Engineer

OpenTelemetry

💖 Ein anbieterunabhängiger Weg zum Senden von Telemetrie

Was es ist

OpenTelemetry ist ein Open-Source-Standard mit passenden Bibliotheken und Werkzeugen, um aus laufender Software Traces, Metriken und Logs zu erzeugen und an ein beliebiges System zur Speicherung und Anzeige weiterzugeben. Die meisten Entwickler nennen es OTel. Es ist ein Projekt der Cloud Native Computing Foundation und entstand aus dem Zusammenschluss der beiden früheren Vorhaben OpenTracing und OpenCensus.

Besonders hilfreich ist zu wissen, was es auslässt. OpenTelemetry ist kein Observability-Backend: Es speichert keine Daten, zeichnet keine Dashboards und verschickt keine Alarme. Diesen Teil übernehmen Jaeger, Prometheus und kommerzielle Plattformen. Die Frage „OpenTelemetry oder Prometheus?“ vergleicht deshalb eine Leitung mit einem Ziel. Meist lautet die Antwort: beides. OpenTelemetry steht dort, wo die Daten entstehen; ein Backend nach Wahl des Teams dort, wo sie aufbewahrt werden.

Das Projekt besteht aus mehreren zusammenpassenden Teilen:

- einer **Spezifikation**, die API, SDK und Daten über alle Sprachen hinweg definiert;
- **OTLP**, dem Protokoll zur Übertragung von Telemetrie über gRPC oder HTTP, wobei die Daten mit Protocol Buffers kodiert werden;
- **semantischen Konventionen**, gemeinsamen Namen für gemeinsame Dinge, damit eine HTTP-Route oder eine Datenbankanweisung in Go, Java und Python gleich bezeichnet wird;
- einer **API und einem SDK pro Sprache**, mit Instrumentierungsbibliotheken und automatischer Instrumentierung für verbreitete Frameworks;
- dem **Collector**, einem separaten Prozess, der Telemetrie empfängt, verarbeitet und weiter exportiert.

Von den übertragenen Signalen benötigen Traces die ausführlichste Erklärung. Ein Trace folgt einer Anfrage durch jeden Dienst, den sie berührt. Jeder Schritt ist ein Span: ein Name, ein Startzeitpunkt, eine Dauer, eine Menge von Attributen und die Kennung des Spans, der ihn ausgelöst hat. Wenn die Anfrage zum nächsten Dienst wechselt, reist diese Kennung in einem Header mit. Standardmäßig ist es der von W3C Trace Context definierte Header:

```
traceparent: 00-a0892f3577b34da6a3ce929d0e0e4736-f03067aa0ba902b7-01
```

Die vier Felder sind die Formatversion, die allen Spans der Anfrage gemeinsame Trace-ID, die ID des aufrufenden Spans und eine Menge von Flags. Ein abschließendes `01` kennzeichnet einen Trace, der zum Sampling ausgewählt wurde. Metriken und Logs vervollständigen das Bild. Ein viertes Signal, Baggage, überträgt kleine Schlüssel-Wert-Informationen mit der Anfrage, damit auch ein drei Stationen entfernter Dienst sie sehen kann.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist OpenTelemetry?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Monitoring und Alarme melden sich, solange ein Fehler noch klein ist; Kennzahlen zeigen, wie oft das gelingt.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Das erste Observability-Dashboard aufgebaut — der erste Einblick des Unternehmens in die eigenen Systeme

Das erste Observability-Dashboard aufgebaut Kontakt aufnehmen →

Warum es wichtig ist

Speicher ist der sichtbare Kostenfaktor von Observability. Schwerer rückgängig zu machen ist die Investition in die Instrumentierung: Jemand muss in jedem Dienst und jeder Sprache die Aufrufe ergänzen und bei Codeänderungen aktuell halten. Als jeder Anbieter seinen eigenen Agenten und seine eigene API lieferte, bedeutete ein Anbieterwechsel, all das erneut zu erledigen. Mit OpenTelemetry ruft der Code eine API auf, während die Wahl des Backends in die Konfiguration wandert: entweder in einen Exporter im SDK oder in eine Route im Collector. Das Projekt formuliert den Grundsatz ausdrücklich: Die von einem System erzeugten Daten gehören dessen Betreiber.

Diese Datenhoheit ist nicht nur eine Vertragsfrage, sondern hat praktische Folgen. Der Collector sitzt zwischen Anwendungen und Backend. Dort lassen sich Attribute filtern, Stichproben auswählen, Attribute umbenennen oder entfernen, bevor etwas das System verlässt. Während einer Migration kann derselbe Datenstrom an zwei Backends gehen. Der Wechsel von einem Anbieter zu einem anderen wird dadurch zu einer Änderung der Collector-Konfiguration statt zu einer Änderung an hundert Diensten.

Das Datenvolumen ist der Kostenfaktor, den es einzuplanen gilt. Jede Anfrage kann Dutzende Spans erzeugen. Bei echtem Datenverkehr kann deren vollständige Speicherung teurer werden als der überwachte Dienst selbst. Der Standard-Sampler des SDK, `ParentBased(root=AlwaysOn)`, wählt jeden neuen Trace aus und folgt bei den übrigen der Entscheidung des Aufrufers. Ohne weitere Konfiguration bleibt daher alles erhalten. Wie viel aufbewahrt wird, sollte das Team bewusst entscheiden.

Der Reifegrad unterscheidet sich nach Signal und Sprache. OTLP ist für Traces, Metriken und Logs stabil, während sich die SDKs der einzelnen Sprachen unterschiedlich schnell entwickeln. Die Statusseite des Projekts zeigt ihren jeweiligen Stand. Lesen Sie sie, bevor Sie einen Dienst in einer weniger verbreiteten Sprache darauf festlegen.

