Engineer

OpenTelemetry

💖 特定ベンダーに依存せずテレメトリーを送る共通の仕組み

どのようなものか

OpenTelemetryは、稼働中のソフトウェアからトレース、メトリクス、ログを生成し、それを保存、表示するシステムへ渡すための、オープンソースの標準です。対応するライブラリーとツールもあります。多くのエンジニアはOTelと呼びます。OpenTracingとOpenCensusという二つの取り組みが合流して生まれた、Cloud Native Computing Foundationのプロジェクトです。

最も役立つのは、何を担当しないかを知ることです。OpenTelemetryは、オブザーバビリティのバックエンドではありません。データを保管せず、ダッシュボードを描かず、アラートも送りません。そこは、Jaeger、Prometheus、商用のプラットフォームが担います。「OpenTelemetryかPrometheusか」という問いは、管と行き先を比べています。通常の答えは両方で、データを生成する場所にOpenTelemetry、保管する場所にチームが選んだバックエンドを置きます。

プロジェクトは、組み合わせて使う複数の部分からなります。

- **仕様**は、すべての言語にまたがるAPI、SDK、データを定めます。
- **OTLP**は、gRPCかHTTPで、Protocol Buffersでエンコードしたテレメトリーを運ぶプロトコルです。
- **セマンティック規約**は、共通のものに共通の名前を付けます。HTTPルートやデータベースの文に、Go、Java、Pythonで同じラベルを使えます。
- **言語ごとのAPIとSDK**があり、主要なフレームワーク向けの計装ライブラリーと、自動計装も含みます。
- **Collector**は、テレメトリーを受け取り、処理し、次の宛先へ送る、別のプロセスです。

運ぶ信号の中で、最も説明が必要なのはトレースです。一つのリクエストを、関わるすべてのサービスにわたって追います。各段階がスパンで、名前、開始時刻、所要時間、属性の集合、それを発生させたスパンの識別情報を持ちます。次

のサービスへ移ると、その識別情報をヘッダーで運びます。既定では、W3C Trace Contextが定めるものです。

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

四つのフィールドは、形式のバージョン、リクエストの全スパンが共有するトレースID、呼び出し元のスパンID、フラグの集合です。最後が `01` なら、そのトレースがサンプリング対象であることを示します。メトリクスとログが全体像を補い、第四の信号Baggageは、キーと値からなる小さな情報をリクエストとともに運び、三ホップ先のサービスでも見られるようにします。

概念を判断に生かす

「OpenTelemetryとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 障害が小さいうちに知らせる監視とアラートを整え、安定した状態をどれだけ維持できているかを数値で示します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 社内初のオブザーバビリティダッシュボードを構築 — 社内システムの状態を初めて可視化

社内初のオブザーバビリティダッシュボードを構築 お問い合わせ →

なぜ重要なのか

オブザーバビリティでは、ストレージが目に見えるコストです。長く付きまとうのは計装です。全サービス、全言語に呼び出しを加え、コードの変化に合わせて保守する人が必要です。ベンダーごとにエージェントとAPIがあった時代は、ベンダーを変えると、すべてやり直しでした。OpenTelemetryでは、一つのAPIを呼び、バックエンド選びは設定へ移します。SDK内のエクスポーターか、Collectorのルートで決めます。プロジェクトは、システムが生成するデータは運用者に属するという原則を、明確に掲げています。

その所有権は、契約だけでなく、実務上も意味があります。Collectorは、アプリケーションとバックエンドの間にあります。外へ出す前に、データをフィルタリング、サンプリングし、属性名を変えたり属性を取り除いたりする場所になり、移行中は、同じ流れを二つのバックエンドへ送れます。ベンダーの変更は、百のサービスでなく、Collectorの設定を変える仕事になります。

計画すべきコストは量です。一つのリクエストで数十のスパンが生まれる場合があり、実際のトラフィックでは、すべて保管する料金が、監視対象のサービスの費用を超えることもあります。SDKの既定のサンプラー `ParentBased(root=AlwaysOn)` は、新しいトレースをすべて採用し、それ以外は呼び出し元の判断に従うため、初期設定ではすべて残ります。どれだけ残すかは、チームが意図して決めることです。

成熟度は、信号と、言語ごとに違います。トレース、メトリクス、ログのOTLPは安定していますが、各言語のSDKは、それぞれの速さで進みます。プロジェクトの実装状況のページに、各SDKの段階が示されています。あまり一般的でない言語のサービスに採用する前に読みます。

