AWS の規範的ガイダンス

AWSクラウドのアプリケーションを最新化するための戦略

AWS の規範的ガイダンス: AWSクラウドのアプリケーションを最新化するための戦略

Table of Contents

アプリケーションのモダナイゼーション戦略AWS雲

Vijay Thumma、Amazon Web Services (AWS)

2020 年 12 月(ドキュメント履歴 (p. 12))

アプリケーションモダナイゼーション戦略を成功させるには、まずビジネスニーズを念頭に置き、次にテクノロジーに焦点を当てます。クラウドへの移行が加速するにつれ、組織はクラウドの採用を加速する方法と、アプリケーションの近代化への規範的なアプローチを模索してきました。Amazon Web Services (AWS) は、モダナイゼーションロードマップを評価、モダナイズ、管理の 3 つのフェーズに焦点を当てた個別の段階に分割して、アプリケーションのモダナイゼーションに取り組んでいます。この記事では、アプリケーションの評価と最新化の戦略について説明し、AWSプロフェッショナルサービスチームの長年にわたる企業支援の経験AWSクラウド導入およびアプリケーション近代化プロジェクトに参加しているお客様

この戦略は、AWS雲。ミッションクリティカルなアプリケーションを特定する方法、さまざまなモダナイゼーションアプローチ (リファクタリング、リアーキテクト、リライトなど) を評価する方法、スケーラビリティ、パフォーマンス、セキュリティ、信頼性の向上によってアプリケーションがどのようにメリットを得るかについて説明しています。

この戦略は、以下が推奨するアプリケーションモダナイゼーションアプローチを取り上げたコンテンツシリーズの一部です。AWS。このシリーズには以下も含まれます。

- におけるアプリケーションのモダナイゼーション準備状況の評価AWS雲
- アプリケーションのモダナイゼーションへの段階的アプローチAWS雲
- モノリスをマイクロサービスに分解
- を使用してマイクロサービスを統合するAWSサーバーレスサービス
- マイクロサービスでのデータ永続性の有効化

概要

アプリケーションを最新化することで、コストを削減し、効率を高め、既存の投資を最大限に活用できます。これには、新しいテクノロジーを採用して使用し、ポートフォリオ、アプリケーション、インフラストラクチャの価値をより迅速に提供し、最適な価格で組織を拡張できるようにするための多面的なアプローチが必要です。アプリケーションを最適化したら、ビジネス運用、アーキテクチャ、および全体的なエンジニアリングプラクティスを簡素化するために、中断することなくその新しいモダナイズされたモデルで運用する必要があります。

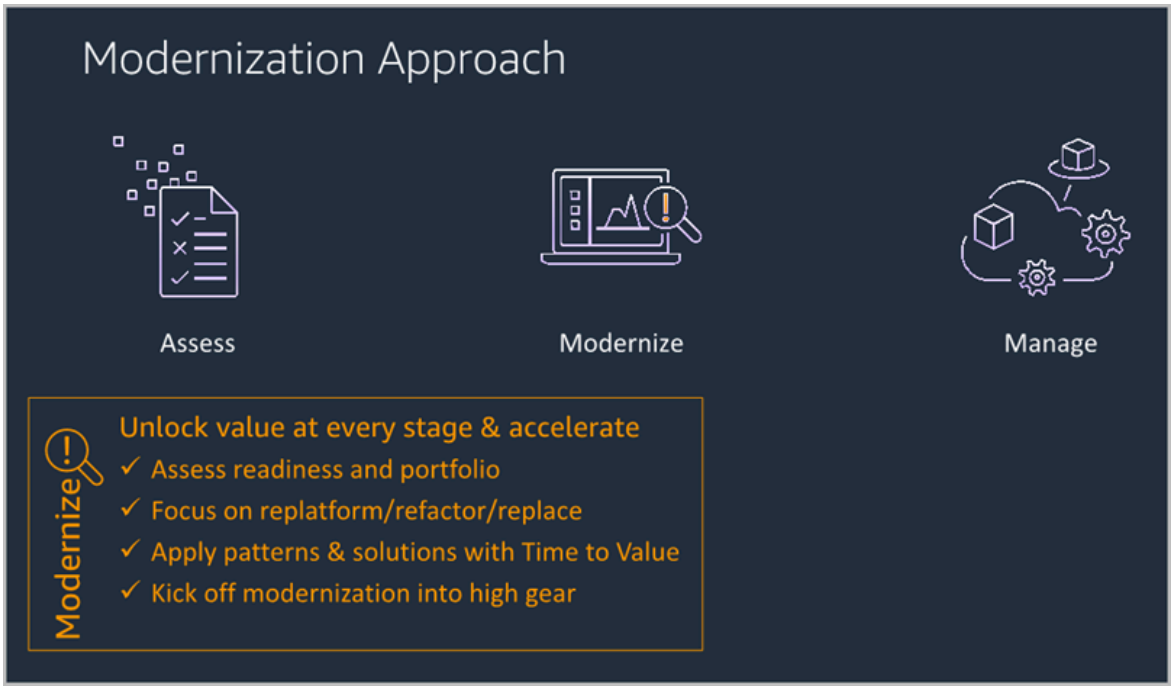
へのアプリケーションの移行AWSリホスティング (リフトアンドシフト) アプローチを使用しても、弾力性、耐障害性、展開と管理のしやすさ、柔軟性などのメリットが自動的に得られるわけではありませんAWSオファー。また、企業文化やプロセスを自動的に近代化して高性能なソフトウェア開発を実現することもありません。モダナイゼーションとは、アプリケーション環境を現在の形式 (ほとんどの場合、レガシーでモノリシック) にして、より俊敏で伸縮自在で可用性の高い環境に変換することを意味します。そうすることで、ビジネスを現代の企業に変えることができます。

