



# アツいIDE環境を生み出す New Relic CodeStream

New Relic, K.K.

Senior Solutions Consultant

Kota Saito ([ksaito@newrelic.com](mailto:ksaito@newrelic.com))

2/17/2022

ソフトウェア開発  
楽しんでますか？

“自分の環境や、自分の道具や、自分の専門分野を  
好きになれなければ、輝かしいことはできない”

*David Heinemeier Hansson*

出典:情熱プログラマー(本書に寄せて)

CODESTREAM

michael ▾

OBSERVABILITY (Preview)   

Errors assigned to me

 com.newrelic.distributedtracingservice.api.model.APIException

Recent errors in distributed\_tracing\_service distributed\_tracing\_se...

 com.newrelic.distributedtracingservice.api.model.APIExc... 47m org.apache.http.client.ClientProtocolException (1) 6h java.util.concurrent.CompletionException (1) 6h java.util.concurrent.CompletionException (1) 11h org.apache.http.client.ClientProtocolException (1) 1dShow All Commands   PGo to File  PFind in Files   FStart Debugging  F5Toggle Terminal  

&gt; FEEDBACK REQUESTS

&gt; CODEMARKS

&gt; ISSUES



# ここがアツいよ！ CodeStream

ここがアツい！① **IDEですべてが完結！**

# ここがアツいよ！ CodeStream

ここがアツい！① **IDEですべてが完結！**

ここがアツい！② **流れるようにコードで会話！**

# ここがアツいよ！ CodeStream

ここがアツい！ ① IDEですべてが完結！

ここがアツい！ ② 流れるようにコードで会話！

ここがアツい！ ③ アプリの問題コードがすぐわかる！

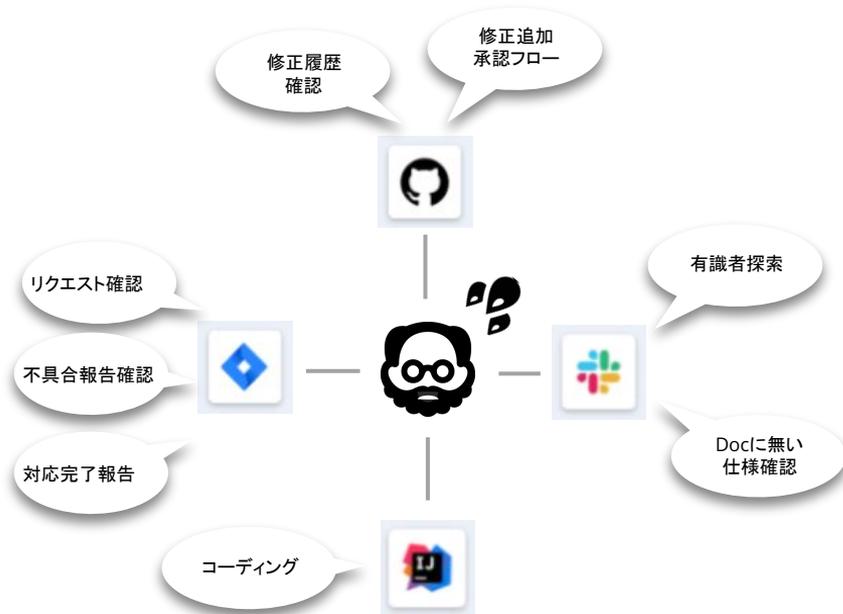
# ここがアツいよ！ CodeStream

ここがアツい！① **IDEですべてが完結！**

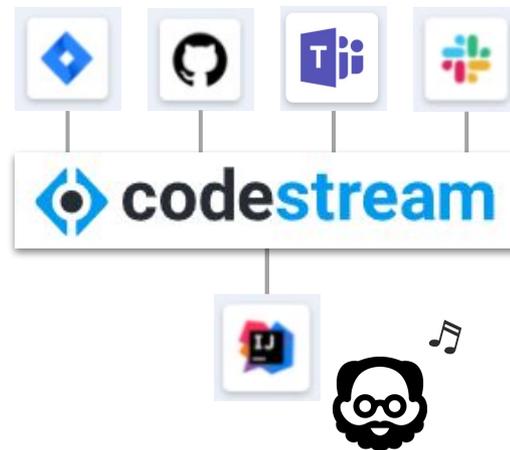
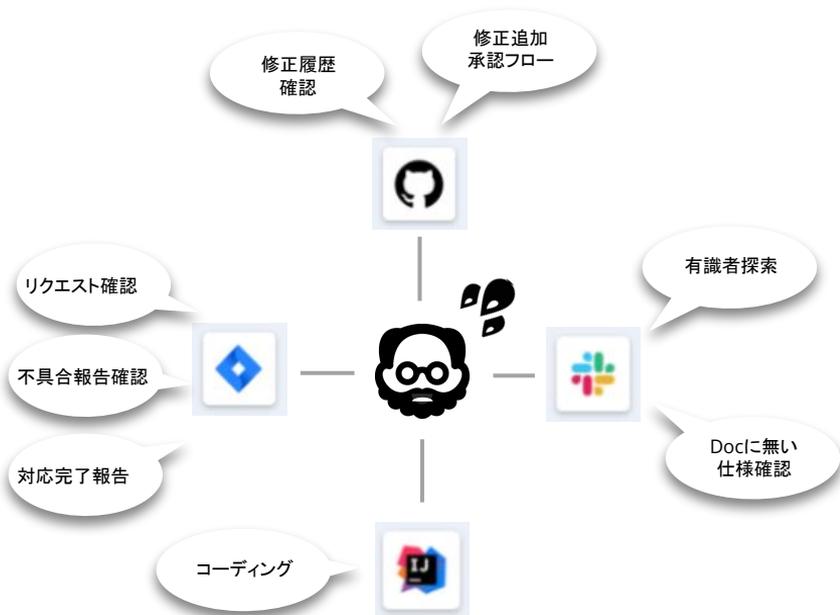
ここがアツい！② **流れるようにコードで会話！**

ここがアツい！③ **アプリの問題コードがすぐわかる！**

# さまざまな便利ツールを駆使した開発...



# 開発環境に全てを統合！



# 多様な開発環境にフィット！

## IDE



VS Code



VS Code  
Insiders



Visual  
Studio



Android  
Studio



IntelliJ  
IDEA



PyCharm



WebStorm



PhpStorm



RubyMine



Rider



CLion



GoLand



DataGrip



AppCode

