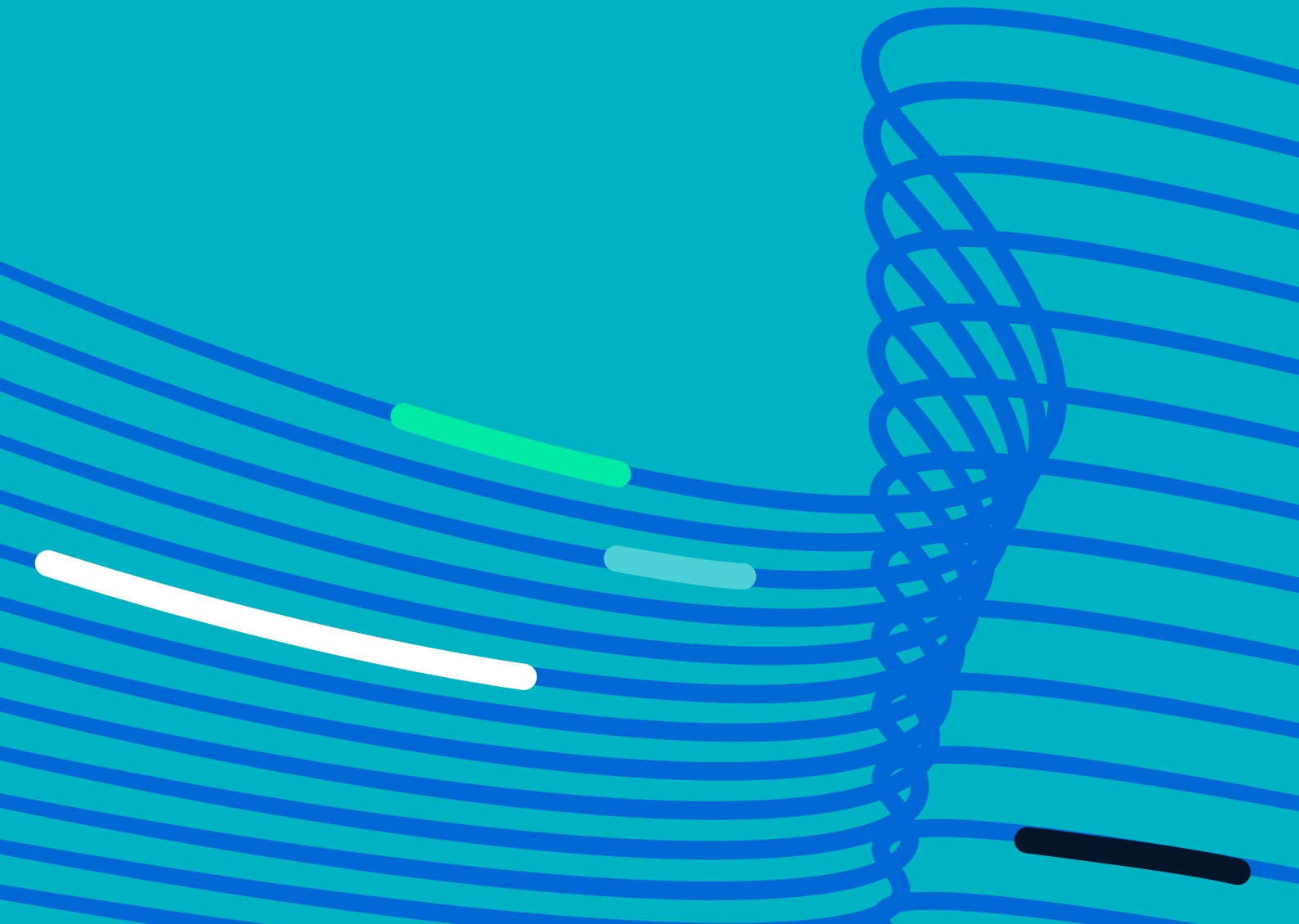
New Relic

White Paper

AWS によるモダン化：

「オブザーバビリティ駆動型」リプラットフォーミングを始める

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 2 部（全 3 部）



概要

組織は基本的に、アプリケーションをアマゾン・ウェブ・サービス（AWS）に移行することによるアプリケーションのモダン化に取り組んでいます。その目的は、既存のアプリケーションのパフォーマンスから信頼性および顧客体験に至るまで、あらゆる面で最適化することにあります。ビジネスの目標と測定可能な結果は通常、以下のような事柄の組み合わせです。

- 信頼性とパフォーマンスを向上させることで顧客体験を向上する
- 運用コストと労力を削減する
- 新機能の市場投入までの期間を短縮する
- 技術とビジネスの機動力を育み、強化する