クラウドの導入と移行を最適化するには、まず企業の準備状況を評価して評価する必要があります。組織の準備状況を評価すると、次のことが可能になります。

- 1 つまたは 2 つのアプリケーションを選択します。

- これらのアプリケーションを最新化して、ビジネスの現在およびfuture ニーズを満たす方法で保守、拡張、展開、管理できるようにします。
- 前の 2 つのステップで得た実践的な経験を通じて、大規模なモダナイゼーションの基盤を確立します。このフェーズでは、サポートするインフラストラクチャ、アプリケーションミドルウェア、ミドルウェアサービス (データベース、キューイングソフトウェア、統合ソフトウェア、その他のテクノロジーなど)、およびその他のコンポーネントを決定することで、完全なモダナイゼーションソリューションを作成できます。

この記事で説明するアプリケーションのモダナイゼーションへの反復的なアプローチは、評価、モダナイズ、管理という 3 つの大まかなフェーズに分けられます。これらのフェーズについては、この記事の後半で詳しく説明します。



ターゲットを絞ったビジネス成果

エンタープライズ規模のアプリケーションのモダナイゼーションには、複数のディメンションをバインドして、加速化されたペースで完全性を実現するための総合的なアプローチ (評価、モダナイズ、管理) が必要です。AWS が推奨するフレームワークオートメーションは、デベロッパーワークフロー、セルフサービスデータ、アーキテクチャの進化、価値のための組織化という 5 つの技術分野にわたるモダナイゼーションを構想しています。これらのドメインについては、「モダナイゼーションの戦略的ディメンション (p. 4)」セクションにで詳細を説明します。AWS プロフェッショナルサービスと AWS パートナーエンゲージメントで使用できるフレームワークには、ソリューション、セルフサービスの技術パターン、プレイブック、テンプレートを含むナレッジベースが含まれます。

モダナイゼーションプロジェクトを成功させることは、次のようなビジネス成果を生み出すのに役立ちます。

- ビジネスの俊敏性 - ビジネスニーズを要件に変換するためのビジネス内の有効性。デリバリー組織がビジネス要求にどの程度対応しているか、および本番環境に機能をリリースする際にビジネスがどの程度制御できるかを示します。
- 組織の俊敏性 - アジャイルな方法論と DevOps セレモニーを含み、明確な役割の割り当て、組織全体の全体的なコラボレーションとコミュニケーションをサポートするデリバリープロセス。
- エンジニアリングの有効性 - 品質保証、テスト、継続的統合および継続的デリバリー (CI/CD)、構成管理、アプリケーション設計、ソースコード管理の改善。