## Issue管理



GitHub



GitLab



JIRA



Azure  
DevOps



Bitbucket



asana

Asana



Linear



Trello



YouTrack

## ソース管理



GitHub



GitLab



Bitbucket

## メッセージング



Slack



Teams

# タスクの管理からコードの マージまで一連の作業 がIDE上で完結

- GitHubやJIRAなどと連携してIDEからIssueやTaskの作成と管理が可能
- 修正ブランチも自動作成し、仕掛かり中のタスクとブランチを自動連動
- PR作成、レビュー、マージもIDE内で完結
- コンテキストを維持しているのでIssueやブランチとも自動で関連付け

エラーチェックルールの外だし **Issue作成**

エラーチェックがハードコードされているので変更に弱い。ポリシー外出しにしておく。

[complex-app-settings] app\_settings/app/controllers/settings\_controller.rb (Lines 13-14)

```
01.
02.     if @setting.key.start_with?('error')
03.         error_loop1("Intended Error: " + @setting.key)
```

<sup>Created from VS Code using [CodeStream](#)</sup>

Set up a branch in `complex-app-settings`

🔄 master  
① 2 commits behind origin **Pull**  
feature/エラーチェックルールの外だし

Update my **Open a Pull Request** **PR作成**

Choose two branches to start a new pull request.



repo: complex-app-settings base: master

← compare: feature/エラーチェックルールの外だし

エラーチェックルールの外だし

もともとハードコードされていた入力チェックのルールを外出しにして、変更耐えられるようにする。

Set upstream to origin/feature/エラーチェックルールの外だし

This PR addresses: 🔄 エラーチェックルールの外だし

Cancel **Create Pull Request**

エラーチェックルールの外だし #19 **PRレビュー、マージ**

**Open**

**ktst79** wants to merge 1 commits into `complex-app-settings:master` from `feature/エラーチェックルールの外だし` 4 minutes ago

🗨️ 0 🔄 1 ± 1 +1 -0

もともとハードコードされていた入力チェックのルールを外出しにして、変更耐えられるようにする。

**This PR Addresses:**

[エラーチェックルールの外だし](#)

Created from VS Code using [CodeStream](#)

🔄 **Put error check outside the code** 6e152c0

Add more commits by pushing to the `feature/エラーチェックルールの外だし` branch on `ktst79/complex-app-settings`.

