aws

でのエージェント AI の基礎 AWS

AWS 規範ガイダンス

AWS 規範ガイダンス: でのエージェント AI の基礎 AWS

Table of Contents

でのエージェント AI の基礎 AWS

Aaron Sempf、Amazon Web Services

2025 年 7 月 (ドキュメント履歴)

ますますインテリジェントで分散した自律的なシステムの世界では、エージェントの概念、つまり環境を認識し、その状態について理由を理解し、意図を持って行動できるエンティティが基盤となっています。エージェントは、単に指示を実行するプログラムではなく、ユーザー、システム、または組織に代わって意思決定を行う、目標指向のコンテキスト対応エンティティです。これらの出現は、ソフトウェアの構築方法と考え方の変化を反映しています。手続き型ロジックと事後対応型オートメーションから、自律性と目的を持って運用されるシステムへの移行です。

AI、分散システム、ソフトウェアエンジニアリングの交差点には、エージェント AI と呼ばれる強力なパラダイムがあります。この新しい世代のインテリジェントシステムは、適応動作、複雑な調整、委任された意思決定が可能なソフトウェアエージェントで構成されています。

このガイドでは、最新のソフトウェアエージェントを定義する原則を紹介し、エージェント AI への進化の概要を説明します。このシフトを説明するために、このガイドは概念的な背景を提供し、ソフトウェアエージェントのエージェント AI への進化をトレースします。

- ソフトウェアエージェントの概要では、ソフトウェアエージェントを定義し、従来のソフトウェアコンポーネントと比較し、確立されたフレームワークを活用することで、エージェントの動作を従来の自動化と区別する重要な特性を紹介します。
- ソフトウェアエージェントの目的は、ソフトウェアエージェントが存在する理由、果たす役割、解決する問題、インテリジェントな委任の有効化方法、認知負荷の軽減方法、動的環境での適応動作のサポート方法を調べます。
- ソフトウェアエージェントの進化は、自律性と同時実行の初期の概念から、マルチエージェントシステムや正式なエージェントアーキテクチャの出現まで、ソフトウェアエージェントを形成した知的財産と技術のマイルストーンをトレースし、生成 AI と収束します。
- エージェント AI へのソフトウェアエージェントは、分散エージェントモデルと基盤モデル、サーバーレスコンピューティング、オーケストレーションプロトコルを組み合わせた数十年にわたる進歩の頂点としてエージェント AI を導入します。このセクションでは、この収束により、自律性、非同期性、真の機関を大規模に運用する新世代のインテリジェントでツールを使用するエージェントがどのように実現されるかについて説明します。

対象者

このガイドは、最新のクラウドソリューションにこのテクノロジーを採用する前に、ソフトウェアエージェントのエージェント AI への歴史、主要概念、進化を理解したいと考えているアーキテクト、デベロッパー、テクノロジーリーダーを対象としています AWS。

目的

エージェントアーキテクチャを採用することで、組織は次のことを行うことができます。

- 価値実現までの時間を短縮する: ナレッジ作業を自動化およびスケーリングし、手動作業とレイテンシーを削減します。
- カスタマーエンゲージメントの向上: ドメイン間でインテリジェントアシスタントを提供します。
- 運用コストの削減: 以前は人間による入力や監視が必要だった決定フローを自動化します。
- イノベーションと差別化を促進する: リアルタイムで適応、学習、競争するインテリジェントな製品を構築します。
- レガシーワークフローをモダナイズする: スクリプトとモノリスをモジュラー推論エージェントに再フレームします。

このコンテンツシリーズについて

このガイドは、AI 駆動型ソフトウェアエージェントを構築するためのアーキテクチャ設計図と技術ガイダンスを提供する一連の出版物の一部です AWS。シリーズには以下が含まれます。

- でのエージェント AI の運用 AWS
- でのエージェント AI の基礎 AWS (このガイド)
- でのエージェント AI のパターンとワークフロー AWS
- でのエージェント AI フレームワーク、プロトコル、ツール AWS
- でのエージェント AI 用のサーバーレスアーキテクチャの構築 AWS
- でのエージェント AI 用のマルチテナントアーキテクチャの構築 AWS

このコンテンツシリーズの詳細については、「エージェント AI」を参照してください。

ソフトウェアエージェントの紹介

ソフトウェアエージェントの概念は、1960 年代の自律型エンティティの基盤から 1990 年代初頭の正式な探索へと大きく進化しました。決定論的なスクリプトから適応的でインテリジェントなアプリケーションまで、デジタルシステムがますます複雑になるにつれて、ソフトウェアエージェントは、コンピューティングシステムで自律的でコンテキスト対応型の目標駆動型の動作を可能にする上で不可欠な構成要素となっています。クラウドネイティブアーキテクチャと AI 拡張アーキテクチャ、特に生成 AI、大規模言語モデル (LLMs)、Amazon Bedrock などのプラットフォームの出現に伴い、ソフトウェアエージェントは新しい能力とスケールのレンズを通じて再定義されています。

この概要は、「Software Agents: An Overview by Hyacinth S. Nwana」(Nwana 1996) から抜粋したものです。ソフトウェアエージェントを定義し、その概念的なルートについて説明し、議論を現代のフレームワークに拡張して、現代のソフトウェアエージェントの 3 つの包括的な原則である自律性、非同期性、機関を定義します。これらの原則は、ソフトウェアエージェントを他のタイプのサービスやアプリケーションと区別し、これらのエージェントが分散型のリアルタイム環境で目的、レジリエンス、インテリジェンスで運用できるようにします。

