New Relic

White Paper

AWS による モダンアプリケーション構築：

「オブザーバビリティ駆動型」 リファクタリングを始める

クラウドにおけるレガシーアプリケーションのモダン化に関するホワイトペーパー第 3 部（全 3 部）

概要

企業が既存のアプリケーションから Amazon Web Services (AWS) へ移行する際は、各アプリケーションについて、正しい方法でリホスティング、リプラットフォーミング、またはリファクタリングにアプローチする必要があります。Application Modernization（アプリケーションのモダン化）シリーズの 3 番目に当たるこのホワイトペーパーは、後者のアプローチを対象とするものです。

リファクタリングは、最も簡単な形としては、アプリケーションコードの改善を意味します。通常、それはパフォーマンス、質および維持・管理の改善を目的とします。AWS のためのリファクタリングは、そのような利点を拡大する他、可用性、スケーラビリティ、信頼性の拡大も可能にするものです。さらにリファクタリングには、区別されていない大量の作業を AWS にオフロードする機能を提供するという大きな利点があります。Amazon Elastic Container Service（Amazon ECS）、Amazon Elastic Kubernetes Service（Amazon EKS）、または AWS FargateTasks など完全なマネージドサービスを使うと、ハードウェアのメンテナンス、オペレーティングシステムのパッチ管理、データベースのアップデートなど多くのタスクを全て管理してくれるので、チームはビジネスの差別化を実現することにフォーカスできるようになります。

確かに、通常リファクタリングでは、基本テクノロジーのロジックまたは大きなシフトを通してアプリケーションの動作方法を変更することになるので、モダン化戦略の中で最もリスクが高いものと言えます。同時に、リファクタリングほど高いレベルで望ましいインパクトを与え、大きな投資効果を実現できるモダン化アプローチは他にありません。

これを読み、リファクタリングに伴うリスクと利点、AWS 環境における異なるタイプのリファクタリング、そして New Relic がプロセスで優れた結果を推進するうえでいかに役立つかをご確認ください。

リファクタリングとは？

Refactoring: Improving the Design of Existing Code の著者 Martin Fowler は、リファクタリングを「既存のコードを再構築し、外部の動きを変えることなく内部構造を変更するための、統制されたテクニック」と定義しています。

リファクタリングを選択する理由とは？

アプリケーションの中には、ビジネスにとって他より重要なものがあります。それは、アプリケーションの戦略的に必要不可欠ですか？それは、収益に貢献しますか？それは、さらなる投資の根拠となりますか？これらの質問のいずれかの答えが「はい」であれば、おそらくリファクタリングの検討対象となるでしょう。

戦略性が低い他のアプリケーションはリホスティングまたはリプラットフォーミングの検討対象となる傾向がある一方、戦略的アプリケーションは、投資がより具体的な効果をもたらすため、リファクタリングの対象として選択するうえで最も適切なものとなる可能性が大きいのです。企業がアプリケーションのリファクタリングを選択する際の、説得性のある理由としては以下のようなものがあります。

New Relic.

- 新しい収益ストリームを推進する、および / または既存の収益ストリームを最適化する
- 将来の収益獲得能力に直接的インパクトを与える改善を生み出す
- 顧客体験の改善を実現する
- 新機能を導入して市場化を加速する
- ビジネスモデルの変更 / 新しいビジネスモデルをサポートする
- 変化する規制 / 新しい規制を遵守する
- 計画された、または予想外のトラフィックがもたらす変化に対応するために、容易にスケールアップまたはスケールダウンする

リファクタリングのリスク / 利益のバランスを管理する

リファクタリングを、リスクと努力に値するものとするには、方程式においてリスク側を最少化し、利益側を最大化することが重要です。

そのためには、AWS へ向けて / においてアプリケーションのリファクタリングを行う場合は、以下のような、企業が直面する可能性があるリスクを理解する必要があります：

