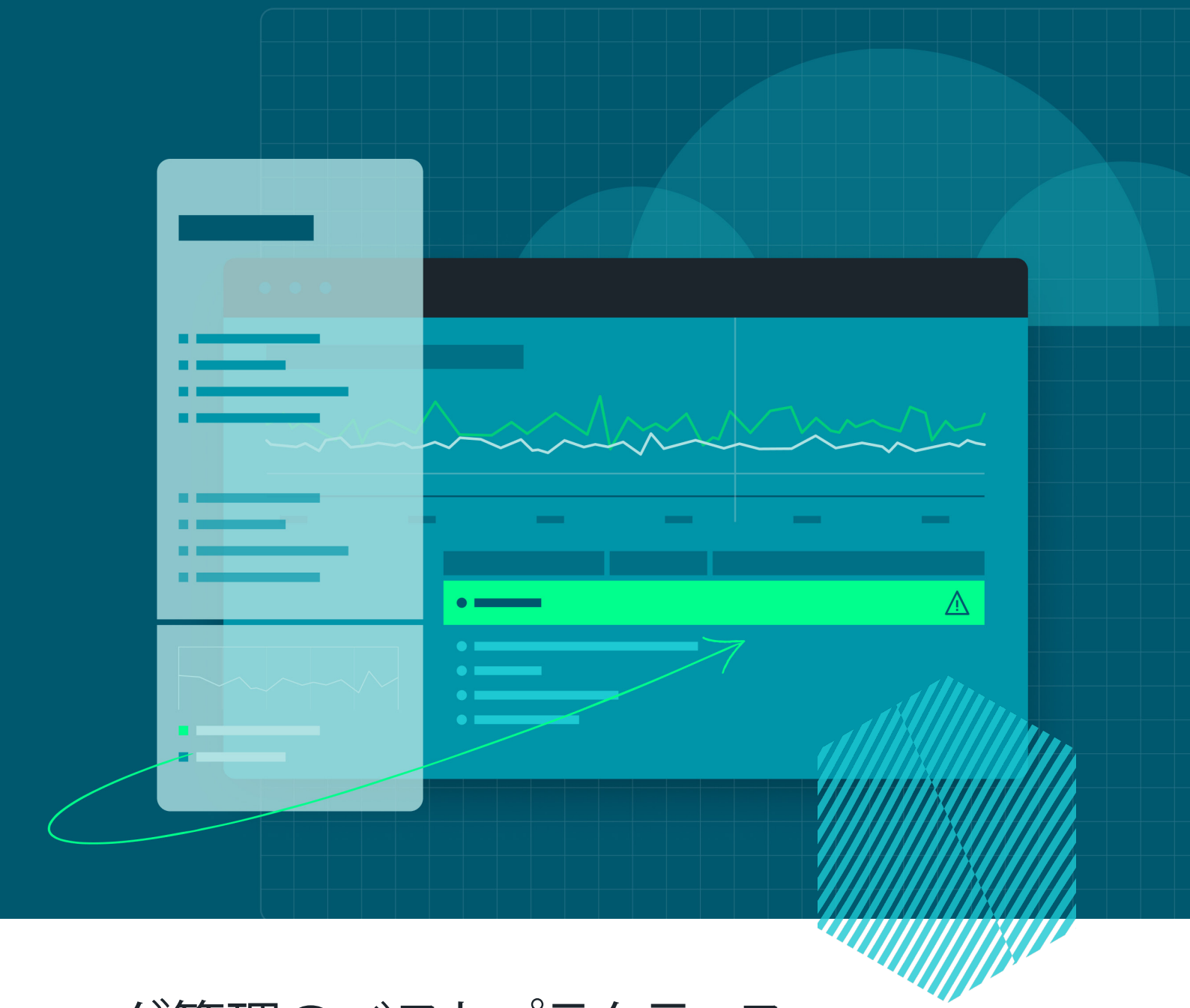
new relic



ログ管理のベストプラクティス

フルスタックオブザーバビリティに向けた効果的なログ管理への移行

ホワイトペーパー / 2022年8月

コンテンツ

概要

ログ管理は進化しています。何かが起きたときにだけ、アプリケーションやインフラのログを生でダンプするのはもはや昔のことです。今ではログは、組織の業務、ビジネスインテリジェンス、マーケティングに必要不可欠な役割を担っています。ログはオブザーバビリティの原動力です。構造化されたログは、組織が、システム全体の稼働状態を迅速かつ容易に把握し、先手を打って問題を阻止することさえ可能にします。

