Estratégias para coleta de Métricas e Logs em WebAPI utilizando Csharp com OpenTelemetry

Nagib Sabbag Filho

Leaders.Tec.Br, 1(13), ISSN: 2966-263X, 2024.

e-mail: profnagib.filho@fiap.com.br

DOI: https://doi.org/10.5281/zenodo.13883405

PermaLink: https://leaders.tec.br/artigo/estrategias-para-coleta-de-metricas-e-logs-em-webapi-utilizando-csharp-com-opentelemetry

Received: 26 Sep 2024 / Accepted: 28 Sep 2024 / Published online: 30 Sep 2024

Abstract:

O artigo apresenta uma introdução ao OpenTelemetry, uma ferramenta essencial para a observabilidade de sistemas distribuídos. Com o aumento da complexidade das aplicações, o OpenTelemetry oferece um padrão unificado para a coleta de métricas, logs e rastreamento. O texto detalha a configuração inicial em aplicações C#, incluindo a coleta de métricas e logs, integração com backends de monitoramento como Jaeger e boas práticas para instrumentação.

Key words:

Estratégias, Coleta de Métricas, Logs, Sistemas C#, OpenTelemetry, Monitoramento, Observabilidade, Instrumentação, Telemetria, Análise de Performance, Diagnóstico de Erros, Integração, API, Dados Estruturados, APM (Application Performance Management), Tracing, Eventos, Contexto de Execução, Exportação de Dados, Visualização, Ferramentas de Monitoramento, Melhoria Contínua, Desenvolvimento Ágil.

Introdução ao OpenTelemetry

O OpenTelemetry é uma coleção de ferramentas, APIs e SDKs que permitem a observabilidade de sistemas distribuídos. Com o aumento da complexidade das aplicações modernas, a necessidade de monitoramento eficaz se tornou crucial. O OpenTelemetry fornece um padrão unificado para a coleta de métricas e logs, permitindo que desenvolvedores e engenheiros de SRE compreendam melhor o comportamento de suas aplicações. Essa ferramenta é um projeto open-source que visa fornecer um caminho para que desenvolvedores implementem observabilidade em suas aplicações de maneira consistente e eficiente.

O conceito de observabilidade envolve não apenas a coleta de dados, mas também a capacidade de entender esses dados e tomar decisões informadas para melhorar o desempenho e a confiabilidade das aplicações. Com isso, o OpenTelemetry se destaca como uma solução que integra rastreamento, métricas e logs, facilitando a depuração e a análise de problemas em sistemas complexos.

Configuração Inicial do OpenTelemetry em Aplicações C#

Para começar a usar o OpenTelemetry em um projeto C#, é necessário instalar os pacotes NuGet apropriados. A seguir, apresentamos um exemplo de como configurar o OpenTelemetry em uma aplicação ASP.NET Core:

```
dotnet add package OpenTelemetry
dotnet add package OpenTelemetry.Extensions.Hosting
dotnet add package OpenTelemetry.Instrumentation.AspNetCore
```

Após instalar os pacotes, você pode configurar o OpenTelemetry em seu método ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddOpenTelemetry()
        .WithTracing(builder =>
        {
            builder
                .AddAspNetCoreInstrumentation()
                .AddHttpClientInstrumentation()
                .AddConsoleExporter();
        });
}
```

Com isso, você habilita a coleta de rastreamento para suas aplicações, permitindo que o OpenTelemetry colete dados de solicitações HTTP recebidas e enviadas. Esta configuração básica é um ótimo ponto de partida para adicionar mais instrumentações conforme necessário.

Coleta de Métricas com OpenTelemetry

A coleta de métricas é uma parte essencial do monitoramento de aplicações. O OpenTelemetry oferece suporte para métricas de forma simples e eficaz. Aqui está um exemplo de como coletar métricas de contagem em uma aplicação C#:

```
using OpenTelemetry.Metrics;

public void ConfigureServices(IServiceCollection services)
{
    services.AddOpenTelemetryMetrics(builder =>
    {
        builder.AddAspNetCoreInstrumentation();
        builder.AddMeter("MyApplication");
        builder.AddConsoleExporter();
    });
}

private static readonly Counter<long> myCounter =
    MeterProvider.Default.GetMeter("MyApplication").CreateCounter<long>("my_counter");

public void SomeMethod()
{
    myCounter.Add(1);
}
```

No exemplo acima, você está criando um contador que contabiliza quantas vezes um método específico é chamado. Isso é útil para entender o volume de chamadas e ajudar na análise de desempenho.

Implementação de Logs com OpenTelemetry

A implementação de logs no OpenTelemetry é igualmente importante. Você pode coletar logs utilizando a biblioteca OpenTelemetry.Logs. Veja como configurar a coleta de logs:

```
dotnet add package OpenTelemetry.Logs
```

Em seguida, configure a coleta de logs em seu método ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddLogging(builder =>
    {
        builder.AddOpenTelemetry();
    });
}
```

Com isso, todos os logs gerados na aplicação serão coletados e poderão ser enviados para os backends de monitoramento configurados. A coleta de logs é crucial para entender o que está acontecendo na aplicação, especialmente em casos de falhas ou comportamentos inesperados.

