SRE・クラウド実践ガイド

高速化とコスト最適化を両立する、オブザーバビリティ戦略

new relic

Chapter 1

SREは オブザーバビリティからはじめる

- そのサービスレベル目標、高すぎませんか？

Contents

近年、デジタルサービスが企業の競争力を左右する重要な要素となる一方で、システムの複雑化・高度化が進み、サービス品質の維持は年々難しくなっています。そこで重要性を増しているのが、システムを理解し、問題の原因特定と改善を迅速に行うために欠かせない「オブザーバビリティ」という考え方です。

今回は、そのオブザーバビリティがどのようにSRE(Site Reliability Engineering:サイト信頼性エンジニアリング)と関係し、なぜ「100%の信頼性を目指すのは得策ではない」のかを解説します。

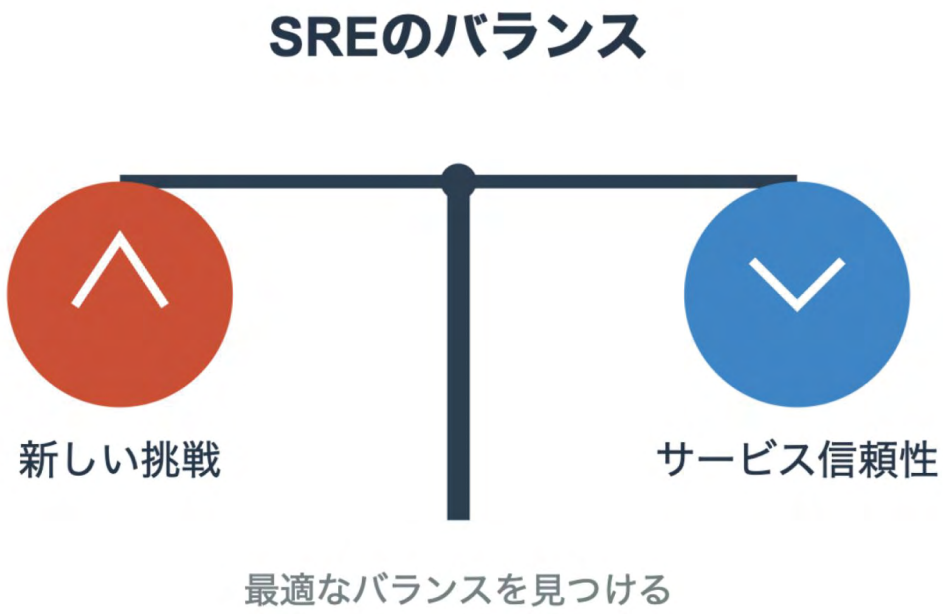
SREとは？ -「信頼性を最適化しながら変化スピードを落とさない」考え方

SREは、Googleが大規模サービスを運用する中で確立した手法で、文字通り「サイト(サービス)の信頼性をエンジニアリングする」というアプローチです。日本ではここ10年ほどで徐々に注目され、大企業からスタートアップまで、その考え方が広がりつつあります。

従来、運用チームは「障害を減らすこと」に専念し、開発チームは「新機能を出すスピード」を重視する構図がありました。SREの真髄は、両者の対立をいたずらに先鋭化させず、両立へと導く考え方にあります。

よく混同されがちなDevOpsには明確な定義こそありませんが、そちらは一般的には「文化やツール、プラクティスを通じて、開発と運用の壁を取り払う」ことをゴールとする手法とされています。

一方、SREでは、それをさらに具体化し、SLO(Service Level Objective:サービスレベル目標)やエラーバジェットなどの定量的な仕組みを通じて、高い信頼性と迅速なリリースの両立を図ります。



