0. OpenTelemetry Concepts

사전 지식

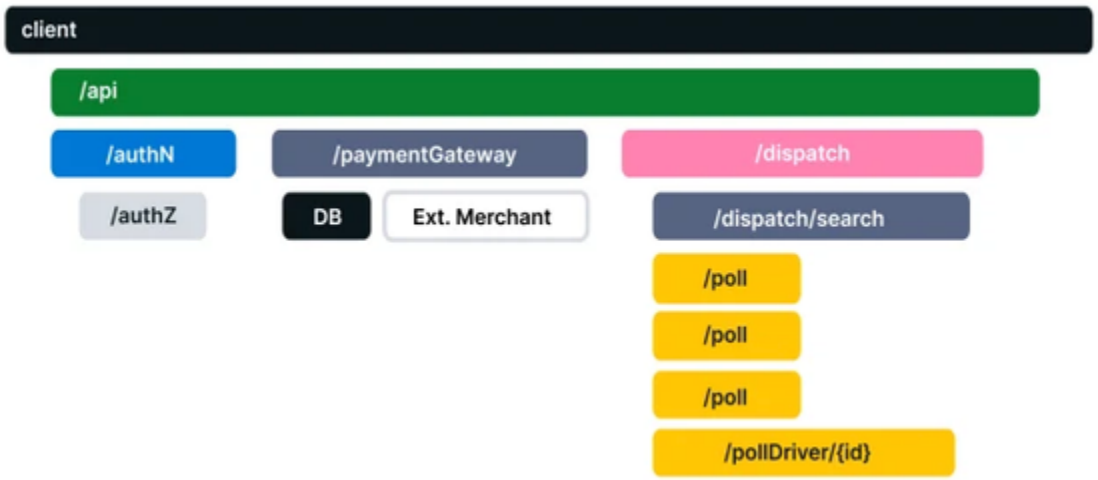
용어 정리집

- Observability(가시성): 시스템의 내부 동작을 실제로 보지 않고도 외부에서 시스템을 이해할 수 있는 속성
- Reliability(안정성): 서비스가 사용자가 의도한 대로 동작하는지의 여부
  예를 들어, A 상품을 장바구니에 담으려고 클릭했지만 계속해서 B 상품이 추가된다면, 해당 서비스는 안정성이 떨어진다.
- Metrics: 특정 주기의 시간 동안 인프라 혹은 애플리케이션의 수치 데이터를 취합한 것.
  예를 들어, 시스템 오류율, CPU utilization, Request rate 등이 있다.
- SLI(Service Level Indicator): 서비스의 행위를 나타낸다.
  이상적으로 SLI는 웹 페이지의 로딩 시간 등 사용자의 시선에서 측정된다.
- SLO(Service Level Objective): 외부에 제공할 안정성을 수치로 나타낸 것. 비즈니스 핵심 가치에 하나 이상의 SLI를 부여함으로써 측정된다.
- Log: 서비스 또는 기타 컴포넌트에 의해 발생하는 특정 시간에 생긴 메시지.
  Trace와는 달리 특정 user request 또는 transaction에 필수로 연관되어 있지 않다.
  Log는 문맥적인 정보들을 포함하지 않기에 log 만으로는 코드의 실행 흐름을 파악하기 어렵다.
- Span: 작업의 단위를 의미한다. 특정 요청이 만들어내는 특정 연산 과정을 추적하며, 시간에 따라 어떤 부분에서 어떤 일이 발생했는지를 파악할 수 있도록 해준다.
- Distributed Traces(Traces): 애플리케이션 또는 end-user가 만들어낸 request를 처리하기 위해 분산 시스템에 관여된 모든 시스템의 기록을 가진다.

  애플리케이션 또는 시스템의 health 및 로컬 환경에서는 재현하기 어려운 문제에 대한 디버깅을 매우 쉽게 할 수 있게 해준다.

  Trace는 하나 이상의 span으로 구성되며, 첫 번째 span을 Root Span이라 한다. 각 Root Span은 하나의 request를 처리하기 위한 시스템의 첫 부분부터 끝까지를 나타낸다.

  아래와 같이 trace를 가시화해 observability를 확보할 수 있다.