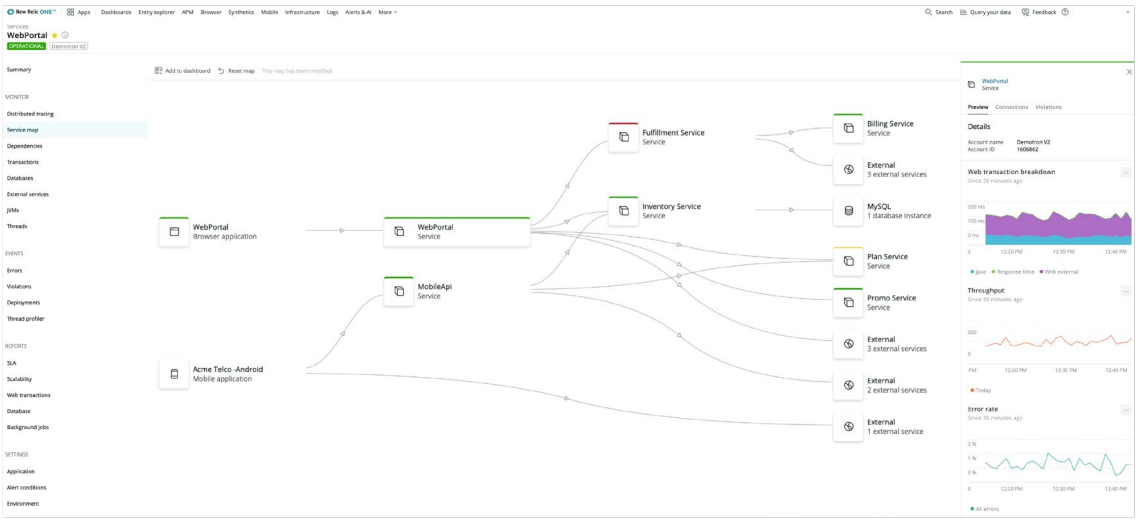
- 社内外におけるアプリケーションの依存関係を完全に理解し損なう
- プロジェクトの複雑さを完全に理解するための分析が不充分であったため、結果として、プロジェクトの目標が達成できなくなる
- 二重インフラストラクチャを伴う大規模なテストおよびデバッギングを必要とする大規模な変更により、一時的にコストが上昇する
- 期待されたビジネス上の結果を達成できない
- 顧客体験における、ナビゲーション面でのインパクト
- 是正を必要とする予想外の新たな問題が発生したために、プロジェクトの所要時間とコストが上昇する

これらのリスクの多くに共通して見られるのは、アプリケーション内部の仕組みに対する理解と可視性の欠如です。データとオブザーバビリティがあれば、いかに、そして何を対象にリファクターを行うか決める前に、アプリケーションを深く理解できるようになるため、リファクタリングの取り組みに伴うリスクを大幅に低減できます。たとえば、New Relic One のサービスマップを使うとアプリケーションの全ての依存関係を特定できるので、リファクタリングの取り組みを計画する際に、それらを考慮に入れることができます。

DevOps がリファクタリングの成功で果たす必要不可欠な役割

リファクタリングの成功にはリファクタリングを成功させるには、敏捷性と DevOps が必要です。数分内にアプリケーションを容易にテストできないなら、それを定期的に改善してゆくことはできません。データ主導型の堅牢な DevOps の戦略、文化、実践の開発に関する詳細については newrelic.com/devops をご参照ください。

New Relic



New Relic Oneのサービスマップで、アプリケーションの依存関係とコンポーネントを調査します。

リファクタリングにおけるオブザーバビリティの価値

現在あるアプリケーションの状況を明確に理解すればするほど、リファクタリングの取り組みの結果がより良いものとなります。モニタリングと測定を通したアプリケーションのオブザーバビリティにより得られる、現在におけるアプリケーションの実行状況に関するインサイトの対象は、目前のリファクタリングプロジェクト全体だけでなく、それを超えるもので、それにより取り組みが与えるインパクトを追跡し提示することができるようになります。

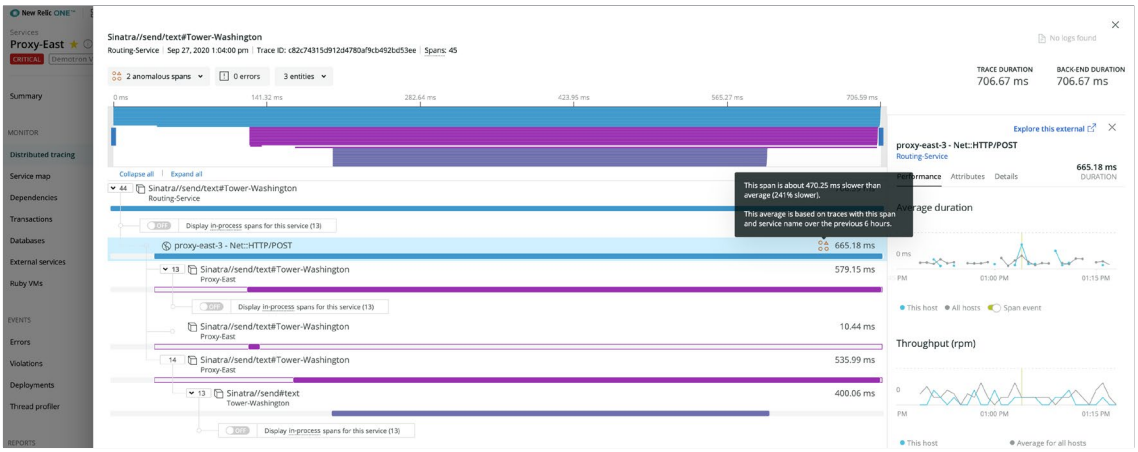
リファクタリングにおけるオブザーバビリティの中核となる原則は、アプリケーションのリファクタリングの前、途中、そして後に、あらゆるものを計装し、測定することです。