このセクションの内容

- 自律性から分散インテリジェンスへ
- Nwana の類型とソフトウェアエージェントの台頭
- 最新のソフトウェアエージェントの 3 つの柱

自律性から分散インテリジェンスへ

ソフトウェアエージェントという用語がメインストリームに入る前に、初期のコンピューティング研究では、独立して行動し、入力に反応し、内部のルールや目的に基づいて決定を下すことができるシステムである自律型デジタルエンティティの概念を調べました。これらの初期のアイデアは、エージェントパラダイムになるものの概念的な基礎を確立しました。(過去のタイムラインについては、このガイドの後半にある「ソフトウェアエージェントの進化」セクションを参照してください)。

自律性の初期概念

人間のオペレーターから独立して動作するマシンやプログラムの概念は、数十年にわたってシステムデザイナーを惹きつけてきました。サイバーネットワーク、人工知能、管理システムの初期の研究では、ソフトウェアが自己制御的な動作をし、変化に動的に対応し、継続的な人間の監督なしで動作する方法を調べました。

これらのアイデアは、インテリジェントシステムの中核的な属性として自律性を導入し、反応や実行だけでなく、決定して行動できるソフトウェアが出現する準備を整えました。

アクターモデルと非同期実行

1970 年代に、A Universal Modular ACTOR Formalism for Artificial Intelligence (Hewitt et al. 1973) という論文で紹介されたアクターモデルは、分散型でメッセージ駆動型の計算について考えるための正式なフレームワークを提供しました。このモデルでは、アクターは非同期メッセージを渡すことによって排他的に通信し、スケーラブル、同時、耐障害性のあるシステムを有効にする独立したエンティティです。

アクターモデルは、最新のエージェント設計に影響を与え続ける 3 つの主要な属性を強調しました。

- 状態と動作の分離
- エンティティ間の非同期インタラクション
- タスクの動的作成と委任

これらの属性は、分散システムのニーズと一致しており、クラウドネイティブ環境におけるソフトウェアエージェントの運用特性を事前に把握しています。

分散インテリジェンスとマルチエージェントシステム

コンピューティングシステムが 1960 年代以降に相互接続されるにつれて、研究者は分散人工知能 (DAI) を調査しました。このフィールドでは、複数の自律エンティティがシステム全体で協力的または競争的に機能する方法に焦点を当てました。DAI はマルチエージェントシステムの開発につながり、各エージェントにはローカルの目標、認識、推論がありますが、より広範で相互接続された環境で動作します。

この分散インテリジェンスのビジョンは、意思決定が分散され、エージェントのインタラクションから緊急の動作が発生するため、最新のエージェントベースのシステムの概念と構築方法の中心をなしています。

Nwana の類型とソフトウェアエージェントの台頭

1990 年代半ばのソフトウェアエージェントの概念の形式化は、インテリジェントシステムの進化の転換点となりました。この形式化に対する最も影響力のある貢献の 1 つは、Hyacinth S. Nwana の半

文「Software Agents: An Overview (Nwana 1996)」であり、さまざまなディメンションにわたってソフトウェアエージェントを分類および理解するための最初の包括的なフレームワークの 1 つを提供しました。

このホワイトペーパーでは、Nwana がソフトウェアエージェントの調査の状態を調査し、エージェントの定義と実装方法の相違が増大していることを特定します。このホワイトペーパーでは、共通の概念フレームワークの必要性を強調し、主要な機能に従ってエージェントを分類する類型を提案しています。代表的なエージェントシステムを学界や業界からレビューし、エージェントを従来のプログラムやオブジェクトと区別し、エージェントベースのコンピューティングの課題と機会の概要を説明します。

Nwana は、ソフトウェアエージェントはモノリシック概念ではなく、洗練と能力のスペクトルに沿って存在することを強調しています。この類型は、この状況を明確にし、将来の設計と研究を導くのに役立ちます。

Nwana は、ソフトウェアエージェントを特定の環境で継続的かつ自律的に機能するソフトウェアエンティティとして定義します。これは、多くの場合、他のエージェントやプロセスに属します。この定義は、次の 2 つの主要な特性を強調しています。

- 継続性: エージェントは、人間による継続的な介入を必要とせずに、時間の経過とともに永続的に動作します。
- 自律性: エージェントには、環境に対する認識に基づいて意思決定を行い、独立して行動する機能があります。

この定義は、Nwana のエージェントタイプと組み合わせて、委任された権限 (自律性による) と積極性をエージェントの基本的な特性として強調します。直接コマンドにのみ応答するのではなく、別のエンティティに代わって独立して行動し、目標を求めて行動を開始するエージェントの能力を強調することで、エージェントとサブルーチンまたはサービスを区別します。

Nwana のエージェントタイプ

さまざまなタイプのエージェントをさらに区別するために、Nwana は 6 つの主要な属性に基づく分類システムを導入します。

- 自律性: エージェントは、人間や他の人から直接介入されることなく動作します。
- ソーシャル能力: エージェントは、通信メカニズムを使用して他のエージェントや人間とやり取りします。
- 応答性: エージェントは環境を認識し、タイムリーに応答します。

- プロアクティブ: エージェントは、イニシアチブを取ることで目標指向の動作を示します。
- 適応性と学習: エージェントは、経験を通じて時間の経過とともにパフォーマンスを向上させます。
- モビリティ: エージェントは、さまざまなシステム環境またはネットワークを移動できます。

