sentryオススメ設定

@vividmuimui

2018/02/06 社内LT資料

sentryとは

ざつにいうと、プログラムのエラーを通知したりその詳細を見れたりするサービス

https://docs.sentry.io/
Ruby, C#, JS, Go etc色々対応してます

はじめに

- Ruby/Rails中心に書きます
  - 他の言語でもほぼ一緒だと思いますが違ったら良い感じに読み替えたり調べたりしてください
- まだ使いこなしたとはいえない状態で書いているので、現時点でのオススメ設定です
- 動かすための基本設定みたいなのは書かないです

ドキュメント系

- ドキュメント
  - https://docs.sentry.io/
  - https://docs.sentry.io/clients/ruby/
  - https://docs.sentry.io/clients/ruby/integrations/rails/
- 設定画面一覧
  - account settings https://your-sentry.example.com/account/settings/
  - project settings https://your-sentry.example.com/sentry/sandbox/settings/
  - organization settings https://your-sentry.example.com/organizations/sentry/settings/

秘匿情報はきちんとfilterする

メールアドレスやパスワードなどがsentry上で表示されないようにするため。
https://docs.sentry.io/learn/sensitive-data/
https://docs.sentry.io/clients/ruby/config/#optional-settings のprocessorsの項目

railsだと以下のようにすると良いとdocumentにある。

```
config.sanitize_fields = Rails.application.config.filter_paramete
```

概ねこれで問題がないが、
railsだと filter_parametersに emailを登録すればxxx_emailやemail_xxxもfilterしてくれる。
sentryは emailしかfilterしてくれないので注意。
sentryでもよしなにやってもらうためには/email/を登録する必要がある。

userのcontextを出す

誰がそのエラーを起こしたかをわかるようにするため。
https://docs.sentry.io/learn/context/#capturing-the-user

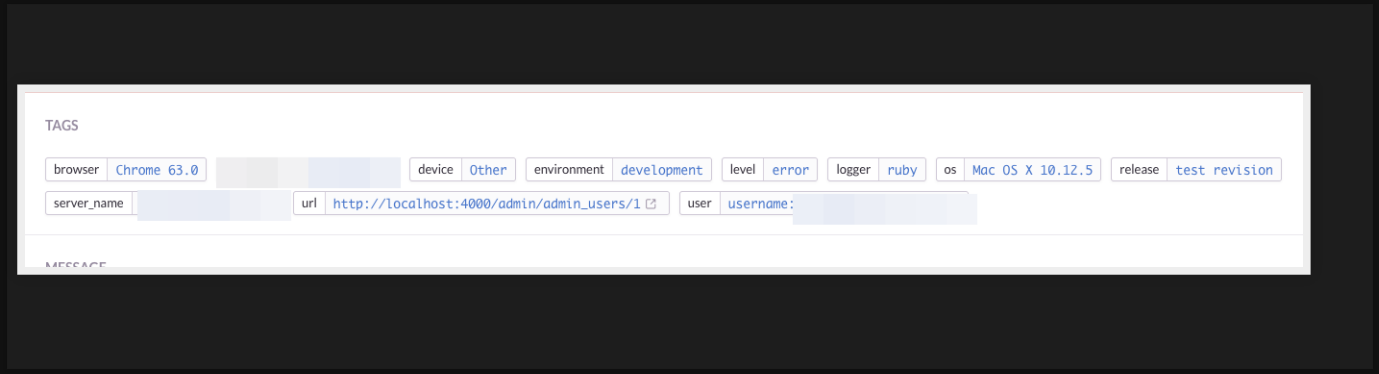
デフォルトではIPアドレスが取得されます。
設定する内容としては、

- 一般ユーザーの場合はそのユーザーを一意に識別できるIDなど
  - Raven.user_context(id: current_user.id) if current_user.present?
- 管理ツール等で社内の人間がエラーを起こした場合は、IDよりは名前やメールアドレスを設定したほうが解決が圧倒的にスムーズで良い
  - sanitizefieldにemailを追加している場合、user contextにemailを設定したときもfilterされてしまうので注意. usernameフィールドに入れる必要がある