既存のアプリケーションをモダン化する方法にはさまざまなものがありますが、このホワイトペーパーでは、コンポーネント・モダン化としても知られる、リプラットフォーミングのアプローチに注目します。AWS におけるアプリケーションのモダン化の 6 つのアプローチ（リタイア、リパーチェス、リテイン、リホスティング、リプラットフォーミングおよびリファクタリング）についての基本を学ぶには、当社の電子書籍「The Enterprise Guide to Continuous Application Modernization（アプリケーションの継続的なモダン化のためのエンタープライズ版ガイド）」をお読みください。

リホスティングではビジネスロジックに変更を加えることなく既存のアプリケーションを AWS に移行しますが、リプラットフォーミングではさらに多くのことが必要となります。具体的には、アプリケーションのコンポーネントを 1 つまたは複数モダン化することで最適化を実現します。コンポーネントとは本質的にはエンドポイントであって、標準化されたされた API を通じてアプリケーションが交信する相手となる、ウェブサーバー、リレーショナルデータベースやメッセジングシステムが該当します。

リプラットフォーミングは、リホスティングよりややリスクが高いものの、リファクタリングほど複雑でなく、リスクと複雑さに関してはモダン化の「中間地点」と分類できます。コンポーネントのモダン化には通常、ある程度の変更が必要であるからです。それにもかかわらず、リプラットフォーミングのアプローチが正しい判断である場合をどのように知ることができるのでしょうか？このアプローチに最適なアプリケーションはどれですか？さらに重要なのは、AWS 上でアプリケーションをリプラットフォーミングする労力を合理化し、すべてのメリットを最大化しながら、プロジェクトのリスクをさらに軽減するにはどうすればよいのか、という問題です。

ここでヒントとなるのが、オブザーバビリティとベストプラクティスです。このホワイトペーパーでは、AWS と密接に連携している New Relic の知識と経験を紹介し、お客様の取り組みと成果を最適化するための情報に基づく意思決定を行うためにはどうすればよいのかについて考えます。

リプラットフォーミングとは、そしてそのメリットは何でしょうか?

リホスティングとは、アプリケーションを基本的に現状のままで AWS に移行するリフトアンドシフトのアプローチであり、コードの変更を一切伴いません。これに対してリプラットフォーミングは、リフトアンドシフトのアプローチをやや進化させたものです。お客様は、アプリケーションの観察結果を基にして、情報に基づく判断を行い、アプリケーションを構成する複数のコンポーネントを変更します。また、それを行うことで最適化します。

誤解のないように言うと、コンポーネントを変更してもビジネスロジックは変更されません。ただし、構成、設定、破棄、および管理のルールを変更する必要はあります。例えば、データ接続や API のコーディングは変更できますが、その変更がアプリケーションのビジネスロジックに影響を与えることはありません。リプラットフォーミングプロジェクトの一環としてコンポーネントを変更する際は、このルールに従う必要があります。一方、リレーショナルデータベースを使用するアプリケーションをオブジェクトストアに変更するにはビジネスロジックの変更が必要です。この場合、アプリケーションの動作の仕方が根本的に変わるため、リプラットフォーミングプロジェクトではなくリファクタリングプロジェクトとなります。

AWS 上でのリプラットフォーミングの例をいくつか見てみましょう。

- Amazon Relational Database Service（Amazon RDS）などの管理されたリレーショナルデータベースサービスへの変更、または Amazon Aurora などの従量制課金データベースへの移行
- AWS Elastic Beanstalk を使用したアプリケーションのデプロイ
- セルフマネージド Apache Kafka（オンプレミス、AWS、またはその他の SaaS Kafka）から Apache Kafka 向け Amazon Managed Streaming（MSK）への移行
- 独自の証明書インフラストラクチャを運営する代わりに AWS Certificate Manager を利用する
- セルフマネージド Kubernetes 環境から Amazon Elastic Kubernetes Service（EKS）への移行

これらの例に共通しているのは、コードがかなり標準化された API を通じてコンポーネントとやり取りしていることです。API を微調整する必要が生じる可能性は高いとはいえ、ビジネスロジックは影響を受けないはずです。

リプラットフォーミングプロジェクトを行う理由は何ですか?メリットは変更するコンポーネントによって異なりますが、以下のような結果が含まれる可能性があります。

- ビジネスの成長に対応するスケーラビリティを高める
- 信頼性とパフォーマンスを高め、顧客体験を向上させる
- ソフトウェアライセンスとリソースの使用にかかるコストを削減する
- アプリケーションのセキュリティ表面積を縮小する
- 管理作業と関連する時間およびコストを削減する
- 情報に基づく意思決定を行う能力を高める

リスクは何でしょうか、そしてどうすればそれらを最小限に抑えられるのでしょうか?