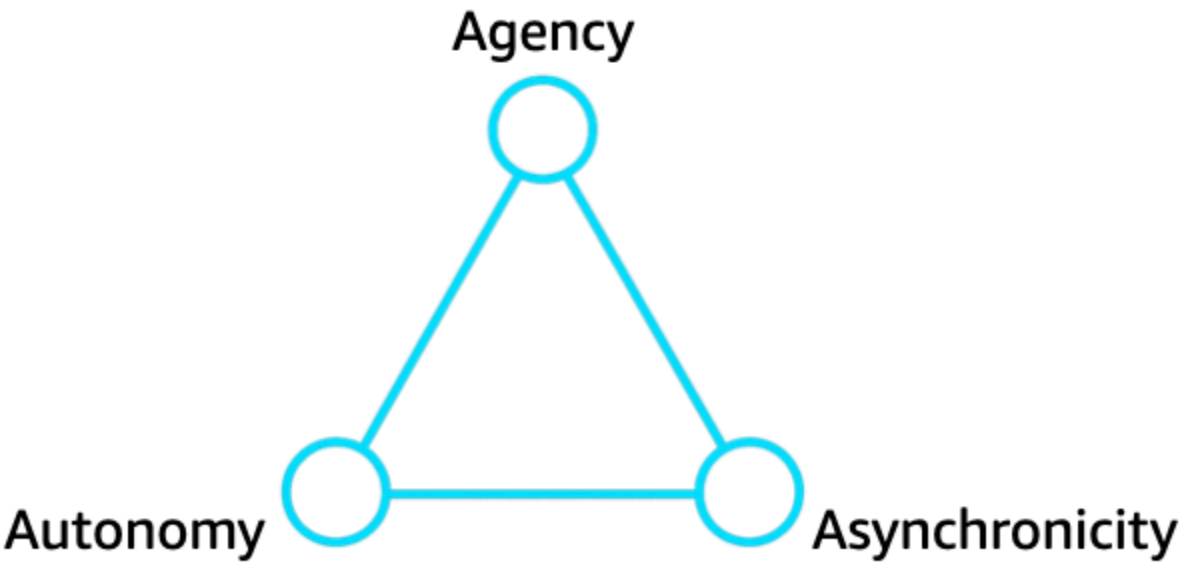
類型学から最新のエージェント原則まで

Nwana の作業は、計算コミュニティがソフトウェアの進化する形態の機関を評価できる分類と基本的なレンズの両方として機能しました。彼は自律性、積極性、およびユーザーやシステムに代わって行動するという概念に重点を置き、現在エージェント行動と見なしているものの基盤を確立しました。

テクノロジーと環境は、特に生成 AI、サーバーレスインフラストラクチャ、マルチエージェントオーケストレーションフレームワークの増加に伴って変化していますが、Nwana の作業からの基本的なインサイトは依然として関連しています。これらは、初期エージェント理論とソフトウェアエージェントの 3 つの最新の柱との間の重要な橋渡しとなります。

最新のソフトウェアエージェントの 3 つの柱

今日の AI を活用したプラットフォーム、マイクロサービスアーキテクチャ、イベント駆動型システムのコンテキストでは、ソフトウェアエージェントは、自律性、非同期性、機関という 3 つの相互依存の原則によって定義できます。次の図と以降の図では、三角形は最新のソフトウェアエージェントのこれら 3 つの柱を表しています。



自律性

最新のエージェントは独立して動作します。人間のプロンプトを必要とせずに、内部の状態と環境コンテキストに基づいて意思決定を行います。これにより、データにリアルタイムで対応し、独自のライフサイクルを管理し、目標と状況に応じた入力に基づいて動作を調整できます。

自律性はエージェント動作の基盤です。これにより、エージェントは継続的な監視やハードコードされた制御フローなしで機能できます。

非同期性

エージェントは基本的に非同期です。つまり、イベント、シグナル、刺激の発生時に応答し、ブロック呼び出しや線形ワークフローに依存しません。この特性により、スケーラブルでノンブロッキングの通信、分散環境での応答性、コンポーネント間の疎結合が可能になります。

非同期性により、エージェントはリアルタイムシステムに参加し、他のサービスやエージェントと迅速かつ効率的に調整できます。

定義原則としての機関

自律性と非同期性が必要ですが、これらの機能だけでは、システムを真のソフトウェアエージェントにするには不十分です。重要な差別化要因は、以下を導入する機関です。

- 目標指向の動作: エージェントは目標を追求し、目標に対する進捗状況を評価します。
- 意思決定: エージェントはオプションを評価し、ルール、モデル、学習したポリシーに基づいてアクションを選択します。
- 委任インテント: エージェントは個人、システム、または組織に代わって行動し、目的意識が組み込まれています。
- コンテキスト推論: エージェントは環境のメモリまたはモデルを組み込み、動作をインテリジェントにガイドします。

自律型および非同期型のシステムは、依然として事後対応型サービスである可能性があります。ソフトウェアエージェントになるのは、意図と目的を持って行動し、エージェントになる能力です。

目的を持つ機関

自律性、非同期性、および機関の原則により、システムは分散環境間でインテリジェントに、適応的に、独立して動作できます。これらの原則は数十年にわたる概念的およびアーキテクチャ的な進化に根ざしており、現在構築されている最も高度な AI システムの多くを支えています。

生成 AI、目標指向のオーケストレーション、マルチエージェントコラボレーションの新しい時代では、ソフトウェアエージェントが本当にエージェント的になる理由を理解することが不可欠です。エージェンシーを定義特性として認識することで、自動化を超えて、目的を持って自律型インテリジェンスの領域に移行できます。

ソフトウェアエージェントの目的

最新のシステムがますます複雑になり、分散され、インテリジェントになるにつれて、ソフトウェアエージェントの役割は、自律運用からユーザー支援テクノロジーまで、さまざまな分野にわたって目立つようになっています。しかし、ソフトウェアエージェントの根本的な目的は何ですか？ スクリプト、サービス、静的モデルを超えるシステムを設計し、代わりに認識、推論、行動が可能なエンティティにタスクを委任するのはなぜですか？

このセクションでは、ソフトウェアエージェントの基本的な目的、つまり、自律性、適応性、意図的なアクションに焦点を当てて、動的環境内でタスクをインテリジェントに委任できるようにする方法について説明します。ソフトウェアエージェントの概念的な基盤を導入し、その認知構造をトレースし、それらが解決するために一意に装備されている実際の問題を概説します。