ログを使用して効果的なオブザーバビリティを得るには、上手くフォーマットされていない大量のログをデータベースやファイルに単に落とし込む方法よりも、さらなる熟慮と注意深さが求められます。どのようにログのプラクティスを変えれば、詳細なログを利用して、アプリケーションやインフラのインシデントをリアルタイムで関連づける能力を高め、異なるアプリケーションおよびツールを切り替える手間を省けるようにできるようでしょうか？どのようにすれば、エンドツーエンドのオブザーバビリティをより良く実現できるでしょうか?組織がフルスタックオブザーバビリティの実現により近づき、ビジネスに役立てるには、どうすればよいでしょうか？

ログの記録方法を変更することで、ログがフルスタックの観測性を高めるようにすることは簡単です。このホワイトペーパーでは、モダンな組織のためのログのベストプラクティスについて解説します。

従来のログ管理

従来、ログは他のシステムとは別に保管されたデータサイロで行われていました。かつてオブザーバビリティは、アプリケーションパフォーマンスモニタリング（APM）とインフラモニタリングに依存していました。モニタリングは重要ですが、アプリケーションやインフラデバイスのログの中身を全体的に把握することはできません。ばらばらで、サイロ化されたモニタリング/ロギングツールはアプリケーション中心主義であり、スタックのごく一部にしか焦点を当てておらず、何が起きているのか、なぜそれが起きているのかを説明するための完全な情報を提供できません。市場化を加速し、顧客の行動を深く洞察し、インシデントのレスポンスタイムを短縮するために必要な情報を、チームが入手することが非常に重要です。

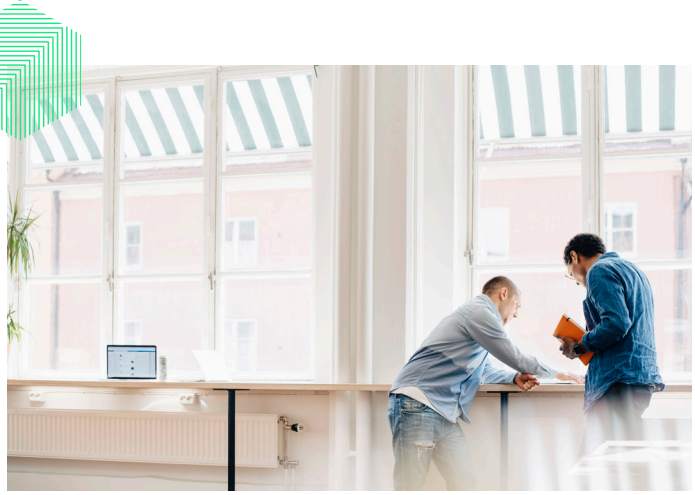
多くの組織は、ログからきめ細かな詳細を取得することを選択しておらず、問題の根本的な原因の特定に苦労するか、あるいは別のツールを使用してログの詳細をエラーとトレースにマッピングしようとしているかのどちらかです。詳細なログを別々のサイロで管理すると、全体像が見えなくなり、製品を市場に展開するためのコストと時間がかさみ、顧客体験の可視性が低下し、平均復旧時間（MTTR）が増大します。

フルスタックオブザーバビリティ

顧客体験に影響を与える可能性のある技術スタック内の全てを表示する機能は、フルスタックオブザーバビリティまたはエンドツーエンドのオブザーバビリティと呼ばれています。それは、全てのテレメトリデータ（メトリクス、イベント、ログ、トレース）の完全なビューに基づいています。

フルスタックオブザーバビリティは、理想的には単一の統合されたソリューションによって複雑なアプリケーションやシステムの動作を完全に視覚化し、インシデントのトラブルシューティング、 MTTRの短縮、顧客体験の理解を可能にします。

フルスタックオブザーバビリティにより、エンジニアや開発者がデータをサンプリングしたり、技術スタックの可視性を低下させたり、サイロ化されたデータをつなぎ合わせるのに時間を無駄にしたりする必要がなくなります。代わりにエンジニアや開発者は、彼らが好む、より優先度が高くビジネスに影響を与えるクリエイティブなコードディング作業に集中できます。



フルスタックオブザーバビリティのためのログ記録

スタック全体のログを記録する作業には労力がかかります。エンジニアや開発者は、何を記録するのか、どの程度の詳細を含めるのか、データ量が大き過ぎると経費がかさむのではないか、など疑問をお持ちのことでしょう。異なるプラットフォームでのログ管理を一元化するために、多くの企業が高いコストをかけており、最終的には、パフォーマンスとコスト次第で送信するログデータの量を制限せざるを得なくなっています。こうしたデータサンプリングがビジネスにもたらす可視性と価値は限定的です。以上のことを念頭に、フルスタックオブザーバビリティを実現するためのログ管理のベストプラクティスをいくつか見ていきます。

