AWS の規範的ガイダンス

サーバーレスサービスを使用したマイクロサービスの統合

AWS の規範的ガイダンス: サーバーレスサービスを使用したマイクロサービスの統合

Table of Contents

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

Hari Ohm Prasath Rajagopal、Tabby Ward、Dmitry Gulin、Amazon Web Services (AWS)

2021 年 1 月(ドキュメント履歴 (p. 18))

組織のソフトウェアをモダナイズするうえで重要なことは、モノリシックアプリケーションをマイクロサービスにリファクタリングする。モノリスを分解すると、1 つのビジネストランザクションのデータを取得するために複数のマイクロサービスが呼び出されます。これらのマイクロサービスがアーキテクチャに誤って統合されると、マイクロサービスアーキテクチャを採用するメリットが損なわれます。これにより、データが失われたり、レイテンシーや整合性の問題が発生したりする可能性があります。これらの問題は解決するのが難しい場合が多く、ユーザーはすぐに影響を受けます。ただし、マイクロサービスが正しく統合されていれば、分散システムの利点が得られ、サービスレベルでのスケーリングが可能になり、効率が向上し、インフラストラクチャコストが削減されます。

このガイドは、アプリケーション所有者、ビジネス所有者、アーキテクト、テクニカルリード、プロジェクトマネージャを対象としています。このガイドでは、新しいマイクロサービスをアーキテクチャに統合するのに役立つ次の 3 つのパターンを紹介します。

- API Gateway パターン (p. 2)
- メッセージングパターンを分離 (p. 4)
- パブ/サブパターン (p. 5)

これらのパターンは自律性とスケーラビリティを提供し、アマゾンウェブサービスのサーバーレスサービスを使用します (AWS) などAWS LambdaとAmazon API Gateway は、マイクロサービスの統合に役立ちます。このガイドは、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。

- アプリケーションの最新化戦略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
- 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
- モノリスをマイクロサービスに分解
- マイクロサービスのデータパーシスタンスの有効化

ターゲットを絞ったビジネス成果

このガイドを使用して新しいマイクロサービスを統合することで、組織のアーキテクチャをマイクロサービスアーキテクチャに効率的に変換できます。これにより、高いスケーラビリティ、耐障害性の向上、継続的デリバリー、障害の切り分けなどのコアアクティビティを中断することなく、変動するビジネスニーズに迅速に対応できます。マイクロサービスアーキテクチャは、各マイクロサービスを個別にデプロイしてテストできるため、耐障害性と耐障害性を向上させ、イノベーションを加速させるのにも役立ちます。

マイクロサービスアーキテクチャは、製品やサービスの市場投入までの時間を短縮するのにも役立ちます。これは、各マイクロサービスには独立したコードベースがあり、新しい機能の追加や反復を簡単かつ迅速に行うことができるためです。

マイクロサービスを統合するためのパターン

マイクロサービスをアーキテクチャに統合するには、次のパターンを使用します。

トピック

API Gateway パターン

API ゲートウェイパターンは、複数のクライアントアプリケーションを使用して複雑な、または大規模なマイクロサービスベースのアプリケーションを設計および構築する場合に推奨されます。パターンは、ファサードパターンオブジェクト指向設計からだが、分散システムリバースプロキシまたはゲートウェイルーティングの一部であり、同期通信モデルを使用する。

このパターンは、内部マイクロサービスエンドポイントにリクエストをリダイレクトまたはルーティングするリバースプロキシを提供します。API ゲートウェイは、クライアントアプリケーションに単一のエンドポイントまたは URL を提供し、内部でリクエストを内部マイクロサービスにマッピングします。抽象化のレイヤーは、特定の実装の詳細 (Lambda 関数の名前とバージョンなど) を隠すことによって提供されます。また、レスポンスとリクエストの変換、エンドポイントアクセスの承認、トレースなどの追加機能をバックエンドサービスの上に追加することもできます。

