Instrumentação com Bibliotecas OpenTelemetry

O envio de sinais para o Priax pode ser realizado através de instrumentação própria, alterando o código da aplicação a ser observada sem inserir componentes de terceiros à sua infraestrutura. Este processo, no entanto pode ser custoso e levar algum tempo. Para que se possa acelerar esse processo, recomendamos, apesar de não ser a única opção, as bibliotecas e SDKs open-source do OpenTelemetry. Tais bibliotecas seguem os padrões OpenTracing e se tornaram a mais difundida solução de instrumentação para o mercado de Observabilidade.

Como o OpenTelemetry facilita a instrumentação
Para tornar um sistema observável, é necessário instrumentá-lo: ou seja, o código dos componentes do sistema deve emitir traces, métricas e logs.

Com o OpenTelemetry, você pode instrumentar seu código de duas maneiras principais:

1. **Soluções baseadas em código**
   Utilizando APIs e SDKs oficiais disponíveis para a maioria das linguagens de programação.
2. **Soluções sem código**
   Ideais para casos em que você não pode ou não deseja modificar o aplicativo que precisa de telemetria.

Soluções baseadas em código

Essas soluções permitem uma telemetria mais profunda e detalhada, gerada diretamente pela sua aplicação. Elas utilizam a API do OpenTelemetry para produzir telemetria personalizada, complementando as informações coletadas por soluções sem código.

Soluções sem código

Perfeitas para começar rapidamente ou quando o aplicativo não pode ser modificado. Essas soluções oferecem uma telemetria rica baseada em bibliotecas utilizadas ou no ambiente em que a aplicação está sendo executada. Elas fornecem dados sobre o que está acontecendo nas "bordas" da aplicação, como interações externas e dependências.

> **Nota**: É possível usar ambas as abordagens ao mesmo tempo para maximizar a observabilidade.

Benefícios adicionais do OpenTelemetry

O OpenTelemetry vai além de oferecer soluções com e sem código. Ele também inclui os seguintes recursos:

- **Bibliotecas compatíveis:** Bibliotecas podem usar a API do OpenTelemetry como dependência, sem impactar aplicações que não importem o SDK.
- **Sinais flexíveis:** Para cada tipo de sinal (traces, métricas, logs), existem várias formas de criá-los, processá-los e exportá-los.
- **Correlação de sinais:** Com a propagação de contexto integrada, é possível correlacionar sinais, independentemente de onde foram gerados.
- **Recursos e Escopos de Instrumentação:** Permitem agrupar sinais por diferentes entidades, como o host, sistema operacional ou cluster Kubernetes.
- **Padrões e especificações:** Cada implementação de linguagem do OpenTelemetry segue os requisitos e expectativas definidos pela especificação oficial.
- **Convenções semânticas:** Fornecem um esquema de nomenclatura comum para padronizar métricas, logs e traces entre diferentes bases de código e plataformas.

Formas de Instrumentação

Instrumentação sem código

A instrumentação sem código adiciona as capacidades da API e SDK do OpenTelemetry à sua aplicação, geralmente por meio de uma instalação de agente ou algo semelhante a um agente. Os mecanismos específicos variam conforme a linguagem, incluindo manipulação de bytecode, monkey patching ou eBPF, para injetar chamadas à API e ao SDK do OpenTelemetry diretamente na aplicação.

Como funciona

Normalmente, a instrumentação sem código adiciona suporte para bibliotecas que sua aplicação utiliza. Isso significa que requisições e respostas, chamadas a bancos de dados, chamadas de filas de mensagens, entre outros, serão instrumentados automaticamente. No entanto, o código da sua aplicação geralmente não é instrumentado. Para isso, é necessário utilizar **instrumentação baseada em código**.

Além disso, a instrumentação sem código permite configurar as bibliotecas de instrumentação e os exportadores que serão carregados.

Configuração

Você pode configurar a instrumentação sem código por meio de variáveis de ambiente e outros mecanismos específicos da linguagem, como propriedades do sistema ou argumentos passados para métodos de inicialização. Para começar, é necessário apenas configurar o **nome do serviço** para identificá-lo no backend de observabilidade escolhido.

