New Relic | NEWS PICKS

NewsPicks の プロダクト開発エンジニアが実践するスキルとしての SRE

©2008–21 New Relic, Inc. All rights reserved

開発者のための
オブザーバビリティ

Tsuyoshi Shimizu

Sr. Solutions Consultant
New Relic K.K.

NewsPicks のプロダクト開発エンジニアが実践するスキルとしての SRE

Yukako Iida

Product division Web Product Unit
NewsPicks, Inc

©2008–21 New Relic, Inc. All rights reserved

New Relic

開発者のためのオブザーバービリティ

Tsuyoshi Shimizu
Senior Solutions Consultant
New Relic K.K.

©2008–21 New Relic, Inc. All rights reserved

Tsuyoshi Shimizu

Package Vender

- CRM Java **Developer**
- E-Commerce **Infra Architect**
- E-Commerce **SaaS SRE**

Public Cloud

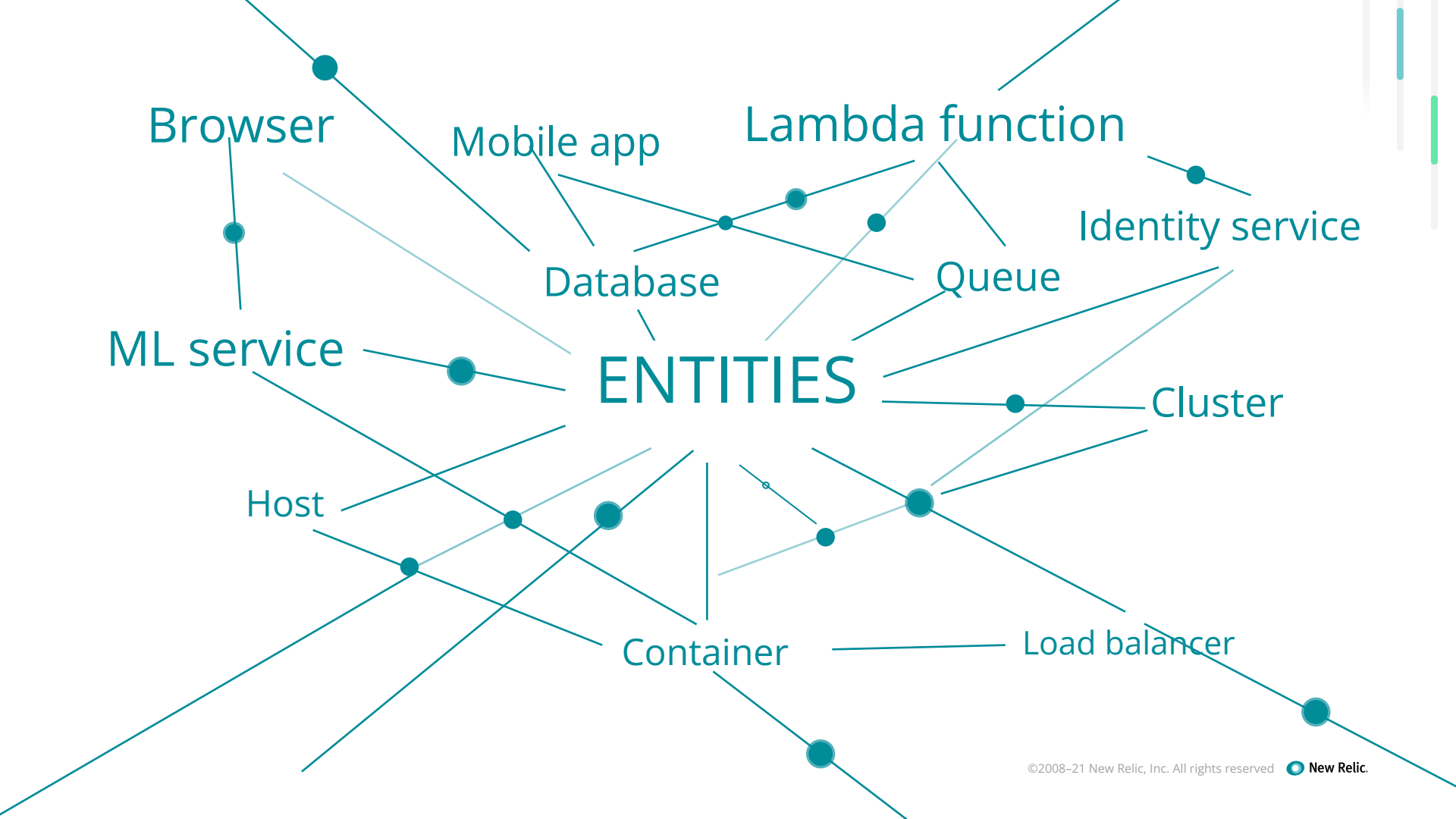
- SaaS **Solutions Architect**

New Relic

- Senior **Solutions Consultant**

©2008–20 New Relic, Inc. All rights reserved

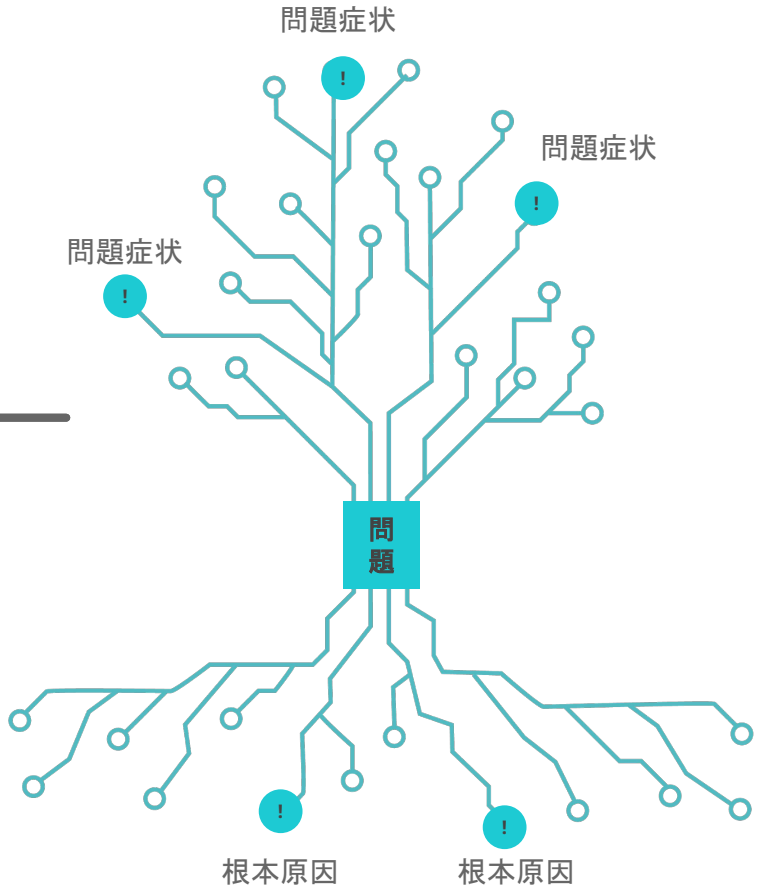
More Perfect Software.



©2008–21 New Relic, Inc. All rights reserved New Relic.