前：候補を適切に特定するため、New Relic One を使用してアプリケーションの全体像を把握します。それは、単なる実行状況にとどまらない可視性と理解をもたらします。そして、実行状況をコードベースと照合し、最大のインパクトがどのようなものになるかを理解することができます。データについて、以下の点を確認すべきです。

- コードベースのどの部分に最も大きな変化があるか、もしくは、それらに対し最も多くの問題が提出されているか?これらは、コンポーネント（すなわち、コンテナまたは AWS Lambda で実行されるマイクロサービス）とするうえで適切な潜在候補対象となります。
- 頻繁に問題が発生するのは（修正を示唆するのは）コードベースのどの部分か？
- コンポーネントで対応できるコードのどの部分で、多大な作業を行っているか？
- アプリケーションでパフォーマンスが良いのは、どの部分か？複雑でエラーが発生しやすいのは、どの部分か?実行するのに最も時間がかかるのは、どの部分か？
- 以下のために、よりモダンなテクノロジーを使用できるのはどこか？
  - パフォーマンス上の問題を緩和できるか？

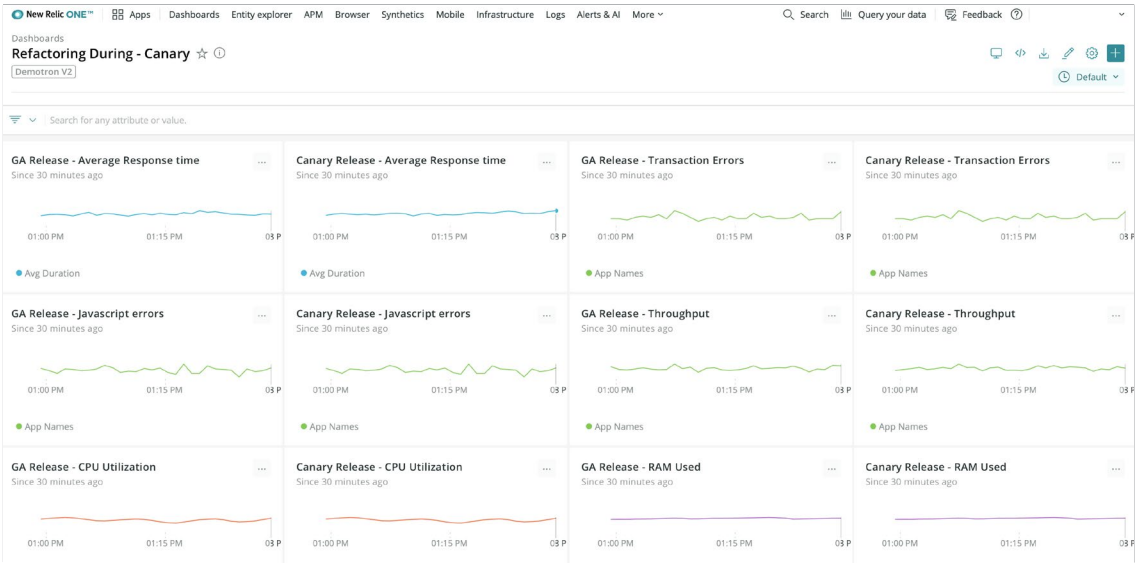
- サーバレステクノロジーを使用するか？
- インフラストラクチャを管理する代わりに、自己回復型インフラストラクチャとサービスを使用するか？
- 競争上の優位性を与えない場合、自己管理型インフラストラクチャを中止するか？

- デプロイメントは、どのくらい上手く行われているか？どのくらい迅速にセットアップされるか？
- ビジネスに与えるインパクトを測定するための主要業績評価指標（KPI）は何か？



ディストリビューティッド（分散）トレーシングでは、リクエストが分散システムを経由するパスを見ることができます。パス上のコンポーネントにおけるレイテンシを発見する、もしくはボトルネックとなっているコンポーネントを理解するために使用します。