🟡 Some checks haven't completed yet [Show all checks](#)

🟢 This branch has no conflicts with the base branch

Merging can be performed automatically.

**Squash and merge**



# ここがアツいよ！ CodeStream

ここがアツい！① IDEですべてが完結！

ここがアツい！② 流れるようにコードで会話！

ここがアツい！③ アプリの問題コードがすぐわかる！

# コードに関する非効率なやりとり...

```
rails.application.config do
  # Settings specified here will take precedence over those in config/initializers/application_controller_defaults.rb

  # To see what configuration options are available, see the Rails Configuration Reference documentation at https://guides.rubyonrails.org/configuring.html
  # In the development environment your application's code is reloaded on every request. This slows down response time but is perfect for development. Since you may not have a web server in production, you can leave config.action_controller.enable_fragments_cache_logging = true
end

# Do not use Rails cache in production
config.cache_store = :memory_store

# See https://guides.rubyonrails.org/action_controller_overview.html for more options
config.action_controller.perform_caching = true

# Do not use Rails cache in production
config.cache_store = :memory_store

# See https://guides.rubyonrails.org/action_controller_overview.html for more options
config.action_controller.perform_caching = true

# Do not use Rails cache in production
config.cache_store = :memory_store

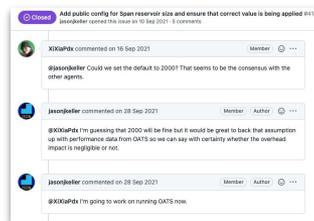
# See https://guides.rubyonrails.org/action_controller_overview.html for more options
config.action_controller.perform_caching = true
```



## チャットで聞いて回る



## Issue/PRを漁る



## Web会議でレビュー

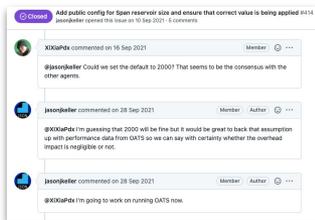


# IDE上でコードについて直接会話！

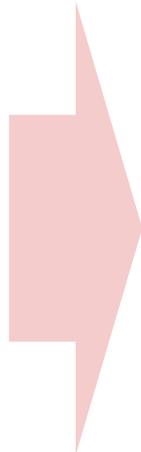
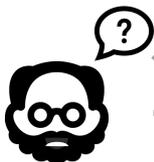
## チャットで聞いて回る



## Issue/PRを漁る



## Web会議でレビュー



```
!!-application.config.yml
# Settings specified here will take precedence over those in config/...

# In the development environment your application's code is reloaded
# every request. This should never be enabled in production.
# Don't care if the server has to restart when you make code changes:
config.cache_classes = false

# Do not eager load code on boot.
config.eager_load = false

# Show full error reports.
config.consider_all_requests_local = true

# Enable/disable caching. By default caching is disabled.
# Full rails dev cache to toggle caching.
# Full rails dev cache to toggle caching.
config.action_controller.perform_caching = true
config.action_controller.enable_fragments_cache_logging = true

config.cache_store = :memory_store
config.public_file_server.enabled = true
  { 'Cache-Control' => 'public, max-age=(2.days.to_s)' }
}
else
  config.action_controller.perform_caching = false
end
config.cache_store = :null_store
end

# Show bundled files on the local file system (use config/storage.yml
config.action_mailer.raise_delivery_errors = false
```



```
!!-application.config.yml
# Settings specified here will take precedence over those in config/...

# In the development environment your application's code is reloaded
# every request. This should never be enabled in production.
# Don't care if the server has to restart when you make code changes:
config.cache_classes = false

# Do not eager load code on boot.
config.eager_load = false

# Show full error reports.
config.consider_all_requests_local = true

# Enable/disable caching. By default caching is disabled.
# Full rails dev cache to toggle caching.
# Full rails dev cache to toggle caching.
config.action_controller.perform_caching = true
config.action_controller.enable_fragments_cache_logging = true

config.cache_store = :memory_store
config.public_file_server.enabled = true
  { 'Cache-Control' => 'public, max-age=(2.days.to_s)' }
}
else
  config.action_controller.perform_caching = false
end
config.cache_store = :null_store
end

# Show bundled files on the local file system (use config/storage.yml
config.action_mailer.raise_delivery_errors = false
```