tags contextを活用する

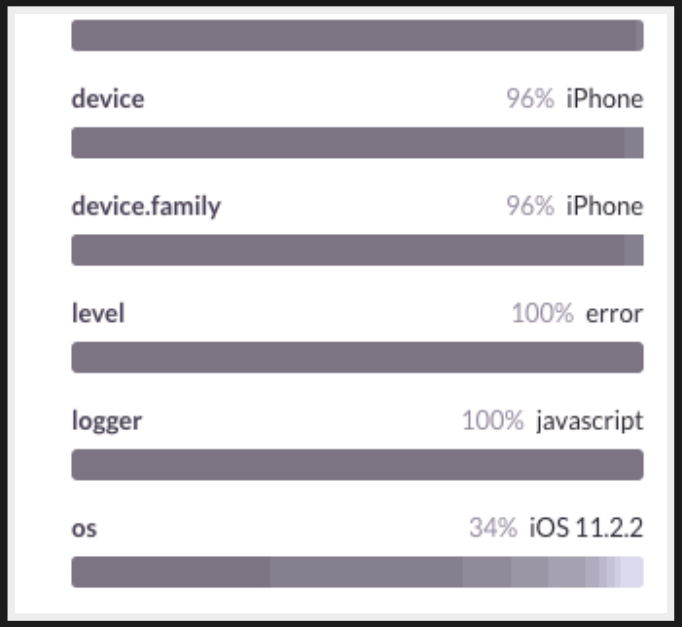
https://docs.sentry.io/learn/context/#tagging-events