必要なものを記録する

ログは、標準的なアウトプットまたはファイルにテキストを書き込むことによって生成されます。最も重要な判断は、何をログに含めるかを選別することです。ログには、調査する際にイベントと根本原因の特定に役立つ全ての必要なメタデータを含めるべきです。ログのメタデータの構成要素には、エラーメッセージまたはスタックトレースと、それらに関連する値、メトリクス、イベントなどが含まれる場合があります。

組織のログの全てのデータには、目的を持たせるべきです。データは、使用率であれ、ユーザーイベントであれ、アプリケーションのエラーと例外であれ、チームにとって価値があるべきです。ログデータ情報の要件：

- 何らかの形ですぐに役立つ
- 根底にある原因を理解し、判断を下すために必要な詳細情報を提供する

共通のシナリオを予想する

ログは、インシデントに対応するためだけのものではありません。業績のプロファイリングや統計データの収集など、ビジネスにも役立つことがあります。

共通のシナリオを幾つか念頭にログ記録を取ることにより、ログは組織に直接的な価値をもたらすようになります。たとえばユーザー同士の行動に関するログは、顧客体験に関する重要な洞察をもたらします。システムログにより、問題やハードウェア障害を監視できます。アプリケーションに関する詳細なログは、パフォーマンスとメモリリークなどの潜在的な問題への洞察をもたらすことがあります。これらは全て、ビジネス上の意思決定を行う上で重要なものとなり得ます。

有意義なメッセージを記録する

ログメッセージは、それが伝える情報やコンテクストと同じだけの価値があります。十分な詳細情報を追加して理解しやすい形に加工することで、チームはログを効果的に活用できます。サードパーティのインフラが必要な詳細情報を捕捉する傾向があります。それでもチームは、社内で作成されたアプリケーションについて、エラー/イベントが発生した理由を診断、特定し、ビジネスに影響を与える必要なアクションを実行できるように、常にログの詳細を記録する必要があります。

アプリケーションエラーの場合、メッセージはその命令行で何が起こっているかを伝える必要があります。たとえば、トランザクション失敗というエラーメッセージは、トランザクション失敗：ユーザー `${path/to/file:line-number}`を作成できませんでしたというエラーメッセージほど有効ではありません。トランザクションに関するデータを含むログは、トランザクションが失敗した理由を開発者やエンジニアが確認するのに役立ちます。

通常、プログラム中のエラーコードまたはステータスコードは、アプリケーションの問題の種類を示すことができます。エラーコードのテキストや番号を出力するだけでなく、ログに簡潔な説明を加えると、トラブルシューティングの際に他の開発者やエンジニアの時間と労力を節約できます。

ログは、組織にとって必要不可欠な情報を提供すべきです。開発者とエンジニアは、チームの限られたメンバーにしか理解できない意味不明のあるメッセージや分かりにくいメッセージを記録しないようにすべきです。

ログは常にシンプルで簡潔にする

ログに十分な情報を含めることは非常に大切ですが、その逆もまた然りです。過剰で不必要なデータをメッセージに含めると、ログのストレージサイズとコストが膨れあがり、ログの検索が遅くなり、本質的な問題から遠ざかり、デバッグがいっそう面倒なものになります。

チームは、ログの簡潔さを維持して最も重要な情報のみを取得すべきです。ログには、不要なノイズを避けつつ、エラーが発生した理由を含めるべきです。

その場合、環境に関する詳細情報は含めず、エラーの根本原因に関する情報を提供すべきです。たとえば、アプリケーションが内部APIに接続できずデータを取得できなかった場合、 APIからのエラーメッセージまたは環境のネットワーク状態に関する情報をログに記録すると役立つことがあります。アプリケーションが使用するメモリの量や実行中のアプリケーションの数を含める必要はないと考えられます。

忘れずに時刻の記録を行う

チームは、ログのタイムスタンプを含める必要があります。当然のことと思われるかもしれませんが、日時を自動的に記録するデータベースにログを書くことに慣れている開発者とエンジニアは、ログメッセージにタイムスタンプを含めることを考慮しない場合があります。その中で、納得のいく最も細かいレベルを選択し、ログに出力する必要があります。高頻度のタスクには、ミリ秒単位まで時間を追跡する必要がありますが、低頻度のタスクであれば分単位か、日単位でも十分かもしれません。大切なのは単なる細かさではなく、組織全体にわたり一貫性のある基準を適用することです。