SREはサービスの信頼性と開発速度のバランスを取ること

そのサービスレベル目標、高すぎませんか？ - SLOとエラーバジェットの考え方

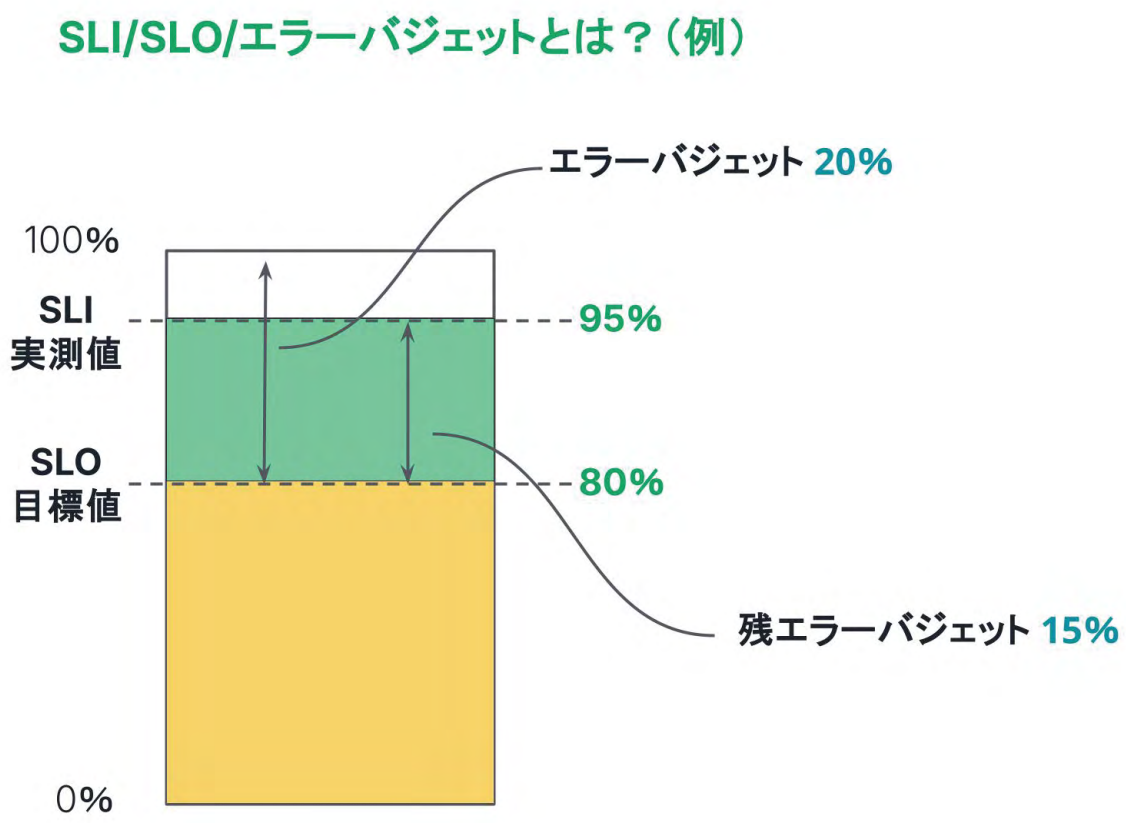
サービスレベルとSLOの本質

SLI(Service Level Indicator:サービスレベル指標)は、サービスの信頼性を数値で測定したものです。たとえばECサイトにおける「決済処理の完了率」といった、ユーザーが実際に体感する品質を具体的な数値として計測します。以下の図では、現在の実測値は95%となっています。

SLOは、このSLIに対する目標値を定めたものです。例えば「APIの応答時間はその99%が1秒以内であること」というように具体的な数値目標として設定します。

この目標は、ユーザーの期待、ビジネスへの影響、技術的な実現可能性、運用コストのバランスを考慮し、ユーザーが不満を感じないギリギリの水準 (以下の図の例では80%) に設定します。SLOを定めることで、チームは明確な行動指針を得られます。実測値 (SLI) が目標値 (SLO) を下回った場合は即座に回復に注力します。