これらのビジネス成果を達成するには、包括的なアプローチと、一連の戦略的な側面に基づくモダナイゼーションプロセスが必要です。

モダナイゼーションへの包括的なアプローチ

アプリケーションのモダナイゼーションへの道のりは、次を含む段階的な取り組みです。

- レガシーワークロードとクラウドのワークロードを分析するためのデータ駆動型の意思決定を行う。
- クラウドに移行するプロセスを評価する。
- コンテナ、サーバーレステクノロジー、最新のデータベースなどの新しい機能を統合して、人工知能 (AI)、モノのインターネット (IoT)、機械学習 (ML) などの新しいテクノロジーをサポートします。

組織のあらゆる分野にわたる継続的なモダナイゼーションは、成功の鍵です。モダナイゼーションの価値を最大限に引き出すには、選択肢とトレードオフを理解し、エンタープライズ、差別化、未分化、およびコモディティアプリケーションを組み合わせて接続する能力に重点を置く必要があります。このプロセスは、ビジネス成果に合わせたアプリケーション評価から始まり、企業がアプリケーションを最適にデプロイおよび管理できるようにします。

今日の企業は、レガシーシステムに次のような複雑さと非効率性が含まれている場合、新しいビジネスモデルや変化するビジネスモデルに適応できない可能性があります。

- 俊敏性の欠如。変化するビジネスや市場の需要に迅速に対応できない。
- 柔軟性の欠如。アプリケーションに必要な変更を加えることはできない。
- スケーラビリティの欠如。新しいアプリケーション機能を導入したり、新しいユーザーやキャパシティを含む既存の機能を拡張したりできない。

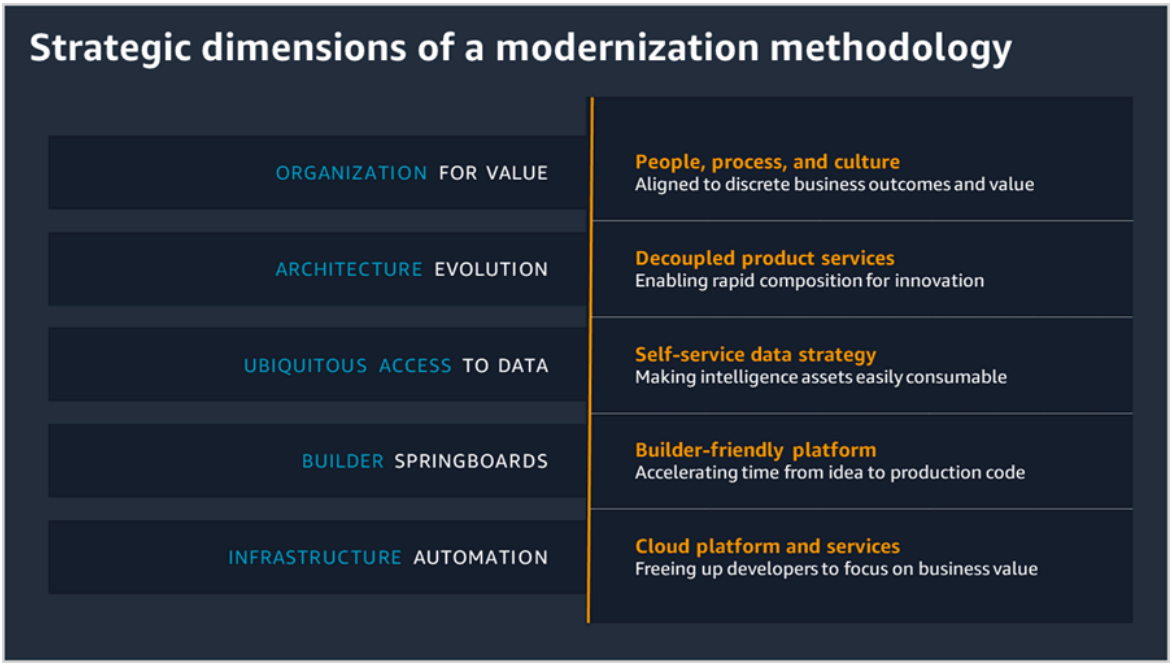
- パフォーマンスの問題。アプリケーションが目的の標準およびメトリクスに対して実行されない。
- データインサイトの欠如。データサイロが多すぎて、デジタルイノベーションが遅い。
- 高セキュリティリスク。セキュリティが組み込まれ、全体で統合されている新しいアプリケーションフレームワーク内に存在しないギャップや脆弱性が、アプリケーションに存在する。
- 新しいアプリケーションやサービスを追加できない。新しいテクノロジーや最新のアーキテクチャの導入が妨げられる。
- コストの増加。レガシーアプリケーションとアプリケーションフレームワークは、多くの場合、より多くのリソースを消費し、モダナイズされたアプリケーションよりも多くの冗長性と非効率性を生み出すため。

