Engineer

オブザーバビリティ

🔭 システムが出すデータから、その動きを外から理解する能力

どのようなものか

オブザーバビリティは、チームが、稼働中のシステムの動きを、そのシステムが出すデータだけで外からどこまで理解できるかを表す性質です。中を開いたり、何が悪かったかを先に知ったりする必要はありません。OpenTelemetryプロジェクトの入門は、問いの形で説明します。観測できるシステムでは、内部の仕組みを知らずに質問でき、ソフトウェアを書いたときに誰も考えなかった問いも含まれます。答えられるべき問いを短く表すと、「なぜ、これが起きているのか」です。

つまり、オブザーバビリティはシステムの性質で、製品ではありません。ツールはデータを保存、表示できますが、先にアプリケーションからデータを出す必要があります。OpenTelemetryのドキュメントは明確です。観測できるシステムにするには、各構成要素を計装し、コードから信号を出さなければなりません。その上で、実用上の基準を置きます。既存の信号を使って、計装を先に変えずに問題を調査できるなら、適切に計装されたアプリケーションといえます。

このデータをテレメトリーと呼び、OpenTelemetryは信号として分類します。

- **トレース**は、一つのリクエストがシステムを通る経路の記録です。トレースはスパンからなり、各スパンは、名前、開始と終了の時刻、HTTPメソッドやルートなどの属性を持つ、一つの作業単位です。最初のスパンがルートスパンで、リクエストを始めから終わりまで覆います。
- **メトリクス**は、実行時に測定し、時間に沿って集約した値です。エラー率、単位時間当たりのリクエスト数、プロセッサー使用率などがあります。
- **ログ**は、時刻を付けたイベントの記録です。特定のリクエストと結び付かないものも多くあります。
- **Baggage**は、ほかの信号が使えるよう、リクエストとともに渡すコンテキスト情報です。

信号のページは、特定の種類のログであるイベントと、コードのレベルで資源使用を記録するプロファイルも挙げています。これらは、開発中、または提案段階とされています。

三つのデータの山を、チームが問いを投げられるものにするのは、相関付けです。OpenTelemetryは、サービス間でコンテキストを伝えます。あるサービスのスパンは、どのトレースに属するかを把握でき、実行中のトレースとスパンのIDを、各ログレコードに付けられます。各信号には、生成した対象を識別する属性であるリソースも付きます。サービス名、ホスト、コンテナーなどです。共有するキーによって、メトリクスから基になったリクエストへ、リクエストからそれが出したログ行へと、チームがたどれます。

オブザーバビリティと監視は、重なる部分がありますが、同じではありません。監視は、先に重要だと決めた状態を見張り、それが起きるとアラートを出します。オブザーバビリティは、アラートが出た後、誰も想定しなかった原因を調べるときに、チームが頼れるものです。

概念を判断に生かす

「オブザーバビリティとは？」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

現代のシステムは、予想しにくい形で故障します。リクエストは、応答するまでに、ゲートウェイ、複数のサービス、キュー、データベースを通る場合があり、どこか一つの不具合が、別の場所の遅さとして現れます。入門は、こうしたシステムに分散トレーシングが不可欠だとします。問題が非決定的だったり、複雑すぎて開発者のマシンで再現できなかったりすることが多いためです。再現できない故障では、発生中に記録したテレメトリーが、唯一の証拠です。

ダッシュボードやアラートは、先に想定した問いに答えます。それは有益で、チームが以前に経験した故障に対応できます。難しいのは、新しいインシデントです。エラー率が上がったかでなく、ある地域から、あるバージョンを通り、ある

決済事業者へ向かうリクエストが、なぜ失敗し始めたかが問題になります。答えるために新しいコードを書き、デプロイし、問題の再発を待つ必要があるなら、その一巡にかかる時間だけ、障害が続きます。

オブザーバビリティは、利用者の視点も保ちます。入門は、信頼性を、サービスが利用者の期待どおりに動くかと定義し、一例を挙げます。常時稼働する店でも、黒い靴をカートに入れる操作が、必ずしも黒い靴を追加しないなら、信頼できません。ホストのメトリクスでは、それは分かりません。リクエスト単位のテレメトリーなら分かり、サービスレベル指標は、そのデータを基に作ります。利用者の視点で測った値を、サービスレベル目標によって、業務上の必要性に結び付けます。

最後に、インシデントのコストが変わります。遅いリクエストを、遅らせた呼び出しそのものまで追えるチームは、推測でなく修正に時間を使えます。後の振り返りも、記憶でなく証拠を引用できます。

実際の活用例

架空のオンラインショップを考えます。14:00以降、決済処理のごく一部が、二秒を超えるようになりました。何も停止しておらず、アラートも出ず、この状況向けのダッシュボードもありません。信号を関連付けているチームなら、次のように調べます。以下の名前、ID、数値は、すべて説明のための架空のものです。

最初に見るのは、決済処理のレイテンシーを秒で測るヒストグラムです。OpenMetricsのテキスト形式では、バケットの `#` の後ろに、エグゼンプラーを付けられます。メトリクス外のデータへの参照で、多くは、そのバケットに入ったトレースのIDです。二つのバケットは、次のようになります。

```
checkout_bucket{le="1.0"} 95
checkout_bucket{le="2.5"} 97 # {trace_id="3f1c9a77e2b04d18a6c5e0b9d4127f60"} 2.3
```

次は、そのトレースIDをたどります。トレースは、2.3秒をどこに使ったか、スパンごとに示します。

```
POST /checkout                         2.30 s   service checkout
  SELECT cart                          0.04 s   service checkout
  POST /authorise                      2.21 s   service payments
    event: provider timeout, retrying
```

```
  INSERT order                      0.03 s   service checkout
```