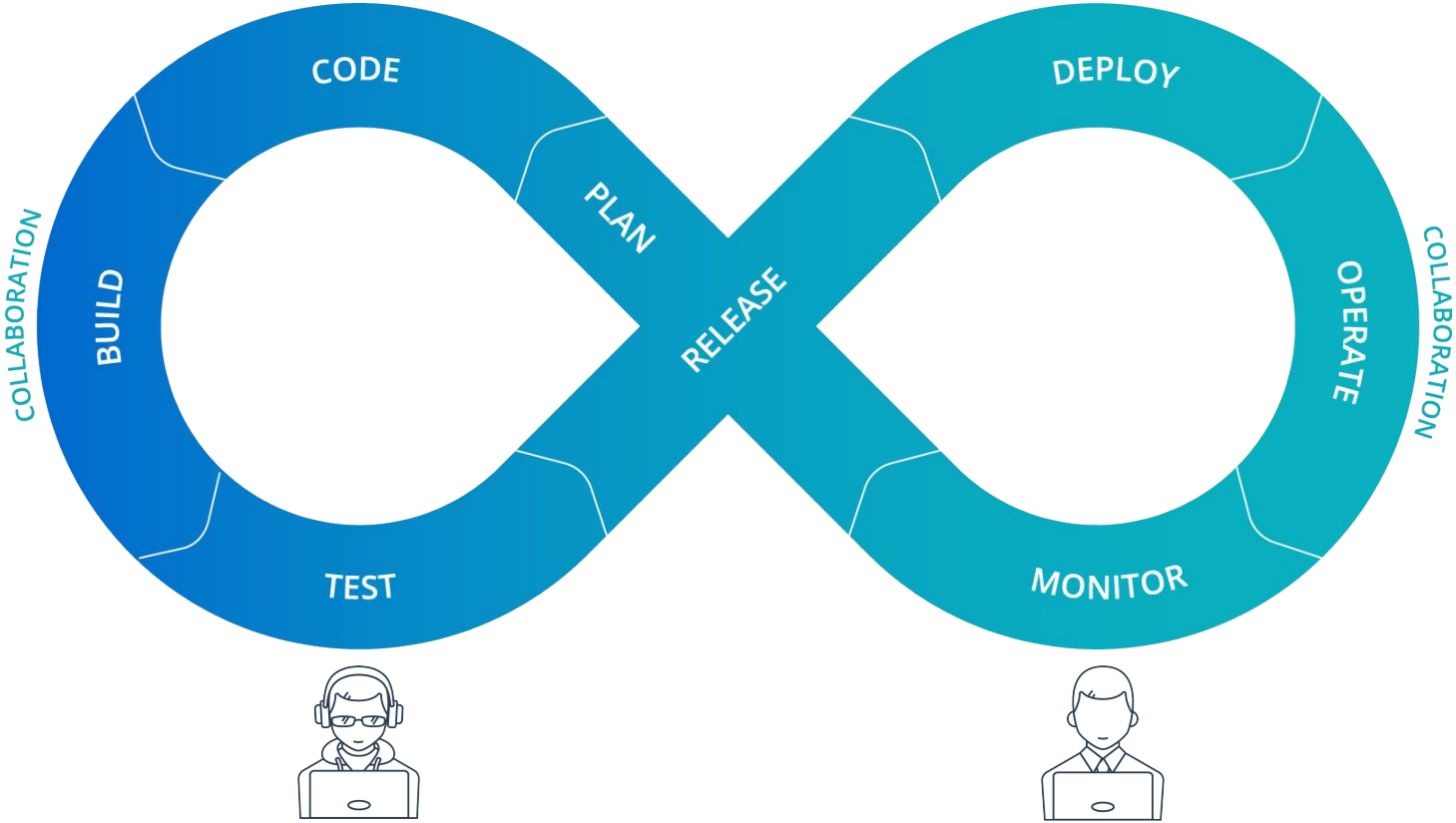
オブザーバビリティ
Observability

システムのメトリクス・イベント・
ログ・トレースのデータを
リアルタイムに取得し続け、
常にシステム全容の
状態把握と改善ができる状態



©2008–21 New Relic, Inc. All rights reserved

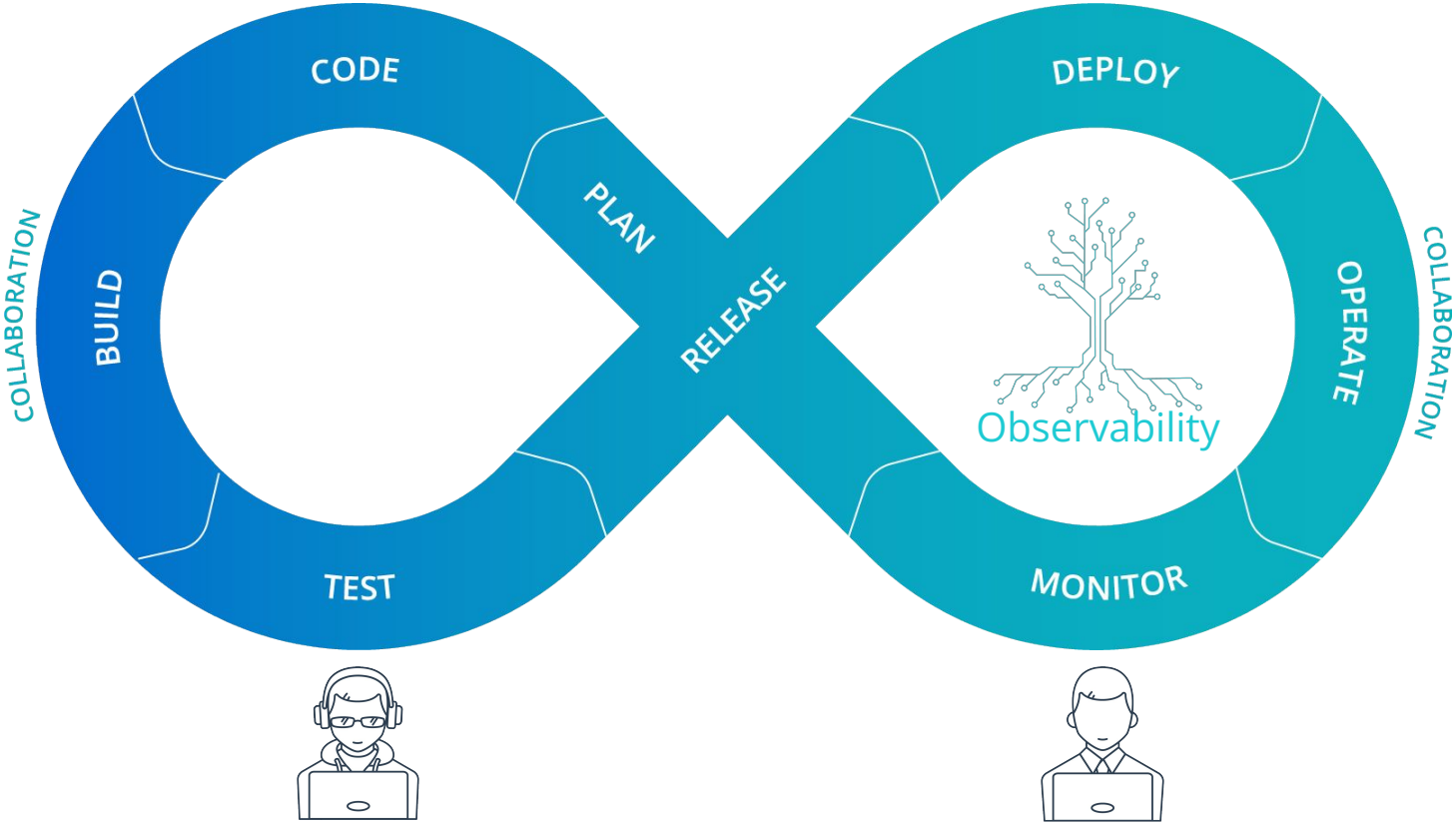
開発者と運用/SREのソフトウェアライフサイクル



開発　　運用/SRE

©2008–21 New Relic, Inc. All rights reserved

SREのオブザーバビリティ

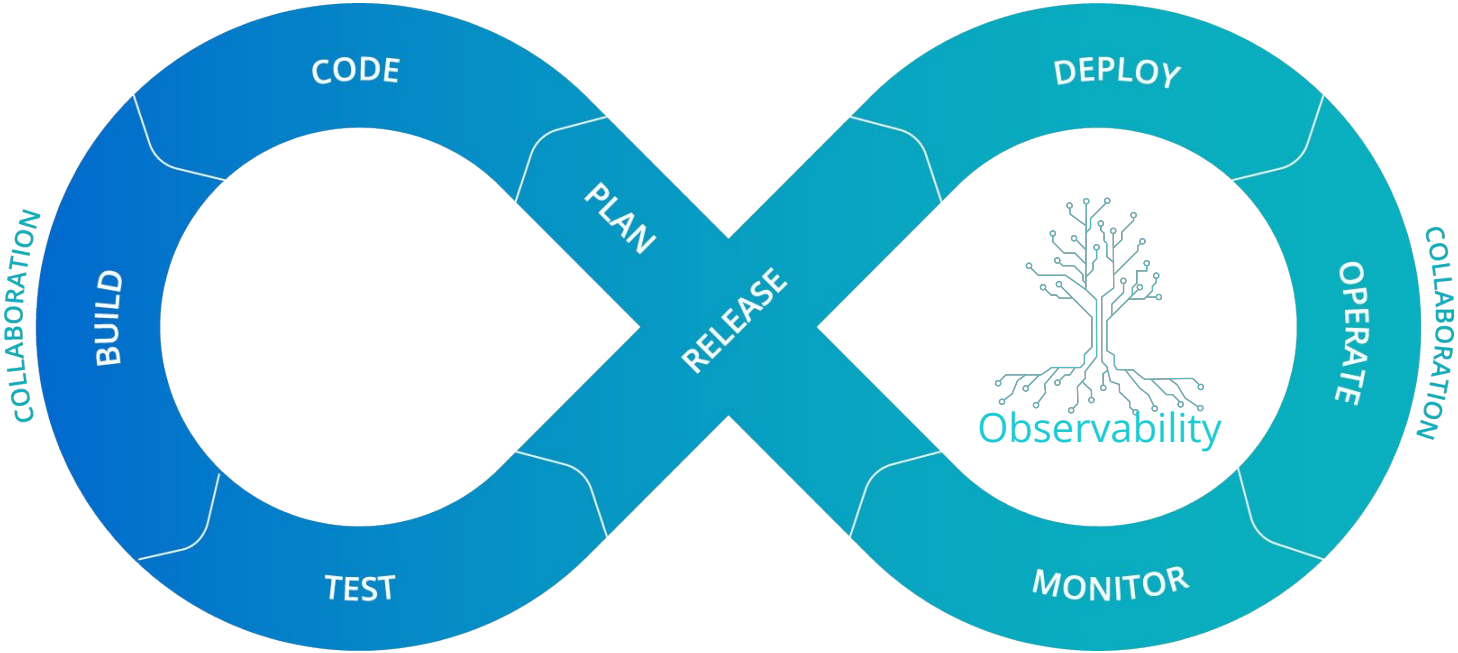


開発

運用/SRE

©2008–21 New Relic, Inc. All rights reserved

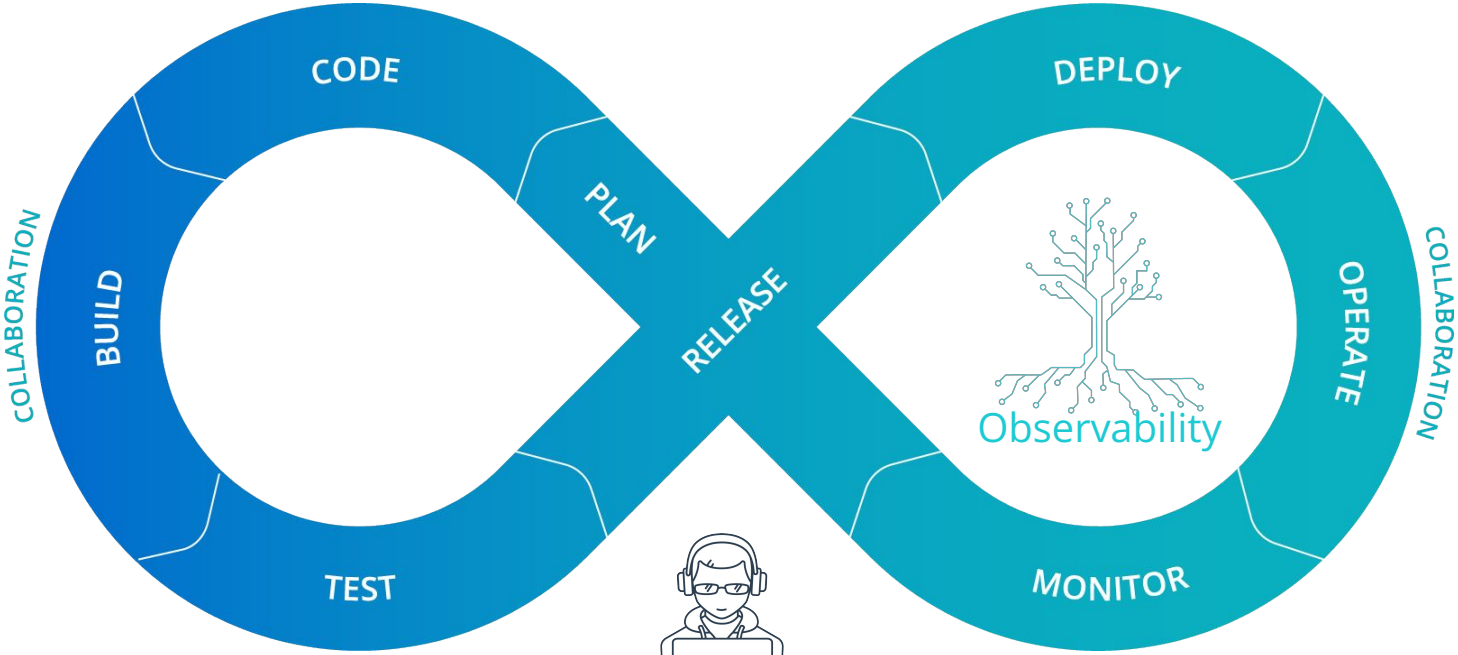
開発者とSREのコラボレーション



開発　運用/SRE

©2008–21 New Relic, Inc. All rights reserved

開発者のためのオブザーバビリティ①運用段階

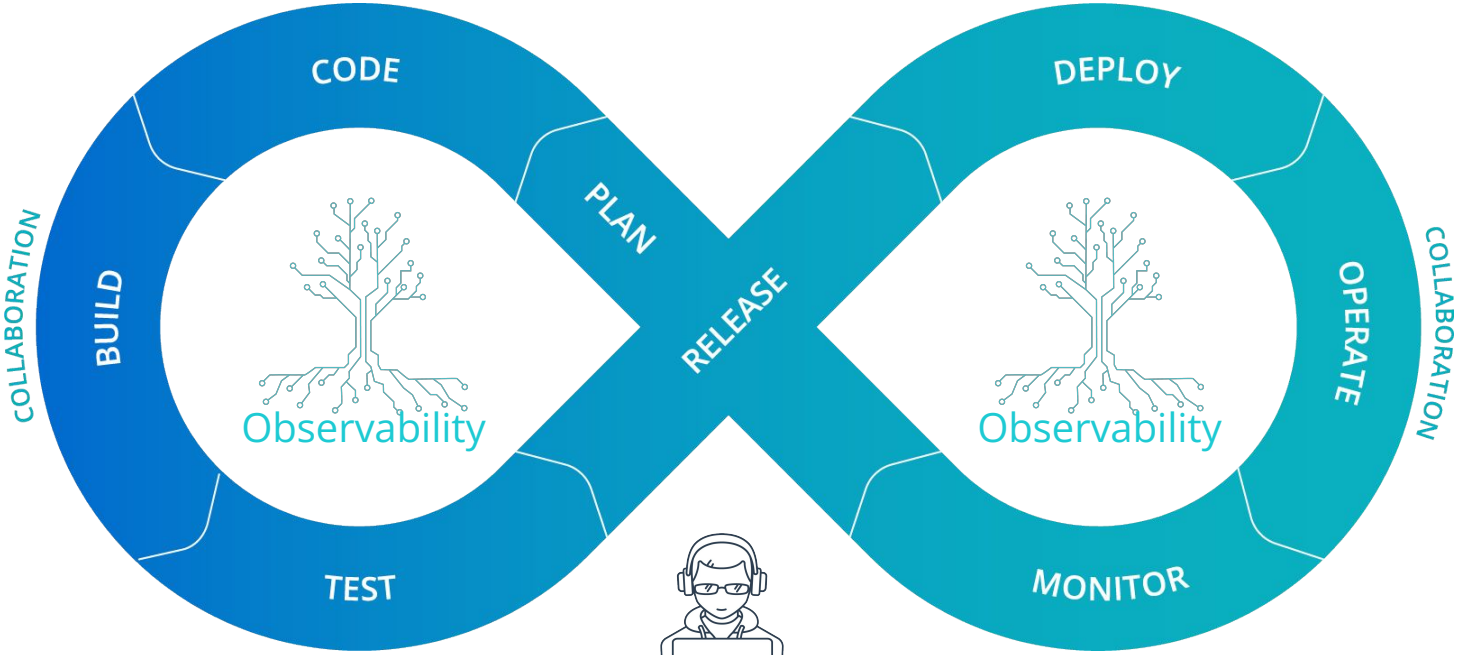


開発

運用/SRE