もう1つの潜在的に明白かつ重要な注意点は、全てのシステムを同時に同期させ、オブザーバビリティプラットフォームがタイムスタンプを使用して、ログイベントを他のテレメトリデータと関連付けられるようにすることです。

解析可能なログフォーマットを使用する

ログからデータを抽出できないオブザーバビリティプラットフォームは、あまり役に立ちません。チームは、開発者とエンジニアが解析して一貫性のあるログ構造を維持できるログフォーマットを使用して、収集と集計を容易にすべきです。 New Relicログ管理機能を使用すると、カスタムログ解析ルール[1]を簡単に定義できますが、ログデータが分かりにくいと解析ルールは魔法のような効果を発揮できません。

解析されていないログフォーマットの良い例は、非構造化テキストを含むデフォルトのNGINXアクセスログです。検索には便利ですが、それ以外はあまり役に立ちません。解析されていないフォーマットの場合、チームは大半の質問に答えるために全文検索を行う必要があります。典型的な命令行の例を以下に示します：

```
127.180.71.3 - - [10/May/2022:08:05:32 +0000]
"GET /downloads/product_1 HTTP/1.1"304 0 "-"
"Debian APT-HTTP/1.3 (0.8.16~exp12ubuntu10.21)"
```

解析されたログはレスポンスコードやリクエストURLなどの属性に整理されます。解析可能なログフォーマットを使用した同じログ情報の例を以下に示します：

```
{
"remote_addr":"93.180.71.3",
  "time":"1586514731",
  "method":"GET",
  "path":"/downloads/product_1",
  "version":"HTTP/1.1",
  "response":"304",
  "bytesSent": 0,
 "user_agent": "Debian APT-HTTP/1.3
 (0.8.16~exp12ubuntu10.21)"
}
```

フォーマットが完全にカスタム化されている場合、ログの種類を設定するとユーザー定義の解析ルールが起動されます。

組織が共通目的に役に立つアプリケーションを複数持つ場合、チームは全てのアプリケーションでログフォーマットを標準化することに注力すべきです。これにより、各アプリケーションに関与するチームが異なる属性を可視化したい場合も、より簡単にオブザーバビリティプラットフォームへのデータの取り込みができるようになります。

[1](New Relic, Inc.、日付不明)

ログフォーマットの詳細

テキストを構造化する方法には、一貫性のある3つのフォーマットカテゴリーがあります。いずれもログ集計ツールがデータを収集した後の使い勝手に影響します。3つのフォーマットカテゴリーは以下のとおりです：

- **構造化**－ログ記録の最も一般的な構造化フォーマットの1つがJSONです。JSONの解析を迅速に行えるツールは多数あります。JSONは柔軟性に富み、軽量です。理想的には、生成される全てのログが構造化フォーマットになっていることです。JSONは階層データの体系化に貢献します。カンマ区切り値（CSV）やタブ区切り値（TSV）など、他の共通フォーマットも構造化ログデータの例です。
- **共通化**－共通フォーマットは構造化されていませんが、よく知られており、定義され、一貫性があります。ログにアクセスするためのApacheという共通ログフォーマットはその一例です。共通フォーマットの利点は、多くのツールが「追加設定なし」でデータを解析できることです。
- **カスタム**－アプリケーションがログ記録を構造化フォーマットや共通フォーマットで行っていない場合、そのアプリケーションはカスタムフォーマットでログを記録しています。ログの転送中に個々のログ行の始まりと終わりを識別するため、チームは解析を行う必要がある場合があります。顧客定義の構文解析規則を作成すると、データの価値をいっそう高める一助となります。

ログのカテゴリー化とグループ化

ログのデータモデルを指定すると、検索がより効果的に行えるようになります。チームは可能な限りいつでも属性を定義して含め、それに応じてログをカテゴリー化およびグループ化する必要があります。

New Relicを含む業界リーダーが集まって策定したログに関するOpenTelemetry標準は、命名規則やフィールド値の定義など多くの要素をカバーしています[2]。これらの標準に正確にフォーマットされたログは、すべてのフレームワークによってネイティブにサポートされているわけではありませんが、ガイドラインとして使用できます。

ログのデータモデルに含めると便利な共通属性には、リソース、コンテクスト化されたログ、およびログレベルなどがあります。

リソース

リソースは以下のようなログの時期と出所を明確にします。

- 日時
- マシン、ホスト名、または識別子
- アプリケーション名またはサービス名

環境に名前があるクラシックなホストベースのアプリケーションでは、ホスト名はログの中で意味を持ちます。ポッドIDやコンテナIDは、コンテナ化された環境や組織化された環境からのログの体系化を改善します。

