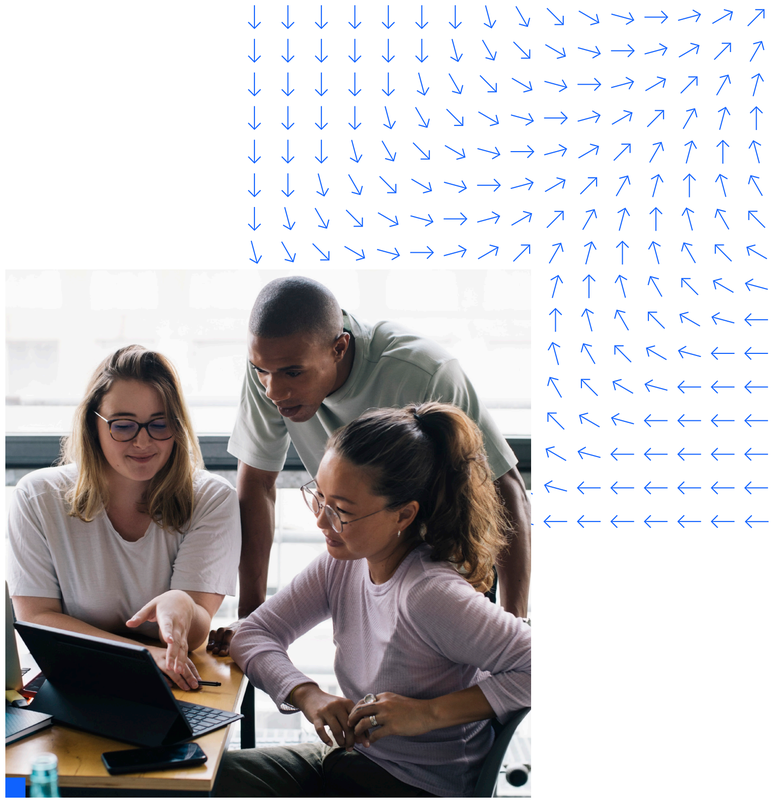
IBM Instana

開発者のための可観測性

可観測性とは

目次



01

概要

デベロッパーは、さまざまな言語やプラットフォームで実行される多数の異種サービスからなるソフトウェアのトラブルシューティングをどのように行うかという、日増しに難しくなる課題に直面しています。重要な変更に気づき、ブラックボックス・サービスを調べ、エラーの本当の原因を識別するにはどうすればよいでしょうか?

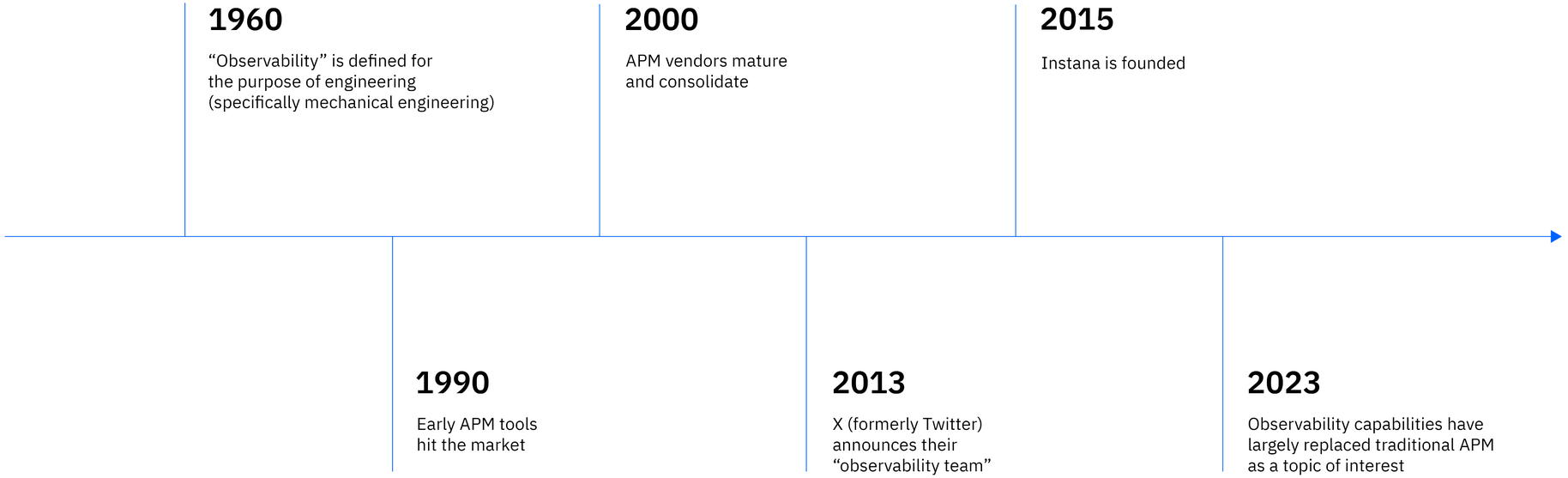
少し前までは、プログラムのデバッグとは、通常、エラー・ログの参照を意味していました。このアプローチは、少数のインスタンスで単純なローカル・プログラムを実行する小規模なチームには適していました。

