リクエストとDB のパフォーマンスを Datadog で監視する

Datadog ユーザ会 #14 @福岡

フューチャー株式会社

市川裕也

自己紹介

■ **名前：** 市川 裕也

■ **経歴**

- 2024/07 フューチャー新卒入社
- 2024/11~ アプリケーションエンジニア@FutureVuls

■ **現在主に取り組んでいること**

現在は、主にアプリのパフォーマンス関連の問題に取り組んでいます（SRE と SWE の中間みたいな…？）

- DB 実行計画を見て、パフォーマンス問題の原因を特定する
- ORM に hint 句を導入してパフォ問題解決
- **Datadog を用いてパフォーマンス監視**

アジェンダ

1. 監視導入前の課題と監視対象の選定
2. 監視ユースケース① : リクエストのレイテンシ (カスタムメトリクス)
3. 監視ユースケース② : DB スロークエリ (DBM モニター)
4. まとめ

サービスの紹介

■ どんなサービス？

⇒ 現在、我々は「FutureVuls」という脆弱性管理サービスの開発・運用をしています。

■ どんな状況？

⇒ ユーザが増えてきて、リクエスト数やデータベースのサイズが日に日に大きくなっている状況です。

今年上半期時点での Datadog 使用状況

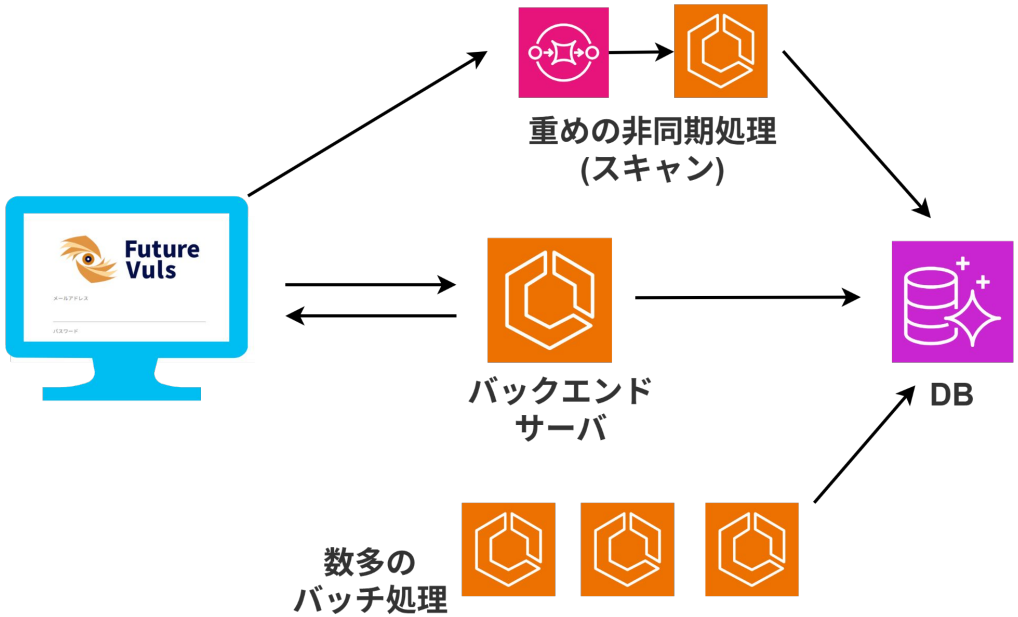
今年 2 月ごろに Datadog が導入され、あらゆる調査に APM が大活躍していました。
一方, 他の機能はあまり使っておらず…　といった状況でした。