©2008–21 New Relic, Inc. All rights reserved

開発者のためのオブザーバビリティ②開発段階

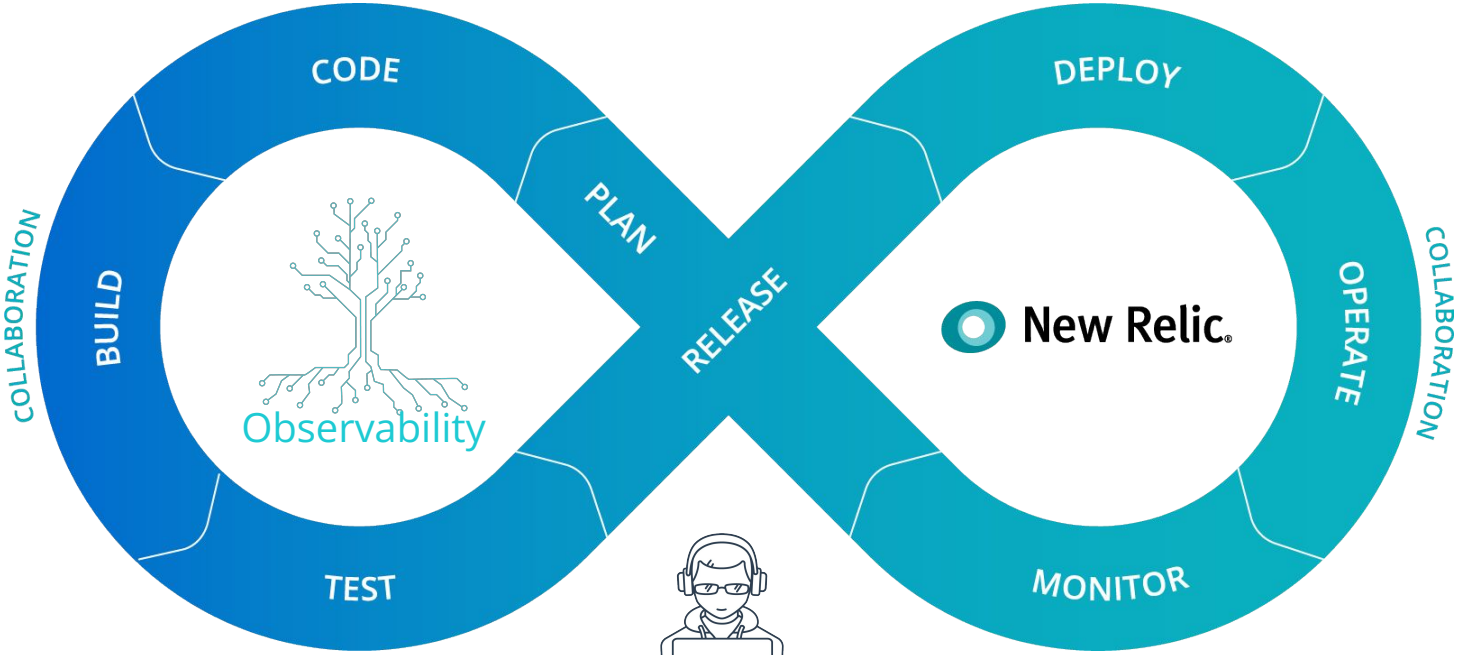


開発

運用/SRE

©2008–21 New Relic, Inc. All rights reserved

運用フェーズにNew Relicの活用



開発

運用/SRE

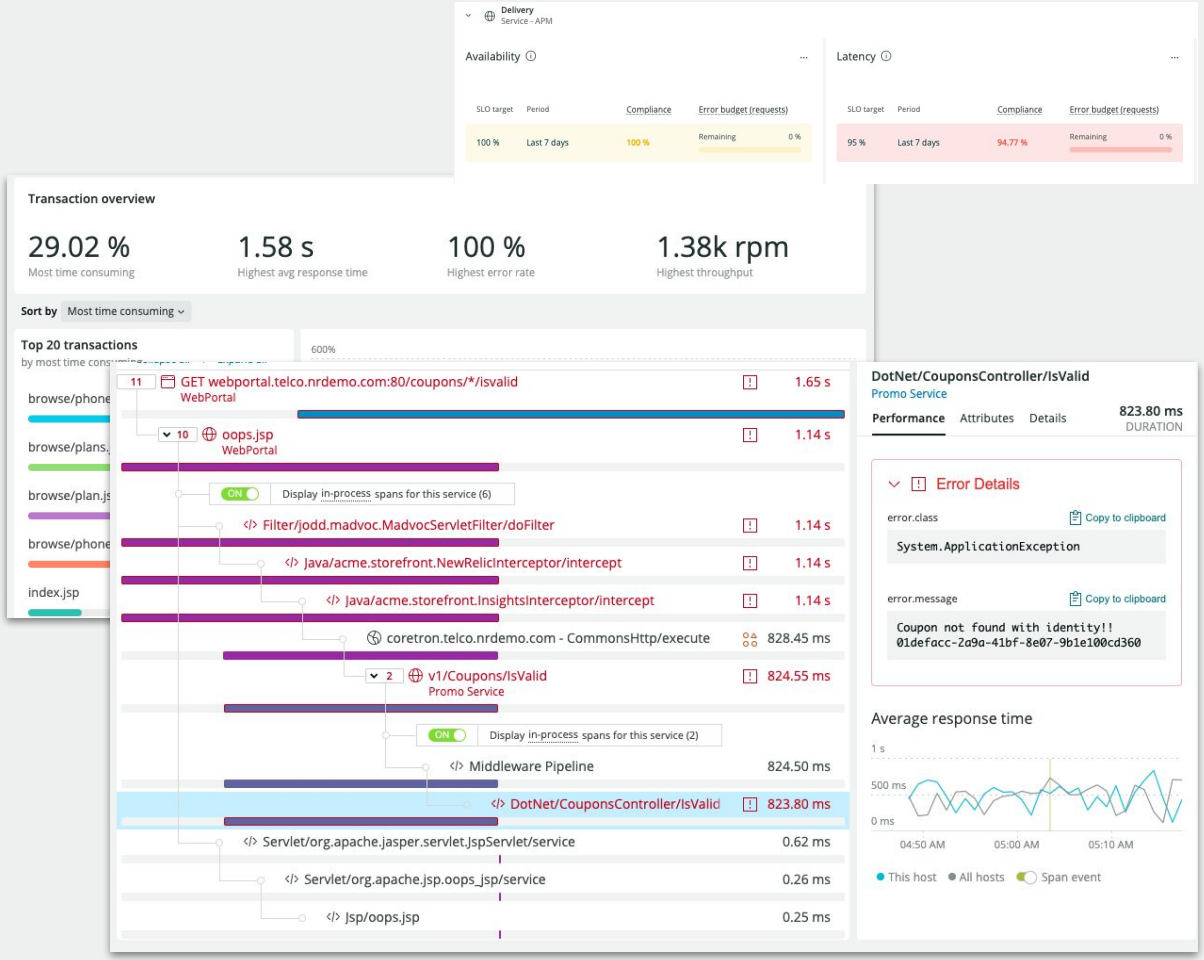
©2008–21 New Relic, Inc. All rights reserved

New Relic®

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**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



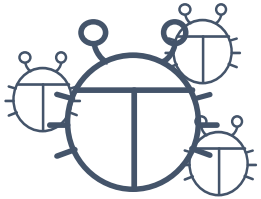
©2008–21 New Relic, Inc. All rights reserved

開発者の課題

ワークフローの分断

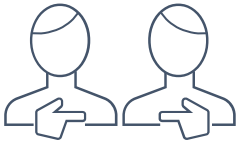
チーム間のワークフローの分断やツールのサイロは、重要なエラーや問題に対するMTTRを遅くすることにつながります