モダナイゼーションの戦略的ディメンション

モダンアプリケーションは、効果的に開発および管理されるときに、多次元の利点を顧客に提供します。俊敏性、回復力、エンジニアリング効率を向上させることで、イノベーションを加速するため、一連の戦略的側面に基づいて、継続的なモダナイゼーションのプロセスを確立できます。これらの実績のあるパターンと手法を継続的にフォローして構築することで、既存のアプリケーションコンポーネントを最新のデプロイプラットフォームにデプロイし、既存の機能を新しいアプリケーションからアクセスできるようにし、アプリケーションアーキテクチャを完全に最新のスタックに更新できます。

次の図に示すように、これらのモダナイゼーションディメンションは次のとおりです。

- 価値のための組織 - 組織構造、ガバナンス、プロセスを再調整して、顧客の成果を通じてビジネス価値を提供できる小規模のフルスタックの製品チームを中心に一元化します。
- アーキテクチャ上の進化 - コアビジネス機能をモノリシックアプリケーションから、デベロッパーがイノベーションの構築ブロックとして使用できる、独立した保守性、進化性、再利用可能なサービスの疎結合化されたコレクションに移行することで、デジタル製品プラットフォームを構築します。
- データへのユビキタスアクセス - 最新のデータアーキテクチャ、ストレージ、アクセスパターンを AWS のサービスと統合し、開発者、データサイエンティスト、およびビジネスユーザーが組織のデータストリームに簡単に利用できるようにします。
- ビルダースプリングボード - 俊敏なソフトウェアエンジニアリングの実践 (DevOps、テストアプリケーション、CI/CD、観測性など)、関連するツール、アプリケーションレイヤサービスを統合されたデベロッパーワークフローに統合します。このワークフローは、開発のパスを定義し、コードをアイデアから本番に移行する時間を短縮します。
- インフラストラクチャのオートメーション — 軽量なインフラストラクチャ基盤を作成するための AWS のサービスの組み合わせを使用します。コンテナと AI/機械学習を使用して、よく使用されるインフラストラクチャプリミティブを抽象化して自動化します。これにより、開発リソースが解放され、顧客のための新しい製品やサービスの作成を通じてビジネス価値の提供に集中できます。



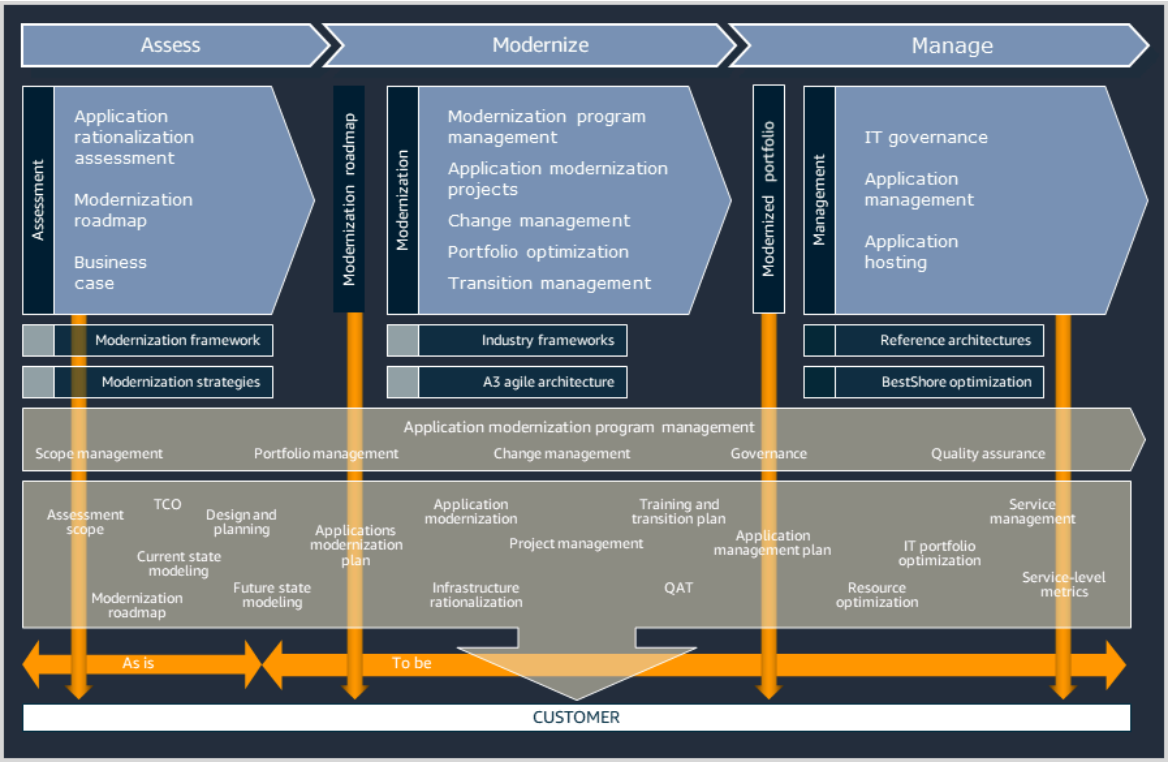
モダナイゼーションの戦略的なディメンションを適用することで、組織の生産性が向上し、測定可能で持続可能な成果を実現できます。組織は次のことができるようになります。