←
弊社の棚井も3 月の
Datadog 会@札幌で発表さ
せていただいています

サービスの状況

バックエンドは複数のコンテナから構成されています。

複数のコンテナサービスが、ひとつの DB を見に行っています。



監視導入前の課題

ユーザからの問い合わせが来るまで、以下のようなパフォーマンス関連の問題に気づけない状態でした。

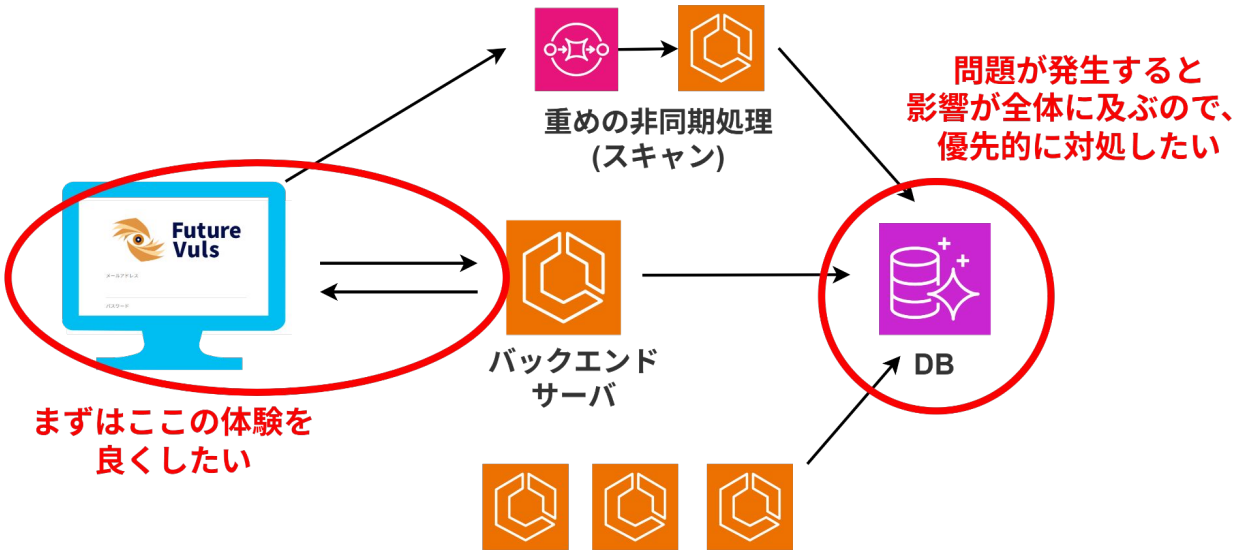
- 裏で走り続けるプロセスの**ロック待ち**が発生している
- サービスの特定画面が**異様に遅い**
- 特定の時間帯に**アクセス負荷が集中**している

⇒ ユーザの拡大に伴い、監視して状況を把握する必要が出てきました。

最初に何を監視対象としたか

① **リクエストのレイテンシ**：ユーザ体験に最も直結する指標のため。

② **DB のスロークエリ**：パフォーマンス問題の多くが DB 起因だったため + 影響範囲が広いため

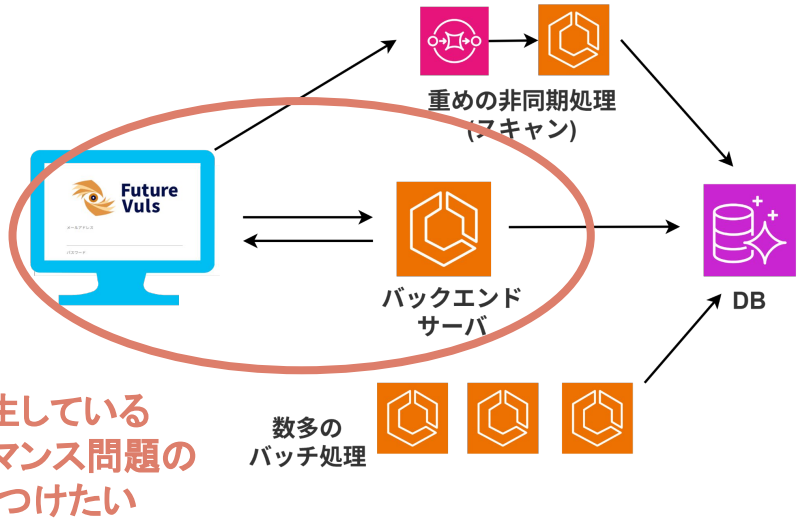


監視ユースケース ①
各リクエストのレイテンシの監視

リクエスト監視で達成したかったこと

リクエスト監視で達成したかったことは
以下のとおりです:

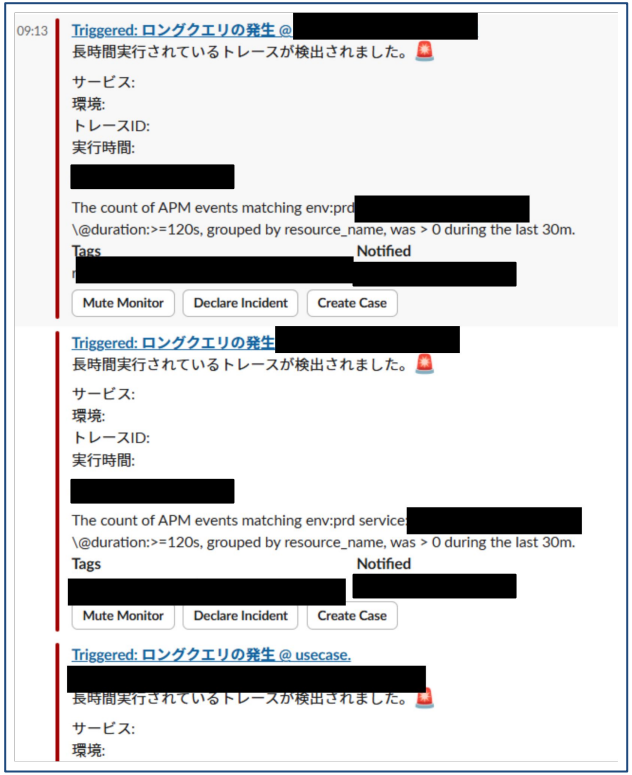
- 各エンドポイントで、どれくらいパフォーマンスの問題が発生しているかを把握する
- 把握した情報を元に、施策の優先度を決定する



アラートの失敗

最初は、30 秒を超えるリクエストが発生する度にアラートが飛ぶようにしたが、アラートが飛びまくりアラート疲れを起こしてしまった。

⇒ レイテンシ大なリクエストの数だけ分かれば十分なことに気づき、**「集計して日次で報告」**というスタイルに変更



↑
当時の私の slack チャンネル

運用方針

■ 現在の運用方針：日次集計

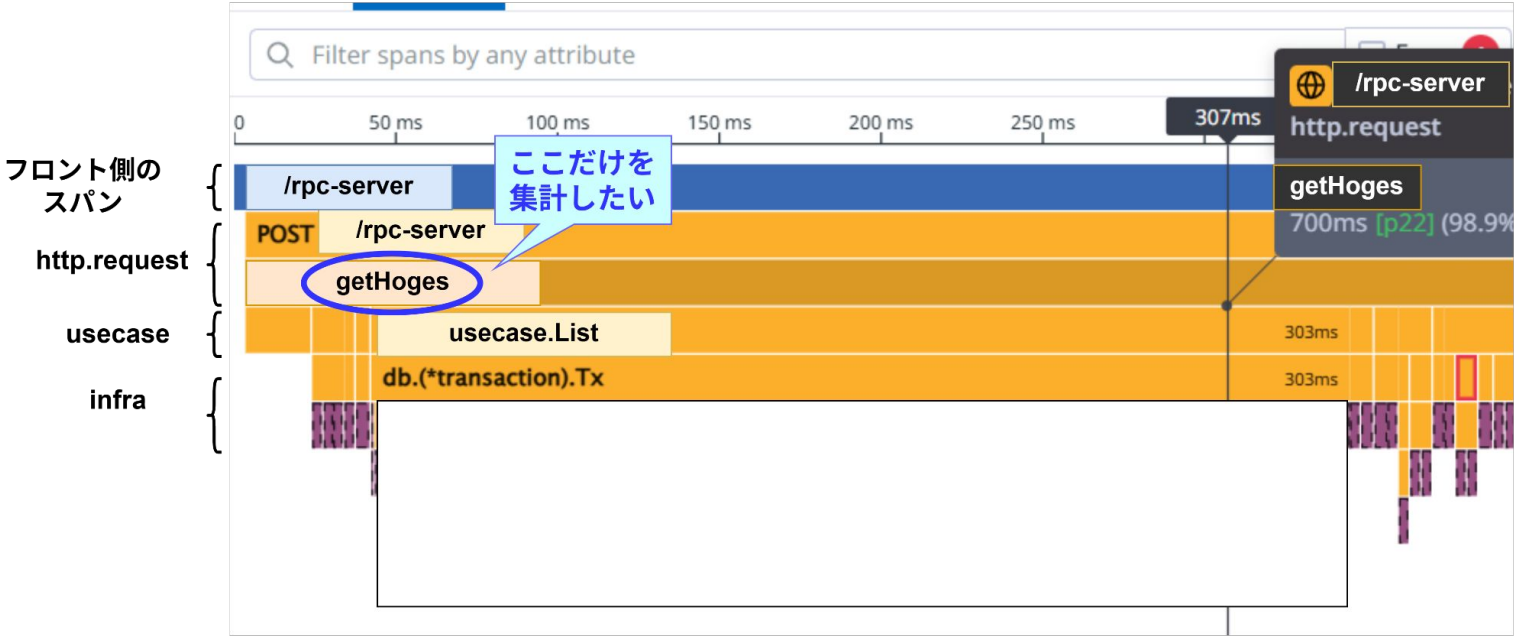
- 30 秒越えのリクエスト数をリソース毎に集計するダッシュボードを作る
- ダッシュボードを日次で slack のチャンネルで共有

