CASE STUDY

GitHub

elastic

課題:

GitHubの400万ものユーザーの検索ニーズを満たすと同時に、顧客サービスの継続的な向上に役立つ運用の洞察を提供するにはどうすればよいか。

ソリューション:

Elasticsearchを利用して、重要なイベントデータと800万以上のコードリポジトリをインデックス化。

エンドユーザーと開発者の両方にとって強力な検索が可能に

- Apache SolrからElasticsearchへの移行により、スケールアウトが可能になり急激に拡大するユーザーのニーズ対応に
- ほとんどのタイプのデータをインデックスして検索可能に
- 開発者アプリケーションでプログラムによる高度な検索が可能に
- 新しいデータのアップロードをほぼリアルタイムでインデックス化

検索データの分析から洞察

- インデックスされたログデータからクエリを実行して不正ユーザーを検出
- すべてのアラート、イベント、ログをインデックスし、特定のコードの例外の割合を追跡して、GitHubプラットフォーム内のソフトウェアバグを発見
- 標準SQL以上のクエリを作成

上級ユーザーのための高度な検索

GitHubは、世界最大のソフトウェア開発プロジェクトのための共有Webサービスであり、その顧客ベースは、要求レベルの高い400万人以上のテクニカルユーザーで構成されています。このGitHubの検索を支えているのがElasticsearchです。GitHubではElasticsearchを利用し、コードリポジトリ数800万以上、ドキュメント数20億以上の拡大し続けるデータを常にインデックス化しています。Elasticsearchを利用したことで、GitHubのユーザーはこのデータを簡単に検索できるようになりました。

GitHubの運用エンジニアであるTim Pease氏は次のように述べています。「検索はGitHubの中核部分です。GitHub.com/searchにアクセスすれば、リポジトリ、ユーザー、イシュー、プルリクエスト、ソース コードすべてを検索できます。」

GitHubがElasticsearchを導入した目的の1つは、GitHub.comで公開・提供されているものすべてをインデックスして、見つけやすくすることです。当然、フルテキスト検索は完全にサポートされますが、幅広い条件に基づく検索も可能で、操作はきわめて簡単です。

「Clojure を主要な言語として使用し、前月アクティビティのあったプロジェクトといった条件で検索ができます。こうした機能はすべて Elasticsearch によって動いているのです。」(Pease 氏)

Elasticsearchの柔軟なアーキテクチャにより、構造化データも非構造化データも区別なく保存・検索できます。また、広範囲にわたる検索機能を組み合わせることで、極めて容易に検索を実現できます。

「Elasticsearchを使用すると、標準のSQLデータベースではサポートされていない多くのクエリをデータに対して実行できます。」(Pease 氏)

ファイアウォール内のセキュリティ分析に関する洞察を提供

GitHubでは、Elasticsearchの検索インデックスと分析の機能を組み合わせて複数のプロジェクトを推進しています。たとえば、GitHubでは、Elasticsearchクエリの分析機能を利用すると、保存されている監査データとログデータを使用してユーザーのセキュリティ関連のアクティビティを追跡できることに気付きました。

「Elasticsearchクエリを使用して、ユーザーが実行したアクションの1つ1つをすぐに確認できます。そのため、アカウントの盗難や乗っ取りがないか、ユーザーが不正な行為をしていないかを的確に監視できます。」(Pease 氏)

GitHubは、GitHub.comに搭載されているさまざまなソフトウェアコンポーネントで生じるコードの例外を追跡・分析する方法を検討し、当初検討したのは一般的なNoSQLデータベースでした。当時、コード例外は2次インデックスに保存され、分析機能を使用して例外の経時分析が行われ、結果は再びデータベースに保存されていました。

GitHubのテクニカルスタッフであるGrant Rodgers氏は次のように述べています。「NoSQLデータベースはGitHubのユースケースには適していませんでした。すべてをElasticsearchに移行してそのヒストグラムファセットクエリを使用したところ、非常にうまく機能しています。」

GitHubではElasticsearchのヒストグラムファセットクエリ機能と、その他の統計ファセットを使用して、特定のタイプのコード例外が占める割合の増加を追跡しています。このプロセスにより、ソフトウェアシステムのバグが明らかになっています。

「Elasticsearch のヒストグラムファセットクエリ機能はきわめて有効であるため、このアプリケーションの使用をさらに拡大することを検討しています。」(Rodgers 氏)

数百万ものユーザーに合わせて拡張

GitHubでは当初、検索にSolrを使用していましたが、Solrは効果的な拡張ができず、管理のしやすさという点でも不十分でした。

「GitHubの利用者が増えるにつれ、1つのSolrクラスタとSolrインスタンスで処理できるストレージ スペースではすぐに足りなくなりました。」(Pease 氏)

社内のデータをSolrでシャーディングして負荷に対処するか、Elasticsearchに移行するかの選択を迫られましたが、迷いはありませんでした。Elasticsearchのシャーディング機能の方が優れていると判断し、移行を決定しました。」(Pease 氏)

Elasticsearchの自動シャードリバランス機能により、パフォーマンスが向上し、フェイルオーバー状況に対応できます。レプリカシャードは、クラスタ内の新しいノードに自動的に分散され、ノード障害の場合は、シャードは自動的に障害ノードから稼働ノードに移行されます。

高度なシャーディングによる高パフォーマンス

20億以上のドキュメントがすべてElasticsearchでインデックスされ、ユーザーによるコードのアップロードや変更が絶えず行われている環境においては、検索のパフォーマンスが重要な指標となります。GitHubでは、平均で1分あたり300件の検索要求を処理しています。

ユーザーがGitHubのリポジトリに新しいコードをプッシュすると、すぐにElasticsearchでインデックスされます。データは直ちに検索可能になり、パブリックリポジトリだけでなく、ログインユーザーであればアクセス権のあるプライベートリポジトリについても検索結果が返されます。

検索データへのアクセスを最適化するために、GitHubでは広範囲にシャーディングを使用しています。GitHubのElasticsearchメインクラスタには、シャードが約128個あり、各シャードにはおよそ120GBが保存されています。

1つのリポジトリ内での検索を最適化するために、GitHubではリポジトリIDに基づいたElasticsearchルーティングパラメータを使用しています。「そうすることで、1つのリポジトリのソースコードを1つのシャードに配置できるのです。ユーザーが1つのリポジトリページでのみ検索を実行すると、実際に1つのシャードだけが検索されます。GitHubのメイン検索ページからの検索に比べ、このようなクエリは速度が2倍になっています。」(Pease 氏)

GitHubにおけるElasticsearch使用のメリット

✓ 効率的な拡張

GitHubは、Elasticsearchの堅牢なシャーディングと高度なクエリを使用して、400万人ものユーザーのコードリポジトリにあるデータを横断的に検索が可能に。

✓ 高パフォーマンス

Elasticsearchのルーティングパラメータと柔軟なシャーディングスキームを使用して、1つのシャードの1つのリポジトリ内で検索を実行することで、検索速度が2倍に。

✓ 高度なクエリによる分析

Elasticsearchのヒストグラムファセットクエリと、その他のElasticsearch分析クエリを使用して、不正使用やバグなどがないか、内部のインフラストラクチャを監視。

✓ ユーザーと開発者のニーズに対応

Elasticsearchは、通常のユーザーに加え、Elasticsearch APIを提供してアプリケーション開発者の検索ニーズに対応。

www.elastic.co/jp

Elasticの詳細については、 elastic-japan@elastic.coまでお問い合わせいただくか、
URLをご覧ください：www.elastic.co/jp

elastic