一方、実測値が目標値を上回っている場合、その差分 (図の例では 95%-80%=15%) は「残エラーバジェット」として、新機能のリリースや実験的な改善施策に活用できます。また、目標値 (SLO) から100%までの差分 (図の例では100%-80%=20%) は「エラーバジェット」として、計画的な施策や改善に割り当てることができます。



SLI/SLO/エラーバジェットとは？

100%の信頼性追求がもたらす弊害、現実的なアプローチ

「すべての機能が常に完璧に動作する」といった目標は魅力的に感じられますが、現実的ではありません。ユーザーが本当に求めているのは「必要な時に必要な機能が適切に動作する」ことであり、たとえばECサイトであれば「商品を探して購入できること」が本質的な価値です。

過度な品質追求は、むしろサービスの成長を妨げます。すでに十分な水準にある機能を完璧にしようとして工数を投じすぎると、新機能の開発や本質的な改善が後回しになり、競合に対して致命的な遅れを取りかねません。

重要なのは「100%」という完璧な数値ではなく、「ユーザーが許容できる最小限の信頼性ライン」を見極めることです。これにより、適切なリソース配分が可能となり、イノベーションと信頼性のバランスを保ちながら持続可能な成長を実現できます。

オブザーバビリティはSREの第一歩

まずは「何を測るべきか」を定める

SREを実践する際の最初のステップは、いきなりSLOを設定することではありません。まず「ユーザーにとって大切な体験は何か」を見極め、その体験を的確に表すSLIを定義することから始めます。たとえばページの表示速度やエラー率、処理の成功率など、ユーザー体験を数値化できる指標を特定し、それを正確に測定できる状態を整えることが重要です。

この計測の仕組み(オブザーバビリティ)がなければ、後続のすべての活動が勘や推測に頼ることになってしまいます。ログやメトリクスがバラバラに管理されていて、サービスの状態を正確に把握できず原因特定もできない状況では、どんなに素晴らしい目標を掲げても意味がありません。

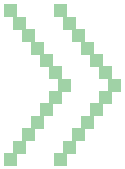
実際のサービスレベルを継続的に測定・把握できる状態になって初めて、ビジネス部門を含む関係者と「どこまで改善するべきか」というSLOの議論が可能になります。こうした段階を踏むことで、実効性のある改善活動を展開できます。

具体例：ECサイトの応答速度をSLOにしたケース

たとえばECサイトにおいて「ユーザーの購入完了までの導線で、95%以上が問題なく完了できる」というSLOを設定したとしましょう。もし障害が起きて顧客が離脱するような事態が発生しても「どの購入プロセスで問題が発生しているのか」「特定の機能だけで問題が起きているのか」が把握できないと、対応が遅れるばかりです。

そこでオブザーバビリティを整備しておけば、「商品の在庫確認プロセスで失敗が集中している」「特定キャンペーンの購入処理で完了率が95%を下回っている」といった問題を素早く特定できます。

さらにサイト上で発生した重要な操作を記録したイベントデータを確認すれば、どのユーザーがいつどんな商品でエラーに遭遇したのかも、追跡可能です。その結果、エラーバジェット内で計画的な改善が進められるだけでなく、新機能のリリースや実験的な施策の実施判断も、データに基づいて適切に行えるようになります。



組織全体で取り組むのがSRE - インフラチームだけの話ではない

「SRE＝インフラのチーム名が変わっただけ」「オブザーバビリティ＝運用部門の取り組み」といった誤解は根強いものの、実際にはビジネス部門や開発チームを含めた全社的な協力が欠かせません。