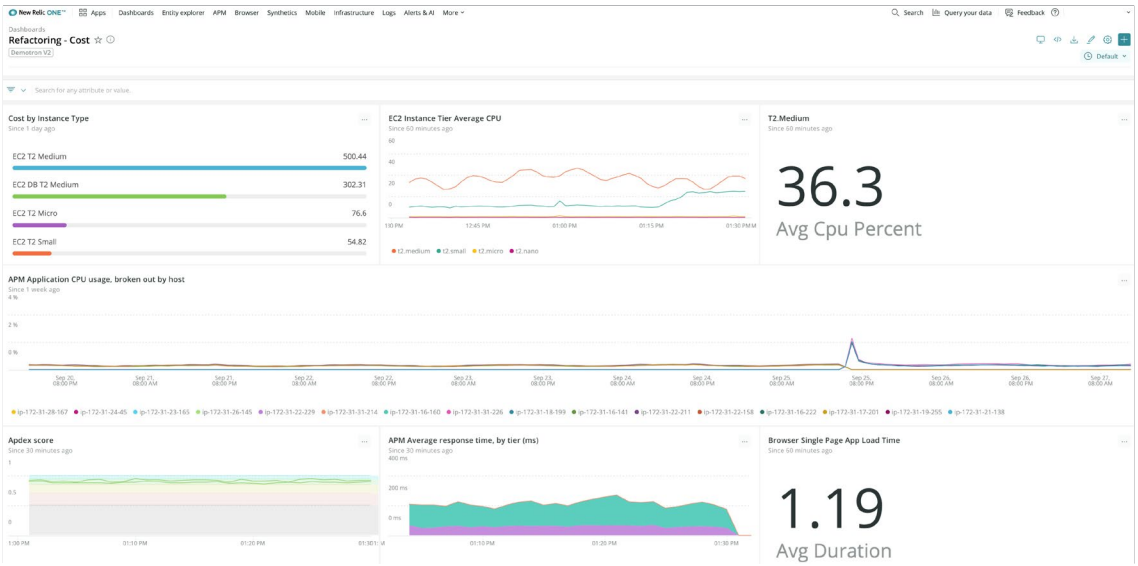
最中：リファクタリングによる問題が発生していないことを確認するため、アプリケーションをテストします。New Relic を使用し、コードと外部サービスのパフォーマンスをモニターして、切替えにおける顧客への影響を把握します。リファクタリングを済ませたアプリケーションを完全にテストし、それに切替えるまで二重インフラストラクチャをどれほどの期間運用しなくてはならないか特定します。



リファクタリング中は、DevOps KPIを注意深くチェックします。

以降：リファクタリングがアプリケーションのパフォーマンスに与えた影響を理解するだけでなく、次のような他の結果も追跡できます：

- **問題 :** 提出されたチケットの数が減ったか?
- **コスト :** このアプリケーションは、クラウドでのコストが低減しているか、および / または、現在の支出を正当化できるか?
- **技術的な負担 :** 修正作業の量が低減したか?
- **DevOps:** 毎日、以前より頻繁にデプロイしているか?どれほど多くなっているか?
- **顧客 :** 顧客体験にどのくらいインパクトがあったか?
- **ビジネス :** KPI に、どのくらいインパクトがあったか ?



ビジネス、アプリケーション、オペレーション、その他のKPIを追跡します。

どのタイプのリファクタリングを使用すべきか ?

コードのリファクタリングをよく知っている人には、AWS 上でのリファクタリングは馴染み深く感じられるでしょう。しかし、それはコードのリファクタリングだけを対象とすることではありません。それは、根本的な形で以下のようなクラウドテクノロジーの利点を活用することを意味するのです :

- 全般的なコードフットプリントを低減させる
- 安全対策が必要な分野の面を減少させる
- システム管理のより多くの部分を自動化し、エラーを低減させる
- コストの最適化を実現する
- 効率性とパフォーマンスを向上させる

AWS 環境では、アプリケーションを対象に、コード、デプロイメント、テクノロジーのリファクタリングという、3 つの異なるタイプのリファクタリングを活用できます。

コードパイプラインの測定に関するベストプラクティス

生産速度を高めることで変更を推進させるために、いかにインストゥルメンテーションを使用できるかについては、New Relic DevOps ソリューション専門家による、このウェビナー をご覧ください。

New Relic.

従来型のリファクタリング：コードを改善

従来の意味に基づく、このリファクタリングの最初のタイプでは、もっぱらアプリケーションコードの改善だけが対象になります。ここでは、質、メンテナビリティ、パフォーマンス、予測可能性のために、コードの再設計が最も意味をもつアプリケーションの部分を特定します。これは、既存の問題を是正すると同時に、コードを簡素化しより合理化するための機会となります。

全体に共通する質問は、リファクタリングの対象とすべきアプリケーションはどの部分であるか、そして関連する取り組みで、何が最も大きな利益を実現するかをいかにして明らかにするか、です。

その答えは、追跡してきたアプリケーションのデータの中にあります。コードパイプラインから得たデータを New Relic One に統合することで、コードベースの変更がオペレーションにどのようなインパクトを与えるかを知ることができます。情報に基づくリファクタリングの「前」段階で収集されたデータを使用すると、問題のある分野およびリファクタリングの機会に関するインサイトが得られます。収集したベースラインをレビューすることにより、改善対象となるアプリケーション分野のリスト作成を開始できます。

