New Relic

ニューノーマルとしてのクラウドネイティブ

成功するために必要な環境の最適化は出来ていますか？

 | US +888-643-8776 | www.newrelic.com | www.twitter.com/newrelic | blog.newrelic.com

目次

はじめに

世界は躍進し続けています。

私たちは、比較的短い期間にクラウド時代からクラウドネイティブ時代へと進化を遂げました。クラウド時代には、アプリケーションの実行に必要なリソースは、クラウド内のサービスとして利用できましたが、クラウドネイティブ時代に入ると、クラウド内で実行するアプリケーションは、用途別に生成・最適化され、弾力性や回復性といったクラウドの利点をフルに活用できるようになりました。

業界ではクラウドネイティブがどこでも話題となり、分散システムやマイクロサービス、コンテナ、サーバーレスコンピューティング、その他の新興技術/アーキテクチャの役割などが取り沙汰されています。多くの組織は、クラウドアプローチの定義づけや最適化の途上にあります。その中でITの現場担当者と意思決定者の間には、クラウドネイティブの真の意味や、組織が最新技術とプラクティスを活用するための目標や期待をめぐって、かなりの混乱があります。

中でも、「クラウドネイティブアプローチによって組織はどのような成功が得られ、この技術はビジネス目標に対してどのようなインパクトを持つか?」という疑問が最大のものでしょう。

このeBookは、クラウドネイティブへの道に不可欠な技術的・文化的取組みを、それがITとビジネスの両方にもたらすプラスの成果と比較し直すことで、この基本的な疑問に答えることを意図しています。貴社のクラウド環境から最大の価値を引き出すためのベストプラクティスについても触れます。

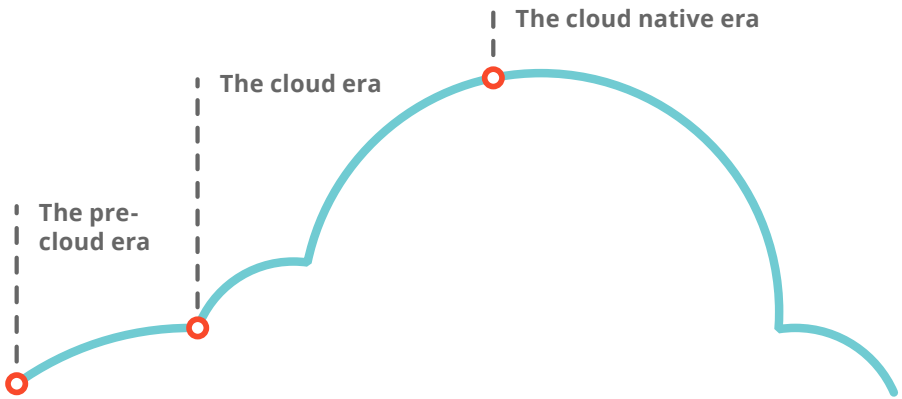
> *「経済的な観点から、クラウドネイティブテクノロジーはアプリケーションをこれまでよりずっと短いタイムラインでスケールして進化できるようにすることでクラウドの真価を発揮します。このスケーラビリティにより、収益の増加、効率の向上、またはより良い顧客体験の提供などのビジネスメリットが生み出されます。」*[1]

1: "The 451 Take on cloud native: Truly transformative for enterprise IT," 451 Research LLC, 2019

第1章

これまでの経緯

第1章 なぜクラウドネイティブが新潮流になったのか



クラウドの意味や最新ソフトウェアシステムから得られる価値をいかに最適化するかについて話す前に、これまでの経緯について振り返ってみることにしましょう。

クラウド前の時代

- **ソフトウェア開発：** ウォーターフォールアプローチを使用し、ソフトウェアの開発には長年かかり、新機能の公開はあまり頻繁ではありませんでした。開発、品質保証、運用は個別のプロセスで、多くの場合サイロ化されています。
- **インフラストラクチャ：** 物理的なハードウェアが構内に置かれ、サーバーの仮想化が登場すると、仮想サーバーの起用により、少数の大型物理サーバー上で運用が統合されるようになりました。
- **運営：** 運用タスクは大きく手作業に依存しており、そのためシステム運用範囲は限定されていました。時が経つにつれて、設定管理ツールにより物理および仮想インフラストラクチャのプロビジョニングおよび保守がしやすくなってきました。

クラウド時代

- **ソフトウェア開発：** アジャイルな開発とDevOpsアプローチがウォーターフォールアプローチに取って代わり、高品質なソフトウェアが短時間で開発できるようになりました。
- **インフラストラクチャ：** パブリッククラウドの出現により、アプリケーション運用に必要なリソースをサービスとして利用できるようになりました。会社はコアビジネスに専念し、ITインフラストラクチャの一部を外部に任せる方法としてクラウドを取り入れ、消費したリソースに対してのみコストを支払うようになります。
- **運営：** プロビジョニング、設定、スケーリング、自己回復を含むクラウドの運用および管理の自動化が普及しました。