どのようなプロジェクトでも、リスクを最小限に抑える第一歩は、関連するリスクを理解することです。この種のモダン化のリスクは比較的低いとはいえ、アプリケーションのリプラットフォーミングに際して特定のリスクに遭遇する可能性があります。こうしたリスクには以下のようなものがあります。

- アプリケーションの依存関係を理解していない、および / または過度に積極的で複雑なプロジェクト目標
- リプラットフォーミングに際して自動化の仕組みが導入されていない、または改善されていない（アプリケーションの運用に引き続き手作業が必要である）ため、コストが削減されない
- 予期しないエラーによるパフォーマンスまたは可用性の低下のせいで顧客体験に悪影響が及ぶ
- 予期しないエラーおよび / または依存関係のせいでプロジェクトに要する時間が元々の計画を超過する
- アプリケーションと新しいプラットフォームとの不適合性を見極められない

これらの、およびその他のリスクは、リプラットフォーミングプロジェクトの前後、および最中に導出されたオブザーバビリティデータによって通知、および導かれるベストプラクティスを使用することで、回避または最小化できます。

互換性分析

既存のアプリケーション、その構造、およびビジネス要件を慎重に評価し、新しいターゲットプラットフォームとの互換性を判断することが重要です。考慮事項には、以下のような要因が含まれる可能性があります。

- 新しいデータベースサービスが以前のデータベースと同じ要件に対応していない
- アプリケーションと新しいプラットフォーム上で利用可能なオペレーティングシステムとの間で互換性がない（この場合、コンテナのリプラットフォーム戦略の使用を検討する必要があるかもしれません）
- 新しいプラットフォームはアプリケーションやビジネスのセキュリティ要件（HIPAA、FedRAMP、GDPR など）を満たしていない

アプリケーションとそのビジネス要件に関連するすべてのアプリケーション要因について、互換性を確認し、確保します。互換性がない場合、アプリケーションにはリプラットフォーミングではなくリファクタリングのほうが適している可能性があります。

オブザーバビリティによってリプラットフォーミングを成功させる

モダン化プロジェクトを成功させるには、アプリケーション、顧客体験、およびビジネス成果を詳細に可視化する必要があります。つまり、包括的なテレメトリデータをすぐに利用できれば、どのアプリケーションをリプラットフォーミングし、どのコンポーネントを変更する必要があるのか、情報に基づいて判断できるようになります。プロジェクト全体のリスクを軽減し、プロジェクトを合理化し、成功を測定して証明するためにも、オブザーバビリティと可視性が不可欠です。

リプラットフォーミングの検討対象となるアプリケーションに完全なオブザーバビリティを確保するには、アプリケーションに計装を施し、行動を起こす前にテレメトリデータを収集する必要があります。プロジェクトの最中も終了した後もデータを追跡し続けることになります。一般に、以下のデータを収集すべきです。

- エンドユーザー体験
- アプリケーションのパフォーマンス
- アプリケーションの依存関係
- アプリケーションの問題
- リソースの使用量
- 既存のコンポーネント

候補を特定するため、アプリケーションのエンドポイント、バージョン、およびアプリケーションに対するパフォーマンスへの影響を確認します。「ブラックボックス化」された特定の機能に対する明確な API を備えたアプリケーションを探すのです。接続をコンポーネントに依存させることが成功の鍵です。New Relic One を使用して、そうしたすべての依存関係を見つけ、確認することが重要です。

最初にやるべきことは、アプリケーションのサービスマップを確認することです。New Relic One のサービスマップは、アプリケーション、データベース、ホスト、サーバー、アウトオブプロセスサービスなどの接続と依存関係を示すアーキテクチャを可視的でカスタマイズ可能な形で表現したものです。

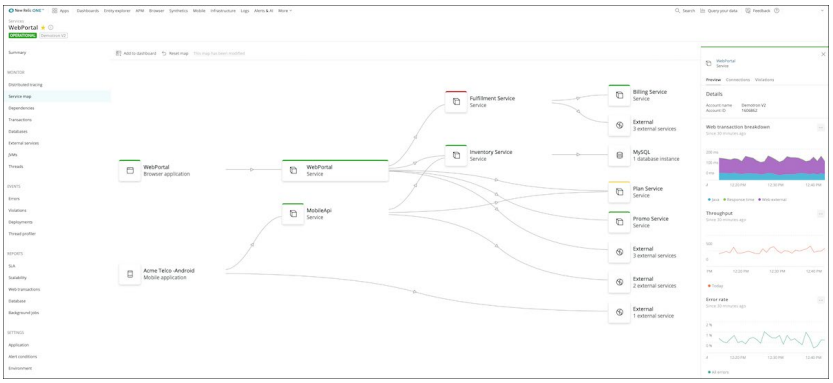


図 1: サービスマップを使用してアーキテクチャ内のアプリおよびサービスがどのように接続し、連携しているかを理解

コンポーネントを深く理解する

AWS 上で New Relic One のホストインテグレーションを使用した既存の構成要素への理解を深めましょう。それにより、AWS サービスに関するデータのフィルタリング、分析、クエリが行えるようになります。AWS インテグレーションの完全なリストと詳細については、当社のドキュメントページを参照してください。