- 差別化されたカスタマーエクスペリエンスを向上させ、創造します。
- イノベーションを加速し、市場投入までの時間を短縮し、新製品を頻繁にリリースします。
- IT インフラストラクチャに費やすことにより、コストを最適化し、回避できます。
- 俊敏性を高め、新しい機能と機能を大規模に追加できます。
- 新機能を迅速にデプロイすることによりスタッフの生産性を向上させます。
- サービスレベルアグリーメント (SLA) を改善し、予期しない停止を減らします。

近代化プロセスのフェーズ

顧客の要求に応え、変化するテクノロジー環境を活用するためにアプリケーションをモダナイズすることは、組織の競争上の優位性と市場シェアを維持するために重要です。このようなビジネスニーズを満たすための重要な戦略は、老朽化したアプリケーションをより現代的なアーキテクチャに変換することで、継続的な使用と真の価値の両方を実現することです。アプリケーションの詳細と他のシステムとの相互関係を包括的に理解することは、アプリケーションの最新化を実行する上で重要なステップです。

AWSアプリケーションモダナイゼーションへのアプローチは反復的であり、次の図に示すように、評価、モダナイズ、管理という3つの高レベルのフェーズに分けることができます。



以下のセクションでは、各フェーズについて詳しく説明します。

トピック

- 評価 (p. 6)
- モダナイズ (p. 7)
- 管理 (p. 8)

評価

組織のモダナイゼーションの第一歩は、既存のアプリケーションポートフォリオを分析し、モダナイゼーションが必要なシステムを評価し、アプリケーションのモダナイゼーションに必要な技術的ソリューションを特定することです。このフェーズでは、アプリケーションモダナイゼーションアンケートを使用して

アプリケーションポートフォリオを評価および合理化し、ポートフォリオ内のアプリケーションのビジネス上、機能上、技術上、および財務上の重要性（戦略的価値）を判断できます。これにより、future 国家アーキテクチャが構築されるときに、その組織がどれだけうまくサポートできるかが決まります。

アクティビティ

- 5 つの観点からアプリケーションを評価します。
  - 戦略的またはビジネスに適合
  - 機能的妥当性
  - 技術的妥当性
  - ファイナンシャルフィット
  - デジタル準備状況評価
- グループ化、ランク付け、および順序付けのアプリケーション。
- 目標および暫定運用モデルを文書化する。
- 主要なテクノロジーと規制要件を理解してください。
- 大規模なデータ移行が必要なアプリケーションを特定します。
- 変換するデータの範囲と量を明確にしてください。

結果

- アプリケーションモダナイゼーション設計図
- 1つまたは2つのアプリケーションのターゲットステートに対応する技術的および機能的なアーキテクチャ
  - 戦略的またはビジネスに適合
  - 機能的妥当性
  - 技術的妥当性
  - ファイナンシャルフィット
  - デジタル対応