次の場合は、API ゲートウェイパターンを使用することを検討する必要があります。

- マイクロサービスの依存関係の数は管理可能で、時間の経過とともに増加しません。
- 呼び出し側システムでは、マイクロサービスからの同期応答が必要です。
- 低レイテンシー要件があります。
- 複数のマイクロサービスからデータを収集するには、1 つの API を公開する必要があります。

ユースケース

このユースケースでは、顧客は Lambda 関数としてデプロイされた 4 つのマイクロサービス (「顧客」、「通信」、「支払い」、「販売」) で構成される保険システムで、定期的な月額支払いを行います。「Customer」マイクロサービスは、毎月の支払い詳細で顧客データベースを更新します。「Sales」マイクロサービスは、セールス・チームがクロスセリング・オポチュニティについて顧客とフォローアップするのに役立つ関連情報でセールス・データベースを更新します。「コミュニケーション」マイクロサービスは、支払いが正常に処理された後、確認メールを顧客に送信します。最後に、「ペイメント」マイクロサービスは、顧客が毎月の支払いを行うために使用するシステム全体です。このパターンは、Web サービスを使用して、「顧客」、「販売」、および「通信」サブシステムを「支払い」マイクロサービスと統合します。

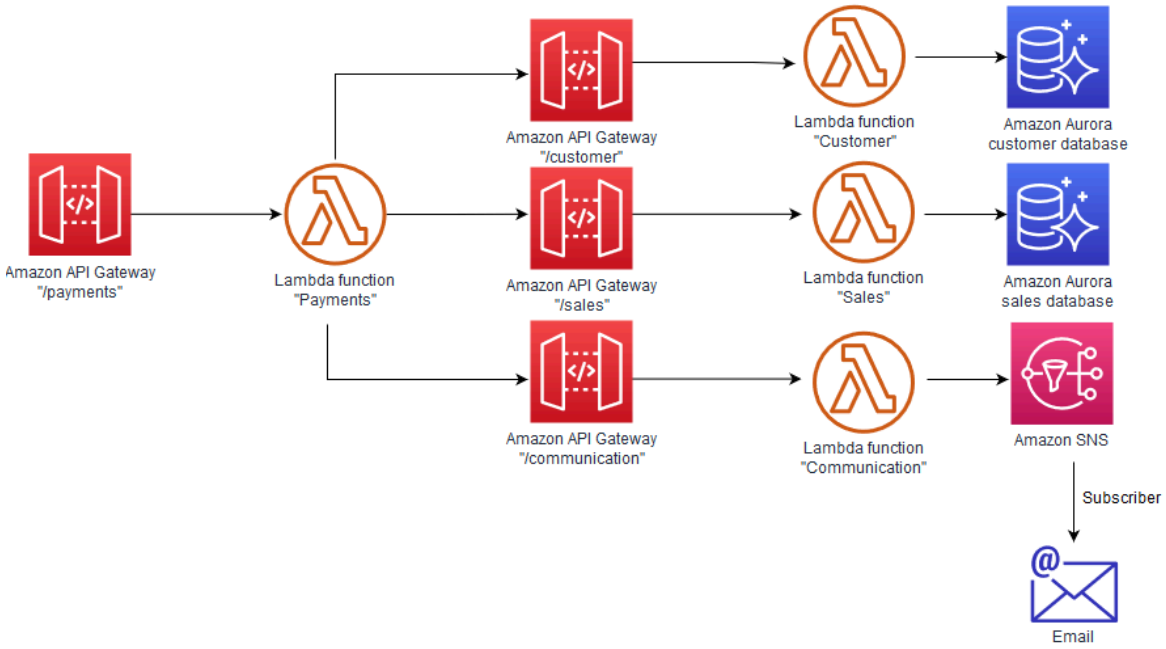
このユースケースでこのパターンを使用するには、次の 3 つの課題があります。

- 同期呼び出しはダウンストリームシステムに対して行われます。つまり、これらのサブシステムによって引き起こされるレイテンシーは全体的な応答時間に影響します。
- 「ペイメント」システムは、呼び出しシステムに応答する前に他のマイクロサービスからの応答を待っているため、ランニングコストが高くなります。したがって、総実行時間は、非同期システムと比較して比較的長くなります。
- エラー処理と再試行は、個々のマイクロサービスではなく、「Payments」システム内のマイクロサービスごとに個別に処理されます。