しかし、ITの状況は常に変化しています。ソフトウェア・アーキテクチャーのパラダイムは、モノリスからマイクロサービスに進化しました。DevOpsの出現により、開発者の責任は変化しました。これは、開発者による配信後のプログラムに対する責任の持ち方の変化を表しています。

可観測性の起源
可観測性という用語は、RE Kalmanが1960年に発表した論文「On the generaltheory of control systems」において、エンジニアリングについて初めて定義されました。[1] 機械工学に関連する可観測性という用語は、出力を測定することによってシステムの内部状態を理解する能力と定義されました。

それから50年経ってからも、開発者やソフトウェアの専門家は、面倒な計測ツールを使用してシステムを監視していました。2013年までさかのぼると、分散システムへの移行はすでに進行中でした。その時、X(旧Twitter)は、Twitterの何百ものサービス全体でテレメトリー・データの収集を一元化および標準化するために、新しい「可観測性チーム」を創設すると発表しました。[2] 可観測性という用語が主流になります。

可観測性の現在
10年以上が経過した現在も、ITチームはさらに細分化されたサービスと、より部門横断的な職務責任に直面し続けています。マイクロサービスはサービスレスに道を譲りつつあります。DevOpsはサイト信頼性エンジニアリング（SRE）につながります。可観測性はこれまでと同様に重要であり、課題の範囲は飛躍的に拡大しています。



02

可観測性とは

科学的な定義はさておき、ソフトウェア開発における可観測性とは、アプリケーションまたはシステムの内部で何が起こっているかを観察し、見ているものを特定することです。要するに、可観測性とは可視性と理解です。