このセクションの内容

アクターモデルからエージェントの認識まで

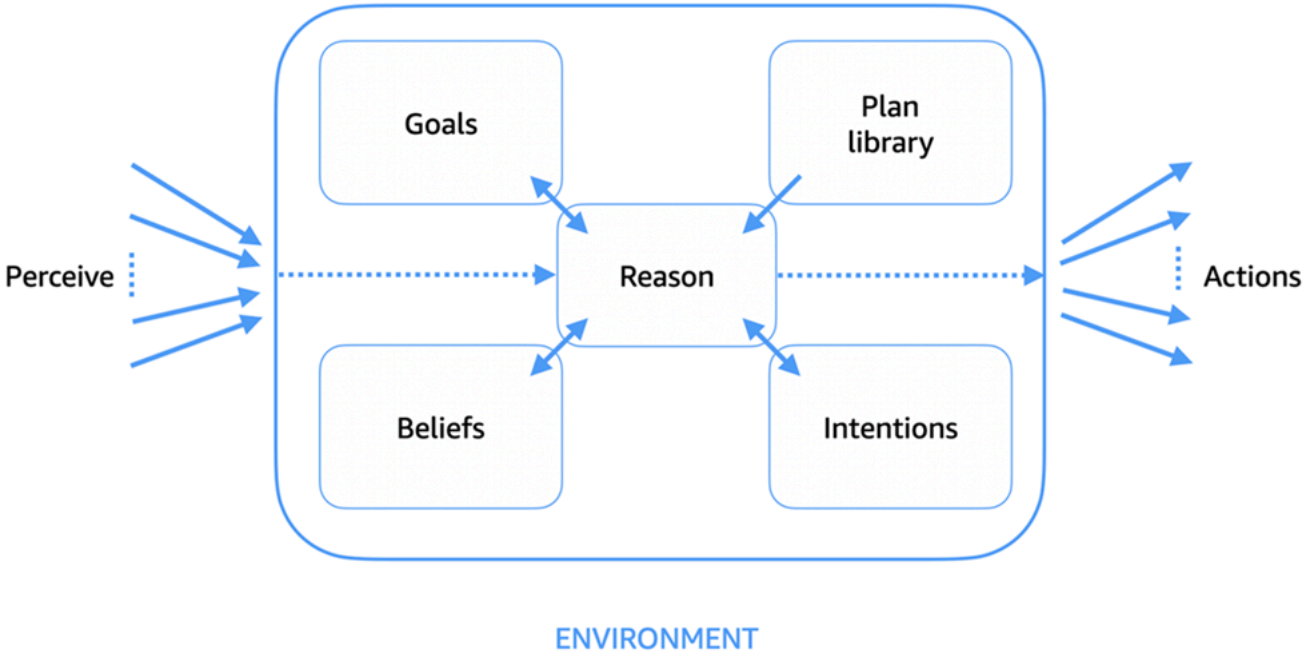
ソフトウェアエージェントの目的と構造は、初期の計算モデル、特に 1970 年代に Carl Hewitt によって導入されたアクターモデル (Hewitt et al. 1973)。

アクターモデルは、計算をアクターと呼ばれる独立した同時実行エンティティのコレクションとして扱います。各アクターは独自の状態をカプセル化し、非同期メッセージパスを介してのみやり取りし、新しいアクターを作成してタスクを委任できます。

このモデルは、分散型推論、反応、分離の概念的な基盤を提供し、これらすべてが最新のソフトウェアエージェントの動作アーキテクチャを支えています。

エージェント関数: 認識、理由、行動

すべてのソフトウェアエージェントの中核となるのは、認識、理由、アクションループと呼ばれる認知サイクルです。次の図表は、このプロセスを示したものです。これは、エージェントが動的環境で自律的に動作する方法を定義します。



- 認識: エージェントは環境から情報 (イベント、センサー入力、API シグナルなど) を収集し、内部の状態や考え方を更新します。
- 理由: エージェントは、プランライブラリまたはロジックシステムを使用して、現在の考え方、目標、コンテキストに関する知識を分析します。このプロセスには、目標の優先順位付け、競合の解決、または意図的な選択が含まれる場合があります。
- アクション: エージェントは、委任された目標の達成に近づくアクションを選択して実行します。

このアーキテクチャは、エージェントが厳格なプログラミングを超えて機能する機能をサポートし、柔軟でコンテキストに敏感な目標指向の動作を可能にします。これは、ソフトウェアエージェントのより広範な目的を導くメンタルフレームワークを形成します。

自律的なコラボレーションと意図

ソフトウェアエージェントの目的は、最新のコンピューティングに自律性、コンテキスト認識、インテリジェントな委任をもたらすことです。エージェントはアクターモデルの原則に基づいて構築され、認識、理由、アクションサイクルに具体化されるため、事後対応型だけでなく、プロアクティブで目的意識の高いシステムが可能になります。

エージェントは、ソフトウェアが複雑な環境で決定、適応、行動できるようにします。これらはユーザーを表し、目標を解釈し、マシン速度でタスクを実装します。エージェント AI の時代が深まるに

つれて、ソフトウェアエージェントは人間の意図とインテリジェントなデジタルアクションの間の運用インターフェイスになりつつあります。

インテントの委任

従来のソフトウェアコンポーネントとは異なり、ソフトウェアエージェントはユーザー、別のシステム、または高レベルのサービスなど、 他の何かに代わって動作します。委任されたインテントを保持します。つまり、次のようになります。