New Relic

データを手に入れれば、情報に基づくリプラットフォーミングプロジェクトの手順を通じて作業を開始する準備が整います。

1. サービスマップ、エンドユーザー体験、およびアプリケーションのパフォーマンスデータを使用して、候補を特定します。
2. 既知の問題を対象にアプリケーションを分析します。
3. コンポーネントのパフォーマンスを詳細に把握するため、コンポーネントをホストインテグレーションサーバーにインストールします。
4. 変更するコンポーネントを絞り込み、選択します。
5. AWS Well-Architected フレームワークのベストプラクティスを適用します。
6. オートメーションを使用して、デプロイ、テスト、トランジションを実行します。

プロジェクトの前後および最中でオブザーバビリティを活用します。

オブザーバビリティは、プロジェクトの意思決定と計画に不可欠であるだけでなく、アプリケーションのコンポーネントをうまく変更するための確信と洞察を得るための最善の方法でもあります。その理由は、プロジェクト全体でアプリケーション、インフラストラクチャ、コンポーネント、エンドユーザー体験、ビジネスの成功に対する可視性を維持することで、複数の利点が得られるからです。適切なデータがあれば、リスクを増大させることなく作業を加速し、問題を予測して回避し、複雑さを最小限に抑え、AWS 環境でのアプリケーションを最大限に最適化できます。

リプラットフォーミングプロジェクトの各段階でのオブザーバビリティの使用方法は以下の通りです。

前 : リプラットフォーミングの候補となる現行の環境のアプリケーションに計装機能を装備し、New Relic One でテレメトリデータを収集してアプリケーションのパフォーマンス、エンドユーザーのエクスペリエンス、リソース消費、エラー率、アプリケーションの稼働率、その他の重要な KPI を把握します。これらの測定はリプラットフォーミングの最中と以降における比較のためのベースラインメトリクスとなります。また、このデータは、アプリケーションについて特定されている体系的な問題への対応するための計画立案も可能とします。収集した情報を活用し、リプラットフォーミングを考えているコンポーネントの既存のプロファイルと、現在の問題を把握します。そして、ビジネスの目標への効果と影響に基づき、どの候補を優先するか考えます。

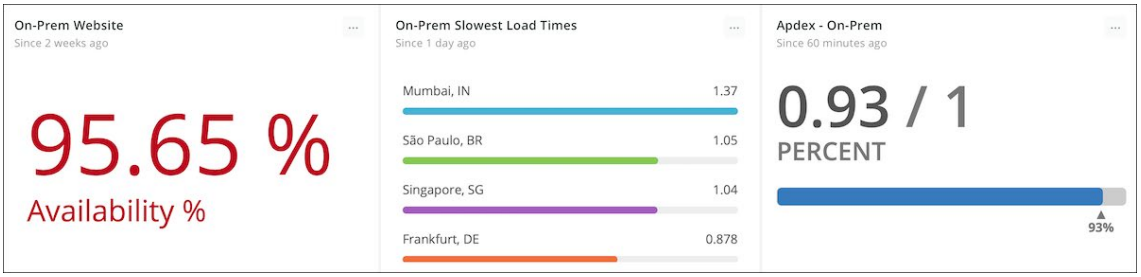


図 2: ベースラインダッシュボードはすべての KPI をまとめて表示。これにより変更前の状態が確定される。

最中 : リプラットフォーミングプロジェクトに着手するときは、移行による問題発生や速度低下が絶対に発生しないよう、アプリケーションをテストすることが重要です。リプラットフォーミングの最中に New Relic One を使用してアプリケーションコードと外部サービスのパフォーマンスをチェックし、「前」のフェーズで確認したリプラットフォーミング着手前のベースラインと比較します。こうすることで、同一条件での比較を徹底することができます。