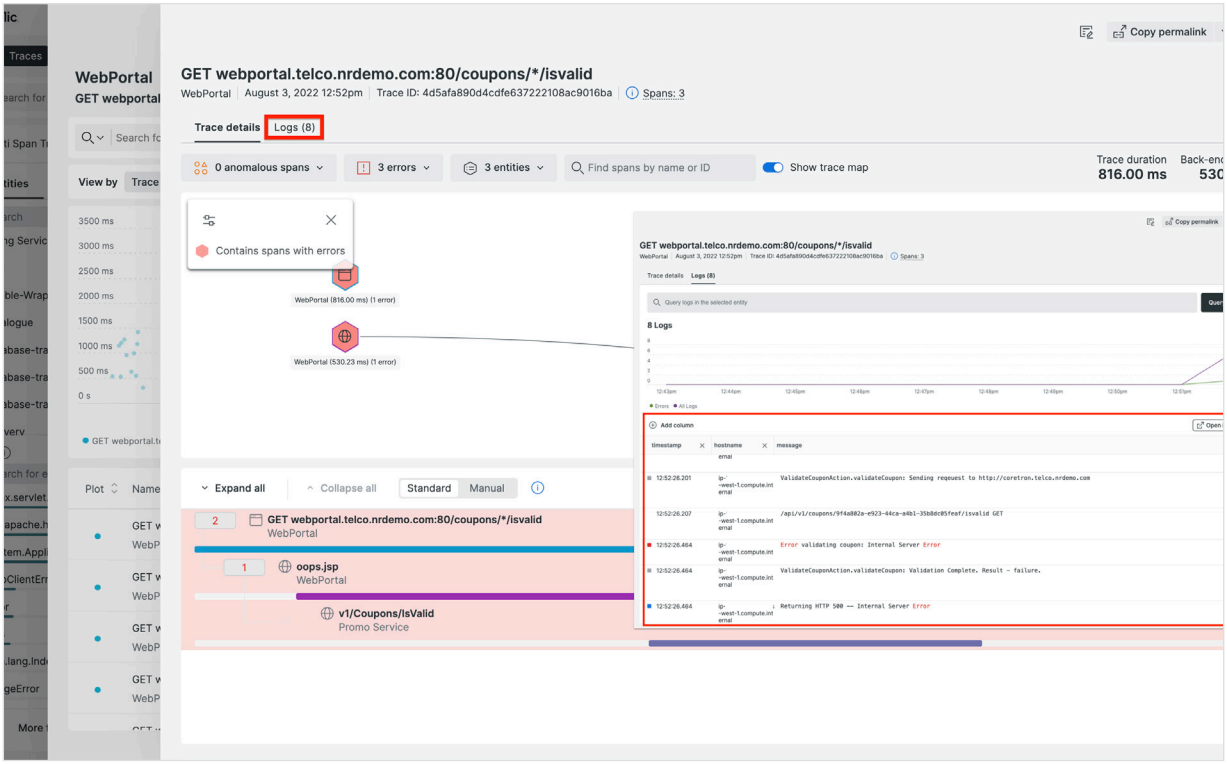
多くの場合、組織化された環境またはサービスとしてのプラットフォーム（PaaS）環境では、ログに大量のメタデータが自動的に入力されます。これは組織にとっては素晴らしいことですが、製品バージョン番号、ステージング環境と運用環境、テストブランチ、 A/Bテストバージョンなど、システムが認識できない有用な修飾語句でログに注釈を付けることも重要です。ログ集計とは、複数ソースから採取した全てのログを同じシステム上に集めることです。チームは、正しいメタデータがなければ、テストランを行う中で失敗となったトランザクションと、運用環境における真のエラーログとを識別できません。

チームによるログソースの識別に役立つもう一つのリソースがログ転送です。たとえば、 New Relicが提供するほとんどのログ転送ソリューションでは、データの搬出に使用したツールのタイプとバージョンを注釈としてデータに自動的に添付します。

[2](OpenTelemetry、日付不明)

Logs in Context

チームにとって、アプリケーションの問題に関するコンテクスト化されたログは有益です。たとえばNew Relic logs in contextの機能は、ログにアプリケーション情報を自動的に追加できます。New Relic APMエージェントは、ロギングフレームワークにアプリケーションパフォーマンス管理データを提供し、アプリケーションログに包含します。その結果、コンテクスト化されたログが関連アプリケーションイベントとトレースにログデータを自動的に関連付けます。APMエラーと分散トレーシングは、エラーやトレースと同じトランザクション中に生成されたログに直接リンクされます。コンテクスト化されたログは、ログメッセージにスパンID、トレースID、アプリケーション名を挿入することにより、こうした関連付けを行います。これによりチームはアプリケーションとログデータを一体化して、トラブルシューティングをより迅速に行うことができます。