- 開始後は個別に操作します。
- 委任者の目標に沿った選択を行います。
- 実行時の不確実性とトレードオフをナビゲートします。

エージェントは、指示と結果のギャップを埋めるため、ユーザーは明示的な指示を必要とせずに、より高い抽象化レベルでインテントを表現できます。

動的で予測不可能な環境での運用

ソフトウェアエージェントは、条件が絶えず変化し、データがリアルタイムで到着し、制御とコンテキストが分散される環境向けに設計されています。

正確な入力や同期実行を必要とする静的プログラムとは異なり、エージェントは周囲に適応し、動的に応答します。これは、クラウドネイティブインフラストラクチャ、エッジコンピューティング、モノのインターネット (IoT) ネットワーク、リアルタイムの意思決定システムにおける重要な機能です。

人間の認知負荷の軽減

ソフトウェアエージェントの主な目的の 1 つは、人間の認知的負担と運用上の負担を軽減することです。エージェントは次のことができます。

- システムとワークフローを継続的にモニタリングします。
- 事前定義された条件または緊急の条件を検出して対応します。
- 反復的で大量の意思決定を自動化します。
- 最小限のレイテンシーで環境の変化に対応します。

意思決定がユーザーからエージェントに移行すると、システムはより応答性、回復力、人間中心になり、新しい情報や中断にリアルタイムで適応できます。これにより、複雑な環境や大規模な環境での

反応のターンアラウンドが短縮され、運用の継続性が向上します。その結果、マイクロレベルの意思決定から戦略的監督、創造的な問題解決まで、人間の焦点が変わりました。

分散インテリジェンスの有効化

ソフトウェアエージェントが個別にまたはまとめて動作する能力により、環境や組織間で連携するマルチエージェントシステム (MAS) の設計が可能になります。これらのシステムは、タスクをインテリジェントに分散し、複合目標に向けて交渉、協力、または競争することができます。

例えば、グローバルサプライチェーンシステムでは、個々のエージェントが工場、配送、倉庫、ラストマイル配送を管理します。各エージェントはローカル自律性で動作します。ファクトリーエージェントはリソースの制約に基づいて本番稼働を最適化し、倉庫エージェントは在庫フローをリアルタイムで調整し、配送エージェントはトラフィックと顧客の可用性に基づいて出荷を再ルーティングします。

これらのエージェントは動的に通信して調整し、一元的な制御を行わずに、ポートの遅延やトラックの障害などの中断に適応します。システムの全体的なインテリジェンスは、これらのインタラクションから出現し、単一のコンポーネントの機能を超える回復力と最適化された物流を可能にします。

このモデルでは、エージェントはより広範なインテリジェンスファブリックのノードとして機能します。これらは、単一のコンポーネントだけでは処理できない問題を解決できる緊急システムを形成します。

反応だけでなく、目的を持って行動する

複雑なシステムでは、自動化だけでは不十分です。ソフトウェアエージェントの目的は、目的を持って行動し、目標の評価、コンテキストの重み付け、情報に基づいた選択を行うことです。つまり、ソフトウェアエージェントはトリガーにのみ応答するのではなく、目標を追求します。経験やフィードバックに基づいて、考えや意図を改訂できます。このコンテキストでは、信条は、認識 (入力とセンサー) に基づいて、エージェントによる環境の内部表現 (たとえば、「パッケージ X はウェアハウス A にあります」) を指します。目的とは、エージェントが目標を達成するために選択する計画を指します (「配信ルート B を使用して受信者に通知する」など) 。エージェントは、必要に応じてアクションをエスカレーション、延期、または適応させることもできます。