## コードに関するやりとりをIDE上で完結

- コード上で直接ディスカッションを開始して会話が可能
- 修正者を探さなくても自動でメンション(もちろん他の人も可)
- 会話に関するコードを探すことなく、コメントから自動表示
- PRやIssueがクローズされてもコードと関連づいて残るので過去の経緯の把握が可能
- SlackやTeamsなどのコミュニケーションツールにも自動で同期

```

app_settings > app > controllers > settings_controller.rb
1  class SettingsController < ApplicationController
2    def index
3      @settings = Setting.all
4    end
5
6    def new
7      @setting = Setting.new
8    end
9
10   def create
11     @setting = Setting.new(setting_params)
12
13     // ktst79: @jWick ここにあるエラーハンドリングってどういう意図ですか?
14     if @setting.key.start_with?('error')
15       error_loop1("Intended Error: " + @setting.key)
16     elsif @setting.key == 'slow'
17       #Some comment...
18       100.times do
19         dummy = Setting.all
20       end
21     end
22     if @setting.save
23       redirect_to settings_path
24     else
25       render 'new'
26     end
27   end
28
29   def edit
30     @setting = Setting.find(params[:id])
31   end
32
33   def show
34     @setting = Setting.find(params[:id])
35   end
36
37   def update
38     @setting = Setting.find(params[:id])
39
40     if @setting.key == 'error'
41       error_loop1("Intended Error")

```

CHAT WINDOW:

ktst79 25 minutes ago  
 @jWick ここにあるエラーハンドリングってどういう意図ですか?

SHARED TO  
 #nr\_cs\_ktst

complex-app-setitngs  
 app\_settings/app/controllers/settings\_controller.rb  
 master 74c6655

13. if @setting.key.start\_with?('error')

14. error\_loop1("Intended Error: " + @setting.key)