たとえばSLOの値を設定する際には、「顧客の期待値」「企業ブランド」「コストとのバランス」など、ビジネス視点が不可欠です。

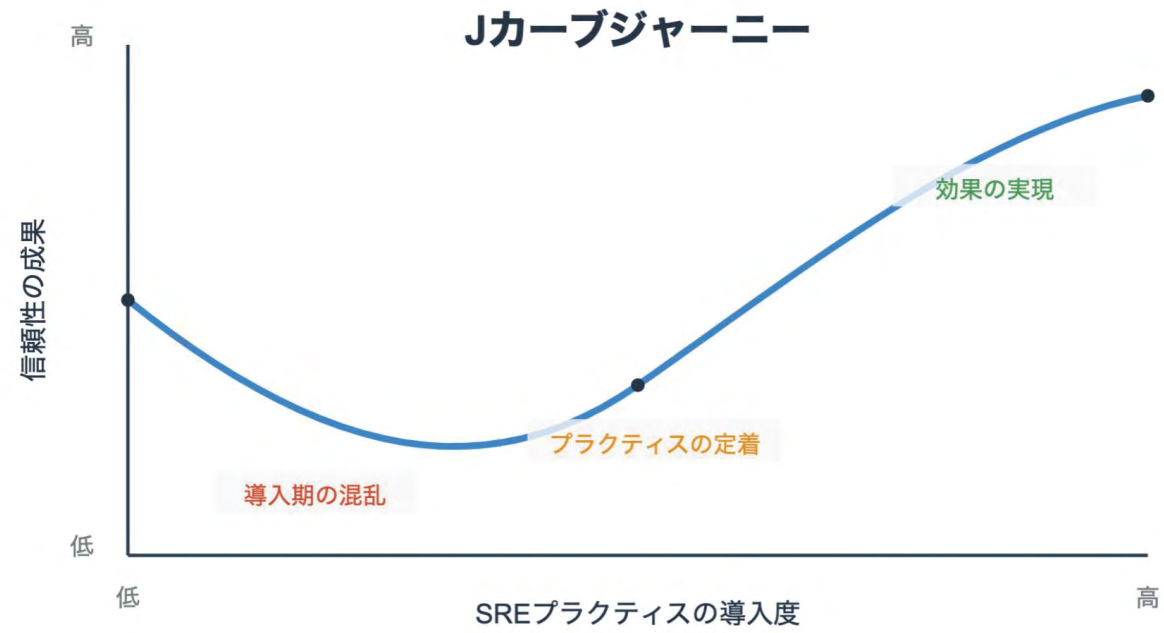
「ユーザーがストレスを感じる応答速度はどの程度か？」「そのSLOを達成するために必要な対応は何か？」「障害を検知した後、どのように対応し、どのようにエスカレーションするか？」といった意思決定を1つの“共通言語”で行うためにも、オブザーバビリティの仕組みが土台となります。全社的にサービスの状態を可視化しておくことで、的確かつ迅速な判断が可能になります。

「Jカーブ現象」に注意、導入初期こそ苦しいが、諦めないで

なぜ一時的に悪化するのか

SREやオブザーバビリティを導入して間もない頃は、ツールの整備や新たなプロセス定義に手間取るため、一時的に障害件数や対応工数が増加しているように見えることがあります。これを「Jカーブ」と呼ぶこともあります。

組織内で「どこまで自動化するか」「どのデータを可視化するか」が定まらず混乱が生じやすい時期であり、新たなツールの導入や学習が追いつかず、現場がストレスを感じやすい段階でもあります。



出典: Jカーブ、2022年の「State of DevOps Report」より
(ソフトウェア開発組織の動向を毎年調査しているレポート)

それでも続ける意義

しかし、ここで諦めてしまうと、SREの真の価値である「迅速なリリースと高い信頼性の両立」を得る前に終わってしまいます。

短期的に苦しくても、オブザーバビリティを活用して障害原因を迅速に把握し、再発防止策を講じられる成功体験を少しずつ積み重ねることが大切です。

適切に導入が進めば、やがて運用効率や障害検知スピードが大きく改善し、結果として“J”の形で上向きのカーブを描くように、サービス品質と開発速度の両方が向上していきます。

ポストモーテムと定期的なパフォーマンス観測の両輪で進める

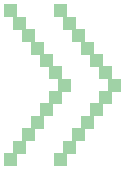
起きたことを「振り返る」だけでは不足、未然に防ぐための定期的なパフォーマンス観測