次の 2 つの例は、API ゲートウェイパターンを使用して、複数の API ゲートウェイまたは 1 つの API ゲートウェイを使用してマイクロサービスを統合する方法の概要を示しています。

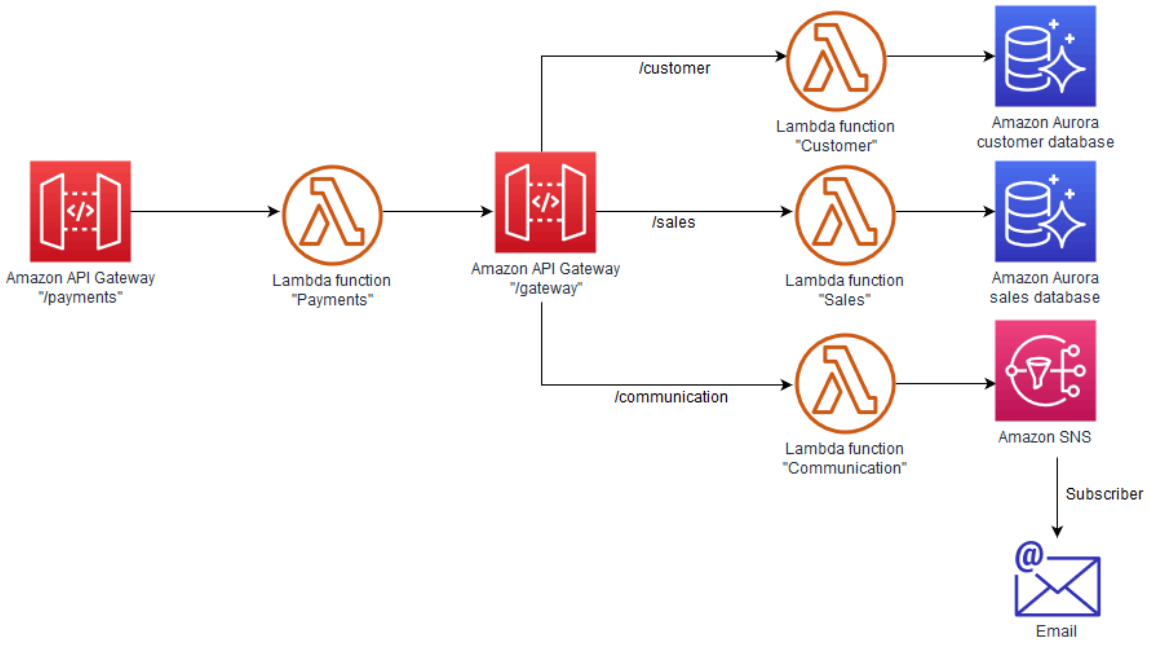
複数の API ゲートウェイ

次の図では、各マイクロサービスには独自の API ゲートウェイがあります。「Payments」マイクロサービスは、個々のシステムを呼び出し、API ゲートウェイパターンを実装します。



シングル API Gateway

次の図では、各マイクロサービスは Lambda 関数としてデプロイされますが、すべてのマイクロサービスは同じ API ゲートウェイで接続されています。



デカップル・メッセージング・パターン

このパターンは、非同期ポーリングモデルを使用して、マイクロサービス間の非同期通信を提供します。バックエンドシステムはコールを受信すると、リクエスト識別子で即時に応答し、リクエストを非同期的に処理します。疎結合アーキテクチャを構築できるため、同期通信、遅延、入出力操作 (IO) によるボトルネックを回避できます。パターンのユースケースでは、Amazon Simple Queue Service (Amazon SQS) と Lambda を使用して、異なるマイクロサービス間の非同期通信を実装します。