ACTIVITY

jWick 24 minutes ago  
 @ktst79 入力フィールドの不正チェックです。

ktst79 23 minutes ago  
 @jWick ハードコードになっちゃっていて仕様変更に対応できないのですが、外だししていない理由とありますか？

jWick 22 minutes ago  
 @ktst79 いや、待たないです。考慮漏れなので直してもらえると助かります。

ktst79 22 minutes ago  
 @jWick ラジャー

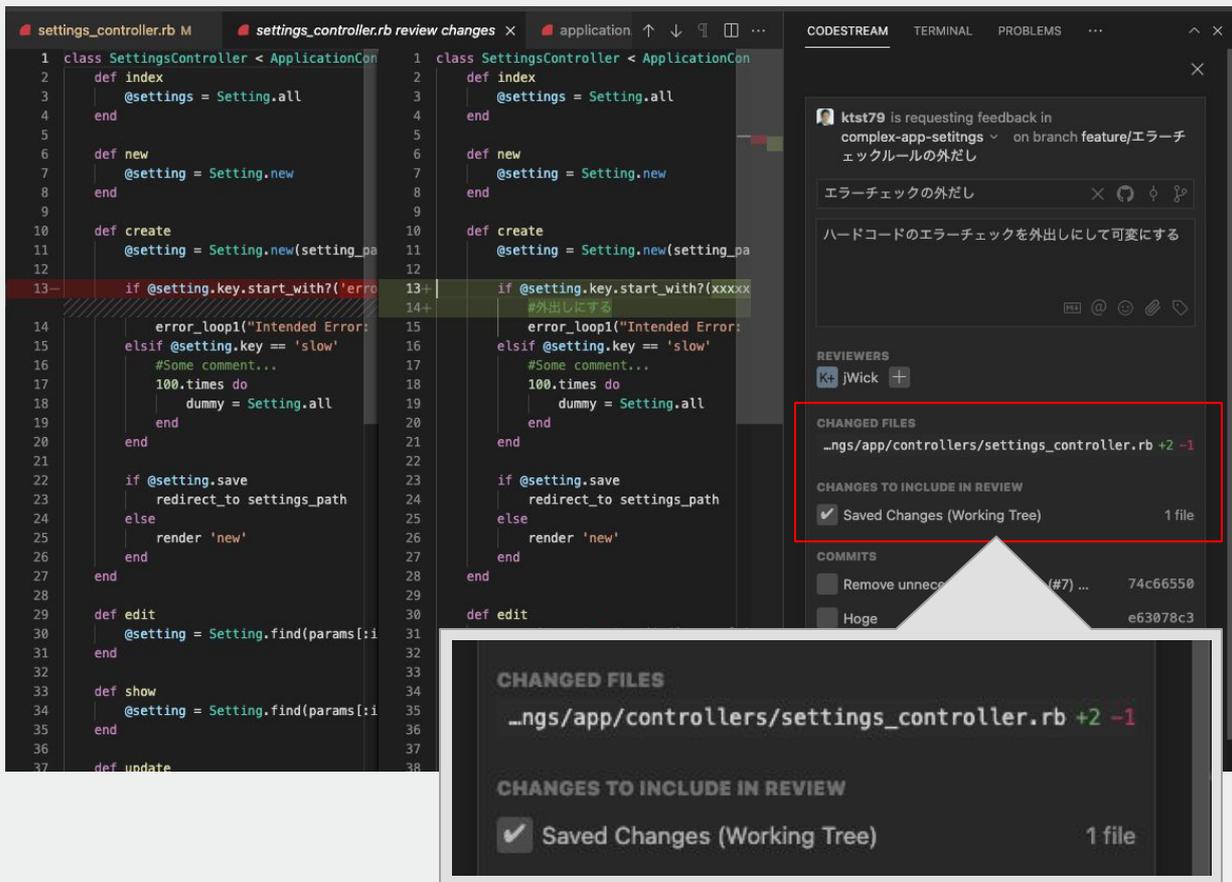
ADD REPLY

Reply...

Resolve & Archive Discussion Comment

# 未コミットの修正の レビューによる レビューの軽量化

- コミット前のローカルの修正のレビューが可能(もちろんコミット済みも)
- レビュープロセスの軽量化と前倒しによる、後フェーズでの手戻り抑止とレビューアの負荷分散