SREの文化では、障害発生中の調査にオブザーバビリティを活用するだけでなく、事後に「ポストモーテム」と呼ばれる振り返りを行い、再発防止策を探るのが定番です。

ただし、この振り返り自体が目的化してしまい、形式的なミーティングに終始してしまうケースもあります。具体的な改善策に結びつかなければ、同じ障害が繰り返されるリスクが高まります。

一歩進んだ取り組みとして、チームを超えた定期的なパフォーマンス観測を行い、潜在的な問題を早期に発見して対策を講じることが重要です。観測データ (オブザーバビリティ) をもとに「負荷が徐々に高まっている箇所はないか？」「ステージング環境やサンドボックスで問題の兆候が見られないか？」などをチェックします。

こうした検証を週次の「パフォーマンス定点観測会」として組織的なシステム運用に組み込むことで、本番環境で重大な障害が発生する前に対策を打ちやすくなり、障害を防ぐ可能性が格段に高まります。



“100%の信頼性”は目指さない。ビジネスを止めないために

SREは、必要十分な信頼性を保ちつつ、迅速なビジネス展開を実現するための全社的な取り組みです。「100%の信頼性」を追い求めることは、コスト面やイノベーションの面で大きなデメリットを伴います。そのため、ユーザーが不満を抱かないぎりぎりのラインをビジネス視点で見極め、それをSLOとして定義することが現実的です。

そして、そのSLOを達成するためには、サービスの状態を正確に把握し問題解決を支えるオブザーバビリティが欠かせません。さらに、ビジネス・開発・運用の各部門が「何が価値を下げる障害なのか」を共有し、全社的な合意のもとでプロセスを確実に回すことで、障害対応に追われる状況から、余裕をもって新機能を開発し続けられる体制へと移行できます。

初期段階ではJカーブによる一時的な混乱が生じるかもしれませんが、継続的な取り組みによって運用効率とビジネス成長の両面で大きなメリットを享受できるはずです。

new relic

Chapter 2

クラウド移行の価値を説明できないあなたへ

- 期待と現実のギャップを埋める戦略

Contents

近年、コスト削減とビジネスアジリティ向上への期待から、自社のシステムをオンプレミス環境からクラウド環境へとリフトし、コンテナやマイクロサービスを導入する企業はますます増加傾向にあります。

しかし、クラウドへの移行を進める過程で、「本当に費用対効果が向上しているのか？」「従来よりも運用がかえって複雑になっていないか？」という根本的な疑問に気づくものの、それらを確認できる手段に乏しく、具体的な対応策に至らないケースが多く見受けられます。

さらに、クラウド運用中の思わぬトラブルやコストの高騰、またユーザー体験の劣化や応答遅延が発生している状況にもかかわらず、どこから手をつければいいか分からずに四苦八苦するといった悩みも珍しくないようです。せっかくクラウドに移行したのに、頻発しがちなこのような状況に対して、どのような手立てを講じればいいのでしょうか？

本章ではまず、クラウド移行を「クラウドリフト」「クラウドシフト」「クラウド運用」の３段階に分け、各段階で発生しがちな課題を整理します。そのうえで、オブザーバビリティを活用することで実現する解決策を見ていきます。

インフラ監視とログ監視の活用やそのモダン化にとどまらずに、新たな武器としてオブザーバビリティを活用し、必要に応じてFinOpsの考え方も取り入れることで、クラウドのメリットを十分に活かしながら無駄な支出を抑え、ビジネスアジリティを維持することができます。運用に変革が起こり、クラウドがもたらす真の価値を享受でき、自信を持って説明もできるようになるでしょう。

クラウドリフト時の課題

オンプレ思考がもたらす過剰リソースの罠

オンプレミスのサーバ環境を大きく作り替えずにクラウドへ移行するクラウドリフトの段階では、物理的なハードウェア管理から解放される利点が大きくあります。しかし、オンプレ時代と同じ台数や同じ設定でインスタンスを稼働させてしまうと、実際には必要以上のリソースを確保しコスト増になっているケースがあります。