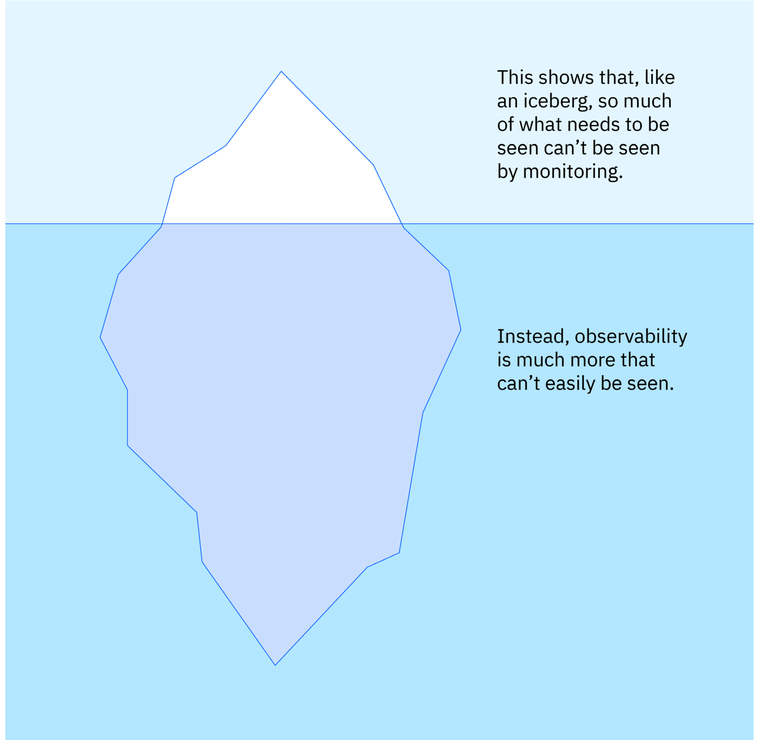
最新の分散ソフトウェア・システムには可観測性ツールが絶対に必要です。開発者にとって、アプリケーションを多数の小さな部分に分割したときに生じる複雑さを常に監視する方法は他にありません。幸いなことに、この複雑なアーキテクチャーこそが、可観測性ツールの存在を可能にするものでもあります。

通常、サービスにはユニバーサル・コントロール・プレーンがあり、HTTP、RPCなどの共通言語に依存して相互に通信するため、ソフトウェア・システムによって統一された可観測性が可能になります。さらに、IT運用を円滑に進めるために古いログに頼るなんて想像もつきません。

可観測性と監視の違い
自分がどこに行ったかを知らずして、これからどこに行くのかを正確に把握することはできません。アプリケーション・パフォーマンス監視（APM）は、最新の可観測性と共存する重要なケイパビリティー一式です。

従来のモニタリングは、既知のコンポーネントの事前定義された側面を測定することに重点を置いています。これは1980年代には大きなアイデアでしたが、今日の分散アーキテクチャーには、常に（場合によっては毎秒）変化するアプリケーション・コンポーネントが含まれています。従来のAPMツールでは、この急速に変化する動的な環境に追いつくことができません。

明らかに、重要な指標が欠落していることに気づく最悪のタイミングは、本番運用停止のトリアージを行っている最中です。ここで、可観測性が重要になります。可観測性ツールは、何を測定するかを事前に決定するのではなく、サービス間で、さらにはサービス内で起こっていることをすべて確認します。この機能を使用すると、未知のものに直面したときなど、予想すらできなかった質問に答えることができます。



モニタリングは役に立つ可能性があります。
時には。モニタリング・ツールやメトリクス・ベースのアラートは、次のような基本的な質問に答えるのに最適です：

- 私のサービスはオンラインで、顧客が利用できますか?
- 要求の何%にエラーがありますか?
- 要求の平均待ち時間はどれくらいですか?
- Redisはどのくらいのメモリーを使用していますか?
- 線形スケーリング・アクションを作成する必要がありますか?

ただし、正常なアプリケーションを維持する過程で、多くの場合、次のようなより深い質問に答えられる必要があります。

- ユーザーはビジネス・クリティカルなパスでエラーを経験していますか?
- この特定の顧客サブセットのエラー率が異常に高いのはなぜですか?
- 特定のエンドポイントの遅延を引き起こしているシステムのボトルネックはどこにありますか?
- このエラーの原因となる可能性のあるシステム内の変更は他にありますか?
- このエラーを修正するにはどのサービスを更新する必要がありますか?