ハウツーガイド

- AWSクラウドでのアプリケーションのモダナイゼーションの準備状況を評価するを参照してください

モダナイズ

このフェーズでは、プロジェクトの目標とリソース要件を決定し、実装ロードマップを作成します。目標は、モダンでアジャイルなアプリケーションアーキテクチャを構築するモダナイゼーションプログラムを使用してアプリケーションを活性化することです。

アクティビティ

- アプリケーションのソースコードとデータを変換するためのマイルストーンを決定します。
- すべての運用領域のマッピングを完了して、新しいターゲット環境の運用と管理に必要な標準と手順が満たされていることを確認します。
- クラウドネイティブなアプローチ、best-of-breed言語、フレームワークを使用して、信頼性、アクセシビリティ、成長の要件に対応できるインフラストラクチャソリューションを実装します。モダナイズされたアプリケーションのコンポーネントには、次のような特徴があります。
  - 軽量容器として包装されています
  - 疎結合のマイクロサービスとして設計
  - インタラクションとコラボレーションのための API が中心

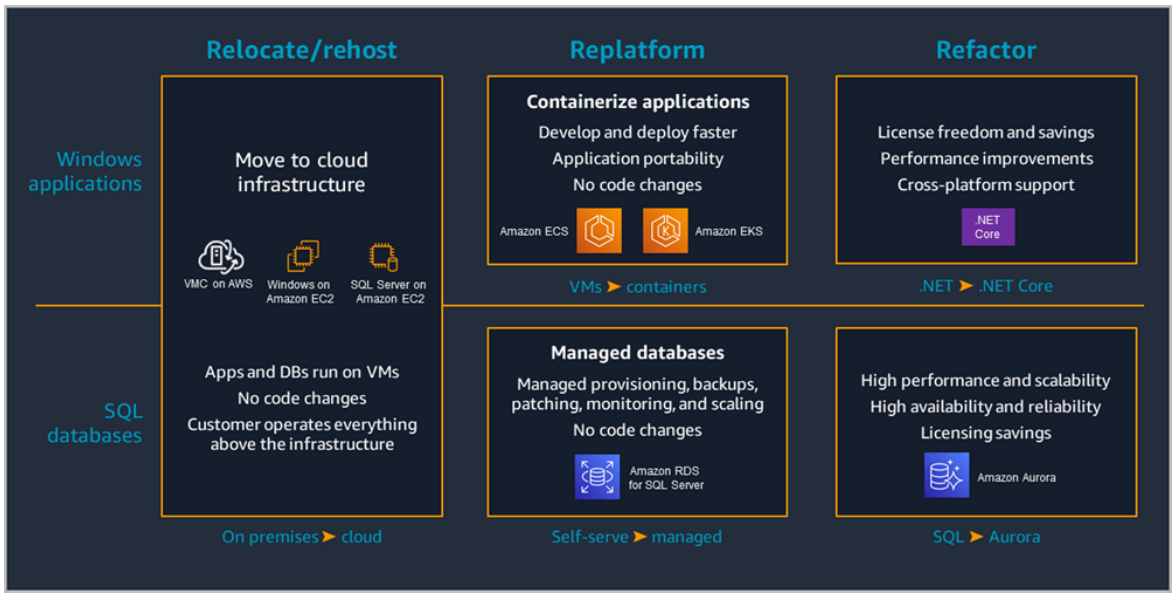
- ステートレスサービスとステートフルサービスを明確に分離した設計
- サーバーやオペレーティングシステムの依存関係から切り離されています
- セルフサービスで伸縮自在なクラウドインフラストラクチャにデプロイ
- DevOpsアジャイルプロセスによる管理
- 自動化機能を含める
- ポリシーに基づいた明確なリソース割り当てを提供

結果

- ターゲットステートデータモデル設計
- トレーニングとツールの改善（変更管理と運用モデル）によって構築された組織の即応性
- 変更活動のための定期的な頻度を設置
- 洗練された運用モデルとデリバリー効果の測定
- 提供された価値について追跡および報告される主要なビジネスケース指標
- 改良と自動化活動の継続
- 各アプリケーションに適用される戦略とその拡張方法を定義するモダナイゼーションロードマップ
- 新しいアプリケーションロードマップと同期した反復テストの実施を含む、最新化の準備と実装

例

次の図は、レガシー Windows アプリケーションの最新化オプションを示しています。



ハウツーガイド

- AWSクラウドでのアプリケーションをモダナイズするへの段階的アプローチ

管理

アプリケーションの特性を詳細に理解し、その後のモダナイゼーションの取り組みによって生じる可能性のあるリスクを軽減するために、すべてのモダナイゼーション活動に再学習の取り組みが組み込まれてい