常に割に合うわけでもありません。構造化ログと少数のメトリクスがある単一のアプリケーションでは、分散トレーシングによる利点は小さいものです。リクエストがプロセスの境界を越え、フロントエンド、API、キュー、データベースを通り、「時間はどこに消えたか」に答えるのに会議が必要になると、OpenTelemetryの導入がコストに見合うようになります。

実際の活用例

あるプラットフォームでは、監視を買わずに構築し、そのトレーシング側をOpenTelemetryが担います。OTLPで、遅いリクエストを、フロントエンド、API、データベースにまたがって追います。推測で調べるのではありません。並行して、errmonサービスと取り込みサービスからなる、自前のエラーパイプラインも動いています。保存前にペイロードを無害化し、エラーを繰り返す問題にまとめ、急増とリグレッションを検知します。一度の悪いデプロイで担当者へ四十回も通知しないよう、通知の休止時間を設け、ミニファイされたフロントエンドのスタックトレースのシンボルを解決し、パイプライン自体が動いていることを証明する、テスト用に生成したハートビートも動かします。

結果は、エラーイベント、エラーグループ、受信箱、APIレイテンシー、スタックのサンプルとして、プラットフォーム自身のデータベースに入ります。運用者は、そのデータに対する11のビューを使い、一つはサービスレベル目標用です。ページは、代償も明記します。構築には実際に時間がかかり、保守対象も一つ増え、市販のエラー追跡ツールなら、その日の午後には動いていたはずです。その代わり、機微なデータはプラットフォーム内に残ります。アラートの規則も、一般的なものではなく、このプラットフォームに合わせて書きました。

問題が起きるところ

境界でトレースが切れます。 HTTP呼び出しでは、計装ライブラリーが両側でヘッダーを読み書きするため、コンテキストを簡単に引き継げます。メッセージキュー、定期ジョブ、未知のヘッダーを取り除くプロキシでは、手でコンテキストを運ばなければ、連鎖が切れます。トレースが二つになり、遅い側と、原因になったリクエストが、無関係に見えます。メッセージのメタデータにコンテキストを入れ、全サービスが同じプロパゲーターを使うかを確認します。

属性が漏れるか、爆発的に増えます。 スパンの属性は簡単に足せますが、最も入れやすい値が、間違ったものです。メールアドレス、クエリー文字列にトークンを含む完全なURL、リテラル値を残したSQLなどです。メトリクスでは、カーディナリティーが危険です。利用者IDのように、異なる値が何百万もある属性は、時系列の数と料金を膨らませます。セマンティック規約に従い、メトリクスに識別子を入れず、第二の防御として、Collectorで機微な内容を除きます。

偶然に任せてサンプリングします。 ヘッドサンプリングは、リクエストの開始時に、遅いか失敗するかが分かる前に決めます。トレースを1%残すと、まれな遅いものも約1%しか残りません。テールサンプリングは、トレースのスパンをまとめて見てから決め、すべてのエラーと、すべての遅いリクエストを残せます。ただし、その処理を行う構成要素は、待っている間、全データを保持しなければなりません。独自の容量計画と監視が必要な、状態を持つサービスです。

パイプラインを、無条件に信じます。 OTLPの到達確認は、一ホップずつで、エンドツーエンドの配信は範囲外です。Collectorがメモリー不足になったり、エクスポートのキューで遅れたりすると、テレメトリーを捨てます。それでも、送信側のアプリケーションは正常のように動き続けます。Collector自身のメトリクスを監視し、複数動かし、トレースの空白を、単に通信の少ない時間帯とは見なさず、原因を調べるべき不具合として扱います。

参考資料

- OpenTelemetryとは — プロジェクト自身の定義、構成要素、意図して対象にしないことです。
- OTLP仕様 — 転送、エンコード、圧縮、到達確認を定める通信プロトコルです。
- Trace Context — `traceparent` ヘッダーの基になるW3C勧告です。
- コンテキスト伝播 — プロセスの境界を越え、トレースがリクエストを追う仕組みです。
- セマンティック規約 — 属性、メトリクス、リソースの共通名です。

- サンプリング — ヘッドとテールのサンプリング、および各方式の運用コストです。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「OpenTelemetryとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く 連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/opentelemetry/