次の場合は、このパターンを使用することを検討する必要があります。

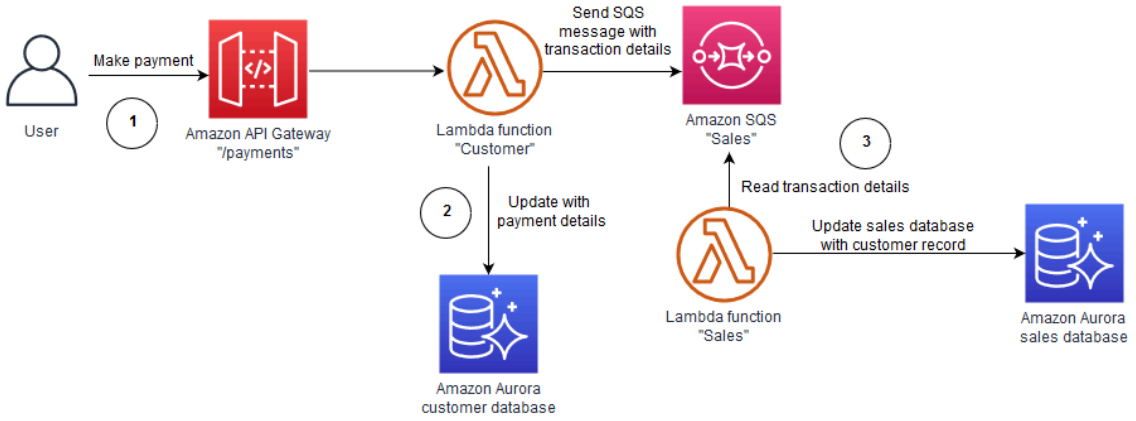
- 疎結合アーキテクチャを作りたい。
- すべての操作は 1 つのトランザクションで完了する必要はなく、一部の操作は非同期にすることができます。
- 1 秒あたりのトランザクション (TPS) の受信レートは、ダウンストリームシステムでは処理できません。メッセージはキューに書き込まれ、リソースの可用性に基づいて処理できます。

このパターンの欠点は、ビジネストランザクションアクションが同期していることです。呼び出し元のシステムが応答を受信しても、トランザクションの一部は引き続きダウンストリームシステムによって処理されることがあります。

> **Important**
>
> このパターンは fire and-forget モデルに適しているため、このサービスを呼び出すクライアントは、リクエスト ID を使用してトランザクションステータスを取得して、実際のサービスをポーリングする必要があります。

ユースケース

このユースケースでは、保険システムには、毎月の支払いが行われた後、顧客取引の詳細で自動的に更新される販売データベースがあります。次の図は、デカップル・メッセージング・パターンを使用してこのシステムを構築する方法を示しています。



ワークフローは、以下のステップで構成されています。

1. フロントエンドアプリケーションは、ユーザーが毎月の支払いを行った後、支払い情報を使用して API Gateway を呼び出します。
2. API Gateway は、支払い情報を Amazon Aurora データベースに保存し、メッセージ内のトランザクションの詳細を「Sales」Amazon SQS に書き込み、呼び出しシステムに成功メッセージで応答する「顧客」Lambda 関数を実行します。
3. 「Sales」Lambda 関数は SQS メッセージからトランザクションの詳細を取得し、売上データを更新します。販売データベースを更新するための失敗と再試行ロジックは、「Sales」Lambda 関数の一部として組み込まれています。

PUB/SUB パターン

プラットフォームが成長すると、相互依存性を持たずに異なるマイクロサービスが相互作用することが困難になる可能性があります。パブリッシュ/サブスクライブ (pub/sub) パターンは、複数の間で非同期通信を提供します。AWS相互依存関係を作成せずに、Amazon SQS、Lambda、Amazon Simple Storage Service (Amazon S3) などのサービス。このパターンでは、マイクロサービスは、ユーザが聴くことができるチャネル内のメッセージとしてイベントをパブリッシュします。たとえば、工場では pub/sub パターンを使用して機器がチャネルに問題や障害を公開できるようにし、加入者はこれらの機器の問題を表示してログに記録できます。