クラウドネイティブ時代

- **ソフトウェア開発：** モノリシックなアプリケーションはマイクロサービスやサーバーレス機能など新しいソフトウェアアーキテクチャに代わり、クラウドコンピューティングの主要特性を活用しています。
- **インフラストラクチャ：** クラウド内では、プラットホームがサービスとして利用可能です。組織は多くの場合、マルチまたはハイブリッドクラウドアプローチを採用し、実行するのが最適な場所でアプリケーションやサービスをデプロイします。
- **運営：** 分散システム上で実行されているソフトウェアは、コンテナやコンテナオーケストレーションプラットホームなど、新しいデプロイメント方法で有効化されます。アプリケーションやインフラストラクチャの抽象化が増えたため、開発者は複雑さと取り組みながらも、改善された点を活用してシステムの健全性や性能に新しい方法でアプローチしています。

クラウドネイティブ時代へのこの進展によって、アプリケーションの単純なクラウド移行や「リフトアンドシフト」では、クラウドの真の利益を活用する上で十分でないことが明らかになっています。DORA（DevOps Research and Assessmentチーム）の2018年の調査によると、組織がどのようにクラウドを採用するかが重要で、以下の要件を満たしたチームが優れた実績を上げる可能性が高いことが示されています[2]:

- クラウドの本質的特徴の採用（オンデマンド・セルフサービス、幅広いネットワークアクセス、リソースの共用、スピーディな拡張性、計測可能なサービスであること）
- コードとしてのインフラ活用（IaC）、最新のデプロイメント、パイプライン実践
- サービスとしてのプラットホーム（PaaS）の利用
- クラウドネイティブデザインのベストプラクティスの採用

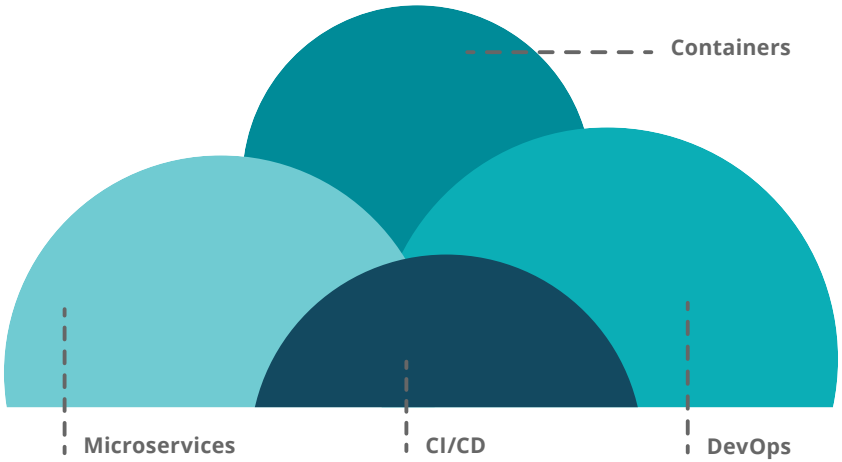
それでは、クラウドネイティブとは何を意味するか、それが組織がビジネス目標を達成するうえで不可欠である理由について話すことにしましょう。

2: Accelerate: State of DevOps, Strategies for a New Economy," DORA, 2018

第2章

クラウドネイティブ時代へようこそ

第2章 クラウドネイティブ時代へようこそ



多くの組織はクラウドネイティブイニシアティブを開始したばかりですが、すでにこれは業界最大のトレンドの一つとなっています。Cloud Native Computing Foundationの報告によると、本稼働でのクラウドネイティブテクノロジーの使用は、2017年12月から2018年8月の間に2倍以上に成長しています[3]。Capsule8、Signal Sciences、Duoからの調査結果によると、62%の会社がその新しいアプリケーションの半数以上においてクラウドネイティブテクノロジーに依存していることが示されています。この数字は2021年までには80%以上の伸びを見せることが予測されています[4]。

クラウドネイティブテクノロジーおよび実践がこのように急成長しているかを掘り下げる前に、まず幾つかの定義から始めましょう。

クラウドネイティブとは

クラウドネイティブの意味を説明できるシンプルな定義はありませんが、この用語の本質と使い方は以下のように定義できます: クラウドネイティブはアプリケーションとその環境を、デジタルおよびビジネス成果を変革する目的でクラウドコンピューティングの中核となる特性を最大利用すべく最適化することです。

クラウド環境で実行するために構築されたアプリケーションは次のようなものです:

- 分散コンピューティングリソース向けにデザイン/デプロイしてスケールメリットを活用する
- 作業負荷の変化に動的に対応できる拡張性
- 基礎的なプラットフォームとインフラの管理がなくなり、オンデマンドによるデプロイ/管理が容易に
- 障害からの回復性の高さ

> *「当社のスケールでは、エンジニアに非常に高速で移行するよう求めながら、サイトの運用を維持するという課題に直面しています。言葉を変えると、当社は、100%の生産性を実現すること、すなわち製品部門の誰もが機能を作成、変更、リリースできることと同時に、100%の可用性を維持すること、つまり当社のサイトがゲスト、ホスト、従業員に対して完全に利用可能であることを望んでいます…。これを達成するために、当社は、チームがサービスレベルのダッシュボードを活用して、ビジネスとシステムメトリックの両方を並べて見ることでサービスの大まかな概要を把握できるようにしています。」*
>
> ***Melanie Cebula、Airbnbのソフトウェアエンジニア***

クラウドネイティブが重要な理由は?