ます。アプリケーション・ワークロードがプラットフォーム・サービスを活用できなければ、アプリケーション・チームはアプリケーション・ワークロードのランタイム特性を理解して最適化できます。つまり、アプリケーションチームはモダナイズされたアプリケーションの運用機能を他のすべてのアプリケーション機能と同様に扱うべきであり、マイクロサービスの運用は事実上エンジニアリングの一部となるべきです。組織内のサイト信頼性エンジニアリング（SRE）能力を構築する一環として、DevOpsこの文化をクラウドネイティブな運用に取り入れることは、モダナイゼーションの採用を成功させるために不可欠です。管理フェーズには、効果的な変更管理、プログラム管理、品質保証、サービスエクセレンスのすべての要素が含まれます。

ハウツーガイド

- AWSクラウドでの運用の近代化

次のステップ

アプリケーションをモダナイズするプロセスを開始する前に、準備評価を実行して、組織がモダナイゼーションの移行を開始する準備が整っているかどうかを判断します。チーム間のアラインメントを作成し、新しい作業方法を導入できるようにします。ほとんどの企業は、最大の収益を生み出し、中核的なビジネス機能を提供するレガシーアプリケーションのフットプリントが大きくなっています。これらのレガシーアプリケーションのアーキテクチャ、インフラストラクチャ、テクノロジーの粘着性は、モダナイゼーションのための複雑さとスピードアップを生み出し、チームのアプリケーションの革新と変革能力をさらに制限してしまいます。

モダナイゼーションへのアプローチは、増分的でありながら継続的である必要があります。この段階的なアプローチに従うことで、技術的な負債を削減するだけでなく、自己修復システムを使用してシームレスにスケーリングし、クラウドネイティブパターンを使用して投資収益率 (ROI) を最大化し、未分化ワークロードと依存関係の負荷を軽減し、アプリケーションパフォーマンスを改善することで、チームがクラウドのメリットを迅速に実現できます。運用モデルを変更し、現在のアプリケーションの詳細な評価から開始するフレームワークに連携させ、効率的なデプロイ、高速なスケール、管理が可能なクラウドネイティブサービスを使用してそれらのアプリケーションをモダナイズする方法の詳細については、関連ガイド (p. 11)セクションの「モダナイゼーションガイド」を参照してください。

Resources

関連ガイド

- AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する
- AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ
- AWS クラウドにおけるオペレーションのモダナイゼーション
- モノリスをマイクロサービスに分解する
- AWS サーバーレスサービスを使用してマイクロサービスを統合する
- マイクロサービスでのデータ永続性の有効化
- AWS クラウドへの移行に関する規範的ガイダンス

AWS リソース

- AWS ドキュメント
- AWS 全般のリファレンス
- AWS 用語集

ドキュメント履歴

このドキュメントは、このドキュメントの大きな変更点をまとめたものです。今後の更新に関する通知を受け取る場合は、RSS フィードをサブスクライブできます。

変更	説明	日付
初版発行 (p. 12)	—	2020 年 12 月 18 日

AWS 規範的ガイダンスの用語集

以下は、AWS規範的ガイダンスによって提供される戦略、ガイド、およびパターンで一般的に使用される用語です。エントリを提案するには、用語集の最後のフィードバックの提供リンクを使用します。

モダナイゼーション用語

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど) 。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、ホワイトペーパー「AWS でのコンテナ化されたマイクロサービスの実行」の「ビジネス機能を中心に組織化」セクションを参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています(ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用する方法の詳細については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS サーバーレスサービスを使用してマイクロサービスを統合する を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、AWS でのマイクロサービスの実装 を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、AWS クラウドのアプリケーションのモダナイズ戦略 を参照してください。

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定されたギャップに対処するためのアクションプランが得られます。詳細については、AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、モノリスをマイクロサービスに分解する を参照してください。

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。詳細については、マイクロサービスでのデータ永続性の有効化 を参照してください。

split-and-seed モデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ 」を参照してください。

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

ツーピザチーム

2 DevOps 枚のピザで養うことができるくらい小さめのチーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。詳細については、の On DevOps 入門AWSホワイトペーパーのツーピザチームのセクションを参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。