デバッグの難しさ



ローカルでのバグの再現は複雑で、どのバグを優先させるかを知るのは大変です

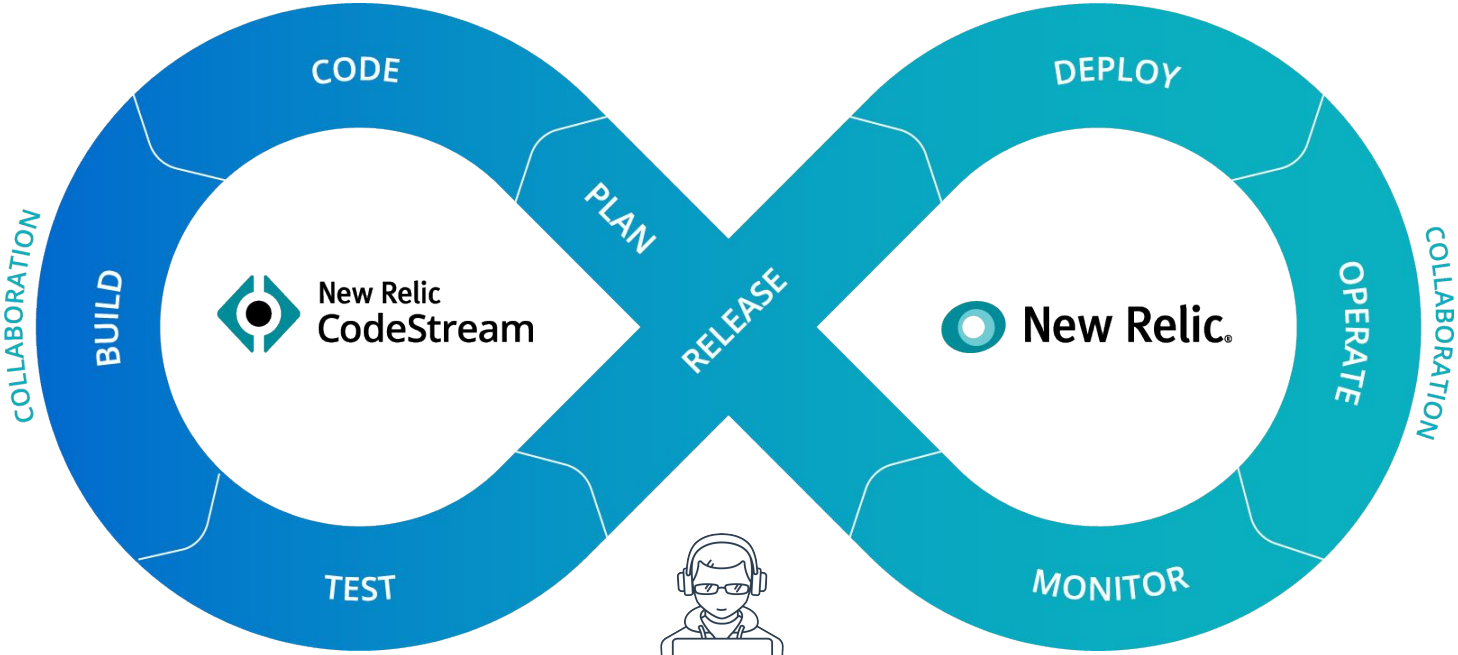
コラボレーションの課題



散発的で刹那的なコミュニケーション方法は、意思決定とリリース速度を遅くします。

©2008–20 New Relic, Inc. All rights reserved

開発者のためのオブザーバビリティ③CodeStream



©2008–21 New Relic, Inc. All rights reserved

多様な開発環境にフィット！

New Relic CodeStream

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


Linear


Trello


YouTrack

ソース管理


GitHub


GitLab

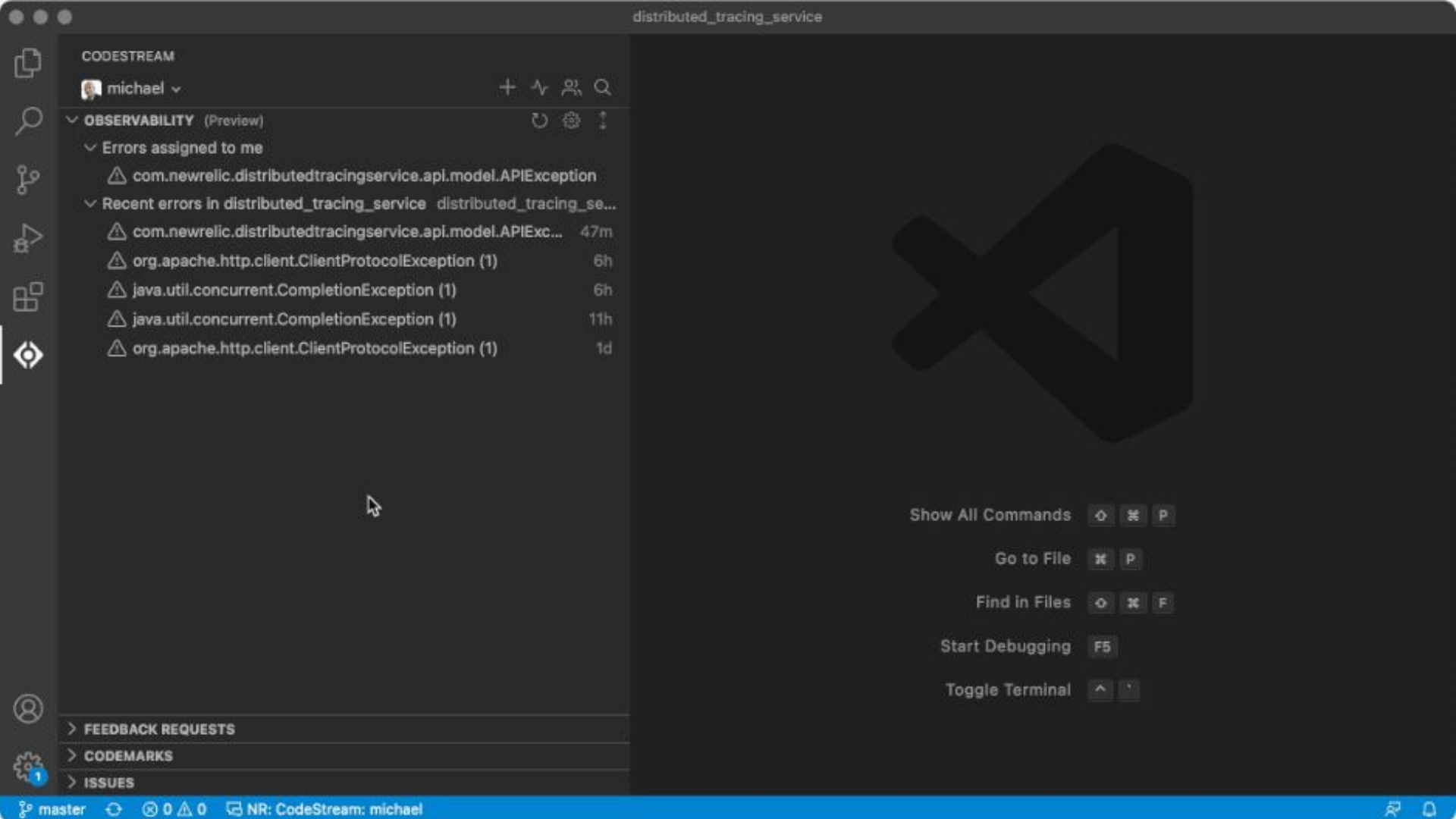

Bitbucket

メッセージング


Slack

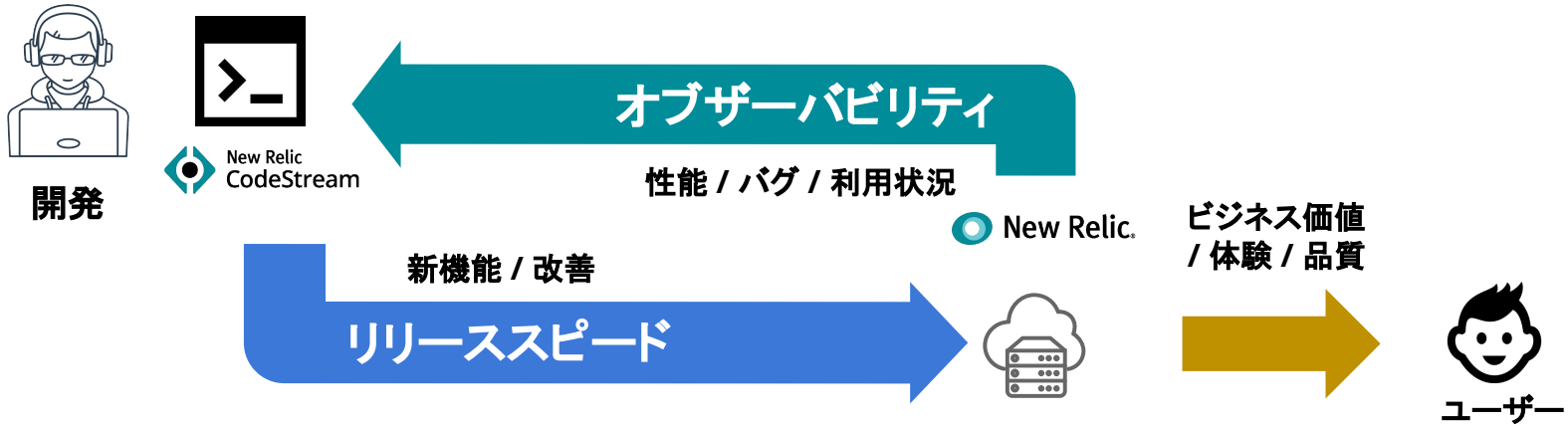

Teams

©2008–21 New Relic, Inc. All rights reserved New Relic.


distributed_tracing_service
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) 1d
Show All Commands
Go to File
Find in Files
Start Debugging
Toggle Terminal
FEEDBACK REQUESTS
CODEMARKS
ISSUES
master
NR: CodeStream: michael

開発者のためのオブザーバビリティ

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



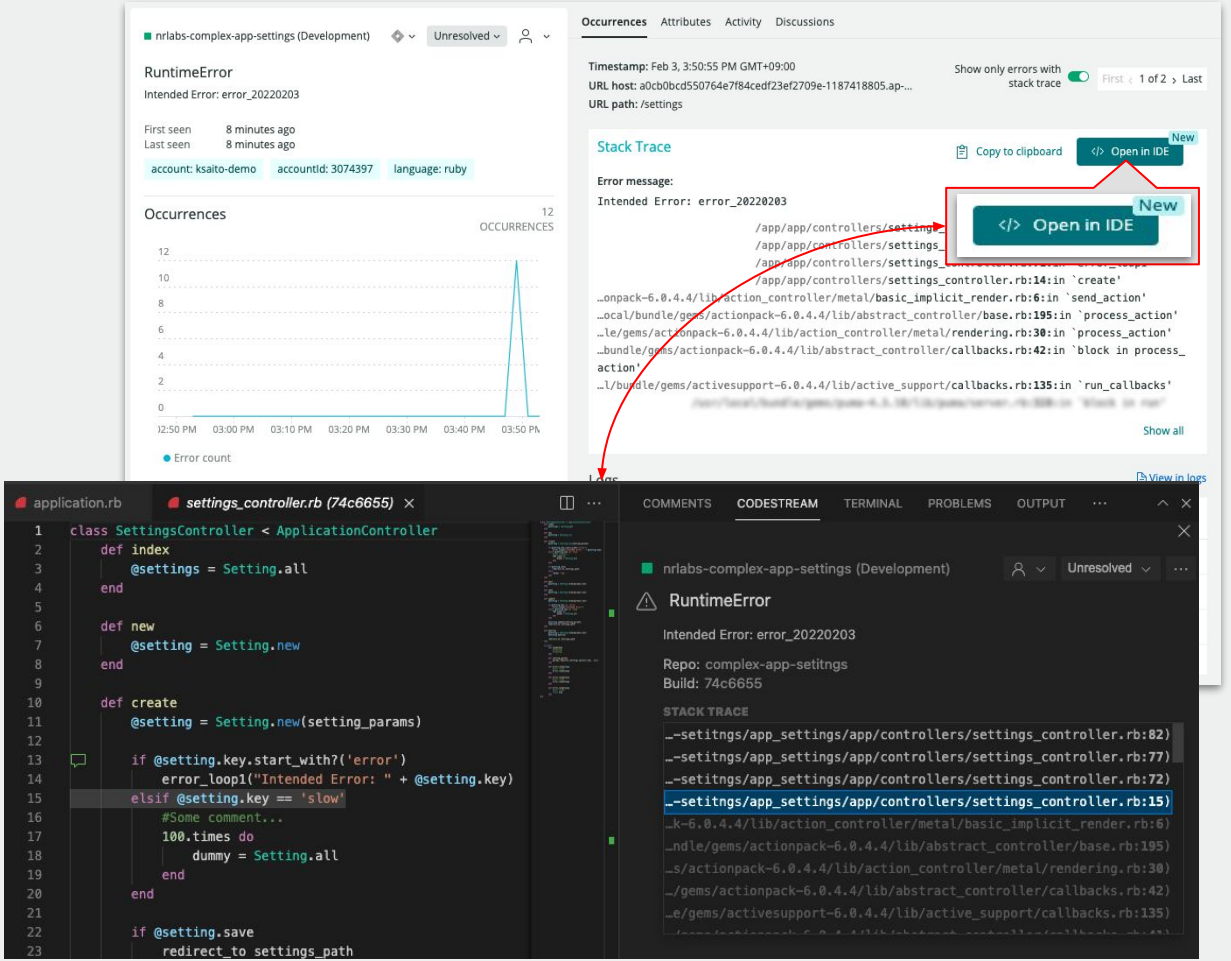
オブザーバビリティと改善の効率性によって次世代の開発者スキルを獲得

©2008–21 New Relic, Inc. All rights reserved

New Relic CodeStream

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

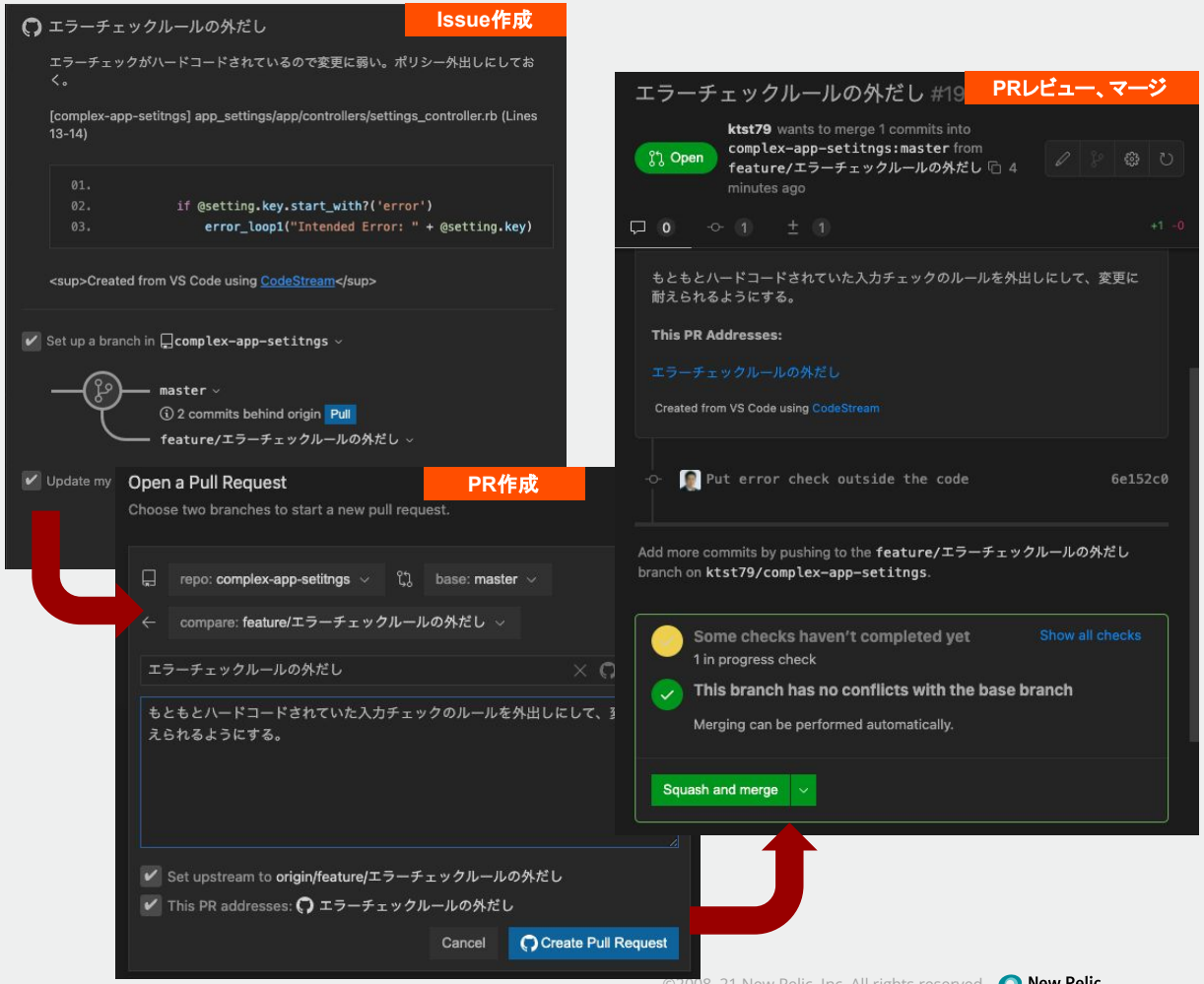


©2008–21 New Relic, Inc. All rights reserved New Relic.