次の場合は、このパターンを使用することを検討する必要があります。

- イベント駆動型のアーキテクチャがあります。
- 疎結合アーキテクチャを有効にできます。
- 呼び出し元のシステムへの応答の前に、トランザクションのすべての操作部分を完了する必要はありません (特定の操作は非同期である可能性があります) 。
- 従来のデータセンターの能力を超えるボリュームに拡張する必要があります。このレベルのスケーラビリティは、主に並列操作、メッセージキャッシュ、ツリーベースのルーティング、および pub/sub モデルに組み込まれているその他の機能によるものです。

このパターンを使用することにはいくつかの欠点があります。たとえば、Amazon Simple Notification Service (Amazon SNS) などの特定のサービスが提供できるものの、通常、Pub/Sub パターンでは、すべてのサブスクライバータイプへのメッセージの配信を保証できません。1 回だけ一部のサブスクライバサブセットへの配信。もう 1 つの欠点は、パブリッシャーが、実際にはチャンネルを聞いていないときに、サブスクライバがチャンネルを聞いていると仮定できることです。

ユースケース

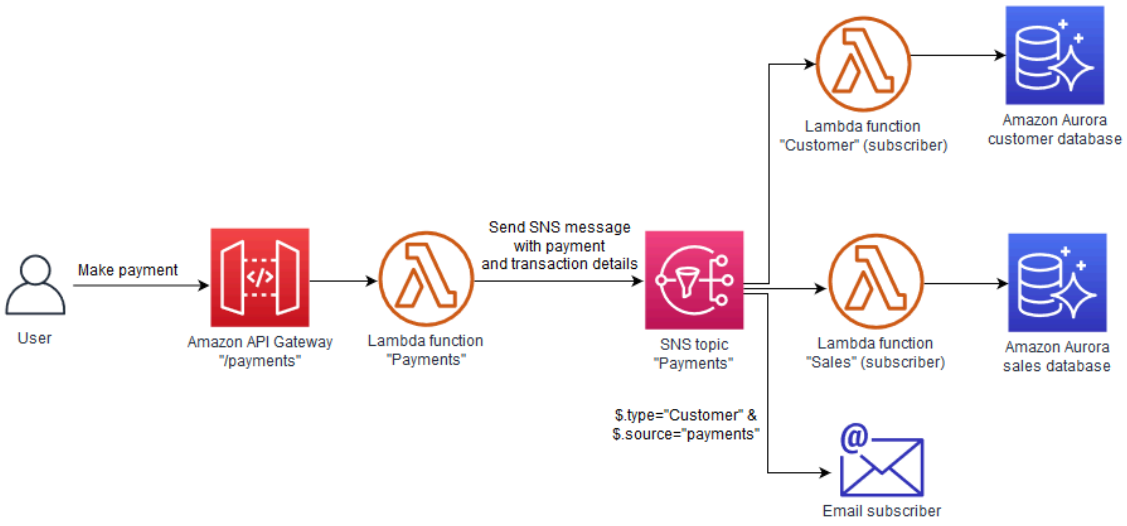
このユースケースでは、SNS トピックを使用して、保険システムの複数の従属マイクロサービスにイベントを公開します。顧客が毎月の支払いを行った後、「顧客」や「セールス」などのサブシステムで情報を更新し、支払い確認を記載した電子メールを顧客に送信する必要があります。このパターンは、Amazon SNS または Amazon EventBridge のいずれかを使用して実装できます。

EventBridge は複数のサブスクライバ間のイベントをフィルタリングします EventBridge の実装には、次の 2 つのオプションがあります。

- 異なるイベントタイプで 3 つのイベントを送信します。遠いターゲットは、イベントルールに基づいてそれらをピックアップします。
- 同じイベントタイプをリッスンする 3 つのイベントルールで 1 つのメッセージを送信します。特定のターゲットが呼び出される前に、不要なデータが除外されます。