# ここがアツいよ！ CodeStream

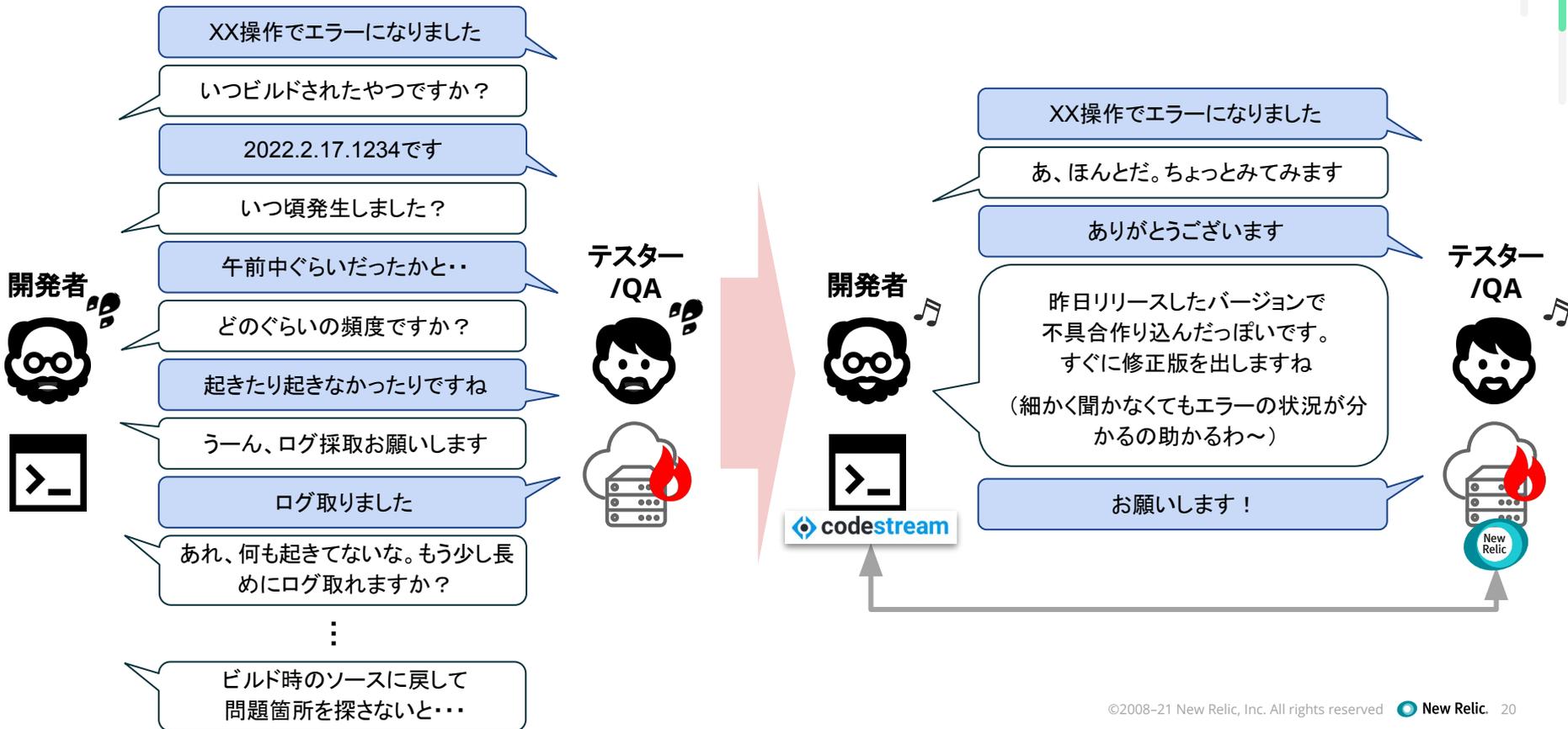
ここがアツい！ ① IDEですべてが完結！

ここがアツい！ ② 流れるようにコードで会話！

ここがアツい！ ③ アプリの問題コードがすぐわかる！



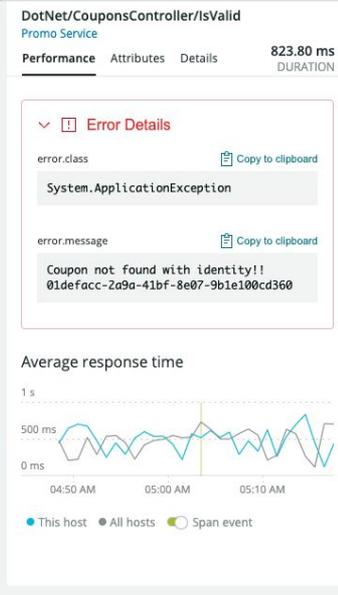
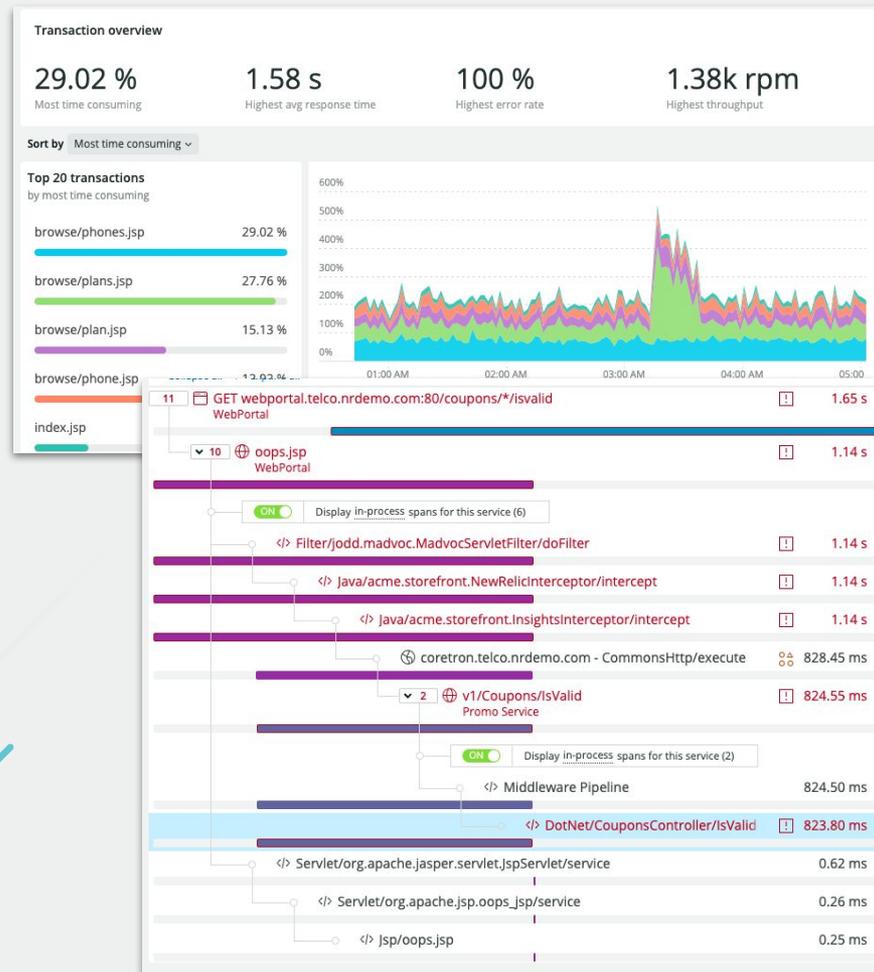
# IDEで問題コードを速攻で特定！





# OBSERVABILITY PLATFORM

- ユーザー体験からインフラまでシステムの全容をリアルタイムに把握できる**オブザーバビリティ**(可観測性)を全てのステークホルダーに提供