New Relic CodeStream

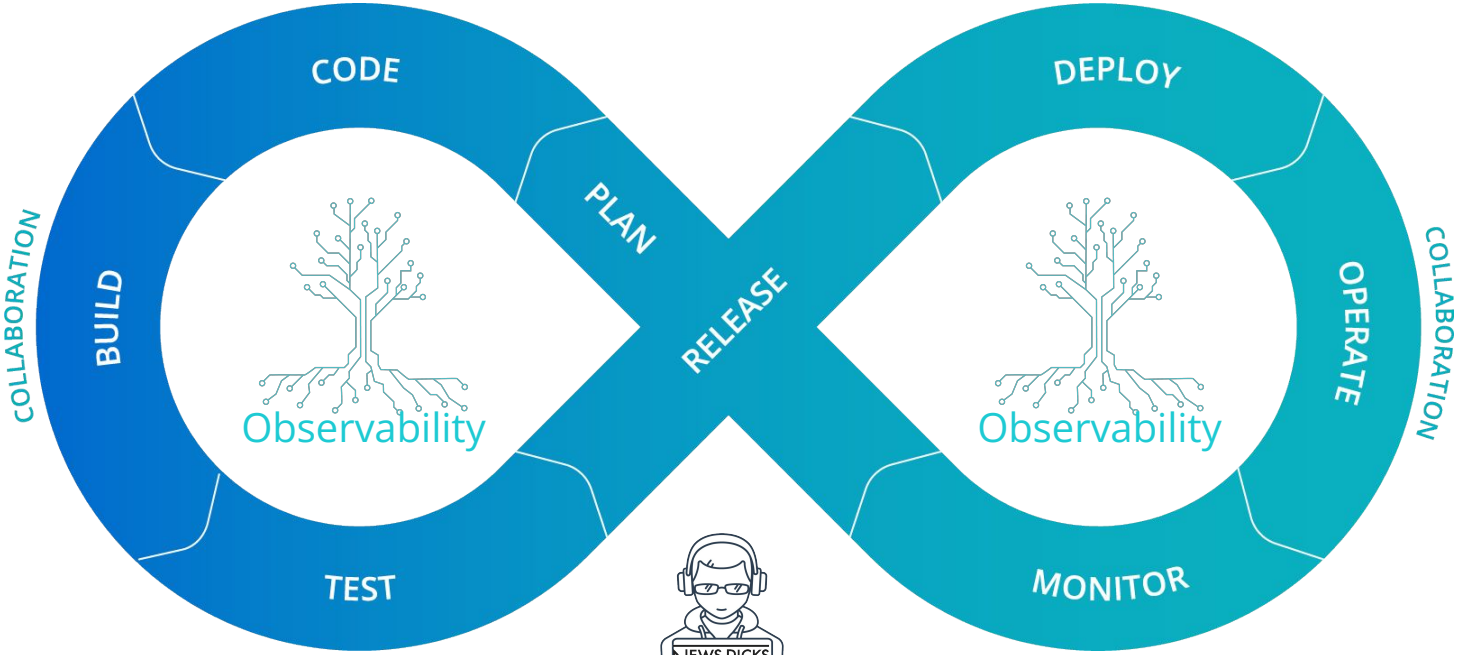
タスクの管理からコードのマージまで一連の作業がIDE上で完結

- 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



©2008–21 New Relic, Inc. All rights reserved New Relic.

まとめ『開発者のためのオブザーバビリティ』



プロダクト開発エンジニア

SRE

©2008–21 New Relic, Inc. All rights reserved

NEWS PICKS

NewsPicks のプロダクト開発エンジニアが実践するスキルとしての SRE

株式会社ニューズピックス
Product division Web Product Unit
イイダユカコ

@becyn on

はじめまして

略歴

2013 年 4 月より
株式会社サイバーエージェントにサーバーサイドエンジニアとして入社。

2015 年 12 月より
AbemaTV への異動をきっかけにフロントエンドエンジニアとして従事。

2019 年 9 月より
株式会社ニューズピックスにエンジニアとして入社。

入社後 API 開発なども行っていたが、
2020年よりフロントエンドエンジニアをメインに従事。

NewsPicks の Web project の Re-architecture の提案をきっかけに、
2021年7月に Web Frontend Unit が発足。
現在は同 Unit の後継である Web Product Unit のメンバーとして開発を行う。

主な興味分野は、
a11y、design system、UI/UX。

安心感高い開発環境を
つくっていきたい！

本日の内容

“SRE” のつく部署の経験がない、開発者としての Web Frontend メンバーが

スキルとしての “SRE” と対峙し、現在の project で実践していく話です。

開発者にとって「スキルとしての “SRE”」がなぜ必要だと考えるか、

具体的に、o11y ツールである New Relic One のどういった機能を利用して、
開発者目線の SRE 的課題をクリアしているか

をご紹介します。

Web Product Unit の責務と「プロダクト開発エンジニア」という働き方

なぜ開発者にスキルとしての “SRE” が必要なのか

division 内の Web Product Unit の立ち位置

* 組織図は、2022年2月時点

division 内の Web Product Unit の立ち位置

* 組織図は、2022年2月時点

Web Product Unit の責務

- Web Product に関する開発
  - Frontend 開発、bff（GraphQL）開発、API 開発を担う
- Web 基盤の開発＆メンテナンスをする**『Web 基盤屋』**
  - 開発部隊である 9 Unit のうち 6 Unit がコミットする基盤の運用
  - Web 基盤環境の全域を、メンテナブルにする責務も持つ

後者の責務により、全関係者にとって「**安心して開発できる環境**」の整備が必要で、
そのために他 Service を含めたプロダクト全体で起こっている事象や因果関係がわかる
オブザーバビリティ(以降、o11y) の向上がとても重要 だと考えている。

Web Product Unit としての理想と課題

- 安心して開発してもらいたい
- ユーザーに価値を提供できることに集中してもらいたい

理想

対して不安になる要素の例とは、

- 何がどうあったら安全なのかをわからないと怖い
- イレギュラー時にどうなるのかわからないから怖い
- リリースした瞬間問題が発覚したら怖い
- リリース後、悪い影響を出していないか不安に感じる
- 不具合が起こった時どこを見ればいいのかわからない
- (もっと言うと) どうなっていけば better なのかがわからない
    - 自身の見立てを立証する根拠やデータがほしい

課題

この課題を解決することでより Unit の Value を発揮できると感じている

➡ みんなでこれらの観測ができるようになることがポイントになる

Web Product Unit としての理想と課題

- 安心して開発してもらいたい
- ユーザーに価値を提供できることに集中してもらいたい

理想

対して不安になる要素の例とは、

FAQ

- 何がどうあったら安全なのかを知りたい
- 今安全なのかを知りたい
- リリースしてよいのかを知りたい
- リリース後、悪い影響を出していないか知りたい
- 不具合が起こった時どうしたらいいのか知りたい
- (もっと)テストしなくていいのか知りたい
  - 自身の見立てを立証する根拠やデータがほしい

課題

そこまでする？
SRE Unit に任せてもよくない？

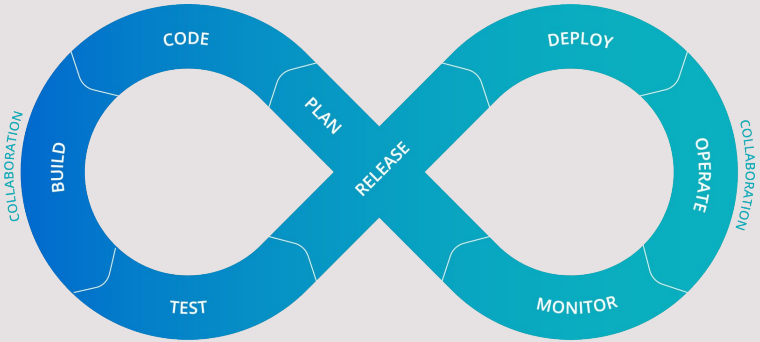
次スライドより、実際どういった理念を持ち働いているのかを
「プロダクト開発エンジニア」の働き方の紹介を交えつつ紹介します。

この課題を解決することでより Unit の Value を発揮できると感じている

➡ みんなでこれらの観測ができるようになることがポイントになる

「プロダクト開発エンジニア」とは？

- NewsPicks の product division 内で根付いている考え方、働き方の名称
- **「ユーザーに価値を届ける」ことを重点に置いている**
  - いわゆる「アプリケーション開発」だけでなく、モニタリングやトラブルシューティングといった運用にもコミット
  - 「問題発見」から「解決」までやることを開発者が担当（もちろん助け合います！）
- よりよい体験、より良いプロダクト開発を実現したいと願い、その責任も持っている



この図の全てをフルサイクルで行うことを

息を吸うようにカジュアルに、

かつ、スピーディにやっていきたい

開発者にとっての o11y の必要性

「プロダクト開発エンジニア」にとっても辛さや不安を感じる要素

- 何がどうあったら安全なのかをわからないと怖い
- イレギュラー時にどうなるのかわからないから怖い
- リリースした瞬間問題が発覚したら怖い
- リリース後、悪い影響を出していないか不安に感じる
- 不具合が起こった時どこを見ればいいのかわからない
- (もっと言うと) どうなっていけば better なのかがわからない
    - 自身の見立てを立証する根拠やデータがほしい

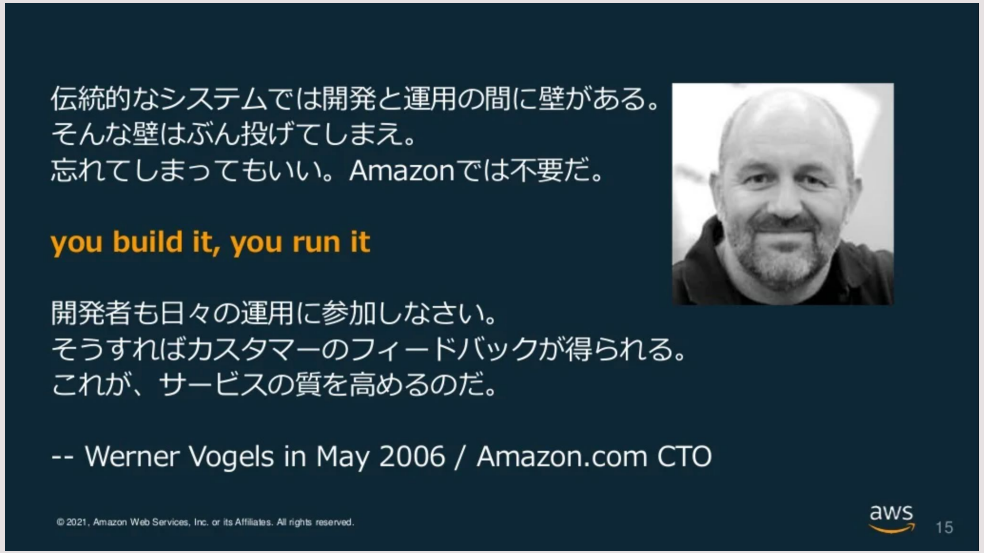
課題

「プロダクト開発エンジニア」としてプロダクトに関わる文化/環境だからこそ
各開発者がプロダクトの健全さを把握するために
「スキルとしての “SRE”（サービスの信頼性を高めるための視点を持ち、価値が想定通り届いているか観測するスキル）」の実践が必要。