Outras opções de configuração incluem:

- Configuração específica para fontes de dados.
- Configuração de exportadores.
- Configuração de propagadores.
- Configuração de recursos.

Suporte de linguagens para instrumentação automática

A instrumentação automática está disponível para as seguintes linguagens:

- .NET
- Go
- Java
- JavaScript
- PHP
- Python

Para realizar a instrumentação de sua aplicação, utilizando o método Zero Code, siga as instruções da página do OpenTelemetry.

Instrumentação Baseada em Código

A instrumentação baseada em código permite criar telemetria personalizada diretamente no código da sua aplicação. Abaixo estão os passos essenciais para configurar essa abordagem:

1. Importar a API e o SDK do OpenTelemetry

- **Serviços:** Dependem da API e do SDK do OpenTelemetry.
- **Bibliotecas:** Apenas dependem da API.

Para mais detalhes sobre a API e o SDK, consulte a especificação do OpenTelemetry.

2. Configurar a API do OpenTelemetry

- **Fornecedores de Tracer e Meter:**
  - Crie um **TracerProvider** para gerar traços.
  - Crie um **MeterProvider** para gerar métricas.
- **Nomeação:**
  - Use um nome que identifique o componente sendo instrumentado.
  - Exemplo: para uma biblioteca, use algo como `com.example.myLibrary`.
  - Inclua uma versão no formato **semver** (ex.: `semver:1.0.0`).

3. Configurar o SDK do OpenTelemetry

- **Exportação de dados:** Configure o SDK para exportar telemetria a um backend de análise.
- **Opções específicas da linguagem:** Verifique as opções de ajuste disponíveis para a sua linguagem.

A configuração pode ser feita programaticamente, por meio de arquivos de configuração ou outros mecanismos.

4. Criar Dados de Telemetria

- **Traços e Métricas:**
  - Gere traços e eventos de métricas com os objetos `Tracer` e `Meter`.
- **Bibliotecas de Instrumentação:**
  - Use bibliotecas de instrumentação disponíveis para suas dependências.
  - Consulte o repositório ou registro da sua linguagem para obter mais informações.

5. Exportar Dados de Telemetria

- **Métodos de Exportação:**
  - **Exportação no processo:**
    - Importe e use **exportadores** para traduzir os objetos de telemetria do OpenTelemetry em formatos apropriados para ferramentas de análise (ex.: Jaeger ou Prometheus).
  - **Exportação via OTLP e Collector:**
    - Use o **protocolo OTLP** para enviar dados ao OpenTelemetry Collector, que pode atuar como um proxy, sidecar ou processo separado.
    - O Collector encaminha os dados para ferramentas de análise.

Para realizar a instrumentação de sua aplicação, utilizando o método baseado em código siga as instruções da página do OpenTelemetry.

Instrumentação de Bibliotecas

A instrumentação nativa de bibliotecas com OpenTelemetry oferece uma experiência aprimorada tanto para desenvolvedores quanto para usuários finais, eliminando a necessidade de expor e documentar hooks personalizados. Aqui está como funciona e os benefícios associados:

Como Adicionar Instrumentação Nativa

- **Instrumentação via Hooks ou *dynamic runtime patching*:**
  OpenTelemetry fornece bibliotecas de instrumentação para várias linguagens, que tipicamente utilizam hooks de bibliotecas ou *dynamic runtime patching* do código.

Vantagens da Instrumentação Nativa

1. **Melhoria na Observabilidade e Experiência do Usuário:**
   - Remove a necessidade de hooks personalizados, facilitando a integração.
   - APIs do OpenTelemetry são fáceis de usar e consistentes para os usuários finais.
2. **Telemetria Consistente e Correlação Aprimorada:**
   - Traços, logs e métricas provenientes do código da biblioteca e da aplicação são correlacionados, criando uma visão coesa dos eventos.
   - Convenções comuns garantem uniformidade entre tecnologias, bibliotecas e linguagens.
3. **Extensibilidade e Ajustes Fáceis:**
   - Sinais de telemetria podem ser ajustados (filtrados, processados, agregados) para atender a diferentes cenários de consumo.
   - OpenTelemetry oferece extensibilidade bem documentada para personalização.