Der Aufwand lohnt sich auch nicht immer. Eine einzelne Anwendung mit strukturierten Logs und einigen wenigen Metriken gewinnt wenig durch verteiltes Tracing. OpenTelemetry zahlt sich aus, wenn eine Anfrage Prozessgrenzen überschreitet, durch ein Frontend, eine API, eine Warteschlange und eine Datenbank läuft und die Antwort auf die Frage „Wo ist die Zeit geblieben?“ eine Besprechung erfordert.

Wie es in der Praxis aussieht

Auf einer Plattform wurde das Monitoring selbst gebaut statt eingekauft. OpenTelemetry übernimmt darin das Tracing. Das Tracing läuft über OTLP, sodass sich eine langsame Anfrage durch Frontend, API und Datenbank verfolgen lässt, statt ihren Weg zu erraten. Daneben läuft eine eigene Fehlerpipeline mit einem errmon-Dienst und einem Ingest-Dienst. Sie bereinigt die Nutzdaten vor der Speicherung, fasst Fehler zu wiederkehrenden Problemen zusammen und erkennt Spitzen sowie Regressionen. Eine Abklingzeit verhindert, dass eine einzige fehlerhafte Bereitstellung jemanden vierzigmal alarmiert. Die

Pipeline löst außerdem die Symbole minifizierter Frontend-Stacktraces auf und führt einen synthetischen Heartbeat aus, um nachzuweisen, dass sie selbst funktioniert.

Die Ergebnisse landen als Fehlerereignisse, Fehlergruppen, Posteingang, API-Latenz und Stack-Stichproben in der eigenen Datenbank der Plattform. Betreiber arbeiten mit 11 Ansichten dieser Daten, darunter einer für Service Level Objectives. Die Seite nennt den Preis offen: Der Bau kostete spürbar Zeit und schuf ein weiteres zu wartendes System. Ein eingekauftes Fehlerverfolgungswerkzeug wäre noch am selben Nachmittag gelaufen. Dafür bleiben sensible Daten auf der Plattform. Auch die Alarmregeln wurden für diese Plattform geschrieben, nicht für eine beliebige.

Wo es schiefgeht

Der Trace endet an einer Grenze. Kontext überquert HTTP-Aufrufe leicht, weil Instrumentierungsbibliotheken den Header auf beiden Seiten schreiben und lesen. Eine Nachrichtenwarteschlange, ein geplanter Job oder ein Proxy, der unbekannte Header entfernt, unterbricht die Kette, sofern niemand den Kontext von Hand weitergibt. Der Trace zerfällt in zwei Teile, und die langsame Hälfte scheint nichts mit der auslösenden Anfrage zu tun zu haben. Legen Sie den Kontext in den Nachrichtenmetadaten ab und prüfen Sie, dass jeder Dienst denselben Propagator verwendet.

Attribute geben Daten preis oder vervielfachen die Datenmenge. Ein Span-Attribut lässt sich leicht ergänzen. Gerade die am einfachsten verfügbaren Werte sind jedoch die falschen: E-Mail-Adressen, vollständige URLs mit Token in der Query-Zeichenfolge oder SQL mit noch enthaltenen Literalwerten. Bei Metriken droht eine hohe Kardinalität. Ein Attribut mit Millionen unterschiedlicher Werte, etwa einer Benutzer-ID, vervielfacht die Zahl der Zeitreihen und damit die Rechnung. Befolgen Sie die semantischen Konventionen, halten Sie Kennungen aus Metriken heraus und entfernen Sie sensible Werte im Collector als zweite Schutzlinie.

Sampling geschieht zufällig statt bewusst. Head Sampling entscheidet zu Beginn einer Anfrage, bevor jemand weiß, ob sie langsam sein oder scheitern wird. Wer 1 % der Traces behält, behält deshalb auch ungefähr 1 % der seltenen langsamen Anfragen. Tail Sampling entscheidet, nachdem es die Spans eines Traces zusammen gesehen hat, und kann jeden Fehler sowie jede langsame

Anfrage behalten. Die zuständige Komponente muss beim Warten allerdings all diese Daten vorhalten. Sie ist ein zustandsbehafteter Dienst, dessen eigene Kapazität geplant und überwacht werden muss.

Der Pipeline wird blind vertraut. OTLP bestätigt die Zustellung jeweils nur für eine Übertragungsstrecke. Eine Ende-zu-Ende-Zustellung liegt außerhalb seines Geltungsbereichs. Ein Collector mit zu wenig Speicher oder einem Rückstand in seiner Exportwarteschlange verwirft Telemetrie. Die sendenden Anwendungen laufen weiter, als wäre alles in Ordnung. Überwachen Sie die Metriken des Collectors selbst, betreiben Sie mehr als einen und behandeln Sie eine Lücke in den Traces als zu erklärenden Fehler statt als ruhige Stunde.

Weiterführende Lektüre

- Was ist OpenTelemetry? — die eigene Definition des Projekts, seine Komponenten und das, was es bewusst nicht ist.
- OTLP-Spezifikation — das Übertragungsprotokoll: Transportverfahren, Kodierung, Komprimierung und Bestätigungen.
- Trace Context — die W3C-Empfehlung hinter dem Header `traceparent`.
- Kontextweitergabe — wie ein Trace einer Anfrage über Prozessgrenzen hinweg folgt.
- Semantische Konventionen — die gemeinsamen Namen für Attribute, Metriken und Ressourcen.
- Sampling — Head und Tail Sampling und deren jeweilige Betriebskosten.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist OpenTelemetry?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/opentelemetry/