よって、全ての「プロダクト開発エンジニア」の運用コストを下げるためにも、o11y の向上が必要。

開発者にとっての a11y の必要性

私達は、開発者が「企画、実装、運用」を含む「フルサイクル」でのプロダクト開発を推進する。
その時、「スキルとしての “SRE”」が必然的になると考えています。



2006 年のワーナー・ヴォゲルス氏の発言である “**you build it, you run it**” の精神を大事にしていきたいです。

開発者にとっての a11y の必要性

私達は、開発者が「企画、実装、運用」を含む「フルサイクル」でのプロダクト開発を推進する。
その時、「スキルとしての “SRE”」が必然的になると考えています。

FAQ

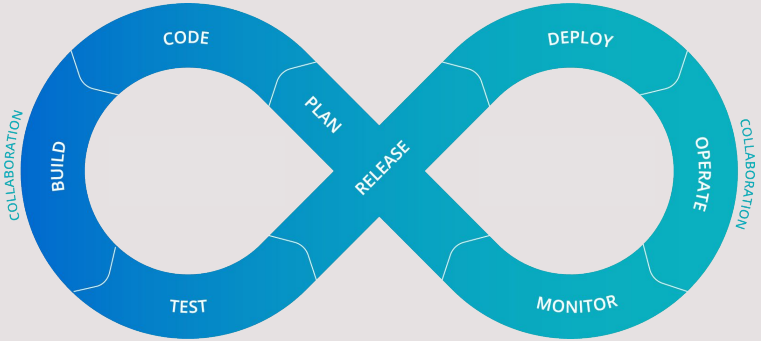
それって開発者が DevOps 頑張ってますっていう意味と何が違うの？

2006 年のワーナー・ヴォゲルスの発言である “**you build it, you run it**” の精神を大事にしていきたいです。

開発者が「スキルとしての “SRE”」を持つこと

よくいわれる「DevOps」とは 開発チーム（Development）と運用チーム（Operations）が
別のチームとして存在する場合が多い。

私達は『「フルサイクル」の流れで開発者が運用と向き合うこと』が
より良いプロダクト開発につながると、強調して伝えたい。



この図の全てをフルサイクルで行うことが

プロダクト開発の上で重要と考える

開発者と o11y の距離が近いことで得られるメリット

特定のロールに依存せずプロダクトを監視できるようになる

→(担当 Unit を越えた) 問題の発見が早まる

→フィードバックループのサイクルが早まる

→結果、実装物の運用コストも下がり、
開発者が目指す「デプロイ頻度の向上」にも寄与する

NewsPicks の開発組織が掲げている指標の一つ

開発者体験向上のためにも ”**o11y は大事な指標の一つ**” と言える

Web Product Unit が
どう o11y を実現しているか

APM を使用した具体例と結果の共有

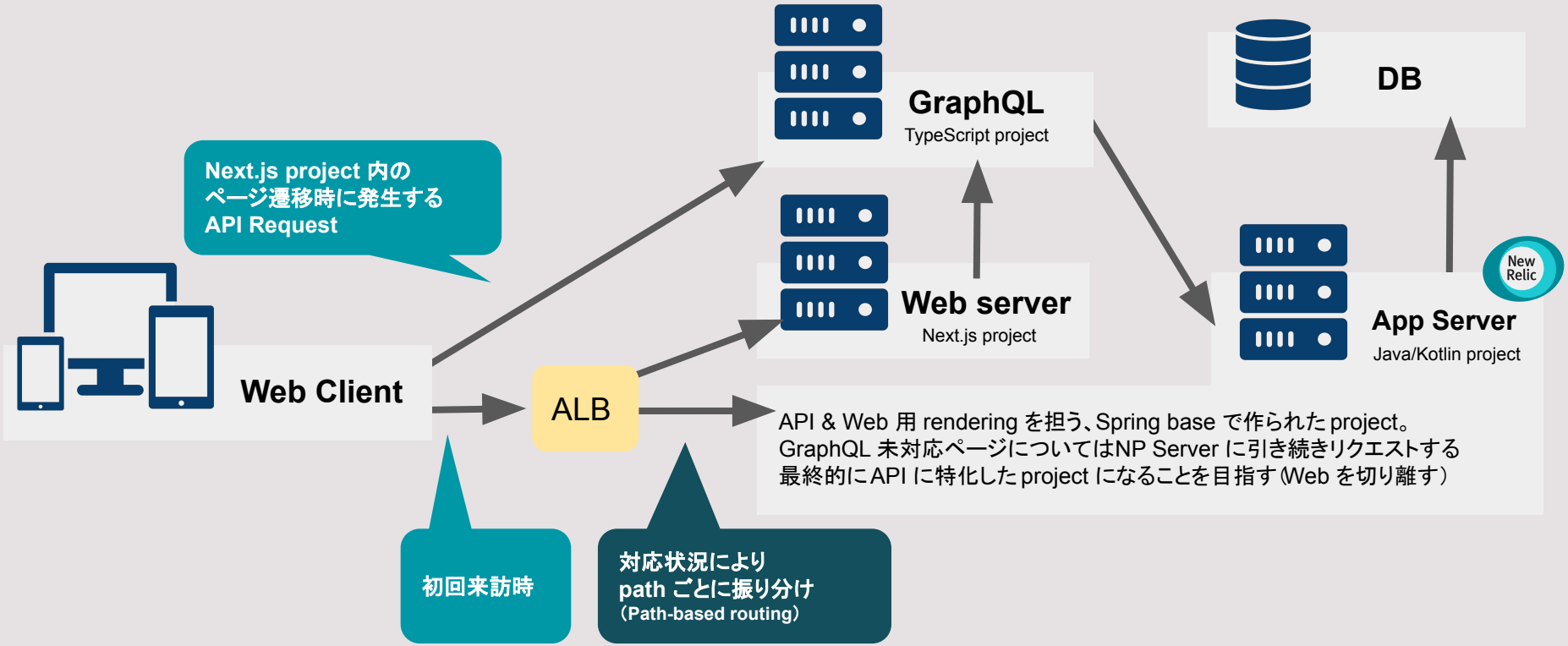
o11y の実現 (APM を利用した具体例) - 問題発生

昨年実際に起こったリリース後の不具合
『リアーキテクチャ対象の1ページ目のリリースを行った後、異常にレスポンスが遅い場合が存在する』

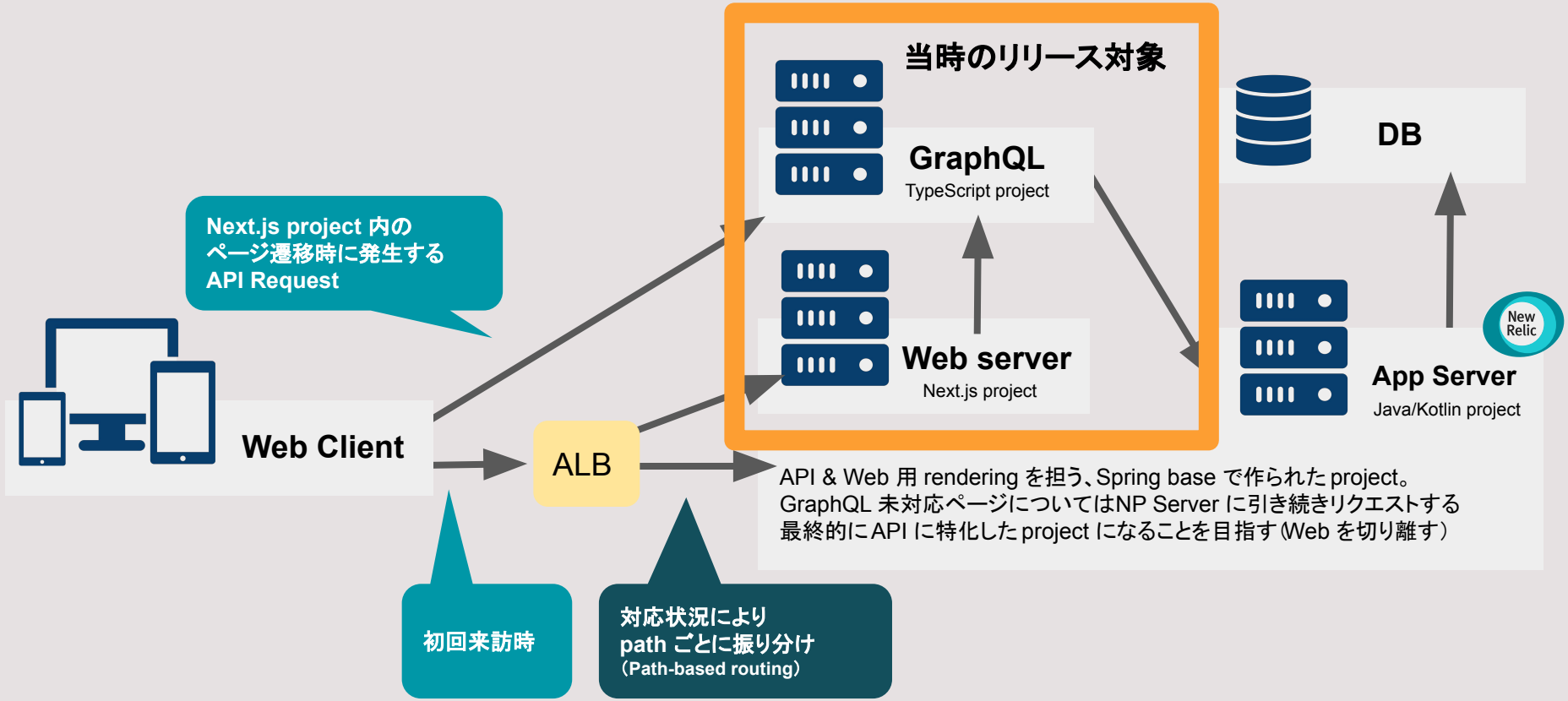
新しい基盤と古くからある基盤の条件が合わさり「推測」したが、「推測止まり」になってしまった。

よりリアルな数値を見る必要を感じ、その場で o11y 未導入環境 (Web Server、GraphQL Server) に対して New Relic One の導入を行い、Application Performance Monitoring(以降、APM) を図った。

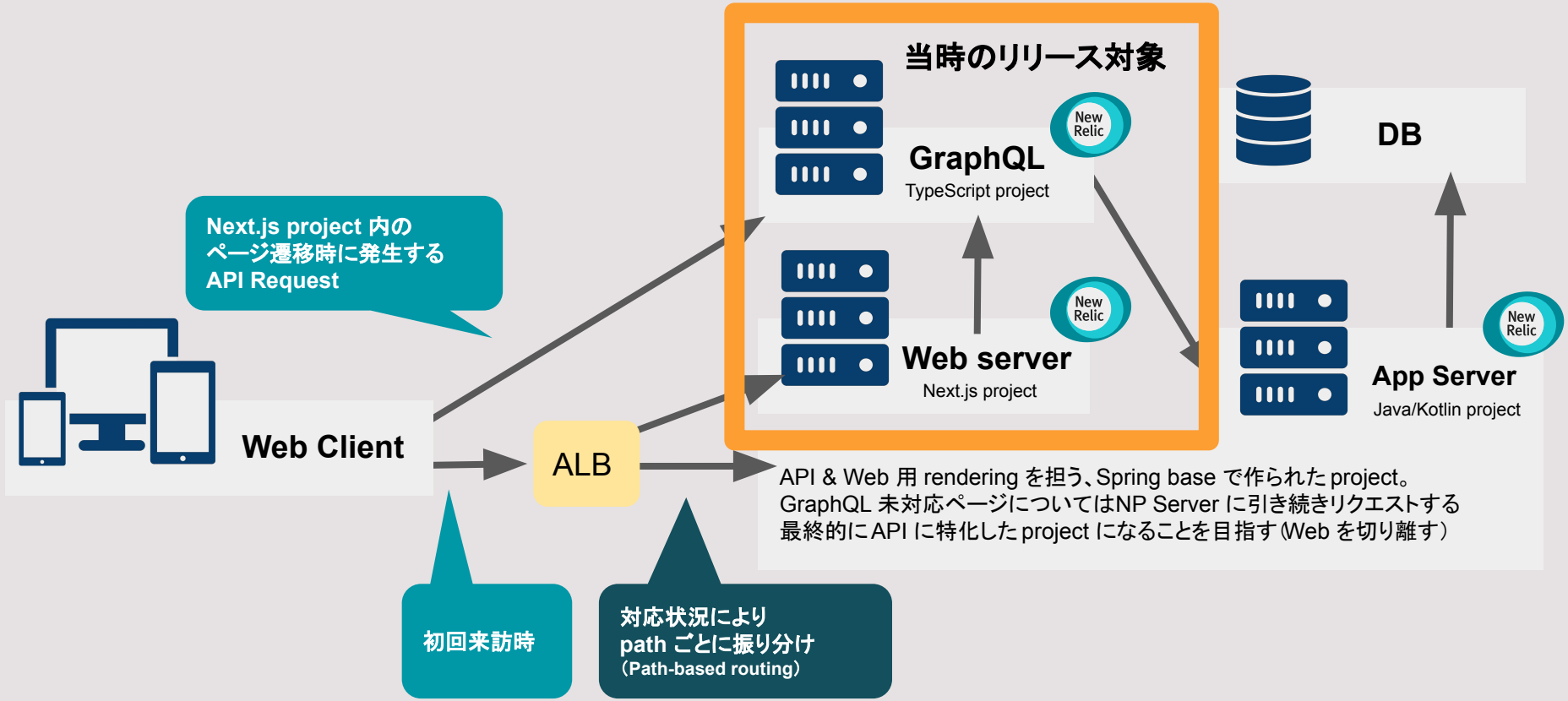
アーキテクチャ(略図)


