/docker/containers:ro
      - ./etc/filebeat/filebeat.yml:/usr/share/filebeat/filebeat.yml:ro
    networks:
      - dotnet

volumes:
   dotnetvolume:
networks:
  dotnet:
    driver: bridge
```

Pronto se estava tudo funcionando nos passos anteriores, basta executar

```
docker-compose up -d
```

e você terá subido sua aplicação.

3 Conclusão

Neste tutorial mostramos como desenvolver uma aplicação do zero usando linha de comando e Visual Studo Code e subila para conteiners docker junto com uma aplicação de filebeat para realizar as entregas de logs para um ELK Stack que estará sendo executado em um servirdor separado.

Caso precise no repositorio desse tutorial, que pose ser encontrado nesse link: https://github.com/flpinheiro/Tutorial-ELK-Beat.git, existe uma pasta "sample" a qual é possui uma cópia dos repositorios do Docker-elk e também uma aplicação mvc implementada seguindo o passo a passo desse tutorial e rodando paralelo ao filebat.