03

完全なエンドツーエンド可観測性を実現する方法

これらのより深い質問に答えるには、クライアントから永続性層まで、ユーザー要求のライフサイクル全体を監視するシステムを構築する必要があります。アプリケーション開発者にとって、これは、クライアントとバックエンドを計測して有用なテレメトリー・データを出力することを意味します。

このテレメトリーを取得したら、データを有用で実用的な洞察に変えるために、それを収集、処理、保存、分析する必要があります。通常、エージェントまたはコレクターはテレメトリー信号を収集し、保存するためにデータベースに転送します。時系列データベースと列指向データベースは、メトリクス・データの効率的な保存とクエリにおいて非常に重要な役割を果たします。

最後に、分析と洞察のために、データベースにクエリを実行するためのUIが必要です。これにより、データをグラフやダッシュボードとして表示できるようになります。アラートは、自動化された根本原因分析やスマート・アラートなど、問題が発生した場合に開発者にすぐに通知するように構成する必要があります。

テレメトリー信号
これらは、アプリケーション・サービスから情報を収集するために使用されるデータ形式です。

- **トレース:** 優れた、要求範囲付きのタイムライン
- **イベント:** デプロイなどの外部変更を追跡する構造化されたログ
- **メトリクス:** イベントやシステムに関する集計可能な数値
- **プロファイル:** ランタイム・レベルのメトリクス
- **ログ:** イベントのタイム・スタンプ記録
- **例外:** エラーを追跡するための構造化ログ

1つの信号や一連の信号が可観測性を構成するものではないという点が重要です。ただし、分散トレーシングはおそらく可観測性において最も重要な信号です。

分散トレースとは何か?
ほとんどの開発者はスタック・トレースに精通しています。スタック・トレースはエラー・ログに毎日表示されます。ただし、分散トレースは、単一サービスの関数呼び出しを分散アプリケーションのネットワーク呼び出しに置き換えたときに得られるものです。

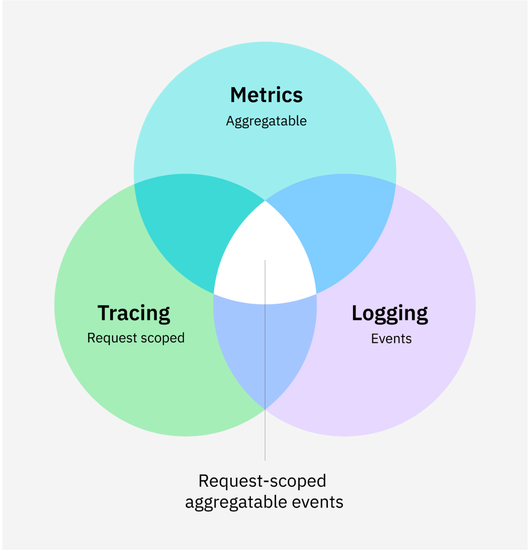
分散トレースは「スパン」(タイムスパンの派生)で構成されています。各スパンには、開始時間、終了時間、任意の数のカスタム属性、およびその親スパンへの参照が含まれます。サービスは要求を処理するたびに、コール側のサービスからのスパンを参照するスパンを作成します。これは、W3C Trace Context仕様に従って要求ヘッダーを通じて受信されます。

カスタム・スパンを作成して、特定のサービス内のトレースの細分度をさらに高めることもできます。たとえば、データベースから結果を受け取った後に多くの処理を実行する関数がある場合、その関数をカスタム・スパンで囲むと便利な場合があります。

しかし、トレースは単なるタイムラインではありません。また、それは単に待ち時間に関係しているだけではありません。個々の要求のパスをトレースすることで、ログ、メトリクス、その他のシグナルを状況に合わせて、ユーザー中心の(要求を範囲とした)観点から質問に答えることができます。トレースには、次のことに役立つデータ構造が含まれています。

- システム全体の要求フローを理解します。
- システム・トポロジーを即座に視覚化します。
- 豊富なトレース・メタデータからメトリクスを導き出します。
- ログを特定のスパンに添付することで、ログにコンテキストを付加します。

すべてをまとめる
これらのシグナルは単独でも役立つ場合がありますが、このデータすべてを関連付けて有意義な方法で検索できる場合、最も価値のある答えが得られます。著名なソフトウェア・エンジニアであるPeter Bourgonは、情報の重要なスイート・スポットを上手く整理しました。そこで、メトリクス、トレース、ログがオーバーラップします。[3] その結果、要求を範囲とした、集計可能なイベントが得られました。