三番目は、同じトレースIDを持つログ行を取り出します。OpenTelemetry自身のプロトコル以外で書くログの仕様は、JSONの最上位フィールド `trace_id` と `span_id` に、小文字の十六進数でIDを記録するよう求めます。

```
{
    "timestamp": 1793110051.402,
    "severity": "WARN",
    "body": "provider timeout after 2000 ms, retrying",
    "provider": "acquirer-b",
    "trace_id": "3f1c9a77e2b04d18a6c5e0b9d4127f60",
    "span_id": "a91c04e7d2f3b658"
}
```

三つの手順で、誰も想定しなかった問いに答えられました。二つの決済事業者の一方でタイムアウトが起き始め、そちらへ送られたすべての決済が、再試行を待っていました。同じ `provider` 属性で、最近のすべてのトレースをグループ化し、コードに手を付ける前に、その傾向を確認できます。

サービスが自分を識別しなければ、どれも成り立ちません。OpenTelemetry SDKのリソース属性 `service.name` は、既定で `unknown_service` になります。ドキュメントは、コードか環境変数で、明示的に設定することを勧めます。ほかのリソース属性も、同じ方法で追加できます。

```
env OTEL_SERVICE_NAME=checkout \
    OTEL_RESOURCE_ATTRIBUTES=deployment.environment.name=production \
    ./checkout
```

このサイトでは、三つのポートフォリオ項目が、オブザーバビリティを段階的に構築する様子を示します。一つは、ある企業の最初のオブザーバビリティダッシュボードです。遠隔のLinuxサーバーから、CPU、メモリー、ディスクなどのシステム指標をSSH経由で集め、後に第二版をGrafanaとPrometheusで構築しました。GrafanaとPrometheusで運用リスクを減らす取り組みでは、主要サービスを計装し、見栄えだけの数字でなく、意味のある指標を選びました。その後の自前のエラー監視とトレーシングは、OTLPによるOpenTelemetryトレーシングを加えました。遅いリクエストを、推測するのでなく、フロントエンド、API、データベースにまたがって追えるようにし、保存前にペイロードを無害化するエラーパイプラインも備えます。

問題が起きるところ

組み込まずに買おうとします。 役立つデータが何も流れ込まない新しいツールは、空の部屋です。オブザーバビリティは、コードやライブラリーの計装から生まれます。ライセンスだけでなく、スパン、属性、ログフィールドを追加する開発時間も予算に入れます。

三つの信号が、三つの島になります。 メトリクス、ログ、トレースが別々のツールにあり、共通のキーがなければ、チームは画面間で時刻をコピーすることになります。すべてのログレコードにトレースとスパンのIDを入れ、技術スタックが対応するなら、レイテンシーのヒストグラムにエグゼンプラーを付け、全信号に同じリソース属性を与えます。

すべてのサービスがunknown_serviceになります。 SDKの既定値で、全サービスが同じ名前では区別できません。各サービスに `OTEL_SERVICE_NAME` を設定します。リソースは、トレーサーとメトリクスのプロバイダーを作るときに付け、その後は変更できないため、起動時点で正しくする必要があります。

外側だけを計装します。 ゼロコード計装は良い出発点で、ライブラリーや実行環境など、アプリケーションの外側で起こることを記録します。しかし、インシデントで問うのは通常、どの顧客ティアか、どの事業者か、どの機能フラグの値か、といった業務のことです。そのためには、コードで属性を追加する必要があります。OpenTelemetryのドキュメントは、コードによる計装をゼロコードの補完と呼び、両者は同時に動かせます。

インシデント中に計装します。 答えるために新しいログ行とデプロイが必要だったなら、入門自身の基準では、適切に計装されていませんでした。すべてのインシデントレビューで「何を追加する必要があったか」を定例の問いとし、答えをそれぞれ後続の対応事項にします。

違う信号に、際限なく詳細を入れます。 利用者ID、完全なURL、リクエストボディーは、スパンやログレコードに置くもので、メトリクスのラベルには置きません。新しい値のたびに、時系列が一つ増えるからです。OpenMetrics標準は、エグゼンプラーでスパンのデータを運べないよう、ラベル名と値の合計を128文字に制限し、受信側がエグゼンプラーをすべて破棄することも認めます。頼る前に、エクスポーターから画面までの全経路で、リンクが保たれるかを確認します。

テレメトリーから情報が漏れます。 スパンやログは、リクエストの内容を記録し、住所、名前、トークンを含む場合があります。記録してよい属性を決め、保存前にそれ以外を除き、対象のデータベースと同じ注意を払ってテレメトリーを守ります。

参考資料

- オブザーバビリティ入門 — OpenTelemetryプロジェクトによる定義と、信頼性、SLI、SLO、スパン、分散トレースです。
- 信号 — トレース、メトリクス、ログ、Baggageと、まだ開発中の信号です。
- 計装 — コードによる計装とゼロコード計装、およびコンテキスト伝播による信号の結び付けです。
- リソース — リソース属性、既定の `unknown_service` 、設定用の環境変数です。
- 非OTLPログ形式のトレースコンテキスト — syslog、プレーンテキスト、JSONのログに、トレースとスパンのIDを記録するための、安定した仕様です。
- OpenMetrics 1.0 — エグゼンプラーと、カーディナリティーに関する助言を含む、メトリクスの出力形式の標準です。

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

「オブザーバビリティとは？」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/observability/