↑
これが日次で流れてくるイメージです

何を監視対象としたか

各リクエストから取得できるトレースは以下のような感じです。

このうち、http.request の部分のみ抽出して、リソース名毎に監視するようにしました。



集計の設定方法

集計には **カスタムメトリクス** を使用しました。(カスタムメトリクスの採用経緯と注意点については Appendix を参照)

Q. カスタムメトリクスとは

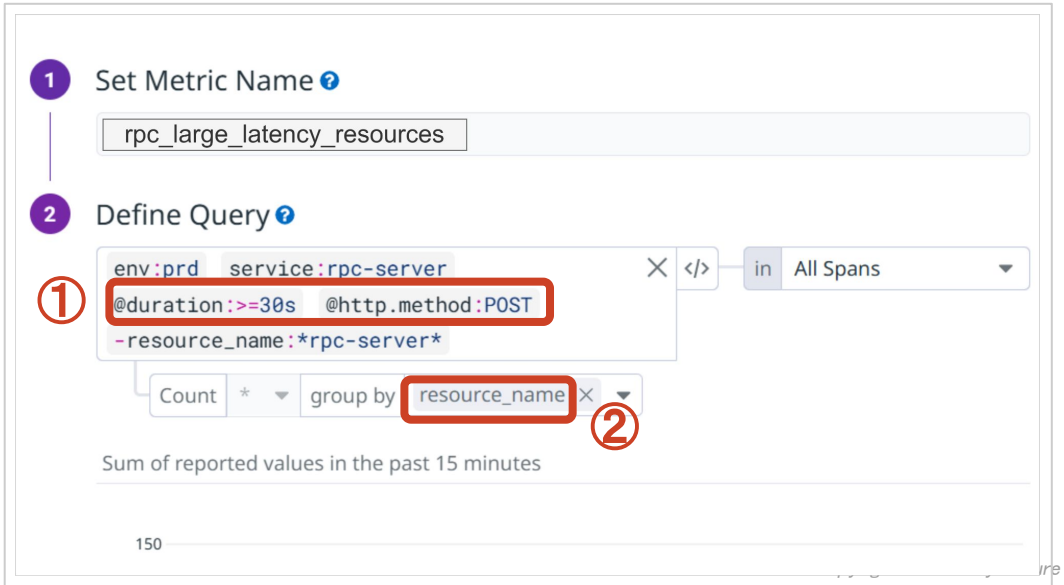
A. APMスパンや DBM 等から 自分で作成できるメトリクスのこと。カスタムメトリクスは、メトリクス名とタグ値 (ホストタグを含む) のユニークな組み合わせによって識別されます。

cf. https://docs.datadoghq.com/ja/metrics/custom_metrics/

集計の設定方法

レイテンシが大きいリクエストのみをAPM スパンから抽出するため、
以下のようなカスタムメトリクスを作成しました

- http.request のスパン && 30 秒以上のスパン　のみに絞る
- `resource_name` で group by する



集計導入の before/after

■ before

- ✕ どのリクエストがどれくらい遅いかが分からなかった
- ✕ ユーザの問い合わせ以外の指標が存在しなかった

⇒ 何から対応すれば良いかを決めづらかった

■ after

- ○ どのリクエストがどれくらい遅いかが**継続的 & 定量的に**分かるようになった
  - □ ⇒ 次に取り組むべき課題が明確に
- ○ 開発メンバに、レイテンシが大きいリクエストについて共有できるようになった