New Relicのオブザーバビリティプラットフォームにおける、トレースのコンテクストの中でエラーを表示するためのフィルタリングされたログ

ログレベル
開発者、 DevOps担当者、マネージャーは、重大性の度合いに従ってログのレベルを参照することがあります。ログのレベルは、イベントの相対的重要性を説明し（デバッグ、情報、警告、エラー、致命的などの観点で）、ロギングフレームワークから得た情報の密度も説明してくれます。重要度に関する属性は、さほど重要でない情報をフィルタリングしたり排除したりするのに役立ち、チームが重大なエラーだけに専念できるようにします。

ログレベルを有効活用すれば、データ量を制限し、集中ログ管理ツールの使用コストを節約し、検索が迅速になります。場合によっては、アプリケーションのログ生成方法を管理できないことがあるかもしれませんが、理想的な状況ではログ管理システムによって不要なデータを排除できます。 New Relicでは、チームは、たとえばログレベルに基づく機械学習パターンを利用して異常値を抽出できます。カラーコード化されたログレベルにより、インジケーターを視覚化して、最も重要な領域に注力することができます。

チームはログレベル、特にデバッグログレベルを注意して使用すべきです。特定の動作に伴う冗長なメッセージを取得するのにデバッグは大いに役立ちます。しかし不必要なデバッグは膨大な量のログをもたらし、付加価値を提供することなくログの取り込みと検索機能の速度を低下させます。より規模の大きなチームとプロジェクトは、一貫性のあるグループ化、カテゴリー化、およびログ記録方法に関するログレベルの基準から恩恵を受けることがあります。

ロギングツールとフレームワークの使用

ロギングソリューションを一から導入して時間やリソースを費やす代わりに、十分にテストされた実績のあるロギングツールとフレームワークを使うことで、チームは時間を節約しトラブルを回避できます。たとえば、 New RelicのAPM言語エージェントは、必要なメタデータをログに付加し、自動的なlogs-in-context機能と転送ログへのアクセスを提供します。サードパーティのソフトウェアをインストール、管理する必要はありません。

一貫性のあるロギングフレームワークを使用すると、エンジニアチームによる導入が簡素化され、ログ出力が正常化されます。コンテクストに応じたログを一様に表示することができます。ロギングフレームワークの導入に際しては、新しいコードを導入する場合と同様に、慎重を期し、パフォーマンスの影響をテストすべきです。

大量のデータは保存場所の参照にとどめ、ログに含めない

状況によっては、ログから大量のデータを取得してより深いコンテクストを提供する必要に迫られることもあります。そうしたデータには、たとえばメモリダンプや一連のファイルや画像などがあります。このようなデータは別途保存するか、指定されたサーバーにアップロードし、ログそのものには含めず、保存場所を参照するにとどめるのが得策です。ログのデータ容量はできるだけ小さくし、データには別途アクセスするようにします。

有用なビュー、クエリ、アラートを共有する

チームは、組織の現在の状態に関する幅広い洞察を提供し、チーム間の可視性とコミュニケーションを向上させるため、ログの標準的な視覚化、クエリ、アラートを作成して共有する必要があります。これがフルスタックオブザーバビリティのパワーです。

ログに加えるべきでないもの

役に立つものやすべてを記録したくなりがちですが、いくつかの例外や落とし穴を避ける必要があります。

機密情報

機密情報は慎重に取り扱う必要があります。個人識別情報（PII）およびクレジットカード番号などの規制データは、欧州連合の一般データ保護規則（GDPR）[3]および米国の医療保険ポータビリティおよび説明責任法（HIPAA）などの規制法に従って保護することが不可欠です。[4]

オープン・ウェブ・アプリケーション・セキュリティ・プロジェクト（OWASP）のロギングガイドラインには、記録すべきでないものが記載されています。例えば、アクセス用トークン、パスワード、機密情報、個人が明かしたくない情報などです。[5]

プライベートサーバーやデータベースに保管されたログの場合、氏名やメールアドレスなどのPIIを偶発的に簡単に記録してしまうことがあります。特定のユーザーの行動またはイベントの追跡に際しては、匿名の識別子を使用する必要があります。ログデータはNew Relicなどのオブザーバビリティプラットフォームに安全に保管されますが、PIIを組織外に送信する場合は細心の注意を払う必要があります。