- アプリケーションのトランザクション処理を観測し、ボトルネックやエラーをソースコードやDBクエリの詳細度で即座に特定



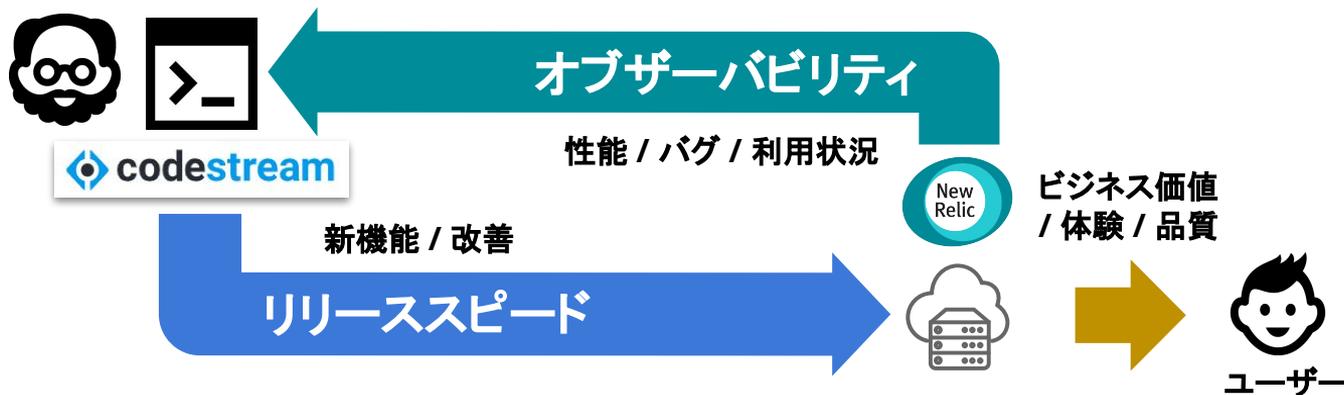
# システムのエラーから該当のソースコードへダイレクトに到達

- 問題の詳細をすぐに把握。トラブルシュートに必要な資料採取やQA/Opsチームとのコミュニケーションを効率化
- 問題が発生しているコードを(リビジョン込みで)自動的に特定してIDEに表示。再現環境を作る手間をかけずに問題発生箇所を把握