Integração com Backend de Monitoramento

Após a coleta de métricas e logs, o próximo passo é integrar essas informações a um backend de monitoramento. O OpenTelemetry suporta várias opções, como Jaeger, Prometheus, e Zipkin. A seguir, um exemplo de integração com Jaeger:

```
dotnet add package OpenTelemetry.Exporter.Jaeger
```

Em seu método de configuração, adicione o exportador Jaeger:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddOpenTelemetry()
        .WithTracing(builder =>
        {
            builder
                .AddAspNetCoreInstrumentation()
                .AddJaegerExporter(options =>
                {
                    options.AgentHost = "localhost";
                    options.AgentPort = 6831;
                });
        });
}
```

Jaeger é uma ferramenta popular para rastreamento distribuído e permite visualizar as chamadas feitas entre serviços em uma arquitetura de microserviços. Integrar o OpenTelemetry com o Jaeger facilita a análise do desempenho e a identificação de gargalos nas comunicações entre serviços.

Estratégias para Criação de Instrumentações Personalizadas

Às vezes, métricas e logs prontos não são suficientes. Para isso, o OpenTelemetry permite a criação de instrumentações personalizadas. Veja um exemplo de como criar uma métrica personalizada:

```
private static readonly Histogram<double> myHistogram =
    MeterProvider.Default.GetMeter("MyApplication").CreateHistogram<double>("my_histogr
am");

public void MeasureExecutionTime(Action action)
{
    var startTime = DateTime.UtcNow;
    action();
    var executionTime = (DateTime.UtcNow - startTime).TotalMilliseconds;
```

```
    myHistogram.Record(executionTime);
}
```

No exemplo acima, você cria um histograma que mede o tempo de execução de uma ação específica. Essa métrica pode ser extremamente útil para identificar quais partes da sua aplicação estão levando mais tempo para serem executadas.

Monitoramento de Aplicações Distribuídas

Em ambientes de microserviços, o monitoramento se torna ainda mais desafiador. O OpenTelemetry facilita a coleta de dados de várias aplicações distribuídas. Ao usar identificadores de rastreamento, você pode correlacionar eventos entre serviços. Aqui está um exemplo de como usar o rastreamento em chamadas HTTP:

```
using System.Net.Http;

public async Task CallAnotherServiceAsync()
{
    using var httpClient = new HttpClient();
    var response = await httpClient.GetAsync("http://another-service/api/data");
    response.EnsureSuccessStatusCode();
}
```

Esse trecho de código demonstra como fazer chamadas a outros serviços enquanto mantém o contexto de rastreamento. Isso permite que você veja a jornada de uma solicitação através de diferentes serviços na sua arquitetura, ajudando na identificação de onde os problemas podem estar ocorrendo.

Boas Práticas para Coleta de Métricas e Logs

Para garantir a eficácia da coleta de métricas e logs, considere as seguintes boas práticas:

• Defina métricas relevantes para o negócio, focando no que realmente importa para os stakeholders.

• Evite coletar dados excessivos que possam causar sobrecarga e dificultar a análise.

• Implemente alertas baseados em anomalias nas métricas, permitindo uma resposta rápida a problemas.

• Utilize tags e atributos para enriquecer os dados coletados, tornando-os mais informativos e úteis na análise.

• Documente suas instrumentações e a lógica por trás das métricas para que outros membros da equipe possam entender e contribuir.

• Realize revisões periódicas das métricas coletadas para garantir que continuam a ser relevantes e úteis.

Considerações Finais

O OpenTelemetry é uma ferramenta poderosa para a coleta de métricas e logs em aplicações C#. Ao seguir as práticas e estratégias discutidas neste artigo, você poderá implementar uma solução de observabilidade robusta, que não apenas ajudará na detecção de problemas, mas também na melhoria contínua de suas aplicações. A observabilidade não é apenas uma questão de coleta de dados, mas de como esses dados são utilizados para guiar a evolução e a manutenção de suas aplicações.

Além disso, ao adotar o OpenTelemetry, você está se alinhando com as melhores práticas de desenvolvimento

moderno, onde a observabilidade é uma parte integrante do ciclo de vida do software. Isso não só melhora a qualidade de suas aplicações, mas também proporciona uma melhor experiência para os usuários finais.

Referências

• OPEN TELEMETRY. OpenTelemetry. Disponível em: https://opentelemetry.io/. Acesso em: 29 set. 2024.

• MICROSOFT. OpenTelemetry for .NET. Disponível em: https://github.com/open-telemetry/opentelemetry-dotnet. Acesso em: 29 set. 2024.

• JAeger. Jaeger, a Distributed Tracing System. Disponível em: https://www.jaegertracing.io/. Acesso em: 29 set. 2024.

Nagib é Professor Universitário e Tech Manager. Possui uma trajetória de conquistas em certificações técnicas e ágeis, incluindo MCSD, MCSA e PSM1. PG em Gestão de TI pelo SENAC e MBA em Tecnologia de Software pela USP, Nagib cursou programas de extensão do MIT e Universidade de Chicago. Outras conquistas incluem a autoria de um artigo sobre chatbots, revisado por pares e apresentado na Universidade de Barcelona.