ソースコードと占有データ

規制およびコンプライアンス情報に加え、アプリケーションのソースコードや組織内の保護されたデータなど、その他の機密情報をログに保存することを避けたい場合があります。

ログを安全に保管することに加えて、ログへのアクセスも安全に行うことが重要です。取引上の秘密や、未公表または未発表のプロジェクトや機能を明らかにする可能性のある情報は、ログの中に含めることはできません。特にサードパーティのサービスにログを外部保存する場合は、ログからこうした情報を削除する必要があります。

情報の複製

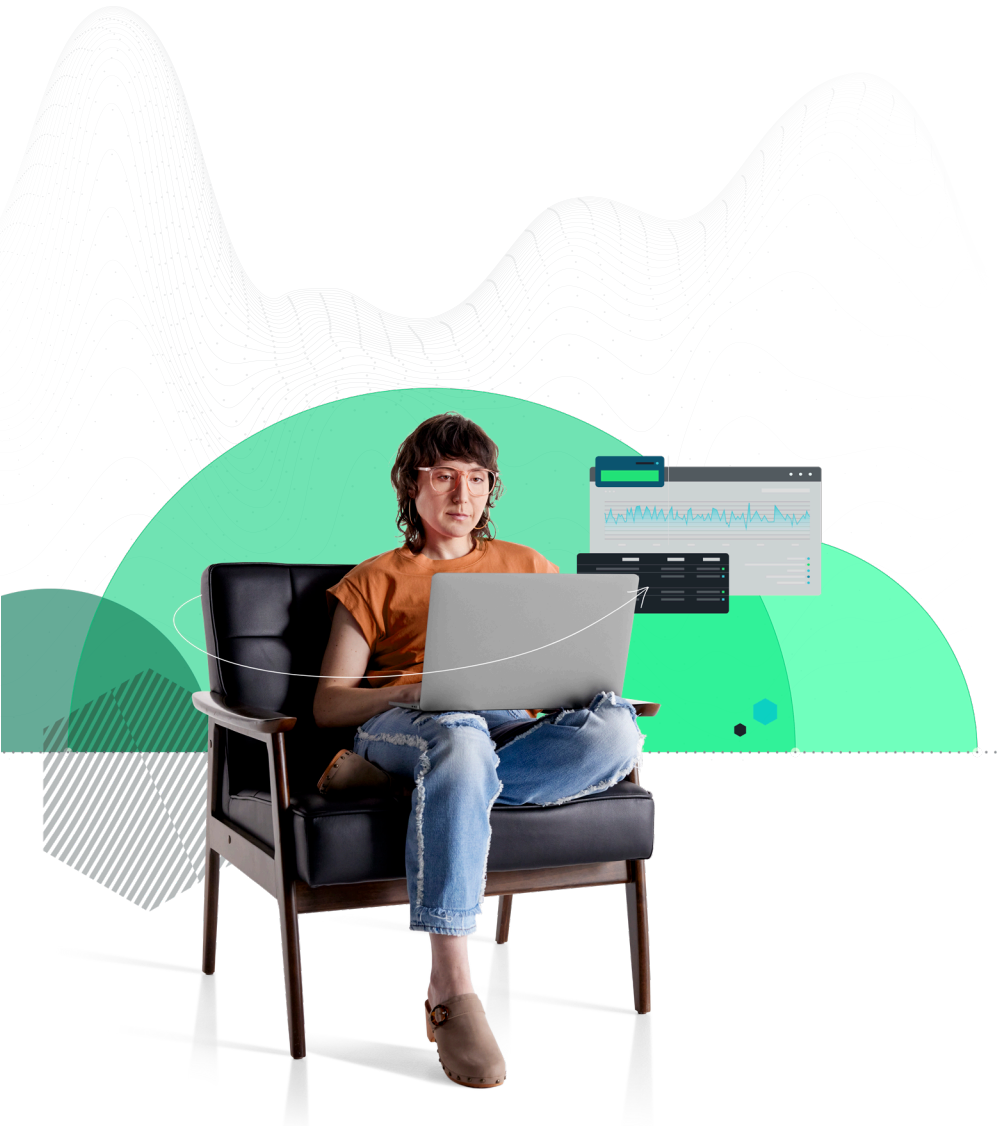
複製した情報をログに追加しても問題はありません。また情報は多いに越したことはありません。ただし、複製された情報を含めると不必要なログが増え、目的は達成されずにコストがかさむことになりかねません。

[3]（欧州委員会、日付不明）
[4]（米国保健福祉省米国（HHS）、日付不明）
[5]（オープン・ウェブ・アプリケーション・セキュリティ・プロジェクト（OWebASP）、日付不明）