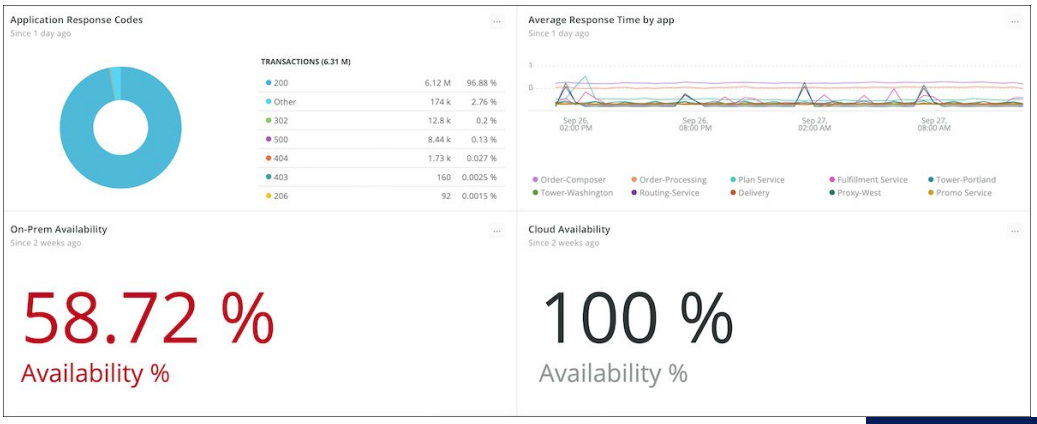


図 3: リプラットフォーミングの最中に問題と障害をすばやく特定

以降 : リプラットフォーミングが完了したら New Relic One を使用して成功レベルを測定して証明し、パフォーマンスの向上、リソースの効率的な使用、および容易な操作性を目的としたアプリケーションのさらなる最適化の可能性を特定します。

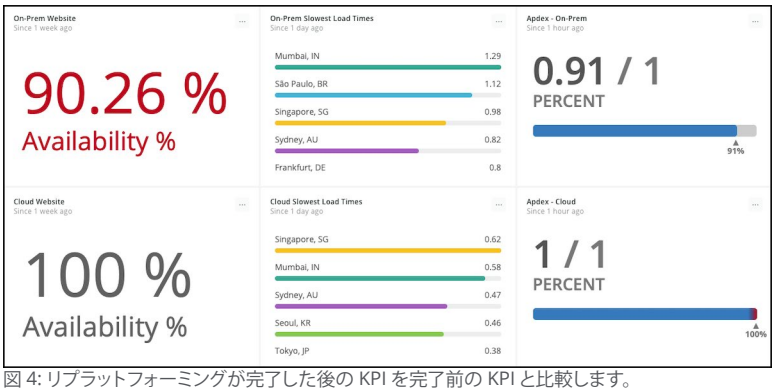


図 4: リプラットフォーミングが完了した後の KPI を完了前の KPI と比較します。

リプラットフォーミングの成功を測定し、トラッキングする 1 つの方法が、New Relic One と New Relic クラウドオブザーバビリティフレームワークを AWS Well-Architected フレームワークのベストプラクティスと共に活用することです。Well-Architected フレームワークは、効果的なパフォーマンス、コスト最適化、信頼性、オペレーショナルエクセレンス、セキュリティからなる、5 つの柱に支えられており、アプリケーションのための、セキュアで、高性能で、高いレジリエンスを備えた、効果的なインフラストラクチャを構築できるように開発されました。2012 年に初めて導入されて以来、Well-Architected フレームワークは 8 回更新されています。

インフラストラクチャ・アズ・コード（IaC：コードとしてのインフラストラクチャ）の重要性を再確認しましょう。

強力なインフラストラクチャ・アズ・コードであるdevopsのベストプラクティスにより、試験運用をスピーディーに行うことができます。その理由は、環境を個別にセットアップおよび破棄できるdevopsの能力により、新しいコンポーネントまたはサービスに移行するメリットがあるかどうかをすばやく判断できるからです。
インフラストラクチャのコード化の詳細については、AWS CloudFormation をお読みください。

New Relic

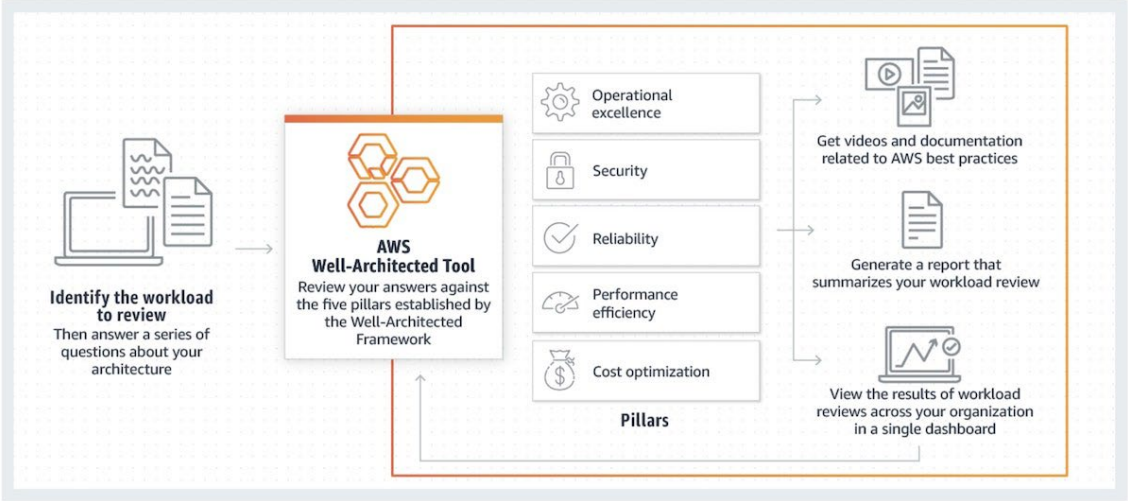


図 5: AWS Well-Architected フレームワークはクラウドアーキテクトが検証されたトラックレコードに基づいてインフラストラクチャの構築をサポート

AWS Well-Architected フレームワーク : 効果的なパフォーマンス

Well-Architected フレームワークの最初の柱に関するベストプラクティスは、リプラットフォーミングプロジェクトの前後と最中におけるリソースの使用効率をモニターおよび観測することです。New Relic One を使用して、リソースの使用状況がどのように変化しているのかを確認できます。どの程度の効率性を達成していますか?