次に、コード自体をレビューします。たとえば、より従来型の E コマースアプリケーションには、追跡しているメトリクスの面から、リファクタリングに適切な候補対象となる分野がある可能性があります。しかし、その分野に大きなインパクトがあるかどうか、自分が目指すそれに伴う望ましい結果が得られるかどうかを見極めるためには、やはり取り組みを、アプリケーションの実行状況およびエンドユーザー体験の面から検討する必要があります。

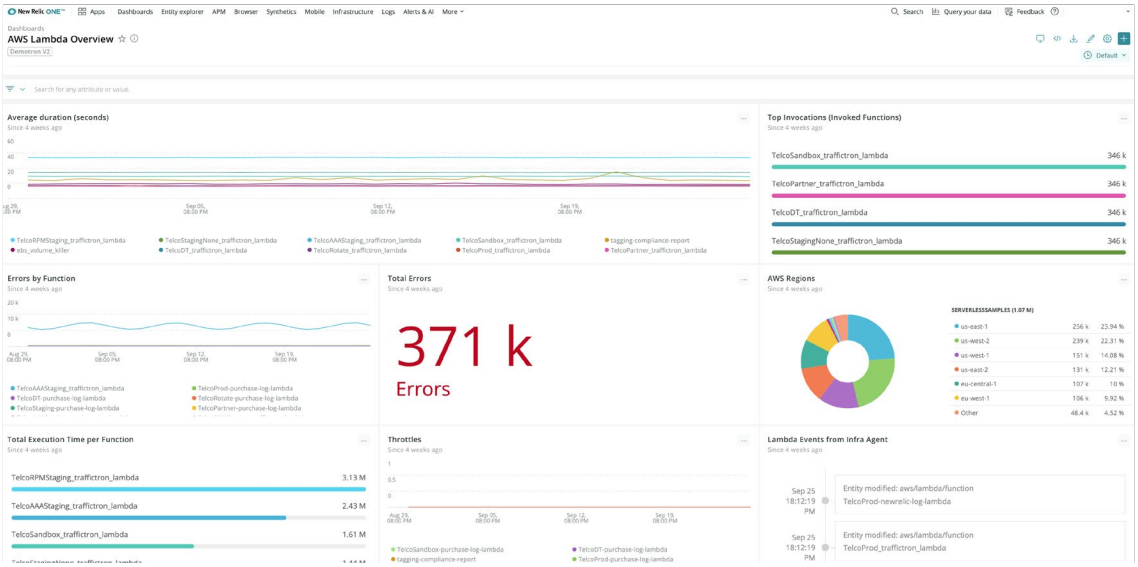
異なるデプロイメントモデルを対象にリファクタリングする

作業の対象とするコードベースの部分を決定した後の次のステップは、新しいデプロイメントモデルを検討することです。IT および将来の方向性の面から、企業にとって最も理に適ったデプロイメントモデルを選択します。

コードのリファクタリングを行う際、それをデプロイして自動管理サーバーに戻すことを選択できます。しかし、代わりに、より多くの低価値のタスクに対応するために AWS サービスを使用することも可能です。たとえば、以下を使用してデプロイすることが可能です：

- 使用、デプロイメント、アプリケーションのスケーリングを容易にするための AWS Elastic Beanstalk
- Kubernetes のコンテナ化のための AWS ECS/EKS
- 完全管理型クラスター環境のための AWS Fargate
- サーバーのセットアップまたは管理無しでコードタスクを実行するための AWS Lambda（サーバーレスモデル）

デプロイメントモデルに関する決定は、将来、どれだけメンテナンスが必要となるかに影響を及ぼします。ワークロードをクラウドに移行させればさせるほど、より進化したサービスや、アプリケーションのサポートに必要な、区別されていない作業を低減させるサーバーレスのようなテクノロジーを導入することが理に適ったことになります。



New Relic Oneを使って AWS Lambda KPIをモニターしましょう。

テクノロジーリファクタリングを見る

アプリケーションのコードベースで何を変え、何を改善すべきかを決めるだけでなく、IT 全体についてしなくてはならない作業の数を減らすのに役立つ新しいテクノロジーを検討することも重要です。インフラストラクチャ最適化はリホスティングまたはリプラットフォーミングの一環として進めるのが得策となります（くわしくはリホスティングとリプラットフォーミングのホワイトペーパーをご確認ください）。その一方で、データストアについては、より新しいテクノロジーの長所を活かすリファクタリングのメリットが数多くあります。

この数年で、データ保存の唯一の方法はリレーショナルデータベース管理システム（RDBMS）であるという考えは時代遅れのものとなりました。現在、アプリケーションがデータを使う方法をサポートするデータストアには、AWS からのものを含め、以下のような様々なものがあります：