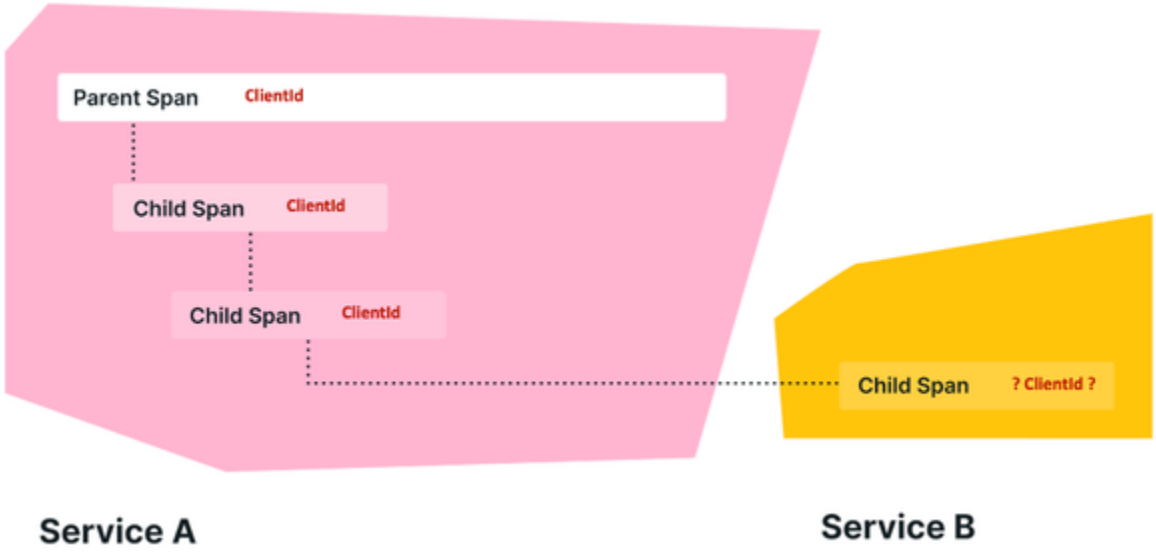


- OpenTelemetry는 개발자가 원하는 부분의 trace, metric, 그리고 log를 찍어낼 수 있게 해주는 도구이다.
  그리고 이런 데이터를 Prometheus 등의 Observability back-end 로 전달해준다.
- OpenTelemetry가 지원하는 기능들은 아래와 같다
  - Traces: 요청이 어떻게 처리되는지의 큰 그림을 보여준다.
    - `Tracer Provider` : `Tracer` 를 위한 factory로, 애플리케이션의 생명 주기와 동일한 생명 주기를 가진다.
    - `Tracer` : 특정 작업에 대한 정보를 담은 span을 생성한다.
    - `Trace Exporter` : Trace를 consumer로 보낸다.
    - Context Propagation: 분산 tracing을 위한 핵심 개념으로, 여러 개의 span들이 모여 하나의 trace를 생성한다.
      - Context: 서로 다른 span을 연관시키는 정보
      - Propagation: Context를 서로 다른 서비스 또는 프로세스로 전달하는 메커니즘
    - Span: 작업 또는 연산의 단위를 나타낸다.
  - Metrics: 런타임에 측정되는 서비스의 특정 수치로, OpenTelemetry는 아래 3개의 metric 수단을 제공한다.
    - counter: 시간에 따라 증가하는 수치로, 예를 들어 자동차의 누적 주행 거리처럼 항상 증가한다.
    - measure: 시간에 따라 취합되는 수치
    - observer: 특정 시각의 값들을 측정한 것
  - Logs
  - Baggage: Span 사이에 전달되는 문맥적 정보를 의미한다.
    Trace 내 여러 span들이 공유하는 key-value store이라고 생각하면 된다.

    예를 들어, 특정 서비스에서만 사용할 수 있는 `CustomerId` 라는 속성이 있고, 이 속성을 trace 내의 모든 span이 가지도록 하고 싶다고 하자. 이

럴 경우에 Baggage를 사용하면 된다.

OpenTelemetry는 Baggage를 전달하기 위해 context propagation을 활용한다.



- Baggage는 trace들의 정보를 전파시키기 위한 종합적인 방법을 제공하며, 3rd-party에 노출시켜도 무방한 민감하지 않은 데이터를 위해 사용되어야 한다. 이는 baggage가 HTTP header를 활용하기 때문이다.
- Baggage는 span 속성과 다르며, span 속성의 부분집합도 아니다. Baggage에 무언가를 추가하면 자동으로 child system의 span에까지 전달되지 않는다.

OpenTelemetry Collector

- 공식 문서

- OpenTelemetry Collector는 수집된 데이터가 데이터베이스에 들어가기 전 거치는 컴포넌트이다.
- 아래 세 가지 컴포넌트들로 구성된다.
  - Receiver: OpenTelemetry가 수집한 데이터가 collector에 어떻게 들어올지를 결정하는 컴포넌트로, push, pull 방식을 지원한다. 예를 들어, fluentbit를 통해 데이터를 받거나 jaeger를 통해, 또는 kafka 등을 통해 받도록 할 수도 있다.
  - Processor: 데이터가 collector에 수집되고 데이터베이스에 보내지기 전, 해당 데이터를 가공하기 위한 컴포넌트이다.
  - Exporter: Collector가 수집한 데이터를 하나 이상의 데이터베이스 또는 backend로 어떻게 보낼지를 결정하는 컴포넌트로, push, pull 방식을 지원한다.