この意図性により、ソフトウェアエージェントは事後対応的なエグゼキュターだけでなく、インテリジェントシステムの自律的な共同作業者になります。

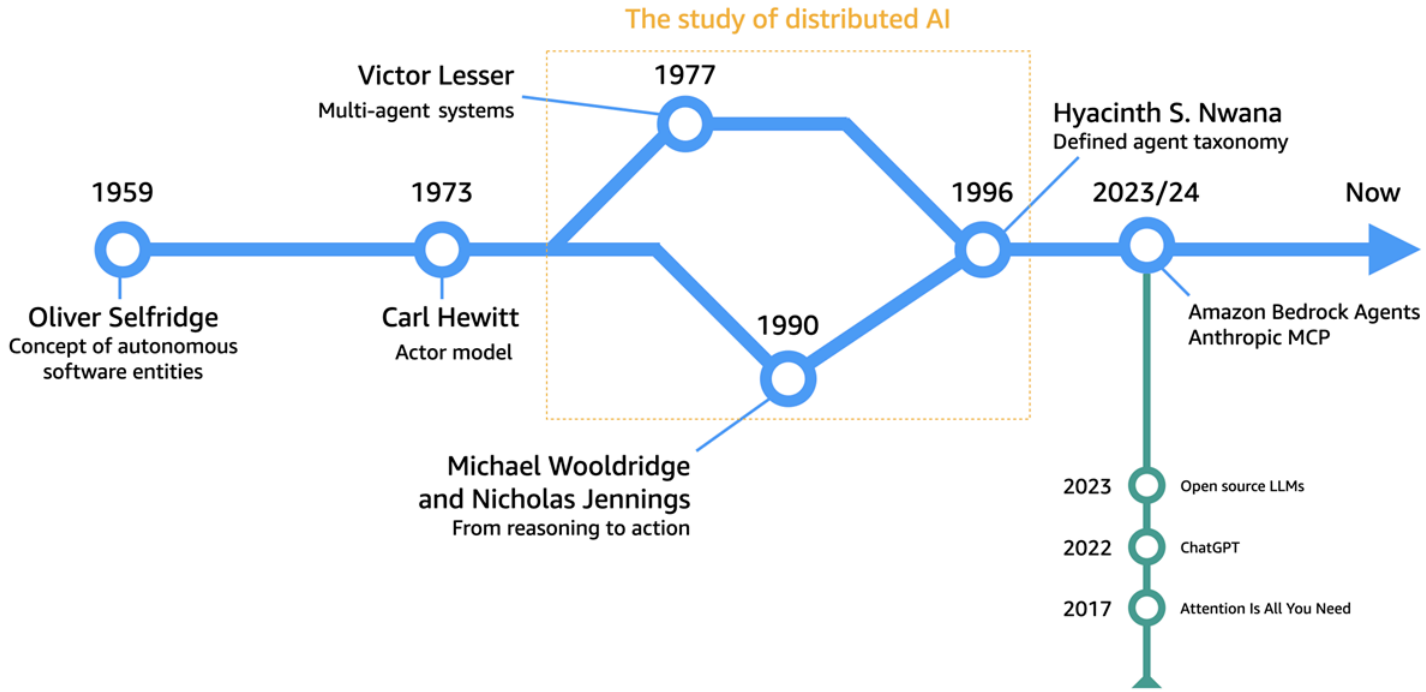
ソフトウェアエージェントの進化

シンプルな自動化システムからインテリジェント、自律型、目標指向のソフトウェアエージェントへのジャーニーは、コンピュータサイエンス、人工知能、分散システムの数十年にわたる進化を反映しています。

この進化の後、機械学習が台頭し、パラダイムが手作業のルールから統計パターン認識に移行しました。これらのシステムは、データから学び、認識、分類、意思決定の進歩を実現できます。

大規模言語モデル (LLMsは、スケール、アーキテクチャ、教師なし学習の収束を表します。LLMs は、タスク固有のトレーニングをほとんどまたはまったく行わずに、タスクの推論、生成、適応を行うことができます。LLMsをスケーラブルなクラウドネイティブインフラストラクチャと構成可能なアーキテクチャと組み合わせることで、エージェント AI の完全なビジョン、つまり自律性、コンテキスト認識、適応性をエンタープライズ規模で運用できるインテリジェントなソフトウェアエージェントを実現できるようになりました。

このセクションでは、次の図に示すように、基盤理論から最新のプラクティスまでのソフトウェアエージェントの履歴について説明します。分散人工知能 (DAI) とトランスフォーマーベースの生成 AI の収束に焦点を当て、エージェント AI の出現を形成した重要なマイルストーンを特定します。



このセクションの内容

- ソフトウェアエージェントの基盤

ソフトウェアエージェントの基盤

1959年 – オリバー・ セルフリッジ: ソフトウェアにおける自律性の誕生

ソフトウェアエージェントの根源は、Oliver Selfridge にまでさかのぼります。Oliver Selfridge は、自律的なソフトウェアエンティティ (デモ) の概念を導入しました。このプログラムは、環境を認識し、独立して動作できます (Selfridge 1959)。機械認識と学習の初期の仕事は、独立したインテリジェントシステムとしてのエージェントの将来の概念のための哲学的基礎を確立しました。

1973 年 – Carl Hewitt: アクターモデル

Carl Hewitt のアクターモデル (Hewitt et al. 1973) は、エージェントを独立した同時エンティティとして記述する正式な計算モデルです。このモデルでは、エージェントは独自の状態と動作をカプセル化し、非同期メッセージパスを使用して通信し、他のアクターを動的に作成してタスクを委任できます。

アクターモデルは、分散型のエージェントベースのシステムの理論的基盤とアーキテクチャパラダイムの両方を提供しました。このモデルは、Erlang プログラミング言語や Akka フレームワークなどの最新の同時実行実装を事前に考案しました。

フィールドの成熟: 推論からアクションへ

1977 年 – Victor Lesser: マルチエージェントシステム

1970 年代後半に、分散人工知能 (DAI) が登場しました。これは、マルチエージェントシステム (MAS) の開拓で広く認識されている Victor Lesser によって推進されました。彼の仕事は、独立したソフトウェアエンティティが協力、調整、交渉する方法に焦点を当てました (「 リソース」セクションを参照) 。この開発により、複雑な問題をまとめて解決できるシステムが生まれました。これは、分散インテリジェンスの構築に不可欠な飛躍です。

1990 年代 – マイケル・ウォルダーリッジとニコラス・ジェニングス: エージェントスペクトル

1990 年代までに、分散インテリジェンス分野は Michael" や Nicholas Jennings などの研究者からの貢献で成熟しました。これらの学者は、事後対応型から熟考型、非認知型システムから目標駆動型の推論エージェントまで、エージェントをスペクトルに沿って分類しました (Wooldridge and Jennings 1995)。彼らの研究では、エージェントはもはや抽象的なアイデアではなく、ロボットからエンタープライズソフトウェアまで、幅広い実用的な分野に適用されていると強調しました。

これらの研究者は、一元的な推論から分散アクションへの焦点の移行も導入しました。エージェントはもはや単なる思想家ではなく、自律性と目的を持つリアルタイム環境で運用された人々でした。

1996 年 – Hyacinth S. Nwana: エージェント概念の形式化

1996 年、Hyacinth S. Nwana は、影響力のある論文 Software Agents: An Overview を公開しました。これは、これまでで最も包括的なエージェントの分類を提供しました。彼のタイプには、自律性、社会的能力、反応性、非アクティブ性、学習性、モビリティなどの属性が含まれ、ソフトウェアエージェントと従来のソフトウェアコンストラクトを区別しました。

Nwana は、現在広く受け入れられている定義、言い換えも提供しました： ソフトウェアエージェントは、委任の概念から派生した、代理関係でユーザーまたは他のプログラムとして機能するソフトウェアベースのコンピュータプログラムです。

この形式化は、ソフトウェアエージェントを理論的なコンストラクトから実際のアプリケーションに移行する上で役立ちました。これにより、通信、ワークフローオートメーション、インテリジェントアシスタントなどの分野でエージェントベースのシステムが生成されました。