移行後にクラウドの請求額が高止まりしていても、「オンプレ時代とどれだけ差があるのか」「移行したことで本当にどれだけ効果が出ているのか」を説明するための基準が曖昧だと、調整が進まないまま割高なコストを払い続ける事態に陥りがちです。

また、インフラ監視を、従来の手法のままでクラウドへ移行するだけ、またはクラウド対応のSaaS(Software as a Service)監視に単に切り替えるだけにとどまって、動的なリソース管理やアプリケーションレベルの問題に手が回らないケースも多く見受けられます。

必要以上にインスタンスを増やしてしまい、コストが予想以上にかさんでしまったり、仮想サーバの再起動だけで対処すると、結果として問題が先送りされ、コストが予想以上に増加することもあるでしょう。

「リフトはしたけれど、結局どれだけメリットが出たのか」と振り返った時に、むしろコストが増えているといったことに気づくという残念なケースがこれにあたります。

マイクロサービス時代の可視性喪失

クラウドを“ネイティブ”に活用しようとするクラウドシフトの段階では、さらにマネージドサービス、コンテナやマイクロサービス、サーバレスといった技術を本格導入することが一般的です。

アプリケーションやアーキテクチャ自体を細かく分割することで、開発と運用のスピードや柔軟性を高められますが、それと同時にログや監視の対象が飛躍的に増えてしまい、部分的に起きている障害や遅延を見落とすリスクが高まります。

たとえば、特定のサービスだけが応答遅延を起こしていても、全体は稼働しているため、その遅延が見逃されることが典型的です。インフラ監視とログ監視を高度に併用していても、コンテナやマネージドサービスで細分化された環境の全体像を把握することは容易ではありません。

顧客がECサイトにて買い物中に、「購入処理だけができない」といった致命的な問題が起き、売り上げにマイナスの影響が出ているのに、その原因がコンテナ内部のアプリケーションコードのパフォーマンスの問題なのか、外部の決済APIの応答遅延なのか、マネージドサービスの問題なのかなど、それらの切り分けが難しいという問題も浮上します。

結果として、マイクロサービス導入をはじめとしたクラウドシフトのメリットを感じる前に、トラブルシューティングの複雑さに翻弄される現場も少なくありません。

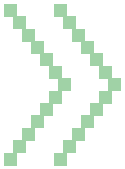
見えざるコストと従来監視手法の盲点

リフトとシフトを終えたとしても、運用中に直面する課題はさらに厄介です。たとえば、円安の影響や従量課金の増加によって、クラウド利用は大きく変わっていないにもかかわらず、クラウド利用の請求額が急激に上がることもあり得ます。

また、マネージドサービスを気軽に追加してしまい、そのまま利用率が低い状態で月額料金だけ発生しているケースも少なくありません。こうした予算やリソースの無駄・浪費を監視するには、単なるインフラ監視とログ監視では不十分です。

クラウドをあくまでインフラとしてのみ監視運用すると、アプリケーション内部の無駄な処理や、特定の処理が高コストを要している状況を把握できず、結果として使用率が高いまま無駄な出費を放置してしまう可能性があります。

たとえば、どのマネージドサービスが実際に高い付加価値を生み出し、どのコンテナが不必要に稼働しているのかを総合的に評価できる仕組みが求められます。



課題の共通点は「見えていない」こと - 全体可視化と運用改善をセットで進める

上記のように、リフトとシフト、そして運用中であっても、クラウド利活用にまつわる本質的な課題は「どこで何が起きているかを捉えられていない」ことにあります。

インフラ監視とログ監視は不可欠な要素ですが、それだけでアプリケーション内部のボトルネックやユーザー体験への影響、マネージドサービスの利用実態といった情報まで把握するのは困難です。