04

可観測性標準とオープンソース

2019年に、OpenTracingプロジェクトとOpenCensusプロジェクトが統合され、OpenTelemetryになりました。Cloud Native Computing Foundation（CNCF）の下にあるこのオープンソース・プロジェクトは、オープンソース・ツールと、IBM® Instana®を含むベンダー可観測性プラットフォームの両方で広く採用されているテレメトリーに関するオープン標準を急速に確立しています。

OpenTelemetryプロジェクトは、仕様と標準、計測ツール、パイプライン・ツールという3つの領域でテレメトリー・データの収集を簡素化します。

仕様には次のものが含まれます。

- OTLP - メトリクス、トレース、ログを効率的に表現するためのバイナリー形式。
- 親のトレースIDをダウンストリーム・サービスに伝播するためのW3Cトレース・コンテキスト仕様。
- アプリケーションまたはライブラリー・コードから呼び出すことができる言語API。

計測ツールには次のものが含まれます。

- – 言語APIを特定の実装に接続する言語SDK。これらは、
- コミュニティーまたはベンダーによって提供され、テレメトリーの処理とエクスポートに使用されるいわゆる標準的なものまたはプラグインのいずれかです。
- – フレームワークまたはライブラリー内で関連イベントが発生したときに、言語APIを自動的に呼び出す自動計測ライブラリー。

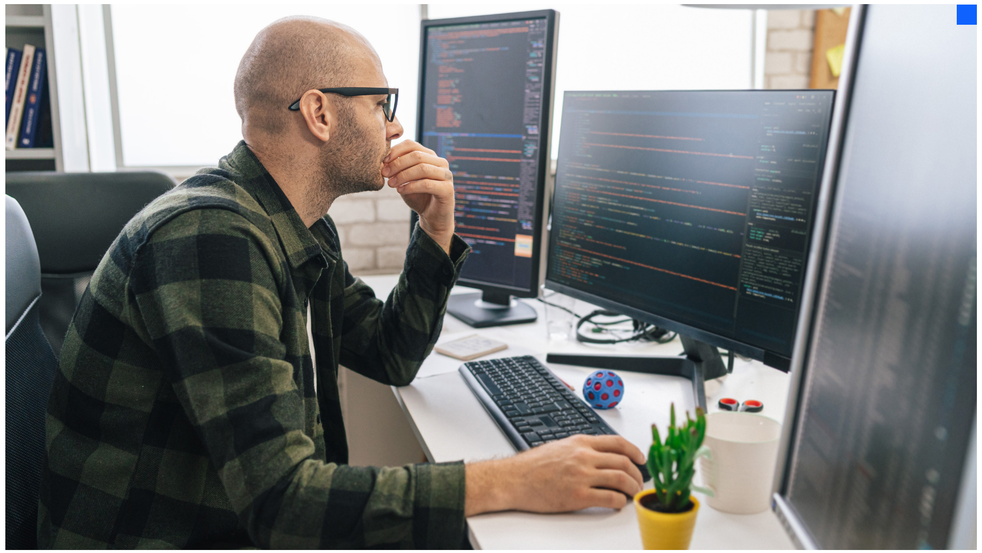
そして最後に、処理と伝送のためのOpenTelemetry Collectorがあります。この拡張的なツールは、ノード上でエージェントとして機能し、すべてのテレメトリー・データを収集して転送できます。プラグインとして利用できるレシーバー、プロセッサー、エクスポーターの大規模なエコシステムもあります。

オープンソース可観測性バックエンド
OpenTelemetryプロジェクトは、モニタリング・データを保存および分析するためのバックエンドを提供しませんが、多くのオープンソース・ツールはOTLPシグナルと互換性があります。