監視ユースケース②
DB のスロークエリ

DB 監視で実現したかったこと

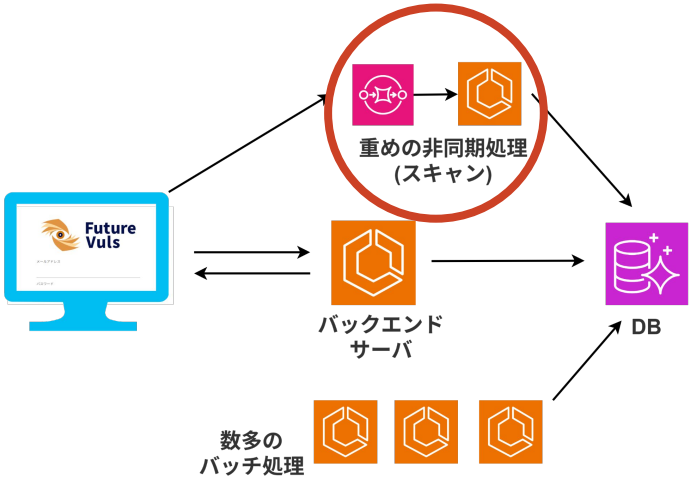
① リアルタイムなアラート

pg_stat_activity を用いた原因特定のため、問題発生にリアルタイムで気づきたい。（どのクエリでロック待ちが発生しているかなど）

後からの調査では原因特定が難しいケースが多かった。

② 過度なスロークエリ検出

「異常に時間がかかっているスロークエリ」が検出された時にアラートが上がってほしい。（主に、右図赤丸の非同期処理によるロック待ちなどを検知したい）



⇒「データベースモニタリングモニター」が最適！

運用方針

■ 設定方針

30 分以上かかっているクエリのみを検出する設定としました。

(∵ アラート疲れを起こさないようにするため)

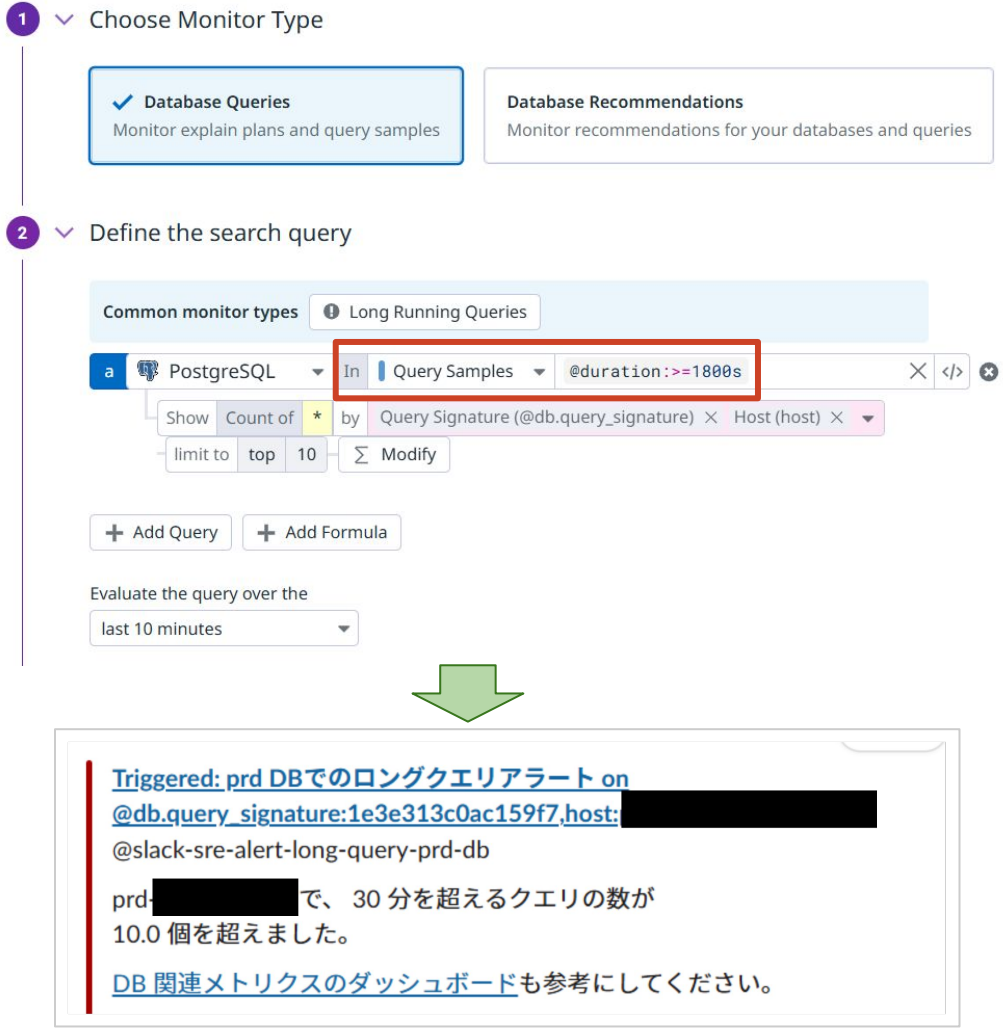
■ 設定方法

Monitor で「Database Monitoring」を選択します。(詳細仕様は Appendix 参照)