リプラットフォーミングプロジェクトのきっかけはパフォーマンスの向上ではないかもしれませんが、それでもエンドユーザーへの影響を理解する必要があります。状況は改善されていますか?アプリケーションに左右される結果に影響を及ぼしましたか?例えば、ページの読み込みが速くなることでクリックスルーやコンバージョンが増加するなどです。この情報を可視化することは、システムの取り組みが収益にもたらす効果を評価するために不可欠です。

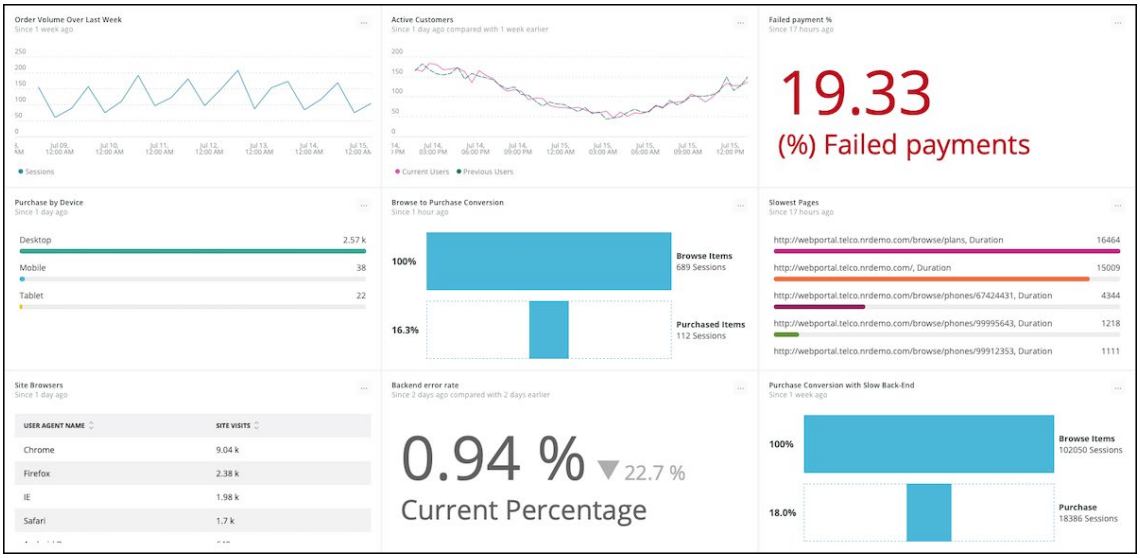


図 6: ビジネス、顧客、および業務に関する KPI を統合

New Relic.

AWS Well-Architected フレームワーク : コストの最適化

Apdex はウェブアプリケーションとサービスのレスポンスタイムについてユーザーの満足度を測定するための業界基準であり、ユーザーがアプリケーションにどれほど満足しているかを確認するために役立ちます。New Relic One を使用すれば、ご自分の Apdex をアプリケーションにかかっているコストを比較することができます。そのコストは、アプリケーションが果たす役割に対して、適切な投資となっているでしょうか?目指すべきエンドユーザーのエクスペリエンスとは、どのようなものでしょう?これらのこととコストのバランスをとり、余計な出費を避けてエンドユーザーのエクスペリエンスを確保することが目標となります。