Para realizar a instrumentação de sua aplicação, utilizando as bibliotecas nativas OpenTelemetry, siga as instruções da página do OpenTelemetry.

Instrumentação com Kubernetes Operator

A instrumentação de aplicações que rodam em workloads Kubernetes é facilitada pelo Opentelemetry Kubernetes Operator que é uma implementação de um Operador Kubernetes que gerencia coletores e a auto-instrumentação das cargas de trabalho usando bibliotecas de instrumentação do OpenTelemetry.

Introdução

O Operador OpenTelemetry é uma implementação de um Operador Kubernetes.

O operador gerencia:

- **Coletor OpenTelemetry**
- **Auto-instrumentação das cargas de trabalho** (pods) usando bibliotecas de instrumentação do OpenTelemetry

Início Rápido

Para instalar o operador em um cluster existente, certifique-se de ter o **cert-manager** instalado e execute:

```
kubectl apply -f https://github.com/open-telemetry/opentelemetry-
operator/releases/latest/download/opentelemetry-operator.yaml
```

Assim que a implantação do `opentelemetry-operator` estiver pronta, crie uma instância do Coletor OpenTelemetry (otelcol) como o exemplo abaixo:

```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: OpenTelemetryCollector
metadata:
  name: simplest
spec:
  config: |
    receivers:
      otlp:
        protocols:
          grpc:
            endpoint: 0.0.0.0:4317
          http:
            endpoint: 0.0.0.0:4318
    processors:

    exporters:
      # NOTA: Antes da versão v0.86.0, use `logging` ao invés de `debug`.
      debug:

    service:
      pipelines:
        traces:
          receivers: [otlp]
          processors: []
          exporters: [debug]
EOF
```

Para mais opções de configuração e para configurar a injeção de auto-instrumentação das cargas de trabalho usando as bibliotecas de instrumentação do OpenTelemetry, continue lendo aqui.

Injeção de Auto-instrumentação

Configurar Instrumentação Automática
Para gerenciar a instrumentação automática, o operador precisa ser configurado para identificar quais pods instrumentar e qual instrumentação automática usar nesses pods. Isso é feito por meio do **CRD de Instrumentação**.

A criação correta do recurso de Instrumentação é fundamental para que a instrumentação automática funcione. Certifique-se de que todos os endpoints e variáveis de ambiente estejam configurados corretamente.

.NET

O seguinte comando criará um recurso básico de Instrumentação configurado especificamente para serviços .NET:

```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```

Por padrão, o recurso de Instrumentação para serviços .NET usa OTLP com o protocolo HTTP/Protobuf. O endpoint configurado deve ser capaz de receber OTLP via HTTP/Protobuf, como no exemplo: `http://demo-collector:4318`.

Excluindo bibliotecas de instrumentação

Se você não quiser usar determinadas bibliotecas, configure as variáveis de ambiente `OTEL_DOTNET_AUTO_[SIGNAL]_[NAME]_INSTRUMENTATION_ENABLED=false`.

Exemplo:

```
dotnet:
  env:
    - name: OTEL_DOTNET_AUTO_TRACES_GRPCNETCLIENT_INSTRUMENTATION_ENABLED
      value: false
    - name: OTEL_DOTNET_AUTO_METRICS_PROCESS_INSTRUMENTATION_ENABLED
      value: false
```

Java

O seguinte comando criará um recurso básico de Instrumentação para serviços Java:

```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```

Por padrão, a instrumentação Java usa OTLP com HTTP/Protobuf.

Excluindo bibliotecas de instrumentação

Para desativar bibliotecas específicas:

- Use `OTEL_INSTRUMENTATION_[NAME]_ENABLED=false`.
- Para desativar todas por padrão e ativar somente algumas:

```
OTEL_INSTRUMENTATION_COMMON_DEFAULT_ENABLED=false
OTEL_INSTRUMENTATION_[NAME]_ENABLED=true
```

Configurar Instrumentação Automática
Para gerenciar a instrumentação automática, o operador precisa ser configurado para identificar quais pods instrumentar e qual instrumentação automática usar nesses pods. Isso é feito por meio do **CRD de Instrumentação**.