Next.js project 内の
ページ遷移時に発生する
API Request
Web Client
ALB
初回来訪時
対応状況により
path ごとに振り分け
(Path-based routing)
GraphQL
TypeScript project
Web server
Next.js project
DB
App Server
Java/Kotlin project
New Relic
API & Web 用 rendering を担う、Spring base で作られた project。
GraphQL 未対応ページについてはNP Server に引き続きリクエストする
最終的にAPI に特化した project になることを目指す(Web を切り離す)

アーキテクチャ(略図)


当時のリリース対象
GraphQL
TypeScript project
Web server
Next.js project
DB
App Server
Java/Kotlin project
New Relic
Next.js project 内の
ページ遷移時に発生する
API Request
Web Client
ALB
初回来訪時
対応状況により
path ごとに振り分け
(Path-based routing)
API & Web 用 rendering を担う、Spring base で作られた project。
GraphQL 未対応ページについてはNP Server に引き続きリクエストする
最終的にAPI に特化した project になることを目指す(Web を切り離す)

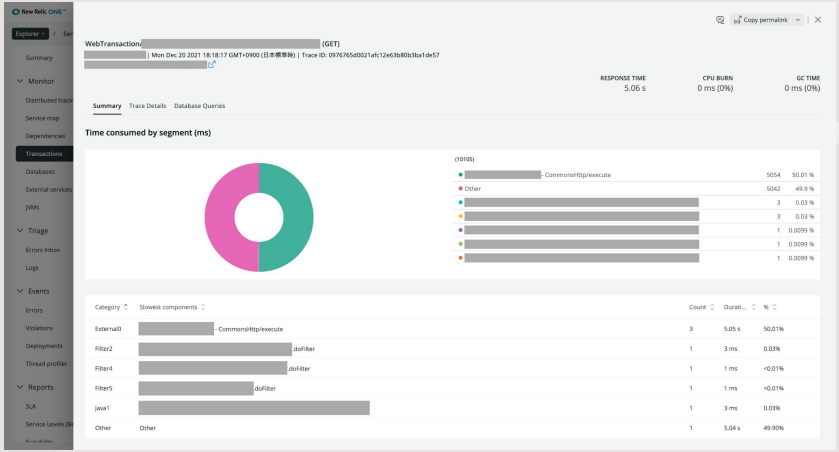
アーキテクチャ(略図)


当時のリリース対象
GraphQL
TypeScript project
New Relic
DB
Next.js project 内の
ページ遷移時に発生する
API Request
Web server
Next.js project
New Relic
App Server
Java/Kotlin project
New Relic
Web Client
ALB
API & Web 用 rendering を担う、Spring base で作られた project。
GraphQL 未対応ページについてはNP Server に引き続きリクエストする
最終的にAPI に特化した project になることを目指す（Web を切り離す）
初回来訪時
対応状況により
path ごとに振り分け
（Path-based routing）

o11y の実現 (APM を利用した具体例)

APM 設定後の流れ

- 各 Service の APM を確認
- 起因となる Service の Transactions を監視し、通信を探る
- 原因の Transaction を確認後、当時実装を担当していたメンバーにヒアリング



実際に起っていることを1ページにまとまっているデータで伝えることができ、スピーディに議論、その後の改善方法の提案、実装にまで繋がりました。

APM の導入を行って感じたこと

Point 1: 共有しやすさ
同 Unit メンバーへの共有はもちろん、Unit を越えたメンバーに見てもらう&伝えることができる

- 不具合内容に関する共有用ドキュメントを書く必要がない
  - ここがほぼ 0 カロリーでできることにとても意味がある

Point 2: 一覧しやすさ
複数 Service を一覧で確認できる (= 調査、確認がしやすい)

- 関係のある Service 全てが健全であることで初めて「サービスの健全性」を担保できる
- マイクロサービス化が進む昨今だからこそ、求められる機能

ただ、APM の導入だけでももちろん十分に不具合を検知できるが、
私達は、もっと「安心して開発できる」かつ、もっと「運用コストが下がった」環境を実現していきたい。

Web Product Unit が
どう o11y を実現しているか

APM の先に進むために行なったことの紹介

APM の次のステップに進みたい

「プロダクト開発エンジニア」にとっても辛さや不安を感じる要素

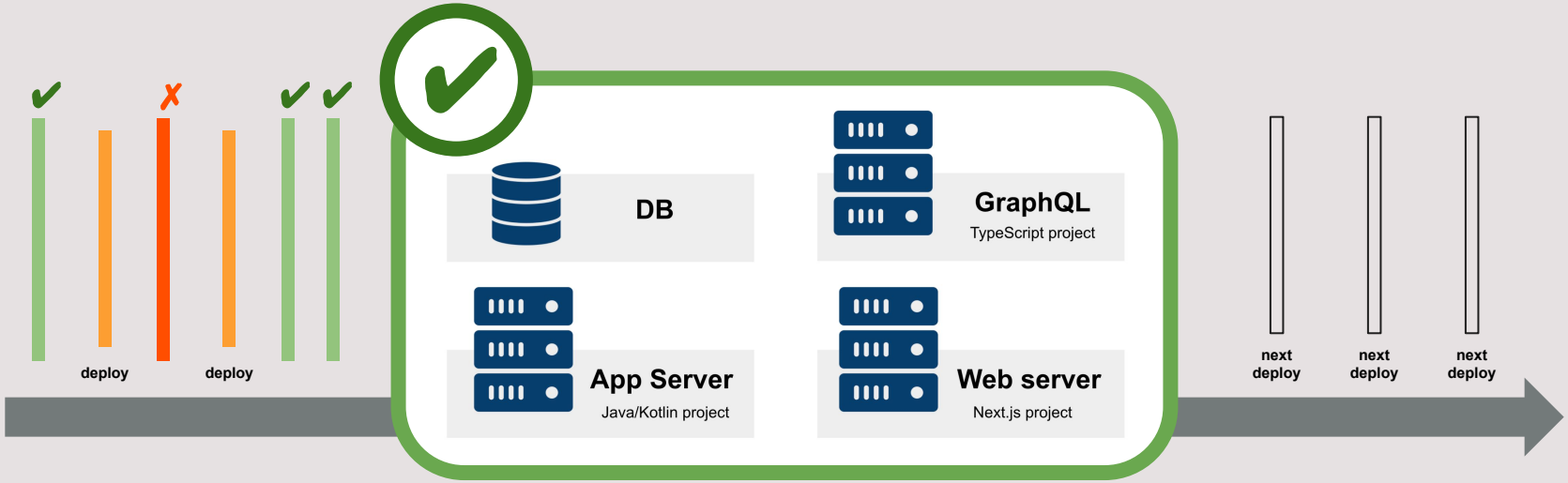
- 何がどうあったら安全なのかをわからないと怖い
- イレギュラー時にどうなるのかわからないから怖い
- リリースした瞬間問題が発覚したら怖い
- リリース後、悪い影響を出していないか不安に感じる
- 不具合が起こった時どこを見ればいいのかわからない
- (もっと言うと) どうなっていけば better なのかがわからない
    - 自身の見立てを立証する根拠やデータがほしい

課題

「開発」「運用 (通常時/問題発生時)」「改善」のフェーズ全てに課題がある状態

どのフェーズの不安材料も全て解消したい

これらを解消するために、



「過去どうあり、どんな変更を加え、今がどうで、未来どうすべきか」の可視化、観測が必要

各フェーズにおける観測したいポイント

開発

1. (リリース前の段階で) 事前にエラーを検知し、より安全に deploy したい
2. イレギュラー時に自身のプロダクトがどうなるのかを知りたい

運用 (通常時/問題発生時)

3. ひと目で「今私達のプロダクトは健全である」を確認したい
4. プロダクト全体でどんな変更が行われているかを確認したい

改善

5. 外的要因を含めた Performance 改善を図るヒントがほしい

この 3 つの異なるフェーズの全てで観測したいポイントを観測できるようにし、課題の解決を図る

事前にエラーを検知し、より安全に deploy したい

「本番環境でエラーを認識」を避けたい。
事前にエラー発生に気づき、その内容を知り、対策を練りたい。

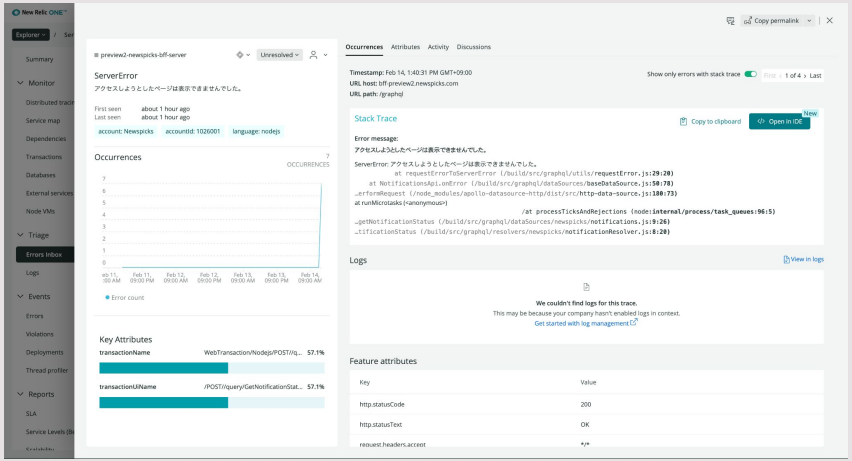
開発環境、テスト環境、本番環境の全てで o11y を測ることで解決
開発段階でのバグの早期発見＆修正が可能。本番前から SLO をより意識できる環境に。