Amazon SNS 実装

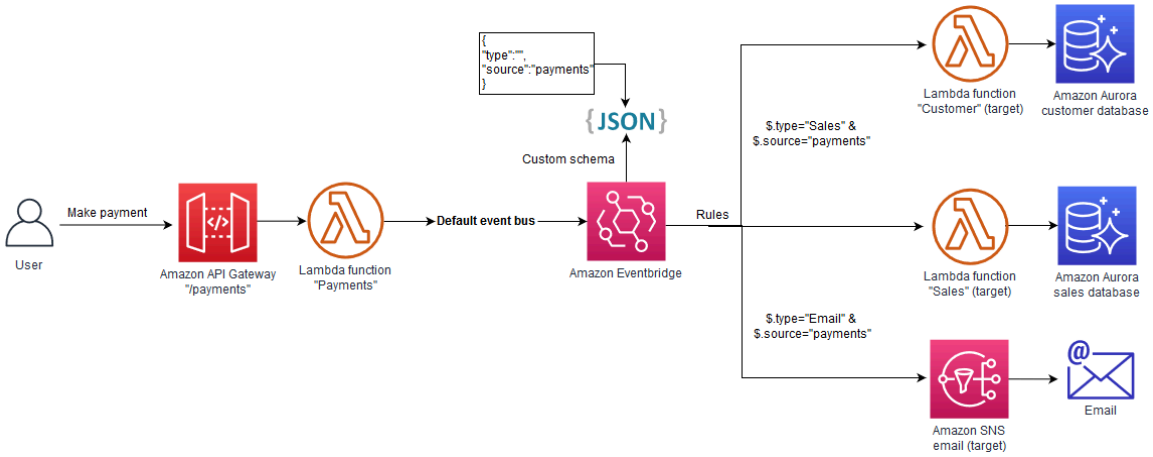
次の図は、Amazon SNS を使用してパブ/サブパターンを実装する方法を示しています。ユーザーが支払いを行った後、「支払い」Lambda 関数によって SNS メッセージが「支払い」SNS トピックに送信されます。この SNS トピックには、メッセージのコピーを受信して処理する 3 人のサブスクライバがあります。



Amazon EventBridge の実装

次の図では、EventBridge を使用して、サブスクライバがイベントルールを使用して定義される pub/sub パターンのバージョンを構築しています。ユーザーが支払いを行った後、「Payments」Lambda 関数は、異なるターゲットを指している 3 つの異なるルールを持つカスタムスキーマに基づいて、デフォルトのイ

ベントバスを使用して EventBridge にメッセージを送信します。各マイクロサービスはメッセージを処理し、必要なアクションを実行します。



マイクロサービスの統合に関するよくある質問

このセクションでは、マイクロサービスの統合に関してよく寄せられる質問に対する回答を示します。

統合パターンを組み合わせることはできますか?

はい、API ゲートウェイパターン (p. 2)およびメッセージングパターンの疎結合化 (p. 4)は、通常、マイクロサービスを統合するときに一緒に使用されます。

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ズ戦略
- 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
- モノリスをマイクロサービスに分解する
- マイクロサービスでのデータ永続性の有効化

その他のリソース

- エンタープライズ統合パターンを AWS メッセージングサービスで実装する: ポイントツーポイントチャネル
- Pub/Sub メッセージング: 非同期イベント通知

AWS 規範的ガイダンスの用語集

AI と ML の用語 (p. 10) | 移行の用語 (p. 11) | モダナイゼーションの用語 (p. 15)

AI と ML の用語

以下は、AWS 規範的ガイダンスによって提供される人工知能 (AI) および機械学習 (ML) に関連する戦略、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。