クラウドネイティブはクラウドがもたらす多くの利点をフルに活用できるため、組織のデジタルトランスフォーメーションおよびビジネス変革を達成する上で重要な役割を担います。

クラウドネイティブ戦略をうまく実践することで以下のことが達成され、より高い価値をより速く得ることができます:

- 顧客体験の改善とロイヤルティの醸成
- アジリティの向上と市場投入までの時間 (TTM) 短縮による競争優位性の確保
- 新しいビジネスモデルの実現による業界革新
- 業務効率の改善によるコストの削減
- 顧客への提供価値向上と新たな収益構造の構築による利益の増加

「クラウドネイティブコンピューティングは、…アプリケーションをマイクロサービスとしてデプロイし、各部分を独自のコンテナにパッケージして、これらのコンテナを動的にオーケストレートしてリソース使用率を最適化します。クラウドネイティブテクノロジーは、ソフトウェア開発者が優れた製品をより速く構築できるようにしてくれます。」

「クラウドネイティブテクノロジーは、組織がパブリック、プライベート、およびハイブリッドクラウドといった動的環境でスケーラブルアプリケーションを構築して実行できるようにしてくれます。」[5]

3: "CNCF Survey: Use of Cloud Native Technologies in Production Has Grown Over 200%," Cloud Native Computing Foundation, August 2018
4: "The State of Cloud Native Security," Capsule8, Signal Sciences, and Duo, 2018
5: Cloud Native Computing Foundation

第3章

クラウドネイティブになる：一朝一夕にはいかない

第3章 クラウドネイティブになる: 一朝一夕にはいかない

クラウドネイティブは既存のアプリケーションをクラウドに移行したり、新しいクラウドサービスのスイッチを入れるだけではありません。これはまた、クラウドで最初からアプリケーションを開発するだけでもありません。クラウドネイティブとは「アプリケーションをどのように開発して実行するか」であり、それらのアプリケーションをどこにデプロイするかだけではありません。

このため、これはクラウドサービスまたはインフラストラクチャを活用した瞬間にクラウドネイティブになれるわけではありません。そうではなく、プロセス、アーキテクチャ、文化、測定の根本的な変化が必要です。これらの変化には時間とコミットメントが必要で、サイクルは一定して繰り返されます。

クラウドネイティブを成功させるための3つのコア機能

ソフトウェア開発ライフサイクルの可変部分をすべて進化させ、クラウドの利点を活用するプロセスは、連続的かつ反復的なプロセスです。その過程を成功させるためには、どの組織もマスターしなければならない3つのコア機能として、可視性・実験・最適化があります。これらの機能は、最新クラウド環境における次の3つの主な課題を克服する助けになります:

- **変革は連続的:** 絶えず出続ける新技術およびアプローチは、終わることのない仕事を意味します。
- **最新環境は複雑:** スタック全体にわたりコンポーネントの数は増え続けており、追跡することは困難です。問題が起きた場合はなおさらです。
- **スケールし難い:** 初期の成功は育成と維持が困難で、妥当かつ予測可能な予算内でビジネスに対する測定可能なインパクトを示すのは困難です。

ベストプラクティスのヒント: 小さく始めて、大きく成長

当社のお客様の多くは、明確に定義されたユースケース、アプリケーション領域、あるいはビジネスユニットから始めて、クラウドネイティブアプローチの経験を積み、短期間で成功を実証することで、組織の残りの部門全体でのサポートを増やすことができることを経験しています。御社の初期の成功に基づいて、クラウドセンターオブエクセレンスを開始して、組織の残る分野がクラウドネイティブを理解して導入する手助けをしてください。