そこで大切になるのが、問題の把握はもちろん、その原因特定ができるように、オブザーバビリティをシステムと組織に導入することです。

アプリケーション動作とリソース使用状況を照合

リフト後、オンプレ感覚のまま稼働させているインスタンスやマネージドサービスが過剰であれば、それを見直すところから始めるのが有効です。

オブザーバビリティが整備されていれば、CPU負荷やメモリ消費だけでなく、アプリケーションがどの程度の処理時間や量なのかを計測できるため、「この時間帯はトラフィックが少ないからインスタンス数を減らせる」「深夜はサーバを停止しても影響がない」など、具体的な改善策を打ち出せるようになります。

さらに、リフト前後の状況を定量的に比較するデータがあれば、「月々のコストがどれだけ下がったか」「どれほどパフォーマンスが向上したか」を関係者に提示しやすくなるでしょう。

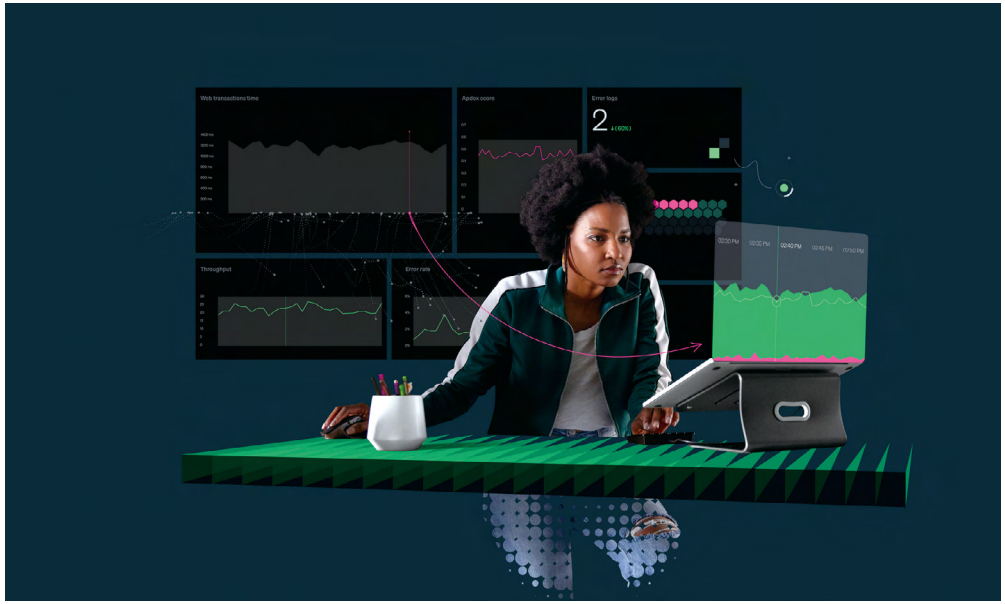
サービス間の通信やAPI遅延をリアルタイムに把握

マイクロサービス化やコンテナ化が進む環境では、サービス同士の通信を横断的に観測するトレース機能や、ユーザーが実際に操作した際のレスポンスタイム、ユーザー体験の状態を的確に表現するサービスレベルを可視化する仕組みが役立ちます。

サービス間の連携状況が一望できれば、遅延やエラーがどこで起きているのかを素早く切り分けられますし、ユーザー体験の劣化が売り上げや顧客満足度にどのような影響を与えているかも定量化できます。そうすることで、単に「サーバやインフラが落ちていないから問題ない」と判断するのではなく、具体的な改善ポイントに直結する運用が可能になります。

オブザーバビリティによって、細分化されたコンテナ内部の挙動やマネージドサービスの利用上の問題などを早期に発見できれば、手段の目的化、すなわち「コンテナを導入しただけでクラウドシフトは完了した」という誤解を払拭できます。運用チームがサービス全体を俯瞰しながら必要な修正やリソース調整を繰り返し実施し、真の意味でクラウドネイティブなスピードを享受できるようになるでしょう。