- Amazon DynamoDB、キー / 値ドキュメントデータベース
- Amazon Elasticsearch Service、迅速なデータ検索のための完全管理サービス
- Amazon Quantum Ledger Database (QLDB)、中央管理され、透明性が確保され、暗号化の検証が可能なトランザクションログ

クラウドにおけるコスト最適化を図るための 5 つの設計原則

1. 消費モデルを採用する（サーバーレス）。
2. 全般的な効率性を測定する。
3. データセンター運用のための支出を中止する。
4. 支出を分析し理由を明らかにする。
5. マネージドサービスを利用して、オーナーシップのコストを低減させる。

情報に基づくリファクタリングのアプローチの一環として、アプリケーションについて収集してきたデータをレビューし、どこで AWS のテクノロジーを活用すべきかを理解できるようにします。たとえば、アプリケーションが膨大な規模とディストリビューティッド（分散）アプローチを必要とする場合は、Amazon DynamoDB のような NoSQL データベースサービスへリファクタリングするのが有利かもしれません。

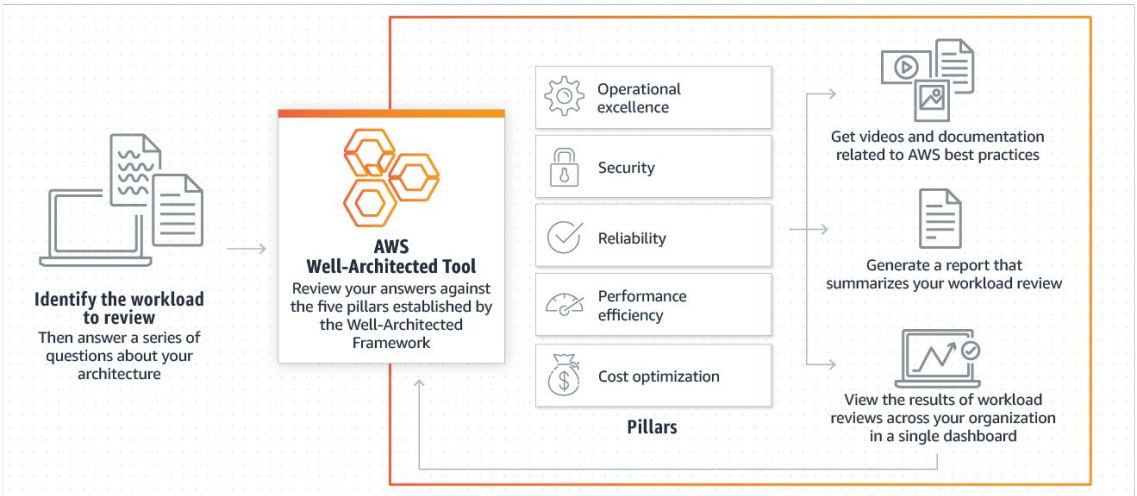
New Relic

同様に、クエリのいくつかが「like(のような)」の言葉を使用する場合は、リファクタリングの選択として Amazon Elasticsearch が適切かもしれません。

どのようなテクノロジーリファクタリングを計画している場合も、ビジネス、IT、エンドユーザーにおける利益が、変更のために要する時間と投資を必ず上回るようにすることが重要です。

AWS Well-Architected フレームワークの環境におけるリファクタリング

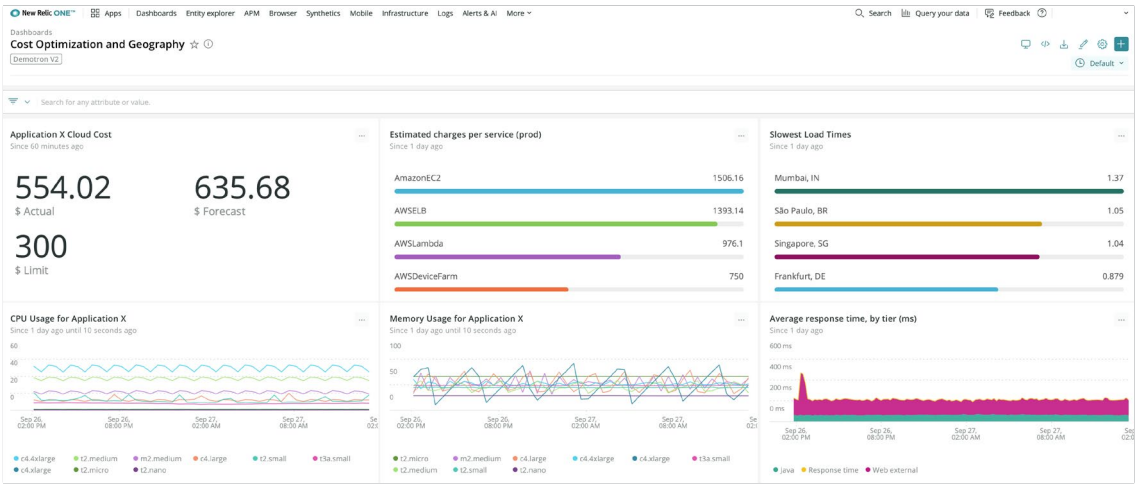
AWS Well-Architected フレームワークは AWS のアプリケーションについての判断のメリットとデメリットを把握するのに役立ちます。このフレームワークには、高い信頼性とセキュリティを持ち、効果的でコスト効率が高いシステムをクラウドでデザインし、運用するための、アーキテクチャに関するベストプラクティスが含まれています。このプラットフォームは、最もパワフルなクラウドベースのオブザーバビリティプラットフォームである New Relic One と共に、ベストプラクティスに照らしてアプリケーションを一貫して測定し、改善の余地がある部分を特定する方法を提供します。