OpenTelemetry Demoプロジェクトは、マイクロサービス・アプリケーションのサンプルを提供します。これは、DockerまたはKubernetesで実行できます。これには、メトリクスとログ用のPrometheusとトレース用のJaegerの3つのオープンソース・テレメトリー・データベースが含まれています。Prometheusからメトリクスの一部を視覚化するためのGrafanaも含まれています。

05

SLO方法論



これまで、サービス・レベル目標（SLO）とサービス・レベル・インジケーター（SLI）について多くのことをお聞きになられたと思います。

SLO方法論は、可観測性と密接に関連するソフトウェアのパフォーマンスと健全性について考える新しい方法を示しています。概念としての可観測性は、機械の（完璧な操作ではなく）最適な操作に焦点を当てた制御理論から来ています。SLO方法論は、ITチームが完璧を目指すと、現実的な目標を設定した場合よりも悪い結果が得られることが多いという知見に基づいています。

すべてはサービス・レベル・インジケーターから始まります。SLIは、パーセンテージに変換できる指標または統計です。実行中のソフトウェアのコンテキストでは、SLIは正常に処理された要求の割合または待ち時間が許容できる要求の割合のようなものです。

SLOはSLIの目標値です。SLOは多くの場合、「9の個数」で表されます。。たとえば、9が4つであれば、SLIが99.99%の確率で目標を満たすことを示します。非常に重要なのは、SLOが「100%」であってはいけないということです。これは、ほぼすべての状況で非現実的です。計画的および計画外の停止に備えて、エラー予算を確保しておくことをお勧めします。

実際、Googleのあるチームは、システム全体の信頼性を向上できることを発見しました。他のチームが100%の信頼性を期待するのを防ぐために、人為的にサービスのダウンタイムを発生させることで、システム全体の信頼性を向上できることを発見しました。[3] 貴社でこれを同じことを行うのは推奨できません。

↓

サービス・レベル・アグリーメント（SLA）はSLOと似ているように見えるかもしれませんが、使用目的はまったく異なります。SLAはビジネス契約の一部であり、目標に違反した場合に何が起こるか（通常はなにかしらの金銭的罰則）を規定します。

SLAは、違反しない限り開発者にとって興味深いものではありません。SLOは、アプリケーションの全体的な信頼性を理解し、新機能に投資しても安全かどうかを判断するのに役立つため、開発者にとって非常に便利です。

06

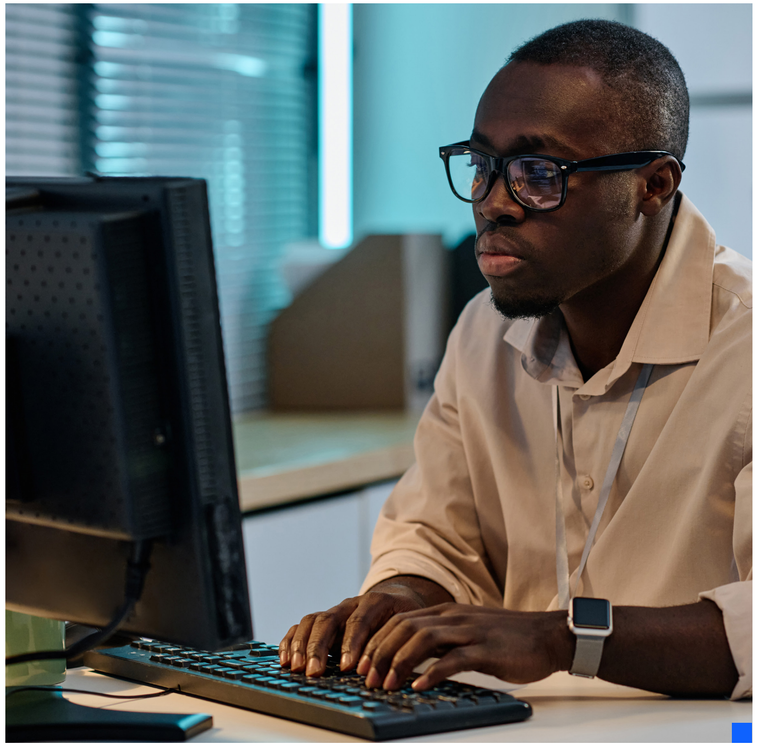
オープンソースを超えて

前章のオープンソース・ツールを使用すると、サービスから大量のテレメトリー・データを収集し、それを有用な方法でまとめ始めることができます。しかし、それだけでは、本当に可観測性を達成したとは言い難いのです。