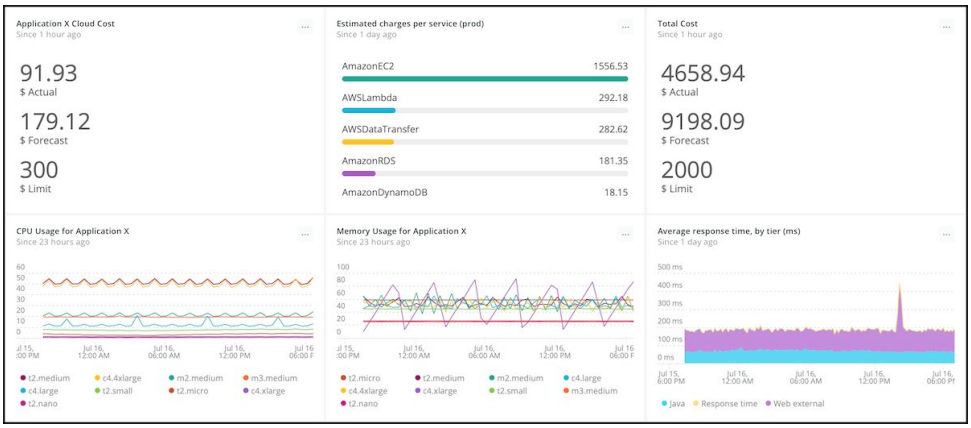


図 7: インフラストラクチャとアプリケーションの KPI を並べて表示し、AWS 関連予算を追跡

リプラットフォーミングとしてのコスト最適化の対象はライセンス料だけではないことに注意してください。コンポーネントを自主管理するために必要な人件費と、そのコンポーネントに関連する生産上の問題や機能停止のためのコストも最適化の対象です。従って、考慮すべきコストには下記のものを含めるべきです。

- 管理されているコンポーネントの数（コンポーネントのバージョンを含む）
- ライセンスコスト
- （パフォーマンス、信頼性、および拡張性のバランスに焦点を当てた）インフラストラクチャのコスト
- 運営コスト
- セキュリティフットプリントへの影響
- エラー管理と、コンポーネントに関連する作業のやり直しおよび問題の削減

AWS Well-Architected フレームワーク : 信頼性

今日のデジタルビジネスでは、重要なアプリケーションの稼働時間に対して厳しい要件が設けられています。最初に、サービス品質保証契約（SLA）と稼働時間の要件を明確に理解しましょう。次に、信頼性に関するメトリクスを比較し、SLA に対するアプリケーションのパフォーマンスレベルを確認します。アプリケーションの稼働率はどの水準にありますか?ユーザーはどのような頻度でエラーの影響を受けるか？

また、アプリケーションの処理能力は、予想される、また予想外の急激な負荷増大を含め、会社の最も忙しい 1 日に合わせてスケーリングして、信頼性を確保する必要があることも考慮してください。SLA の目標を達成し、顧客を満足させるには、協調して機能するスケーリングと信頼性の両方が必要です。

スケーリングはこの目的でどの程度有効に機能していますか?大半のアプリケーションでは、わずか 6 ヵ月後までであっても、スケーリングの要件を予測することは困難です。ユーザー数が少ない場合は、アプリケーションがどれだけ有効にスケーリングするかについてそれほど心配する必要はありませんが、ユーザー数が多い場合は、スケーリングは自動的に行われなければなりません。自動スケーリングのルールの微調整に時間を空費するべきではありません。ユーザー数からみて、適切なテクノロジーを採用していますか?使用量の増減への準備はできていますか?

New Relic One のホストインテグレーションを使用すると、コンポーネントの詳細なスケーリング情報を取得できます。拒否された接続、キューの長さ、再配置されるシャードなどのメトリクスに注目することができます。これらはいずれも、システムの規模が小さ過ぎることを示しています。一方、コンポーネントが過剰投資され、負荷のピークに対応するためのリソースが多過ぎ、負荷の谷間では無駄になる可能性があります。AWS のマネージドサービスを利用すると、こうした作業の多くはバックグラウンドで処理されます。

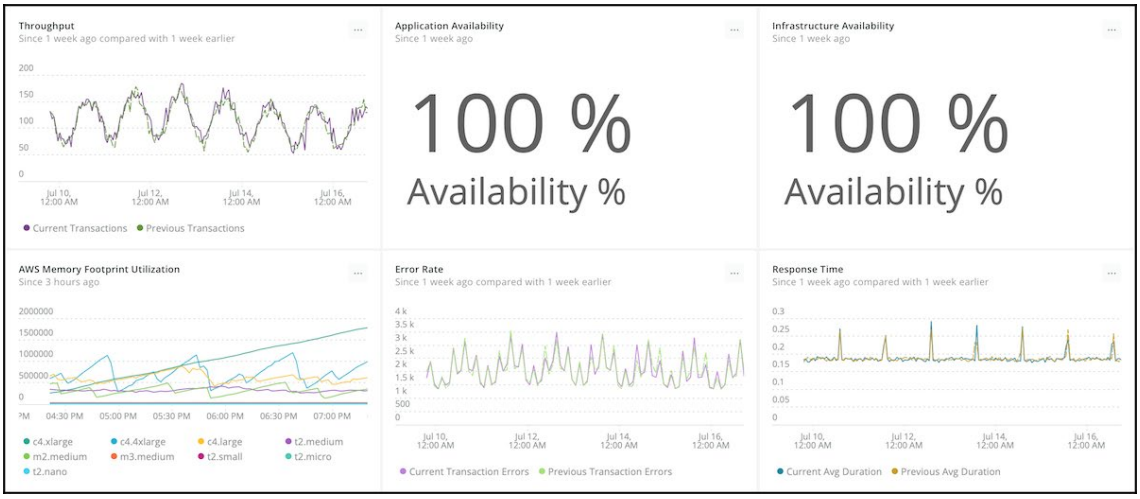


図 8: 可用性に関する KPI とバックエンドのメトリクスを表示するハイレベルのダッシュボード

Well-Architected フレームワークの主要な機能であるダッシュボードは、現在の信頼性を測定するだけでなく、将来の拡張性と信頼性を確保するために適切なテクノロジーを使用しているかどうかを評価することに留意してください。現在のユーザー基盤と予想される将来のユーザー基盤を比較して、シームレスでスケーラブルなテクノロジーがアプリケーションにどの程度使用されているかを把握できます。すべてのアプリケーションが完全にサーバーレスになるわけではありませんが、適切なデータがあれば、自分で管理しているアプリケーションに関してより良い選択ができます。