エラー発生を知ることだけでなく、IDE と連携することで「エラーイベント」を「手元」に持ってくることができる

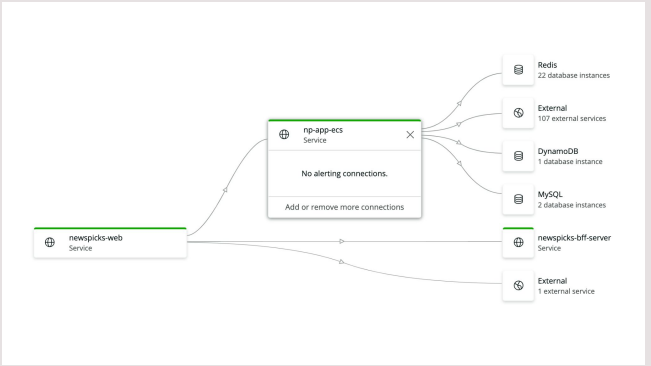
イレギュラー時に自身のプロダクトがどうなるか知りたい

全ての関係者が安心を得るため「何が起こるかわからない」を潰していきたい。
その時どう対応するのが良いかを事前にチームで共有しておきたい。

イレギュラーが起こる負荷テスト環境等にも入れることで、「体感」できるようにすることで解決



全ての環境でエラー検知を行ない、エラーのトレースができる状態に

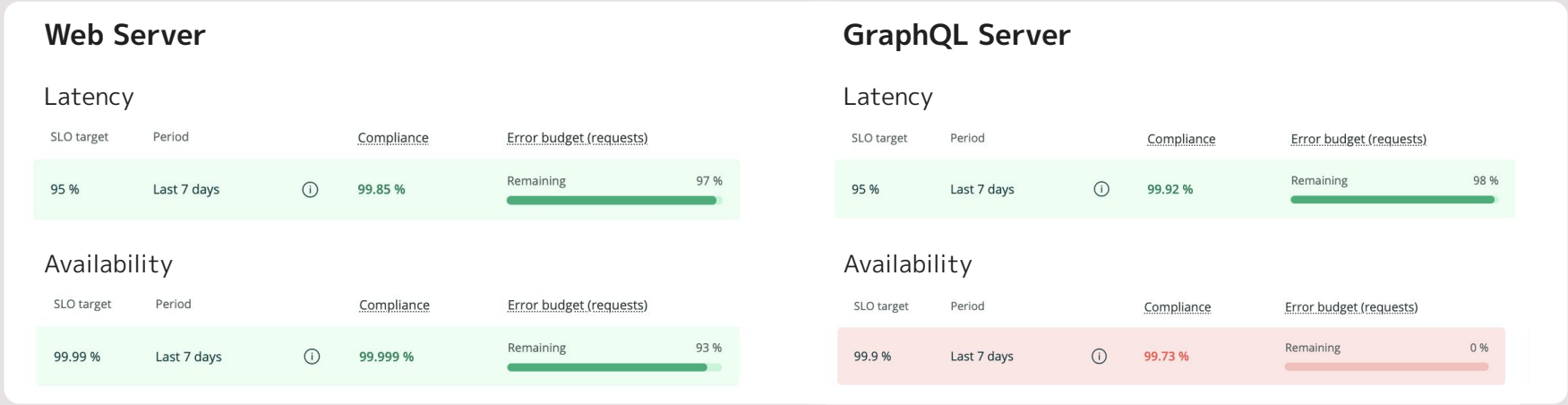


どのタイミングでどこが落ちたかがわかる状態に

ひと目で「今私達のプロダクトは健全である」を確認したい

「健全だ」と、実装者自身にも、チームメンバーにも立証できる形で伝えたい。

各環境ごとに SLO を o11y ツール内で設定し、監視データから計測＆描画することで解決
(副次効果として、新規メンバーへの説明もしやすく、SLO 達成の良い文化づくりにも繋がる)

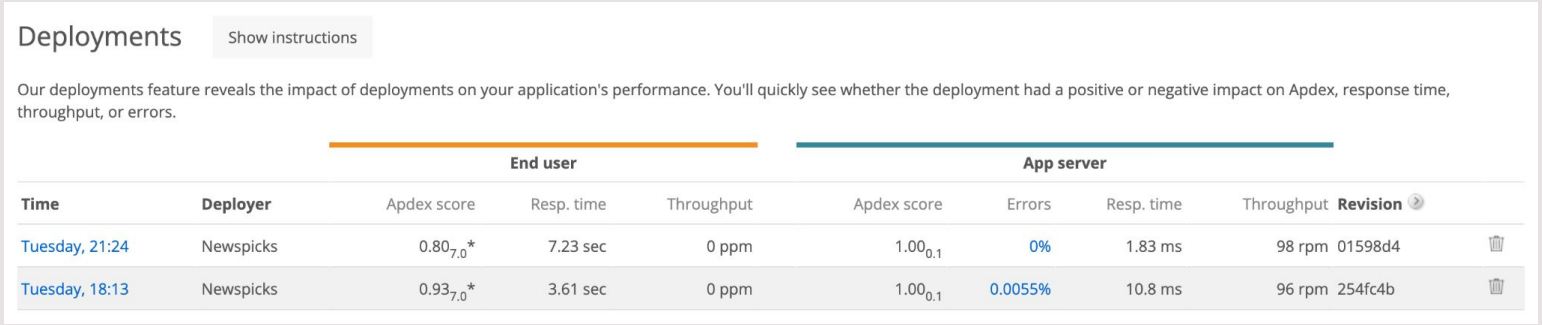


サービスの健全レベルをひと目で確認でき、SLO を守れていない時に「予期していない状態」を認識しやすい

プロダクト全体でどんな変更が行われているかを確認したい

自分たちの Code Deploy 以外の Deploy (画像基盤の変更、インフラの改修など) も追うことで、
より正確な、状態の向上低下の原因究明を実現することで、より効果的な　PDCA を回したい。

Deployments 履歴を o11y ツール内で設定し、追っている数値の変化を可視化することで解決
Apdex score やエラー率を一覧に載せることで　deploy 履歴と状態の推移を確認できる

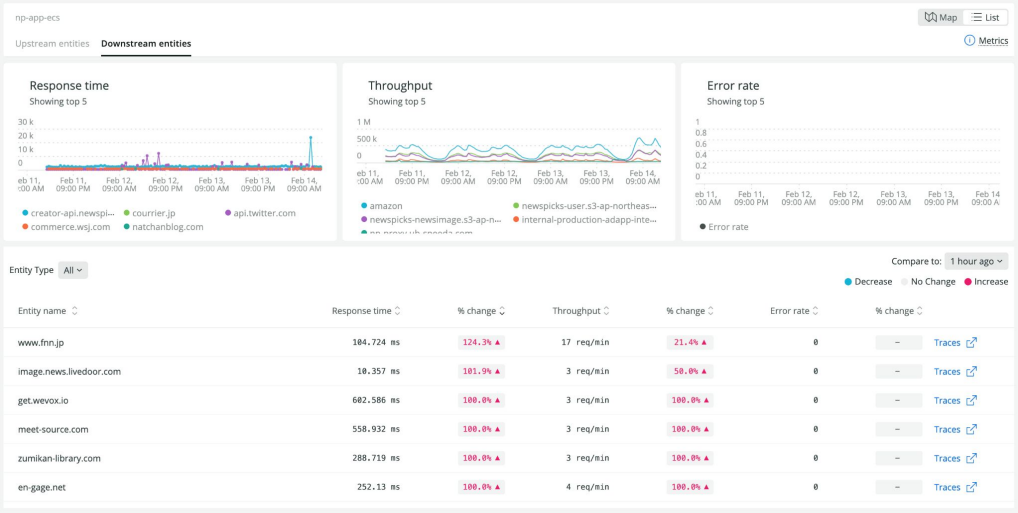


Deployment Marker で deploy ごとに Apdex score や Error 率などを比較することができる

外的要因を含めた Performance 改善を図るヒントがほしい

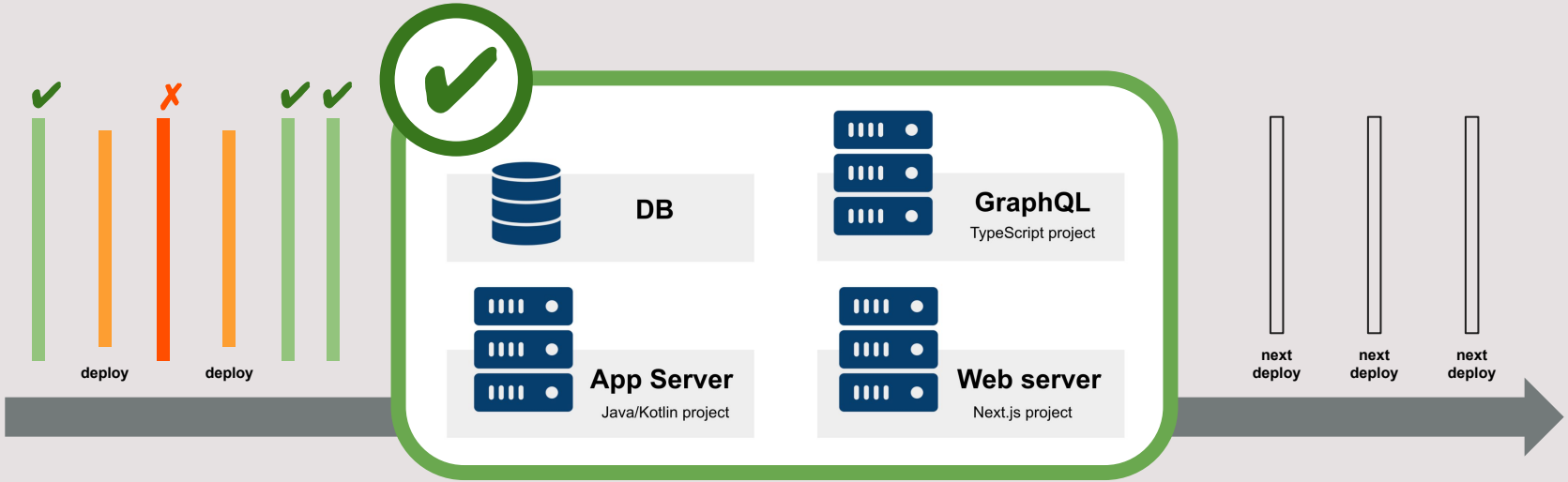
我々のサービスの特性上、他サービスからの情報取得がクリティカルになるので、実はとても重要。
測りにくい外のサービスの情報を正確なデータで取得し、判断材料にしたい。

o11y ツール内の外的 Service の監視機能を使用し、解決。
直近の Performance 的変化などを可視化することでより正確な対策を練ることができる。



外のサービスをドメインごとにResponse time がどう変化しているかを一覧することができる

APM の先に進むため、o11y 向上を図った



あらゆる環境で o11y を図ることで、
「開発」「運用 (通常時/問題発生時)」「改善」の各フェーズの不安要素を解決することができた

まとめ

まとめ

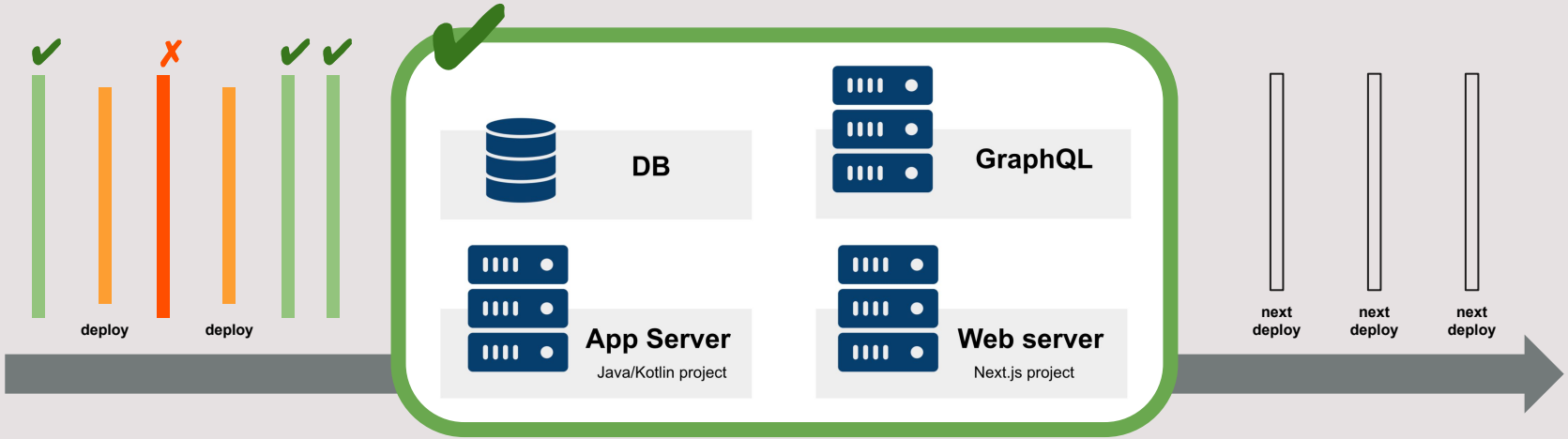
今回 Web Product Unit のいちエンジニアが実体験を元に「スキルとしての “SRE” の必要性」と、
そこで直面した課題を o11y を実践することで解決に至った話を共有しました。

「ユーザーに価値を届ける」ことを評価軸にサービス開発しているエンジニアにとって、
開発、運用 (通常時/問題発生時)、改善のあらゆるタイミングの不安解消を図るために o11y の向上が有用でした。

これからもより開発に集中できる環境にすべく o11y 観点を意識した開発、環境強化をしたいです。

また、この経験を通して、本当の意味での「サービス監視」に向き合い、その価値を体感できました。

スキルとしての “SRE” を実践し、出た課題を o11y で解決



「過去どうあり、どんな変更を加え、今がどうで、未来どうすべきか」を
o11y ツールを用いることで「**チーム**」で体感し、全体の理解度が増した

Thank You

New Relic.

©2008–22 New Relic, Inc. All rights reserved