タグはこんな感じでそのエラーがどういう環境で起きたかとかを見れる。



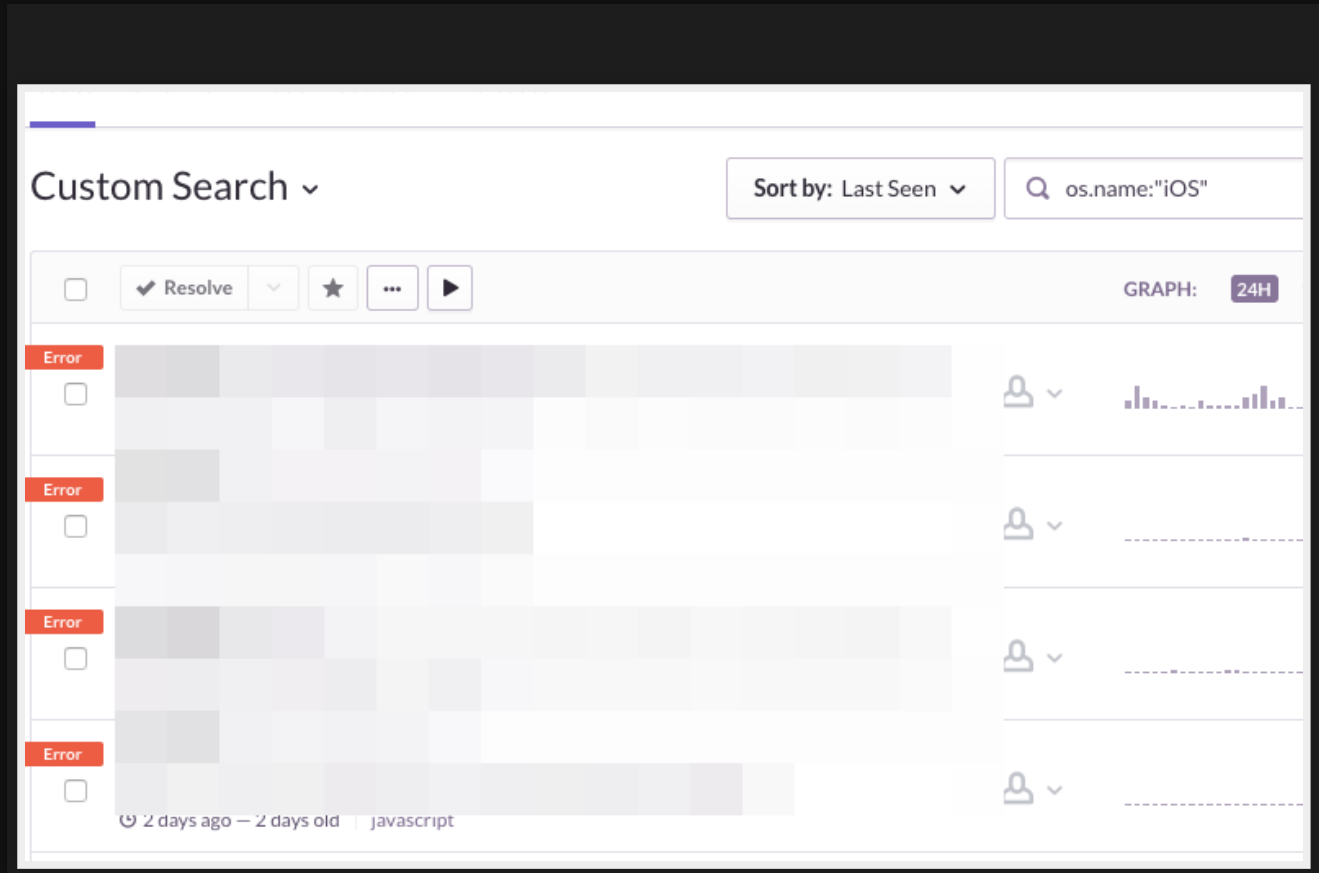
tags contextを活用する

そのエラーでどういう端末・サーバーで起きているのかを割合でみれる



tags contextを活用する

そのタグで起きているエラーを絞込できる



extra contextを活用する

そのエラーが起きた時に追加したい詳細な情報をなんでも詰め込める場所。
https://docs.sentry.io/learn/context/#extra-context

- request_id, paramsあたりを出しておくと良さ
  - Raven.extra_context(params: params.to_unsafe_h, url: request.url, request_id: request.request_id)

contextの種類の補足

https://docs.sentry.io/learn/context/
tags, user, extraの3種類ある。

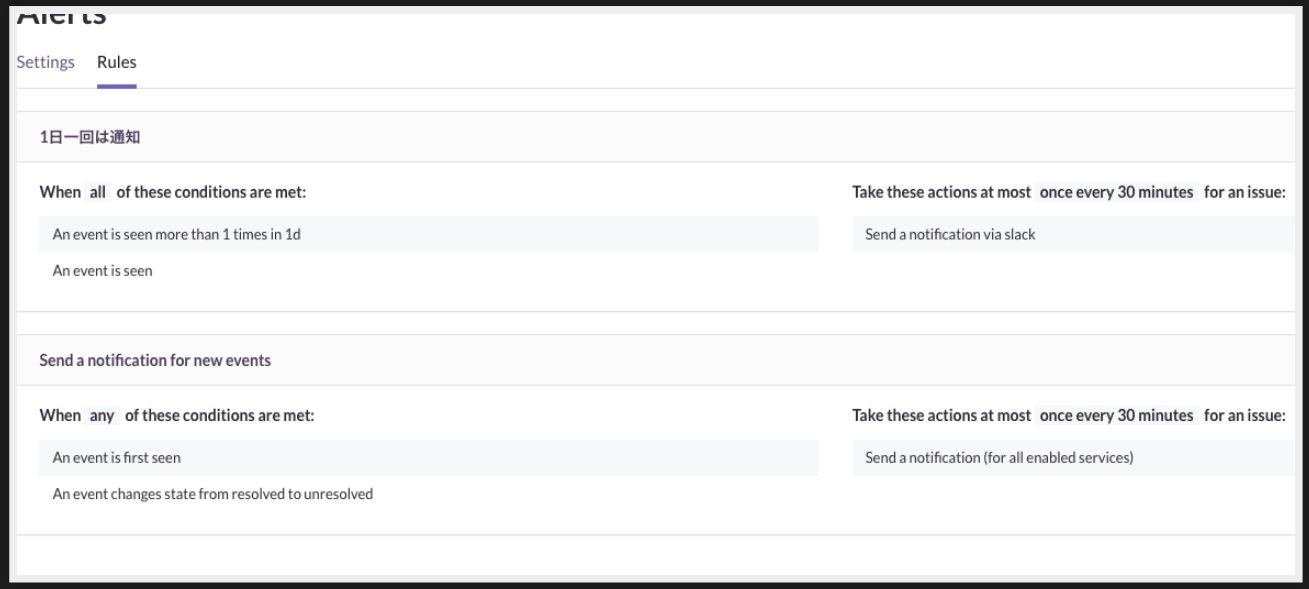
- tagsは、tagsでfilterしたりできる
- extraは、そのエラーに表示されるだけ。filterできたりしない。
  - リクエストparamsだしたり、http header出したりするのが良さそう
- user はtagsとextra両方の扱いになる

という認識。

通知ルールのカスタマイズ

https://your-sentry.example.com/sentry/your-project/settings/alerts/rules/

- デフォルトでは初めてそのエラーが発生した時に通知がされるだけ
- 通知ruleは複数設定できる
- 「指定時間内に指定回数起きたら」や「特定のタグがついていたら」などの条件が設定できる
- また、１つのruleの中に条件を複数書くことが出来る