The screenshot displays the CodeStream interface for a runtime error. On the left, the error details for 'Intended Error: error\_20220203' are shown, including the time it was first and last seen (8 minutes ago) and associated account information. A graph shows the error count over time, with a peak at 03:50 PM. On the right, the stack trace is visible, with a red box highlighting the 'Open in IDE' button next to the relevant code frame. Below this, the IDE window shows the source code for 'settings\_controller.rb' with the error location highlighted.

# 開発者にオブザーバビリティを

スピーディかつ高品質に顧客価値を提供し続けるために、  
開発者自身がフィードバックを得て改善できるスキルが重要に



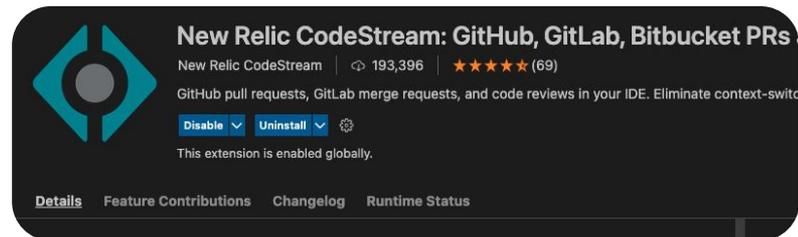
オブザーバビリティと改善の効率性によって次世代の開発者スキルを獲得

# まとめ

- **ここがアツいよ！ CodeStream**
  - IDEですべてが完結！
  - 流れるようにコードで会話！
  - アプリの問題コードがすぐわかる！
- **開発者にオブザーバビリティを**
  - 開発者自身がフィードバック(アプリの品質、バグ、利用状況)を自ら理解して改善するスキルを身につけることが重要に
  - アプリケーションの迅速、かつ継続的改善のために必要な、オブザーバビリティと効率性をCodeStream / New Relicが提供します

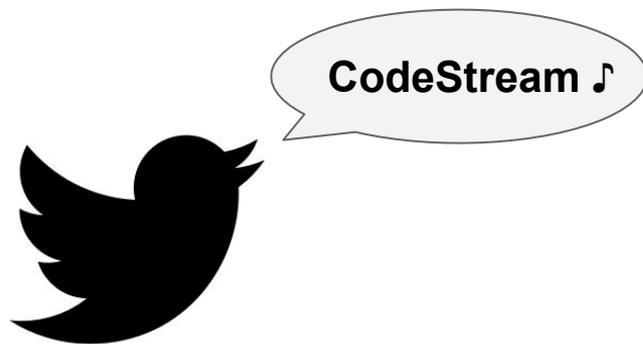
# Next Step

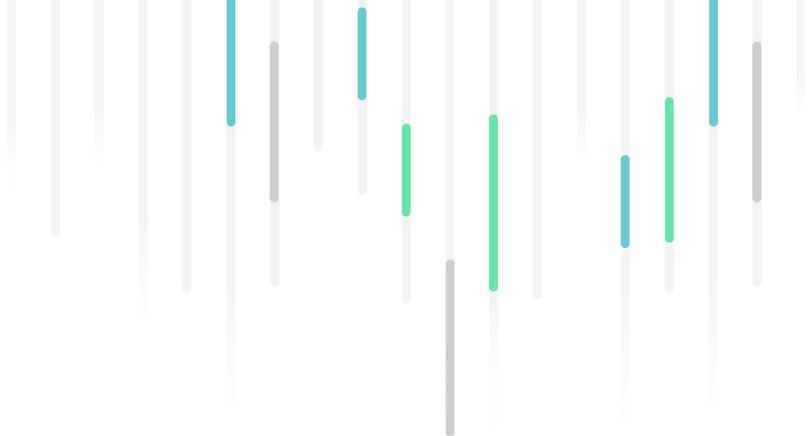
- IDEにCodeStream Pluginを入れてみましょう



- New Relicアカウントセットアップしてアプリを観測しましょう







# Thank You