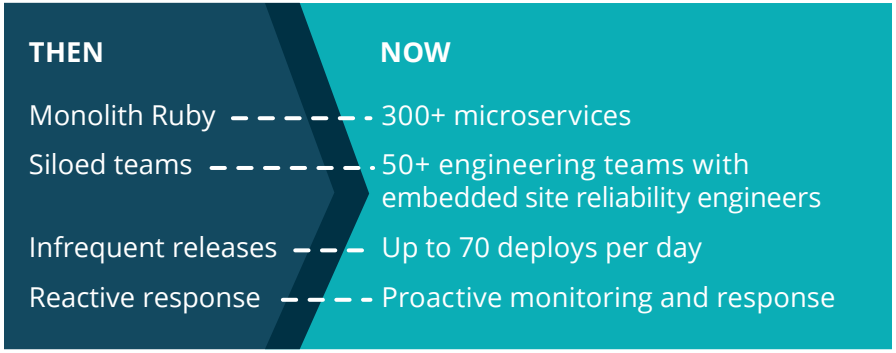
課題	コア機能	ITへのインパクト	ビジネスへのインパクト
変革が連続的	可視性	以下についての洞察と理解の深化をサポートします: • 効果的に評価して最新のクラウド技術を取り入れる • クラウドネイティブアプローチを活用するための最善の候補となるアプリケーションを決定する • モノリシックアプリケーションをマイクロサービスコンポーネントに分割する	現在の環境の可視性は、以下の明確な理解を助けます: • ビジネスに及ぼすアプリケーションパフォーマンスの現在の影響度 • ビジネス成果の改善に最大の機会をもたらすアプリケーションはどれか • アプリケーション/インフラストラクチャの各部間の依存関係とそれがエンドユーザーに与える影響 • クラウドネイティブに移行するIT/企業目標の評価方法
最新環境は複雑	実験	メトリクスの活用、迅速なフィードバック、詳細分析による以下の実現: • 実験と革新の文化を促進する • 複雑な分散システムですばやく失敗してすばやく修復する • 複雑な環境で成果を上げるためのDevOpsチーム用の閉じたフィードバックループを作成する	データドリブンアプローチは以下のようにチームを支援します: • エンドユーザー体験の最適化を目的として、各デプロイメントの前・最中・後に問題を迅速に見つけて解決する • 新しいソフトウェア/機能が顧客体験とビジネス成果に与える影響を理解する • 開発リソースと企業目標の整合性を確保する
スケールし難い	最適化	以下により分散システムの拡張性と性能を改善する: • 分散したコンテナ化された環境を自動化して管理する • アプリケーションの信頼性と拡張性を最大化して、ピーク時でもシームレスな顧客体験を確保する • ビジネスの進化を促進する取り組みに予算とリソースを集中する	クラウド環境の最適化は以下の面で貴社を支援します: • 迅速かつコスト効果のあるビジネス目標を達成する • 顧客体験の向上 • ITコストの正当性を証明し予測する • 競合他社を圧倒する

主な成功要因: クラウドネイティブ測定

測定の必要性は、最新環境の最適化の成功を支えるものです。この複雑な分散クラウドネイティブ環境ではコンテキストが重要です。データドリブンなアプローチなしではこれら3つの機能のどの成熟度や成果も測定することは不可能です。

これらは測定と状況が重要なだけでなく、測定のベストプラクティスはこれらの最新システムの要求を満たすように進化することが必要です。クラウドネイティブ環境では、従来方式の監視アプローチは3つの重要な方法に分かれる傾向があります:

- **複雑なシステムにまたがる可視性の欠如:** 新技術やサービスを採用するとき、エンドツーエンドの監視機能と複雑な環境内でのコンテキストが失われます。多くの場合、チームは個別のツールにわたる膨大なデータを手作業で相関して異常や依存関係を理解しようとします。
- **原因究明に対する洞察の弱さ:** 環境の複雑さと実験速度が増すにつれて、分散スタックの異なるレベルにまたがる異常を特定することが困難になるため、平均解決時間(MTTR)も増す傾向にあります。



New Relic の沿革: 当社のクラウドネイティブおよびDevOpsへの取り組みの経緯

ベストプラクティスのヒント: プラットフォームスキルを雇用するか育成する

最新技術とインフラストラクチャのシフトにより、新しい役職が出現しています。プラットフォームエンジニアとも呼ばれることがあるこれらの職務は、企業のクラウドネイティブプラットフォームの構築と進化の責任者です。スキルの面からは、この職務にある人は通常、インフラストラクチャソフトウェアおよびシステム管理に強い経歴がある必要があります。そのような職務を最終的に作成するかに関わらず、チームの誰が組織を率先して適切なクラウドツールとインフラストラクチャをデプロイするかを考慮すべきです。

- **システムおよびチームにまたがる監視ベストプラクティスの拡張性の困難さ:** サイロ内での技術的およびプロセスの成功を組織全体のベストプラクティスに変換することは困難です。測定可能で維持できるビジネスインパクトを持つ拡張性を妥当なコストで成功させることは困難になってきています。

このため、最新環境には最新測定方法が必要です。

「信頼性のすべては、何が行われているかを測定する能力に起因しています。つまり、プロセスまたはシステムの信頼性を改善するためには、最初に本稼働での性能を測定する必要があります。まとめると、私たちが実施するどの信頼性プロジェクトでも監視が第一であるということです。」

Karthik Nilakant, XeroのSREチームリーダー

第4章

マイクロサービスへの移行

第4章 マイクロサービスへの移行

クラウドネイティブアプローチの中心となる方針は、分散システムの活用です。これには、モノリシックアプリケーションを分解されたサーバーに返還するか、分散システムを最初から開発する作業が含まれます。いずれの方法も、大型で複雑なアプリケーションの連続的な提供とデプロイメントを可能にするという概念に基づいています。

マイクロサービスのアーキテクチャは、アプリケーションを合理的な最小サービスに分解するという優れたアプローチです。マイクロサービスアーキテクチャーの背後にある主要概念は、アプリケーションは互いにシームレスに機能する小さいサービスに分解したときに簡単に構築、変更、保守しやすくなるという概念です。

各マイクロサービスは独自のコンテナ、サービス、または機能内で実行されるため、開発者は各サービスを独立して構築、管理、スケールできます。適切に計装されると、マイクロサービスアーキテクチャーは以下が可能です:

- ソフトウェアデプロイメントの速度を上げ、拡張する
- アプリケーションの拡張性を改善し、拡張するかどうかの判断を最適化する
- ビジネスアジリティ(適応能力)を改善する
- 障害分離を改善する
- 小型で機敏なチームを可能にする

モノリスからマイクロサービスへ

モノリシックアプリケーションを識別してより基本的なマイクロサービスコンポーネントに分解することは重要な作業です。マイクロサービスへ移行するのにどのアプリケーション(および、もっと重要なことはそれらのアプリケーションのどのコンポーネント)が最も適しているかに関するコンテキストを得るために、定量および定性データを収集する必要があります。

サービスメッシュの理解

アプリケーションとそれをホストする環境が一層分散されるにつれて、主要な課題はすべての異なるサービス間のネットワークを維持することです。サービスメッシュは、サービスが相互に通信し合い、サービスの検出、負荷分散、認証機能をサポートできるようにするソフトウェア定義のネットワークレイヤーを提供することでこれを支援します。同時に、サービスメッシュはマイクロサービス環境にそれ自身のレイヤーを追加することになります。この複雑な環境の可視性と理解を得る一つの方法は、分散トレーシングによるもので、これはクラウドネイティブ監視ソリューションが分散システム全体を通して計装、コンテキストの伝播、記録、および要求の可視化を実現できるようにします。

アプリケーションが分解されたら、新しく作成されたマイクロサービスのうち、他のアプリケーションやサービスとの相互依存関係が多すぎるのはどれであるかを探るための洞察が必要です。また、特定のメトリックを分析して移行工程の効力と成果を測ることも必要です。そして、すべてのクラウドネイティブの場合と同様、これもボトルネックが移動する可能性があるため、繰り返し実施すべきプロセスとなります。追加データを収集して分析し、次の最適化候補を識別する、このプロセスを繰り返す必要があります。

> *「監視業務はマイクロサービスを採用するにつれて進化させる必要があります。サービスの不良行動が見られ、それが自分のコードでない場合もあるでしょうが、それに影響を及ぼしているのは下流または上流のサービスです。障害が起きて、サービスをすばやく回復する必要があります。改善された監視基準によって迅速な回復を達成します。」*
>
> *Melanie Cebula、Airbnbのソフトウェアエンジニア*

トランザクションコールボリュームやレスポンスタイムなど、アプリケーション性能を低下させユーザー体験に影響し得るモノリシックアプリケーションのコンポーネントを指し示す主要メトリックを収集することから開始できます。これらはマイクロサービスへの分解に関する主要候補です。ベースラインデータが提供するコンテキストを補助するために、アプリケーションを作成、デザイン、および/または保守した同僚に相談することも重要です。彼らの経験と提案は分解工程のガイドとなります。

主な成功要因: 可視性で複雑さを単純化する

マイクロサービスアーキテクチャの利点は多いものの、欠点も幾つかあります:

- マイクロサービスアーキテクチャは複雑
- 依存関係の追跡が困難
- 適正なサイジングが重要だが容易ではない
- 実装の成功には、DevOpsチーム間の思慮に富んだタイムリーなコミュニケーションが必要です

データによりマクロサービスアーキテクチャの利点を最適化し弱点を最小化できるということが、クラウドネイティブへの移行プロセスにおけるコア機能としての可視性から説明を始める理由です。技術およびビジネス目標の達成を可視性によって可能にするためには、モノリスおよびマイクロサービスに分解されたもの両方のアプリケーション性能、ならびにアプリケーションをサポートするインフラストラクチャの完全なビューが必要です。

クラウドネイティブで優れた監視ソリューションによりマイクロサービス間の相互接続依存関係を追跡することで、それらの接続の性能、運用特性、およびサービスレベル目標 (SLO) を監視して、問題が検出されたらアラートを受け取るようにできます。

ベストプラクティスのヒント: メトリクスを使用してマイクロサービスを最適化する

アプリケーションをマイクロサービスに分解した後、メトリクスを使用してまだ完全に独立していないマイクロサービスを識別します。たとえば、同じ下流サービスへの大量の繰り返し要求を持つマイクロサービスがあるかをチェックします。あれば、それはマイクロサービスが完全に分解されていないことを示します。スループットも重要な指標です。あるマイクロサービスへの毎分のコール数がアプリケーション全体のスループットに比べて著しく多い場合、これはサービスが分解されていないことを明確に示しています。