A criação correta do recurso de Instrumentação é fundamental para que a instrumentação automática funcione. Certifique-se de que todos os endpoints e variáveis de ambiente estejam configurados corretamente.

.NET

O seguinte comando criará um recurso básico de Instrumentação configurado especificamente para serviços .NET:

bash
```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```
Por padrão, o recurso de Instrumentação para serviços .NET usa OTLP com o protocolo HTTP/Protobuf. O endpoint configurado deve ser capaz de receber OTLP via HTTP/Protobuf, como no exemplo: `http://demo-collector:4318`.

Excluindo bibliotecas de instrumentação

Se você não quiser usar determinadas bibliotecas, configure as variáveis de ambiente `OTEL_DOTNET_AUTO_[SIGNAL]_[NAME]_INSTRUMENTATION_ENABLED=false`.

Exemplo:

yaml

```
dotnet:
  env:
    - name: OTEL_DOTNET_AUTO_TRACES_GRPCNETCLIENT_INSTRUMENTATION_ENABLED
      value: false
    - name: OTEL_DOTNET_AUTO_METRICS_PROCESS_INSTRUMENTATION_ENABLED
      value: false
```

Java

O seguinte comando criará um recurso básico de Instrumentação para serviços Java:

bash
```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```

Por padrão, a instrumentação Java usa OTLP com HTTP/Protobuf.

Excluindo bibliotecas de instrumentação

Para desativar bibliotecas específicas:

- Use `OTEL_INSTRUMENTATION_[NAME]_ENABLED=false`.
- Para desativar todas por padrão e ativar somente algumas:
  bash
  ```
  OTEL_INSTRUMENTATION_COMMON_DEFAULT_ENABLED=false
  OTEL_INSTRUMENTATION_[NAME]_ENABLED=true
  ```

Node.js

O seguinte comando criará um recurso básico de Instrumentação para serviços Node.js:

```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4317
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```

Por padrão, a instrumentação Node.js usa OTLP com gRPC.

Gerenciar bibliotecas de instrumentação

- Para ativar bibliotecas específicas:

```
nodejs:
  env:
    - name: OTEL_NODE_ENABLED_INSTRUMENTATIONS
      value: http,nestjs-core
```

- Para desativar específicas:

```
nodejs:
  env:
    - name: OTEL_NODE_DISABLED_INSTRUMENTATIONS
      value: fs,grpc
```

Python

O seguinte comando criará um recurso básico de Instrumentação para serviços Python:

```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
```

```
kind: Instrumentation
metadata:
  name: python-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```

Por padrão, Python usa OTLP com HTTP/Protobuf.

Logs em Python

Para ativar a instrumentação automática de logs:

```
python:
  env:
    - name: OTEL_PYTHON_LOGGING_AUTO_INSTRUMENTATION_ENABLED
      value: 'true'
```

Excluindo bibliotecas de instrumentação

Defina `OTEL_PYTHON_DISABLED_INSTRUMENTATIONS` com os pacotes a serem excluídos.

Go

Para serviços Go, crie o recurso básico de Instrumentação:

```
kubectl apply -f - <<EOF
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
```

```
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: "1"
EOF
```

A instrumentação Go usa um agente eBPF em um sidecar, que requer permissões elevadas.

Adicione anotações para ativar a instrumentação automática:

```
instrumentation.opentelemetry.io/inject-go: 'true'
instrumentation.opentelemetry.io/otel-go-auto-target-exe: '/path/to/executable'
```

Habilitar Instrumentação Automática

Adicione anotações ao seu deployment para ativar a instrumentação automática:

- .NET: `instrumentation.opentelemetry.io/inject-dotnet: "true"`
- Java: `instrumentation.opentelemetry.io/inject-java: "true"`
- Node.js: `instrumentation.opentelemetry.io/inject-nodejs: "true"`
- Python: `instrumentation.opentelemetry.io/inject-python: "true"`
- Go: `instrumentation.opentelemetry.io/inject-go: "true"`

Revision #11
Created 2024-08-16 01:34:58 UTC by Wagner B. Simonato
Updated 2024-09-12 02:53:28 UTC by Wagner B. Simonato