可観測性には、未知の事態についての質問に答えられることが求められます。テレメトリーは、サービスの変化に合わせて適応し、変化できるものでなければなりません。今のところ、オープンソース・ツールでそのレベルの自動化を提供できるものはありません。しかし、IBM Instanaのようなソリューションならそれが可能です。

可観測性ソリューションは適応性があり、動的です。
アプリケーションは、さまざまな言語やテクノロジーを使用する数十、数百、さらには数千のサービスで構成されている可能性があります。アプリケーションのサーフェス・エリア全体にわたって一貫した手動計測を維持することは事実上不可能です。

動的なサービス検出や自動計測などの自動化を利用することで、インシデントが発生する前にシステムをより完全に理解できます。それが重要なのは、本番運用停止のトリアージを行っている最中に、パズルの重要なピースが欠けていることに気づいても遅いからです。



自動相関の利点

エンタープライズ可観測性ソリューションの主な利点は、マシン、インフラストラクチャー、アプリケーションまたはサービスのメトリクスとトレースを相互に関連付けることができることです。分散トレースは要求のフローを把握し、メトリクスは必要なパフォーマンス・ポイントを提供します。しかし、これらを手作業で関連付けるのは非常に面倒です。さまざまなサービスに対して複数のダッシュボードを使用することが嫌がられる主な理由は、データを照合して問題の全体的なコンテキストを収集することがほぼ不可能であるためです。

自動相関の最大の利点は、洞察をすぐに得られることです。問題やインシデントを調べる場合、ソリューションはすべての検出作業を行います。これにより、コンテキスト上の証拠として重要な情報が提供され、関心のある領域に直接誘導されます。

07

IBM Instanaはあなたに適しているか?

IBM Instanaは、自動化されたアプリケーション・パフォーマンス・モニタリング機能を含む、エンタープライズ可観測性 プラットフォームです。複雑な最新のクラウド・ネイティブ・アプリケーションを、オンプレミス、パブリッククラウド、プライベートクラウド、モバイル・デバイス、IBM Z® 環境など、場所を問わず運用する企業向けに設計されています。

IBM InstanaはAIを利用してハイブリッド・アプリケーション内の深いコンテキスト依存関係を検出することで、最新のハイブリッド・アプリケーションの制御を支援します。また、IBM Instanaは開発パイプラインの可視性も提供するため、DevOpsの自動化を循環させることができます。

これらの機能は、クライアントがアプリケーション・パフォーマンスを最適化し、イノベーションを可能にし、リスクを軽減するために必要な実践的なフィードバックを提供します。これらのフィーチャーにより、サービスおよびビジネス・レベルの目標を達成しながら、DevOpsチームの効率性を向上させ、ソフトウェア配信パイプラインに付加価値を与えることができます。

IBM Instanaのパワーをご自身でご確認ください。今すぐサインアップして、製品のフルバージョンの14日間の無料トライアルを体験してください。クレジット・カードは必要ありません。

IBM Instana無料トライアル →

Explore IBM Instana →

1. R. E. Kalman, “On the general theory of control systems,” 1960年8月

2. Twitter blog, “Observability at Twitter,” 2013年9月9日, 2023年7月アクセス

3. GoogleオンラインSREブック, “Service Level Objectives,”2023年7月アクセス

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21
IBM Corporation
New Orchard Road
Armonk, NY 10504

2023年8月アメリカ合衆国で制作