AWS Well-Architected フレームワーク

1. コストの最適化

コストの最適化とは、単に費やされた額だけではなく、アプリケーションが使われたことによるインパクトに関するものでもあります。カスタムイベントを通して New Relic One にビジネスデータを統合することにより、ウェブサイト上で顧客からもたらされる収益を追跡できます。これを New Relic One で地域ごとに分類することで、パフォーマンス改善の投資のどの部分が顧客に最も大きなインパクトを与えているかに関するインサイトが得られます。



AWSの支出を、インフラストラクチャ、アプリケーション、地域別KPIと並べて追跡しましょう。

たとえば、ブラジルでのパフォーマンスは望ましいものではありませんが、それは主要ターゲットの一つではないので、投資をするのは理に適ったことでないかもしれません。現在、日本にはごく少数のエンドユーザーしかいませんが、ビジネスを成長させるため、企業はこの地域に大きな投資をしたいと思うかもしれません。そのためには、日本のユーザー体験を優先させる必要があります。こうしたパフォーマンスデータと企業の目標の組合せは、適切な投資をする方法に反映されなければなりません。

コストが最適化されているかどうかを知るもう一つの方法は、New Relic One でアプリケーションの Apdex スコアを見ることです。アプリケーションの経費の面から、それは理に適ったものか?もしくは、現在の Apdex では、顧客の満足を維持するために過剰提供（そして過剰支出）しているか？

2. パフォーマンス効率

フレームワークの一環となっているこの柱は、アプリケーションのパフォーマンスというより、むしろリソースの使用状況および、どのくらい迅速かつ容易に物事を変更できるかに関するものです。デプロイメントの方法に基づき、開発者が新しいテクノロジーを試すことはどのくらい容易か？迅速にリファクタリングできるか？

リファクタリングを上手く機能させるためには、自動化およびインフラストラクチャのコード化を導入し、日々厳しさが増す今日の要求に対応し続けてゆく必要があります。以下をモニタリングし追跡することで、これらの分野における企業のパフォーマンスを評価できます：

- デプロイメントの頻度
- デプロイメントの成功
- 環境セットアップまでの所要時間

クラウドにおけるオペレーショナルエクセレンスを実現するための 6 つの設計原則

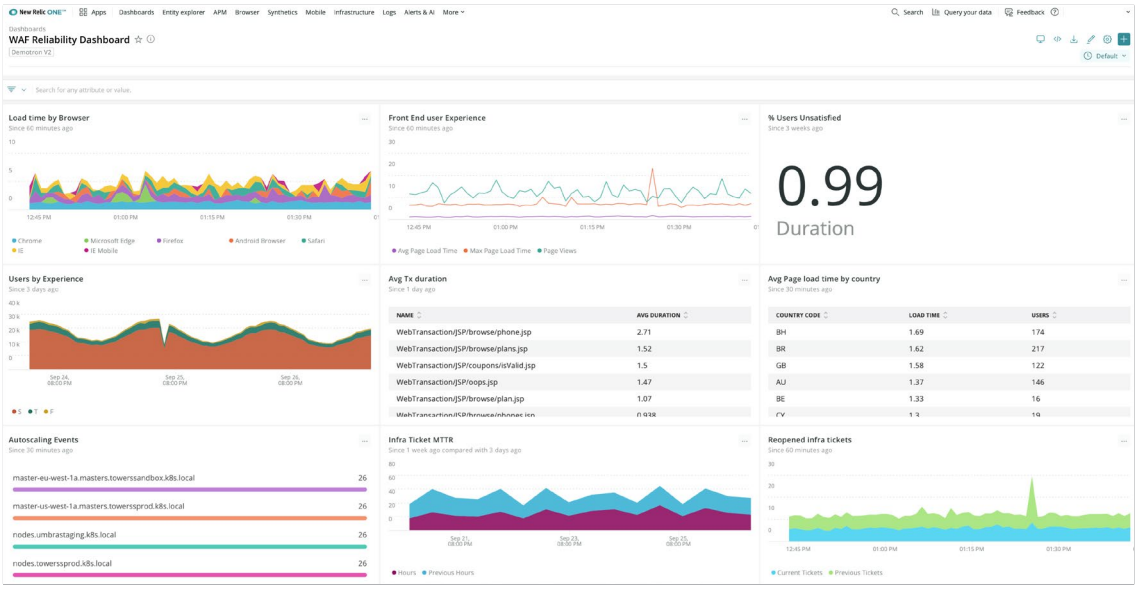
1. オペレーションをコードとして実施する。
2. ドキュメントに注釈をつける。
3. 小さい、可逆性のある変更を頻繁に行う。
4. 頻繁にオペレーション手順を改良する。
5. 失敗を予測する。
6. オペレーション上のあらゆる失敗から学ぶ。