第5章

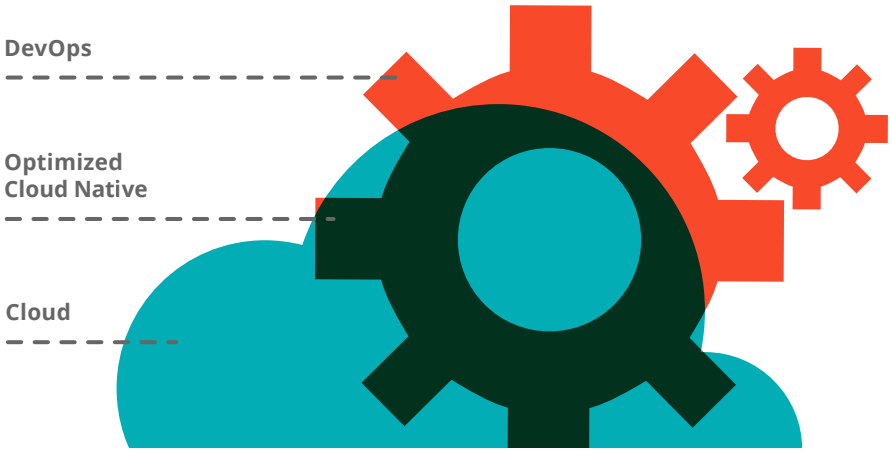
クラウドネイティブ DevOpsの導入

第5章 クラウドネイティブDevOpsの導入

クラウドネイティブについてのeBookでDevOpsについて話す理由は何でしょうか? それは、ツール、テクノロジー、アプリケーションアーキテクチャーを含むクラウドネイティブはIT文化、組織、プロセスの変化を必要とするからです。

テクノロジーメディアの*The New Stack* は、クラウドネイティブDevOpsを「クラウドネイティブアプローチで人々が一緒にアプリケーションを構築、デプロイ、管理する方法を説明するDevOpsツールコミュニティ内の一連の新たな原則」としてわかりやすく定義しています。[6]

すでにDevOps原則を採用しているか、始めたばかりかに関わらず、最新環境で成功するには実践、役割、ツール、ワークフローを進化させる準備ができている必要があります。これは、すでにDevOpsプロセスを使用している場合を含め、クラウドネイティブインフラストラクチャ、アーキテクチャ、およびデプロイメント方法により人々が働く方法が変化しているためです。



組織が技術および文化的な成熟度を増すに連れて、クラウドとDevOpsの公差部分は大きくなります。

すばやく失敗してすばやく回復する

クラウドネイティブDevOpsアプローチの重要な側面は、実験の文化を育成することです。最新アーキテクチャーをフルに活用することで、チームはインターネットスケールの機敏性とスピードを備えたソフトウェアを構築、テスト、デプロイできます。実験の文化を促進、推進すること(およびそれに伴う失敗)により、以下を可能にすることでビジネスに対するクラウドの価値を最大化できます:

- 新機能/体験を導入して市場化を速める
- インサイトドリブンのイノベーションで競合他社に挑戦する
- 新技術の採用を速める
- 大胆なアイデアを奨励する組織文化
- ビジネス目標および成果との緊密な連携

とはいうものの、以下のような減速要因に注意する必要があります:

- 開発者の疲労とアンバランスなコールローテーション
- コミュニケーション/コラボレーションの一時的中断やストレスの増大
- 正しい洞察や脈絡のない性能、用途、プロセスに対する変更

主な成功要因： 相関データ

実験の原動力はデータです。しかも、どのデータでも良いというわけではなく、*相関* データです。これは、分散システムが生成するテラバイトのデータ間を構文分析するのではなく、すべての実験、デプロイメント、配布に関する「それでは?」という問いに対する包括的な洞察とコンテキストを提供する 精選されたビューを持つことを意味します。

相関データと、それによる的を絞った行動は、チームがクラウドネイティブDevOpsの性能を測定・追跡し、最新環境におけるソフトウェア配布およびビジネス成果を最適化する過程を支援します。最も重要なことは、それにより、実験文化を育成するために必要な洞察をチームが得られることです。データにより、感情的になったり名指しで責任を擦り合ったりすることを排除し、スキル、経験、役割を超えた共通言語を創造します。

現実世界でのDevOpsの使用についての詳細は、DevOps の正しい使用をご覧ください。

ベストプラクティスのヒント： DevOpsの成功に関する以下のメトリクスの監視

DevOpsチームは開発、配布、本番環境で生じる問題への対応スピードを監視します。以下のメトリクスに関する目標の作成、トラッキングを検討してください:

- 変化に対するリードタイム
- コードリリースの頻度
- 平均解決時間 (MTTR)

6: “Guide to Cloud Native DevOps,” The New Stack, 2019.

第6章

クラウド環境の最適化

第6章 クラウド環境の最適化

最新クラウド環境は複雑で、絶えず変化し続けています。クラウドネイティブであることは、単純な方式に従う離散的な「AからB」への移行ではありません。それは、配布速度、スケーラビリティ、柔軟性、回復性において一層の向上を推進しながら、クラウドリソースのより効率的で効果的な使用を可能にする継続的な最適化ループを必要とする繰り返しプロセスです。

組織は通常、継続的に最新環境を最適化し続けるためにオートメーション、オーケストレーション、インストゥルメンテーションおよび可視性に依存します。

オートメーションとオーケストレーション

クラウドネイティブ環境のオートメーションは通常、クラウドインフラストラクチャーのプロビジョニング、設定、管理を行います。オーケストレーションは自動化したタスクをワークフローの一部として実行します。これは、環境全体を通してのオートメーションフローの組織化と見なすことができます。オートメーションとオーケストレーションの統合を実現するツールやプラットホームは多数あります。ソリューションには、クラウドベンダーが提供するものや、独立ソフトウェアベンダーが提供するもの、およびオープンソースのものがあります。