戦略的なリソース分析：インフラからアプリケーションコードまで投資対効果を最大化



運用中にコストが膨らむ一因は、為替リスクとともにマネージドサービスを含めたクラウドサービスの過剰利用にあります。

オブザーバビリティを活用して、各サービスやコンテナがいつ・どれだけ使われているのか、アプリケーションの動作に対してどの程度の付加価値を生み出しているのかを把握できれば、利用していない時間帯だけスケールダウンしたり、アクセスがほとんどないサーバやDBを夜間には停止したりといった調整策を講じやすくなります。

ここでFinOpsの概念を取り入れると、投資とリターンのバランスを見失わずにクラウド環境を最適化できます。FinOpsは、コスト削減だけを目的とするのではなく、必要な投資にはリソースをかけながらビジネスアジリティを落とさずに運用するための枠組みです。

オブザーバビリティが示すデータをもとに、「この部分は冗長でコストがかかりすぎている」「該当APIのコード自体を最適化すればサーバ台数を減らしつつレスポンス向上が望める」といったインサイトを活用し、最終的に“必要なところは残し、不要なところは削る”という理想的な運用を続けることができます。

たとえば、無駄なインスタンスを止めたりサイズを縮小したりする簡単な対応は当然として、オブザーバビリティを活用してキャッシュストアを活用するなどのアーキテクチャ自体の変更判断や、アプリケーションコードの改善によるサーバやクエリの改善、データベースのリソース削減など、リソースを俯瞰したうえでトータルで負荷削減を実践することができます。コスト削減とユーザー体験の両方の改善を実現することで、結果的にビジネスの成長へと繋がります。

クラウド採用だけで満足する「手段の目的化」を脱却し、活用の価値を証明

これまで見てきたように、クラウドへのリフトやシフト時、そしてその後の運用時においても、オンプレミス環境とは異なる可視性の喪失や監視要件の複雑化を背景とした、多くの課題が付きまといます。

為替リスクによるコスト増大はもちろん、コンテナ・マネージドサービスのブラックボックス化、インフラ監視とログ監視だけでは見抜けないアプリケーション内部の課題など、問題点を放置してしまうと「クラウドを導入したのに思ったほど効果が出ない」という残念な結果を招いてしまいます。

そうした状況を脱する鍵が、組織とシステムにオブザーバビリティを採用することです。インフラだけでなく、アプリケーション挙動やユーザー体験も観測できる仕組みを整えれば、クラウドならではの動的リソース管理やサービス細分化が引き起こすトラブルを素早く捉えられます。加えて、必要なデータを得られるようになれば、FinOps的な手法を用いて“ビジネスアジリティを失わずにコストを抑える”取り組みを継続できるはずです。

クラウド導入はあくまで手段であり、最終的な目的はビジネス成果の向上にあると考えられます。リフトとシフトによって環境を整えたあとこそ、オブザーバビリティを活かして稼働実態と費用対効果を丁寧に確認し、投資優先度を見極める姿勢が必要です。

そうすることで「なぜクラウドに移行したのか」の根拠を明快に示し、“コスト削減とビジネスのスピードアップを両立している”ことが確認できるようになり、より高いビジネスアジリティを獲得できるようになるでしょう。

※本記事は、TECH+連載「いまさら聞けないオブザーバビリティ」の一部を引用し、再構成したものです。 ➡連載ページはこちら

〈著者〉

清水 毅 Tsuyoshi Shimizu

New Relic株式会社 上席エヴァンジェリスト

オブザーバビリティの導入・利活用の支援や、市場に新しいアイディアを伝える啓蒙・啓発活動を通じて、日本のIT業界でのエンジニアの地位とビジネス貢献度を向上させることに深い情熱を持っている。これまでソフトウェア開発とSaaS化、インフラ設計・運用、SRE/セキュリティCoE、パブリッククラウドのソリューションアーキテクトの経験を積み、インラ、パフォーマンス、セキュリティの設計から運用までを幅広く得意分野としている。好きな食べ物はアボカド。