結論

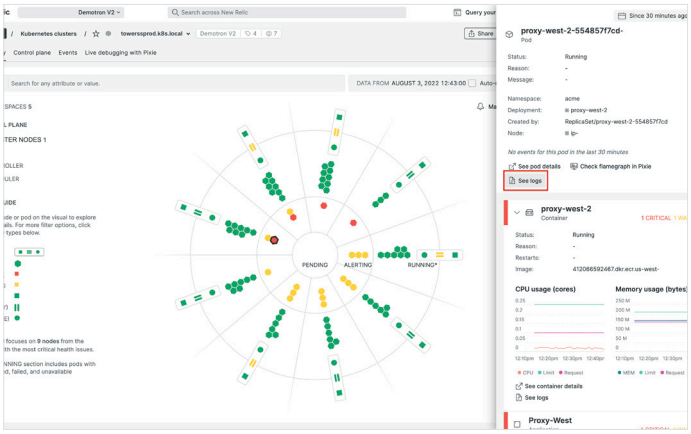
ログを利用してフルスタックオブザーバビリティを強化すると、ビジネスにインパクトを与えるリアルタイムの意志決定が下せるようになり、開発者やエンジニアのデバッグやインシデント対応時間が短縮され、その時間を有効活用できます。

このようなベストプラクティスを導入すると、顧客向けの業務を円滑に進めるための必要情報をログから得られ、スタック全体の視覚化が改善され、問題を速やかに解決し、開発スピードを上げることができます。



New Relicオブザーバビリティプラットフォーム

New Relicは、詳細なログを含むすべてのテレメトリデータのための単一の統一されたプラットフォームを提供します。New Relicのオブザーバビリティプラットフォームには、ログ管理、APM、インフラストラクチャーの監視、サーバーレスモニタリング、モバイルモニタリング、ブラウザの監視、合成モニタリング、分散トレース、Kubernetesモニタリングなどが組み込まれています。これらの機能により、組織はソフトウェアスタック全体を視覚化、分析、トラブルシューティングを行うことができます。このプラットフォームの一部であるNew Relicログ管理を使用すると、組織はロギングデータとアプリケーションおよびインフラストラクチャーのモニタリングデータを組み合わせることができ、強力なオールインワンの観測可能なプラットフォームが実現されます。



APM、インフラ、イベント、ログアクセスを一覧表示

New Relicは、AIOps（IT運用のための人工知能）と統合されたソフトウェアスタック全体の指標、イベント、ログ、トレースをつなぎ合わせます。これにより、組織はより迅速にログを検索でき、異なるレガシーソリューションを使用するよりも手頃な価格で利用できます。開発者とエンジニアは、スタックの異なる部分に別々のツールを使用する代わりにNew Relicを使用することで、特定のエラーに関する詳細なログを全て容易に把握することができます。

レガシーなログ記録ソリューションでは、スピードとスケーラビリティに問題があり、遅延データの実行に数分、または数時間もかかるため、詳細なログへのクエリが厄介になります。これに対して、New Relicのログ管理検索はわずか数秒しかかかりません。したがって、ソフトウェアのフルスタックにおけるインシデントの調査と対応が可能な限り速く行うことができます。

New Relicのオブザーバビリティプラットフォームには、ログ管理、データ量の少ない顧客のための無料ティア、チームが必要な詳細なログをすべて取り込むことができる、GBあたりの抑制された価格が含まれています。

New Relicログ管理の使用を開始するには、今すぐ無料アカウントにサインアップしてください。無料アカウントには、1月あたり100GBのデータ取込み、1名のフルプラットフォームユーザー、および人数無制限の無料ベーシックユーザーが含まれます。

今すぐサインアップ

参考資料

European Commission. n.d. “EU data protection rules.” European Commission. Accessed July 19, 2022. https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en.

New Relic, Inc. n.d. “Parsing log data.” New Relic Documentation. Accessed July 28, 2022. https://docs.newrelic.com/docs/logs/ui-data/parsing/#custom-parsing.

OpenTelemetry. n.d. “OpenTelemetry Logging Overview.” OpenTelemetry. Accessed July 18, 2022. https://opentelemetry.io/docs/reference/specification/logs/overview/.

Open Web Application Security Project (OWASP). n.d. “OWASP Logging Guide.” https://owasp.org/www-pdf-archive/OWASP_Logging_Guide.pdf.

U.S. Department of Health and Human Services (HHS). n.d. “Summary of the HIPAA Security Rule.” HHS.gov. Accessed July 19, 2022. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html.

new relic.