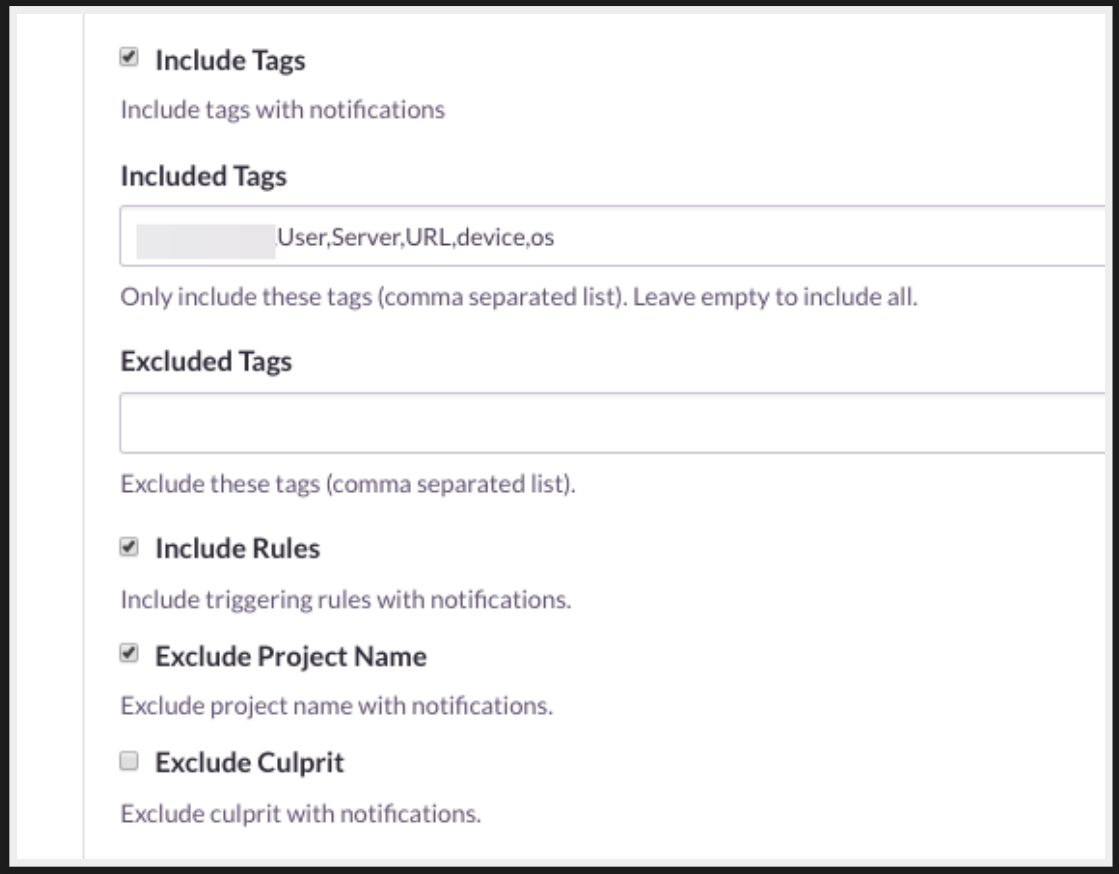


既知のエラーでも1日1回は通知するようにルールを設定する場合はこんな漢字。

通知内容のカスタマイズ

https://your-sentry.example.com/sentry/your-project/settings/alerts/

通知のintegration(slackなど)が設定されていると以下のような設定項目が出てくる



- include tagsは設定すべき
- incude rule は通知ruleが複数ある場合は設定したほうが良い
- ほかはよしなに。

auto resolveする

該当エラーが一定期間発生しなければresolve、という設定ができる。
https://your-sentry.example.com/sentry/your-project/settings/

古いエラーを溜め込んでしまっても割れ窓っぽくなってしまうので、
勇気を持って見て見ぬふりしていくのも大事そう(プロジェクトやチームの都合によりけり)

releaseを設定する

該当エラーがどのリリースで発生したのかを特定できるようにするため。
https://docs.sentry.io/learn/releases/
https://docs.sentry.io/clients/ruby/config/ のreleaseの項目

- githubリポジトリだったらそのSHA
- リポジトリルートにREVISIONファイルがあればその中身を見る
  - => capistranoでデプロイしてれば勝手に読んでくれる
- config.release = "xxx"
- apiを叩く

などで設定できる。
そのリリースでどういうエラーが起きたのか、次回リリースのタイミングでresolve、githubとの連携がうまく行ってればそのコミットのautherやgithubのページへのリンクなどいいことといっぱい

bashのエラーを検知する

https://docs.sentry.io/learn/cli/send-event/#bash-hook
出来るらしい