その後の設定は右のとおり

↓

スロークエリが検知された際にアラートが飛ぶようになります。



アラート導入の before/after

before

- ✕ DB で発生している問題に気づけず、調査が後回しになってしまっていた

after

- ○ 問題のあるクエリに即時で気づけるようになり、リアルタイムな状況を調査しやすくなった。
- ○ 閾値を大きめ（30分）に設定したことで、アラート疲れを起こさず、本当にヤバいクエリに絞って調査を行えるようになった。

まとめ

- やりたいことに合わせた監視方法を選択することで、アラート疲れを起こさず必要な情報を取得・共有できる
- 「リアルタイム性が必要か」は、重要な考慮ポイント（必要ないなら日次集計で OK）
- DB モニタリングモニター、便利

監視対象	やりたいこと	運用	導入後
①：リクエストのレイテンシ	各エンドポイントで、どれくらいパフォーマンスの問題が発生しているかを把握する ⇒ 優先度付け	カスタムメトリクスを日次集計	以下が可能になった - メンバーへの共有 - 定量的な傾向を掴む
②：DB のスロークエリ	リアルタイムで問題に気づきたい (その場で原因調査したいため)	DBM モニターでアラート	その場で原因調査しにいけるようになった

まとめ

ご清聴いただき
ありがとうございました

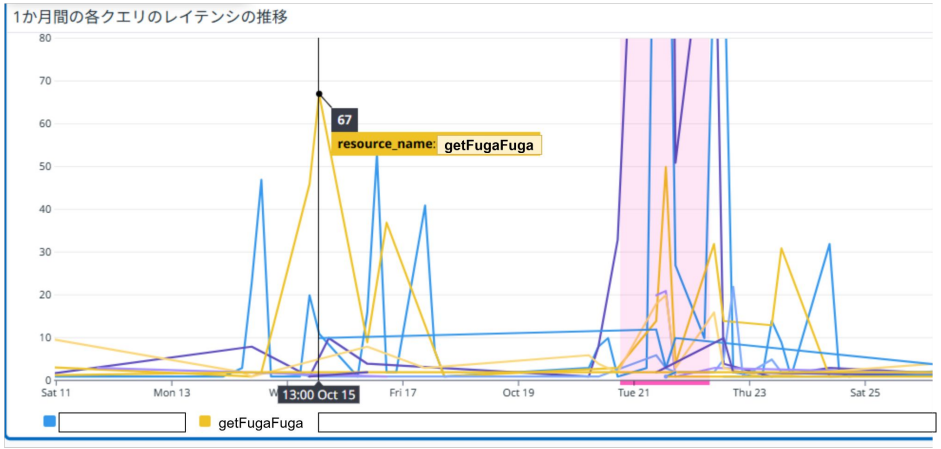
Appendix：なぜカスタムメトリクスを使用したか

A.　長期間の推移も見たかったから

実は、日次集計のみであれば、カスタムメトリクスを使用する必要はありませんでした。
（ダッシュボードで直接先ほどの条件を指定すれば良いだけ）

しかし、メトリクス化することで、右図のように**長期の傾向**を見ることができるようになります。

推移も見られると嬉しかったため、カスタムメトリクスを使用しました。



Appendix : カスタムメトリクスの料金面の注意点

カスタムメトリクスは、タグの組み合わせの数に応じて課金が発生します。

group_by の単位を 3 つ以上設定したりすると、組み合わせ数が爆発して破産する可能性もあるのでご注意ください。

今回は、以下の 2 点よりほぼ課金が発生しないだろうと判断し、カスタムメトリクスを採用しました。

- group by の単位を `resource_name` のみに絞った
- カスタムメトリクスを取得するリクエストを、30 秒以上かかったもののみに絞った

Appendix : DBM モニターの内部処理

PostgreSQL のクエリパフォーマンスを調査したい場合は、以下のいずれかを用いることが多いです。

- `pg_stat_activity` : 現在走っているクエリの情報を取得できます。リアルタイムの調査に便利です。
  - クエリの wait 状況、クエリのスタート時間 等
- `pg_stat_statements` : クエリのトータル実行時間などの統計情報を取得できます。

データベースモニタリングモニターで「In Query Samples」を選択した場合、**pg_stat_activity の情報**が用いられるようです。リアルタイムな情報を用いてアラートを上げてくれるはとても嬉しいなと感じました。

（こちらの仕様については、サポートの方にお伺いしました。この場を借りて感謝を申し上げます。）