AWSを使用してウェブアプリケーションを大規模にデプロイする

このプレゼンテーションをAWSオンラインテックトークスで閲覧し、よりスケーラブルなウェブアプリケーションの作成方法をご覧ください。

New Relic.

AWS Well-Architected フレームワーク : オペレーショナルエクセレンス

リプラットフォーミングの実践に伴い、自動化する作業量を増やすための方法を探るべきです。例えば、マネージドサービスを採用すると開発者の時間が解放され、開発者は実験を行いやすくなります。新しい環境を簡単に立ち上げて新しいアイデアを試すことができるからです。開発作業は時間の経過とともに改善されていますか?機能の提供に要する期間をどの程度短縮することができますか?ビジネスに真の違いをもたらすタスクに費やす時間があることが分かるはずです。

特定のコンポーネントに関連するコードが様々な領域に分散している場合、変更が必要なすべての領域を特定すべきです。次に、以下のようなデータのモニターを開始します。

- コードの行数
- データのコンポーネント化の進捗度合い
- 実環境におけるデータに関連する問題
- どれくらい短期間に環境を整えられるか

データアクセスレイヤーをコンポーネント化すると、維持する必要のあるコードの行数が減ります。コードの行数が少ないと、エラーが減り、一般的にはより厳格にテストされたコードとなり、潜在的なセキュリティリスクとなり得る領域が縮小します。これを実現するには、DevOps の強力なプラクティスの活用に着手する必要があります。

Apache Kafka などのセルフマネージドコンポーネントのデプロイをスクリプト化するのは通常、AWS 上の CloudFormation テンプレートを使用する場合ほど容易ではありません。コンポーネントのデプロイメントと保守プロセスを合理化すると、管理しなければならない変動要素が少なくなります。

AWS Well-Architected フレームワーク : セキュリティ

セキュリティに関して理解すべき重要なメトリクスは、組織がマネージドサービスを使用することによりセキュリティの表面積と労力をどの程度削減しているかということです。セキュリティの観点から追跡、管理、およびパッチの適用が不要となったデータベースはどのくらい存在していますか?データベースの数に加えて、データベースの種類（MySQL、Oracle、Microsoft SQL Server など）と、トラッキングがすでに不要になっているバージョン（1.2、2.7、5.4.2 など）の数はそれぞれいくつありますか?保護する必要のあるデータベースの種類とバージョンの数が増えるほど、セキュリティ上の課題はより大きくなります。

例えば Amazon Aurora では、多くのセキュリティタスクが自動的に処理されるため、利用者は他の領域に集中できます。データを保護し、アクセス可能な人物を制御するための適切なルールを提供する必要は依然としてありますが、オペレーティングシステムのパッチや更新など低レベルのセキュリティ問題は AWS が管理します。サイバーセキュリティは技術と法制度の両方の問題であるため、組織のセキュリティおよびガバナンス要件と、AWS が契約で提供しているサービスに対してこうした要件をどのようにマッピングするかを理解することが重要です。

次のステップ

アプリケーションのモダン化は反復的なプロセスであり、実際には完了することはありません。リプラットフォーミングのメリットが得られる可能性のある追加コンポーネントを見い出すため、アプリケーション全体を継続的に精査すべきです。また、アプリケーションのモダン化における次のステップがリプラットフォーミングであるとは限りません。次の最善のステップは、アプリケーションをリファクターして（最新のクラウドサービス、アーキテクチャ、テクノロジーを利用するためにコードレベルの変更を加えて）、新たなレベルの品質、パフォーマンス、スケーラビリティ、信頼性、および柔軟性を実現することです。

New Relic One は、お客様がモダン化への取り組みのどの段階にあっても、アプリケーションをより高速化し、より費用対効果が高く、またよりリスクの少ない方法での最適化をサポートします。New Relic は、AWS アドバンストテクノロジーパートナーとしてとして 6 つのソリューション分野（移行、DevOps、コンテナ、モバイル、リテール、公的機関）で、技術的熟練および立証された顧客成功に基づき、AWS コンピテンシーの資格を付えられています。

newrelic.com/aws にアクセスし、リホスティ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 がいかに役立つかご確認ください。

New Relic