リファクタリングの後にこれらのメトリクスがどのように変化したかを理解することは、リファクタリングの適切な選択をしたことを示す主な指標となります。

3. 信頼性

信頼性とは、アプリケーションが機能しているか故障しているかだけに留まることではありません。ピークタイムにアプリケーションの機能をどれだけ上手く拡大できるかを追跡し理解することも、同様に重要なことです。トラフィックが急増した時に、何が起こるか？テスト結果は、どのくらい良好か?開発、テスト、生産で問題が見つかったか？解決までの平均時間（MTTR）はどのくらいか？

リファクタリングでは、常にプロジェクト後の安定性を向上させることが目標となります。自動化は人的エラーのリスクを低減させるため、安定性の向上に大きく役立ちます。リファクタリングの取り組み後に、企業は単に問題の数が減ったことを確認するだけでなく、アプリケーションのライフサイクルの初期に、それらの問題を特定しなければならないのはそのためです。



Well-Architected フレームワーク、アプリケーション、エンドユーザーKPIを表示したダッシュボード。

4. オペレーショナルエクセレンス

この柱は、オペレーションにおけるプロセスおよび手順、チームのパフォーマンスおよびコラボレーションの改善に関するものです。さらに、実験を受容する企業文化の育成に関するものでもあります。改善を追跡するために、ここではアプリケーションのパフォーマンスだけでなく、アプリケーションを自動運用するためのアプリケーションのコードベースと、インフラストラクチャのコード化の使用状況をモニターすることが必要になります。

チームがプロセスの改善と自動化のために、インフラストラクチャのコード化をいかに上手く活用しているかに注目します。これは、特定のテンプレートにおける AWS CloudFormation のデプロイメントの数と、それらのデプロイメントに関連するエラーの数など、クラウド環境における指標をモニターすることにより実行できます。さらに、Terraform または CloudFormation のコードのリポジトリが変更される頻度にも、常に注意を払うようにします。開発、テスト、生産過程のどこで問題が見つかったか？

5. セキュリティ

この原則は、セキュリティ態勢の改善に関するものです。リファクタリングの目的の一つは、維持するコードの量を減らし、ひいては複雑性、テスト、セキュリティ上の欠陥が発生する可能性を低減させることです。AWS でのリファクタリングにおけるもう一つの目的は、企業が自分で管理しなければならないテクノロジーの数を減らすことです。これは、企業が可視性とパッチ管理の責任をもつインフラストラクチャの数を減らすことになります。維持するコードとパッチするインフラストラクチャの数が減れば、企業が保護すべき攻撃対象となる面が低減され、セキュリティ態勢が改善されます。

結論

AWS 環境のためのアプリケーションのリファクタリング序論として、最も高い結果を達成すると共に、その取り組みがアプリケーション、顧客やビジネスに与えるインパクトを提示するうえで、オブザーバビリティに基づくリファクタリングのアプローチが、なぜ役立つか、いかに役立つかに関する基本事項を説明しました。以下のデータを活用しながら、継続的にアプリケーションのモダン化を図り素晴らしい利益を実現するために、何をいつリファクタリングするかを決定するうえで、New Relic One をいかに活用できるかについて述べました :

- インフラストラクチャ
- アプリケーションの実行状況
- エンドユーザー体験
- ビジネス KPI
- DevOps プロセス
- コードベースの変更と問題

リファクタリングに関して最後に言いたいのは、それが単一かつ一度だけのプロジェクトではないということです。それは、パフォーマンス、顧客体験、そしてビジネス結果の向上を図るために、アプリケーションを常に改善し続けてゆくプロセスだと考えてください。

newrelic.com/aws にアクセスし、リファクタリングまたはアプリケーションおよびインフラストラクチャモダン化のフェーズで、New Relic One がいかに役立つかに関する情報をご覧ください。

New Relic