二項分類	バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。
分類	予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。
データの前処理	raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。
ディープアンサンブル	予測のために複数の深層学習モデルを組み合わせる。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。
ディープラーニング	人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。
探索的データ分析 (EDA)	データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。
特徴量	お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。
特徴量重要度	モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「AWS を使用した機械学習モデルの解釈」を参照してください。

機能変換	追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。
解釈可能性	機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「AWS を使用した機械学習モデルの解釈」を参照してください。
多クラス分類	複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。
回帰	数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。
.training	お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。
ターゲット変数	監督された機械学習でお客様が予測しようとしている値。これは、結果変数 のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。
チューニング	機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。
不確実性	予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には 2 つのタイプがあります。認識論的不確実性限られた不完全なデータが原因ですが偶然性の不確実性データに内在するノイズとランダム性が原因です。詳細については、深層学習システムにおける不確実性の定量化ガイドを参照してください。

移行の条件

以下は、AWS 規範的ガイダンスによって提供される移行関連戦略、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。

7 Rs	アプリケーションをクラウドに移行するための 7 つの一般的な移行戦略。これらの戦略は、ガートナーが 2011 年に特定した 5 Rs に基づいて構築され、以下で構成されています。 • リファクタリング/アーキテクチャの再設計 — クラウドネイティブ特徴を最大限に活用して、俊敏性、パフォーマンス、スケーラビリティを向上させ、アプリケーションを移動させ、アーキテクチャを変更します。これには、通常、オペレーティングシステムとデータベースの移植が含まれます。例: オンプレミスの Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行する。

	• リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: オンプレミスの Oracle データベースを次の Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。AWSCloud。 • 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: 顧客関係管理 (CRM) システムを Salesforce.com に移行する。 • リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: オンプレミスの Oracle データベースを次の EC2 インスタンス上の Oracle に移行する。AWSCloud。 • 再配置 (ハイパーバイザーレベルのリフトアンドシフト) — 新しいハードウェアの購入、アプリケーションの書き換え、お客様の既存のオペレーションの変更を行うことなく、インフラストラクチャをクラウドに移行できます。この移行シナリオは AWS の VMware Cloud に固有のもので、お客様のオンプレミス環境と、および AWS の間の仮想マシン (VM) 互換性とワークロードの移植性をサポートします。インフラストラクチャを AWS の VMware Cloud に移行するときに、お客様のオンプレミスのデータセンターから VMware Cloud Foundation テクノロジーを使用できます。例: の Oracle データベースをホストしているハイパーバイザーを VMware Cloud 上に再配置する。AWS。 • 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。 • 使用停止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。
アプリケーションポートフォリオ	アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の需要要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。
AI オペレーション (AIOps)	機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。
AWS クラウド導入フレームワーク (AWS CAF)	組織がクラウドに正常に移行するための効率的で効果的な計画を立てるのを支援する AWS からのガイドラインとベストプラクティスのフレームワーク。AWSCAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用の観点と呼ばれる 6 つの重点を置く分野にガイダンスを編成しています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、AWS CAF は、クラウドの導入を成功させるための組織の準備を支援するために、人材開発、トレーニング、コミュニケーションに関するガイダンスを提供します。詳細については、AWS CAF ウェブサイト と AWS CAF のホワイトペーパー を参照してください。
AWS ランディングゾーン	ランディングゾーンは、Well-Architected の、スケーラブルで安全なマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロードとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、安全でスケーラブルなマルチアカウント AWS 環境のセットアップ を参照してください。
AWS ワークロード資格フレームワーク (AWS WQF)	データベース移行ワークロードを評価し、移行戦略を推奨し、作業の見積もりを提供するツール。AWSWQF は AWS Schema Conversion Tool (AWS SCT) と共に