クラウド環境でのマイクロサービスおよび分散システムのスコープが増大するにつれて、それらの管理方法を最適化する方法が必要となります。分散システムの最適化およびオーケストレーションには様々な方法がありそれらは進化を続けていますが、そのうち現在よく使われている2つの方法は次のとおりです:

- **コンテナおよびコンテナオーケストレーションシステム：** マイクロサービスを使用するクラウドネイティブアプリケーションアーキテクチャーは多くの場合、アプリケーションのライブラリおよびデプロイメントプロセスのパッケージ化においてコンテナに依存します。コンテナ内のアプリケーションは他のコンテナ内のアプリケーションからは独立して稼働しますが、それでもカーネルやその他のオーケストレーションツールからの指示に応答します。そのようなオーケストレーションツールは、膨大な数のコンテナの管理には不可欠です。様々な代替方法の中で、Kubernetesは多数の組織およびクラウドサービスプロバイダーの中で優れたコンテナオーケストレーションプラットフォームとなっています。
- **サーバーレスサービスまたは機能：** サーバーレスとは、サービスを使用するために専用サーバーをスピンアップする必要のないすべてのクラウドサービスを含む一般用語です。サーバーレスマイクロサービスまたは機能（1つのことのみを行う非常に細分化されたマイクロサービス）は、必要なときにのみスピンアップされます。そして、これらが稼働しているときに使用したリソースに対してのみ支払いを行います。定常的に使用されていない機能やマイクロサービスに対しては、サーバーレスは真のコスト削減を実現しながらスケーラビリティおよび柔軟性を可能にします。

ベストプラクティスのヒント：Kubernetes環境で以下のプラクティスに従ってください

Kubernetes環境で最初のコンテナを構築してアプリケーションをデプロイする前に、以下のベストプラクティスに留意してください：

- コンテナイメージを小さく保ち構築時間とプル時間を改善する
- リソース要求と制限を使用してクラスタの誤用を回避する
- ネームスペースを利用して管理性、セキュリティ、性能を改善する

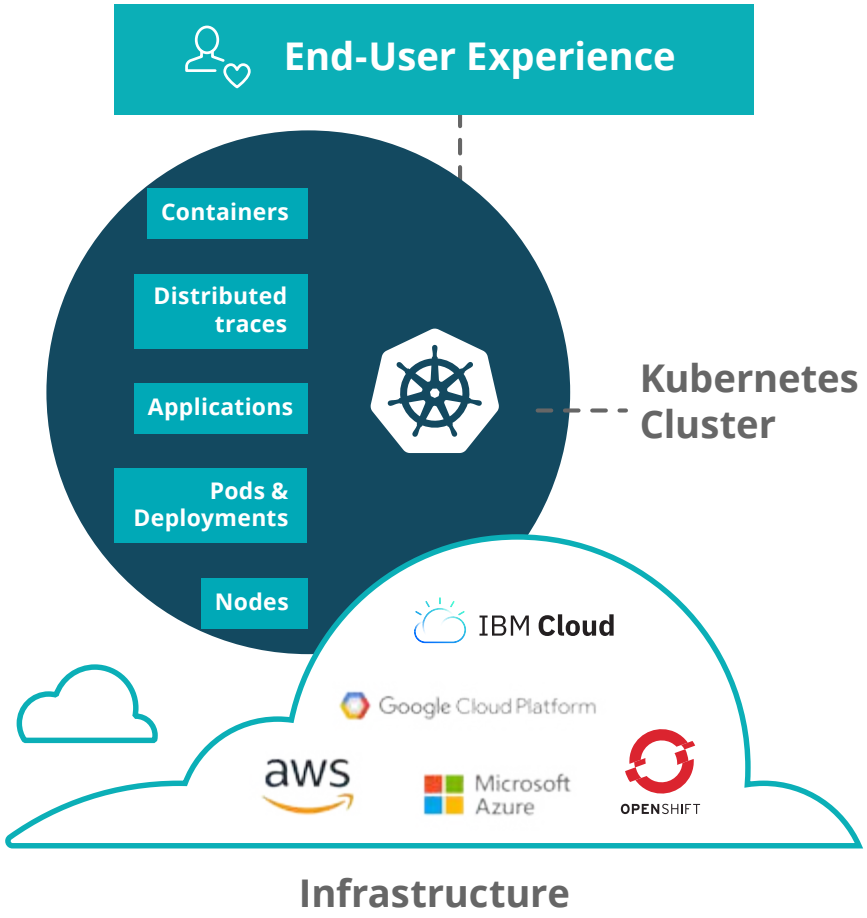
なぜクラウドネイティブアプローチにおいてコンテナオーケストレーションシステムまたはサーバーレス機能を使うのでしょうか? それは以下の実現ができるからです:

- 水平スケーリングの実現
- チームの動きの高速化
- クラウドリソースの最適化によるコスト効率の向上

インストゥルメンテーションと可視性

クラウドネイティブシステムがダイナミックに拡大するにつれ、使用するリソースとそれが予算に与える影響に対する支配力と可視性を維持するのはより困難になります。ここでも、最新のモニターが重要な役割を果たします。Kubernetesクラスタ内で実行されているノード、デプロイメント、ポッド、コンテナ、およびフロントエンドとバックエンドアプリケーション、分散トレース、ホストのデータとメタデータをプロアクティブに監視する必要があります。

また、アプリケーションやサービスが構築され使用される方法を日常的に緊密に監視する必要もあります。このデータ主導のアプローチにより、インスタンスを適切なサイズに設定し、データベースを微調整して、ストレージ用途を変更し、ロードバランサーの設定を改善し、アプリケーションを再構成することさえ可能です。クラウドの活用および消費の仕方に関するこうした考え方は、貴社の環境を最適化しクラウド予算正当化するのに役立ちます。



第7章

フィードバックループを閉じる

第7章 フィードバックループを閉じる

クラウドネイティブテクノロジーおよびアプローチを活用することで、会社はソフトウェアアップデートを1日に数千件までもより速く配布できるようになります。ただし、成功はバグ、新機能、性能、顧客体験、ターゲットのビジネス成果に関する緊密かつ迅速なフィードバックループに依存しています。

著作者、研究者、およびDevOpsのエキスパートであるGene Kimは、増幅されたフィードバックループを、DevOpsのプロセス、手順、実践を表現する「3つの方法」の1つとして挙げています。彼は、継続的に訂正できるようにフィードバックループを短縮および増幅するために、右から左へフィードバックループを作成する重要さを説いています。

この点では、クラウドネイティブの3つのコア機能に関する主要業績指標（KPI）を追跡し、チームがシステムを分析し改善するための基準として共通フレームが持てるようにすべきです:

コア機能	サンプルKPI
可視性	可用性、レスポンスタイム、Apdex、エラーレート、スループット、クラウドコスト
実験	デプロイ数、変更にかかる時間、平均解決時間（MTTR）
最適化	クラウドコスト、ポッド別エラー、コンテナメモリ使用率、欠如ポッド数、カスタマーセグメント別レスポンスタイム

成果に着目する

フィードバックループを一歩進めて、Mobius Frameworkは測定を使用して、DevOpsチームが提供する必要のあるビジネス成果に焦点を当てることができるようにするための成果への洞察を提供します。これは、前出のインフラストラクチャおよびアプリケーションの変更が最終的に顧客体験およびビジネス成果にいかにインパクトを及ぼすかの話題に戻ります。会社が測定値により成果を実感できなければ、取り組みが成功したと主張することは困難です。

速度を上げることはクラウドネイティブ環境の有効化の一つの目標ですが、その速度を適切な問題に向けるための適切なデータを使用して、誰もが確認できる測定可能な成果を上げなければ、速度を上げても無意味または非生産的となる可能性があります。

主な成功要因：インテリジェンス

最新クラウド環境では、かなりの量の「ノイズ」、つまり膨大な量のゴミデータがお使いの分散システムのあらゆる場所で生成される可能性があります。クラウド環境でエンドツーエンドの計装を行う場合、手作業のみでは、生成されたデータ量に追いつくだけでも人間には不可能です。フィードバックループ内で意味を成す方法でコンテキスト化や分析するなど、考えにも及びません。

そのため、チームが何が意味を成すかを分析し理解して、成果に関連付ける助けとなるすべてのデータを管理し相関するためのクラウドネイティブ監視ソリューションが必要となります。そのソリューションは以下を提供することが理想的です：

- **可視性の拡張：** 環境全体にわたって複雑さを単純化する
- **アナリティクスとインテリジェンス：** クエリされたイベント、分散トレース、集計データからのソースデータにより、チームが必要とする迅速なトラブルシューティングと最適化のコンテキストを提供する
- **処方的ソリューションとサポート：** 分散環境のオートメーションおよび最適化のための専門知識とガイダンスを提供する

次のステップ

クラウドネイティブアプローチの採用は常に迅速で容易であるとは限りませんが、得られるメリットによりその工程がITとビジネスの両面からやりがいのあるものとなります。

クラウドネイティブへの道を先に進むにつれて、これまでに説明した3つのコアコンピテンシーに関する組織の位置を評価するための4つの質問を自問してください：

1. エンドツーエンドの可視性があるか?
2. 自社のエンジニアはインフラストラクチャの変更、アプリケーションデプロイメント、およびエンドユーザー体験を結びつけることができるか?
3. 自社のシステムの信頼性と拡張性を改善するためにどのようなステップを取る必要があるかを知っているか?
4. クラウド成熟度の次のレベルに到達する明確な目標があるか?

クラウドネイティブ時代に突入する準備はできていますか?

newrelic.comで始めましょう

「プラットフォームの正常性は顧客体験に影響します。システムがオーバーロードされていたり、加えた変更が悪影響を及ぼしているかを理解する必要があります。システムの相互接続が与えられた場合、インフラストラクチャ、アプリケーション、顧客体験からメトリクスの監視をすべて一箇所に集めることができれば、問題を区分しやすくなります。」

***Karthik Nilakant**、Xeroのサイトリライアビリティエンジニア*