Nwana の仕事は、初期の分散 AI 研究と最新のエージェントの運用アーキテクチャの収束点にあります。これは、エージェントの認知理論と今日のシステムへの実際のデプロイとの間の重要な橋渡しです。

並列タイムライン: 大規模言語モデルの台頭

エージェントフレームワークが進化するにつれて、自然言語処理と機械学習で並行して収束的な改革が行われました。

- 2017 – トランスフォーマー: ホワイトペーパー「Attention Is All You Need」 (Vaswani et al. 2017) では、トランスフォーマーアーキテクチャが導入されました。これにより、機械による言語の処理方法と生成方法が劇的に改善されました。

- 2022 – ChatGPT: OpenAI は ChatGPT と呼ばれる GPT-3.5 へのチャットベースのインターフェイスをリリースしました。これにより、汎用 AI システムとの自然なインタラクティブな会話が可能になりました。
- 2023 年 – オープンソース LLMs: Llama、Falcon、Mistral のリリースにより、強力なモデルが広くアクセス可能になり、オープンソース環境とエンタープライズ環境でのエージェントフレームワークの開発が加速しました。

これらのイノベーションにより、言語モデルはコンテキストの解析、アクションの計画、レスポンスの連鎖が可能な推論エンジンに変わり、LLMs はインテリジェントなソフトウェアエージェントの主要なイネーブラーになりました。

タイムラインの収束: エージェント AI の出現

2023-2024 – エンタープライズグレードのエージェントプラットフォーム

分散ソフトウェアエージェントアーキテクチャとトランスフォーマーベースの LLMs の収束は、エージェント AI の台頭に至りました。

- Amazon Bedrock エージェントは、Amazon Bedrock の基盤モデルを使用して、目標駆動型のツールを使用するソフトウェアエージェントを構築する完全マネージド型の方法を導入しました。
- Anthropic の Model Context Protocol (MCP) では、大規模言語モデルが外部ツール、環境、メモリにアクセスして操作する方法を定義しました。これは、コンテキスト的、永続的、自律的な動作の鍵です。

これら 2 つのマイルストーンは、機関とインテリジェンスの合成を表しています。エージェントは静的ワークフローやリジッドオートメーションに制限されなくなりました。複数のステップにわたって推論を行い、ツールや APIs と連携し、コンテキスト状態を維持し、時間の経過とともに学習して適応できるようになりました。

2025 年 1 月 ~ 6 月 – エンタープライズ機能を拡張

2025 年前半、エージェント AI の状況は新しいエンタープライズ機能で大幅に拡大しました。2025 年 2 月、Anthropic は市場初のハイブリッド推論モデルである Claude 3.7 Sonnet をリリースし、MCP 仕様は広く採用されました。

Amazon Q Developer、Cursor、WindSurf などの AI コーディングアシスタントは MCP を統合して、コード生成、リポジトリ分析、開発ワークフローを標準化します。2025 年 3 月の MCP リ

リースでは、OAuth 2.1 セキュリティ統合、多様なデータアクセスのためのリソースタイプの拡張、Streamable HTTP による接続オプションの強化など、エンタープライズ対応の重要な機能が導入されました。この基盤に基づいて、2025 年 5 月に MCP ステアリング委員会に参加し、 AWS 新しいagent-to-agent通信機能に貢献したことを発表しました。これにより、エージェント AI 相互運用性の業界標準としてのプロトコルの位置がさらに強化されます。

2025 年 5 月、Strands Agents フレームワークをオープンソース化することで、エージェント AI ワークフローを構築するための顧客オプション AWS を強化しました。このプロバイダーに依存しないモデルに依存しないフレームワークにより、デベロッパーは深い AWS サービス統合を維持しながら、プラットフォーム間で基盤モデルを使用できます。AWS オープンソースブログで強調されているように、Strands Agents は、基盤モデルをエージェントインテリジェンスの中核とするモデルファーストの設計哲学に従います。これにより、お客様は特定のユースケースに合わせて高度な AI エージェントを簡単に構築してデプロイできます。

エマージェンス – エージェント AI

ソフトウェアエージェントの進化は、初期の自律性から最新の LLM 対応オーケストレーションまで、長くレイヤー化されています。「Oliver Selfridge」の「プログラムを認識するというビジョン」から始まったことは、コラボレーション、適応、理性を可能にする、インテリジェントでコンテキストに対応した目標駆動型のソフトウェアエージェントの堅牢なエコシステムに成長しました。

分散人工知能 (DAI) とトランスフォーマーベースの生成 AI の収束は、ソフトウェアエージェントがもはやツールだけでなく、インテリジェントシステムの自律アクターである新しい時代の始まりを示しています。

エージェント AI は、ソフトウェアシステムの次の進化を表します。自律型、非同期型、エージェント型のインテリジェントエージェントのクラスを提供し、委任されたインテントで動作し、動的で分散された環境で意図的に動作できます。エージェント AI は以下を統合します。

- マルチエージェントシステムとアクターモデルのアーキテクチャリネージュ
- 認識、理由、行動の認知モデル
- LLMs とトランスフォーマーの生成能力
- クラウドネイティブコンピューティングとサーバーレスコンピューティングの運用上の柔軟性