	含まれます。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。
ビジネス継続性計画 (BCP)	大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。
Cloud Center of Excellence (CCoE)	クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、AWS クラウドエンタープライズ戦略ブログの CCoE の投稿 を参照してください。
導入のクラウドステージ	組織が、AWS クラウドへの移行時に通常実行する 4 つの段階。 • プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する • 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など) • 移行 — 個々のアプリケーションの移行 • 再発明 — 製品とサービスの最適化、クラウドでのイノベーション これらのステージは Stephen Orban が AWS クラウドエンタープライズ戦略ブログの クラウドファーストジャーニーと導入ステージ というブログ記事で定義したものです。これらが、AWS 移行戦略とどのような関係があるかについては、移行準備ガイドを参照してください。
構成管理データベース (CMDB)	企業のハードウェアおよびソフトウェア製品、構成、相互依存関係に関する情報を含むデータベース。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。
エピック	アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。例えば、AWS CAF セキュリティエピックには、アイデンティティとアクセスの管理、検出型制御、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。AWS 移行戦略のエピックの詳細については、プログラム実装ガイド を参照してください。
異種混在データベースの移行	別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCT を提供します 。
同種データベースの移行	お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。
アイドル状態のアプリケーション	90 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。
IT 情報ライブラリ (ITIL)	IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。
IT サービス管理 (ITSM)	組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、「オペレーション統合ガイド」を参照してください。

大規模な移行	300 台以上のサーバの移行。
Migration Acceleration Program (MAP)	組織がクラウドへの移行のための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。
移行ポートフォリオ評価 (MPA)	AWS クラウドに移行するためのビジネスケースを検証するための情報を提供するオンラインツール。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての人に無料で利用できる AWS コンサルタントと APN パートナーコンサルタントです。
移行準備状況評価 (MRA)	組織のクラウド対応状況に関するインサイトを獲得し、長所と短所を特定し、AWS CAF を使用して特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイド を参照してください。MRAは、AWS 移行戦略 の第一段階です。
大規模な移行	アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。
移行ファクトリー	自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー DevOps スプリントで働く専門家。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、『』を参照してください。マイグレーション・ファクトリーの議論そしてクラウド移行ファクトリーガイドこのコンテンツセットで。
移行メタデータ	移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例として、ターゲットサブネット、セキュリティグループ、AWS アカウントが挙げられます。
移行パターン	移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: Amazon EC2 への移行を再ホストするAWSApplication Migration Ser
移行戦略	ワークロードを AWS クラウドに移行するために使用するアプローチ。詳細については、この用語集の「7 Rs (p. 11) エントリー」と、「組織を動員して大規模な移行を加速する」を参照してください。
オペレーショナルレベルアグリーメント (OLA)	サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。
オペレーション統合 (OI)	クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイド を参照してください。
組織変更管理 (OCM)	人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。AWS 移行戦略では、クラウド導入プロジェクトに必要な変更の

	スピードから、このフレームワークは 人材の高速化 と呼ばれます。詳細については、OCM ガイド を参照してください。
プレイブック	クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。
ポートフォリオ評価	移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行準備状況ガイド 」を参照してください。
実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス	プロジェクト内のロールと責任を定義して割り当てるマトリックス。例えば、お客様が RACI を作成して、セキュリティコントロールの所有権を定義したり、移行プロジェクト内の特定のタスクの役割と責任を特定したりできます。
ランブック	特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。
サービスレベルアグリーメント (SLA)	サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。
タスクリスト	ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。
ワークストリーム	特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。
ゾンビアプリケーション	平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

モダナイゼーション用語

以下は、AWS 規範的ガイダンスによって提供されるモダナイゼーション関連戦略、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。

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。このコンセプトは、エリック・エバンスの著書で紹介されました。ドメイン主導型設計: ソフトウェアの中心にある複雑さへの取り組み (ボストン：アディソン・ウェズリー・プロフェッショナル、2003年) 。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 型	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	小さいな DevOps 2 枚のピザで養うことができるチーム。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、『』を参照してください。2 枚ピザチームのセクションについて
DevOps オンAWSホワイトペーパー。

ドキュメント履歴

このガイドは、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. 18)	—	2021 年 1 月 11 日