エージェント AI へのソフトウェアエージェント

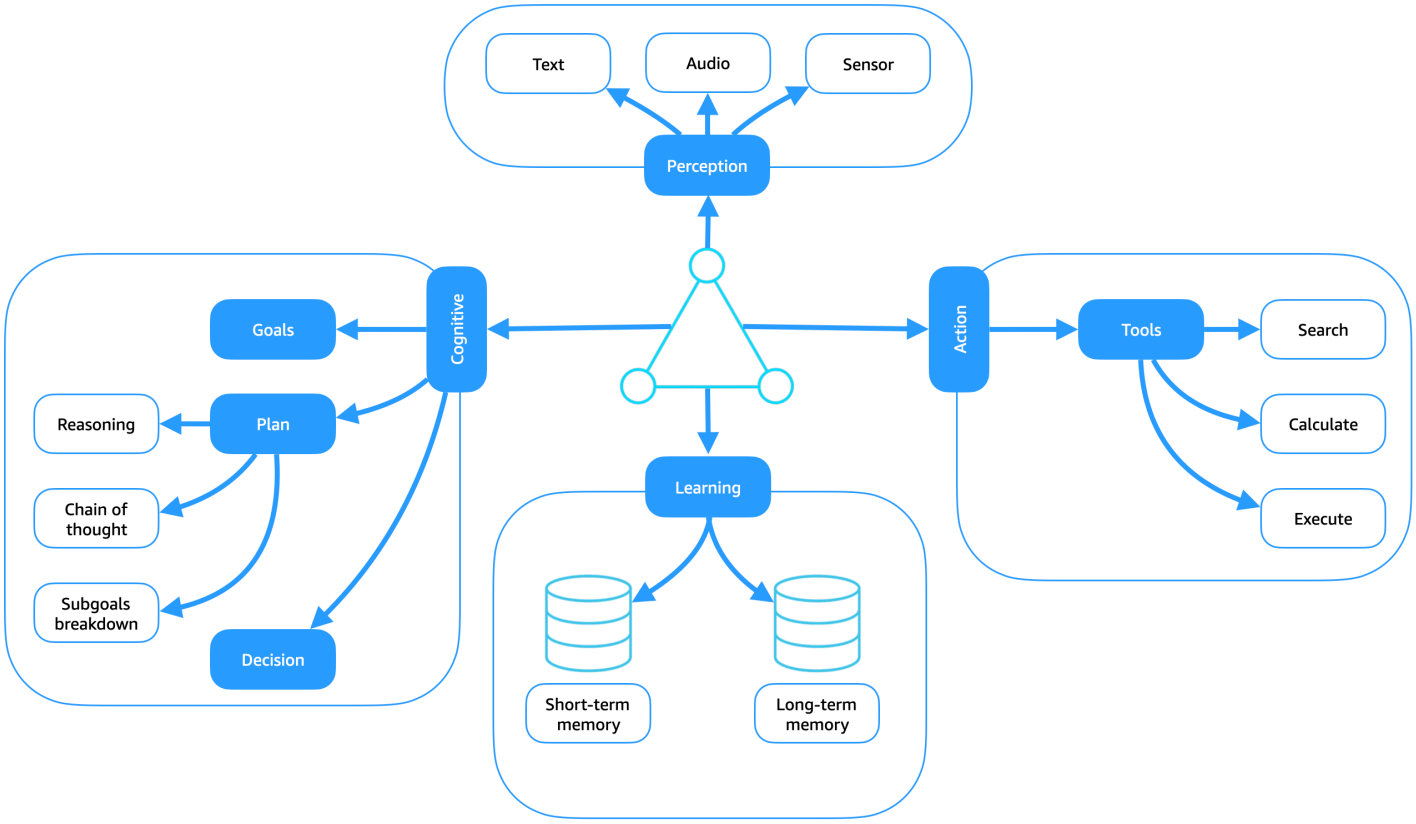
ソフトウェアエージェントは、環境、目標の理由を認識し、それに応じて行動するように設計された自律型のデジタルエンティティです。固定ロジックに従う従来のソフトウェアプログラムとは異なり、エージェントはコンテキスト入力と決定フレームワークに基づいて動作を適応させます。これにより、クラウドネイティブシステム、ロボット、インテリジェントオートメーション、現在は生成 AI オーケストレーションなどの動的な分散環境に最適です。

このセクションでは、ソフトウェアエージェントの中核的な構成要素を紹介し、認識、理由、行動モデルに基づいて、これらのコンポーネントが従来のアーキテクチャ内でどのように相互作用するかについて説明します。ここでは、生成 AI、特に大規模言語モデル (LLMs) がソフトウェアエージェントの推論と計画の方法をどのように変革したかについて説明します。これは、ルールベースのシステムから、エージェント AI のデータ駆動型の学習済みインテリジェンスへの根本的な移行を示しています。

このセクションの内容

ソフトウェアエージェントのコア構成要素

次の図は、ほとんどのインテリジェントエージェントで使用されている主要な機能モジュールを示しています。各コンポーネントは、複雑な環境で自律的に動作するエージェントの能力に貢献します。



認識、理由、アクションループのコンテキストでは、エージェントの推論機能は認識モジュールと学習モジュールの両方に分散されます。メモリと学習の統合を通じて、エージェントは過去の経験に基づく適応的推論を開発します。エージェントが環境内で動作すると、緊急のフィードバックループが作成されます。各アクションは将来の認識に影響を与え、結果として生じるエクスペリエンスは学習モジュールを通じてメモリと内部モデルに組み込まれます。この認識、推論、アクションの継続的なループにより、エージェントは時間の経過とともに改善し、認識、理由、アクションサイクル全体を完了できます。

認識モジュール

認識モジュールを使用すると、エージェントはテキスト、オーディオ、センサーなどの多様な入力モダリティを通じて環境とインターフェイスできます。これらの入力は、すべての推論とアクションが基づく raw データを形成します。テキスト入力には、自然言語プロンプト、構造化コマンド、またはドキュメントが含まれる場合があります。音声入力には、話し方や環境音が含まれます。センサー入力には、ビジュアルフィード、モーションシグナル、GPS 座標などの物理データが含まれます。認識の中核となる機能は、この未加工データから意味のある特徴と表現を抽出することです。これにより、エージェントは現在のコンテキストについて正確で実用的な理解を構築できます。このプロセスには、特徴抽出、オブジェクトまたはイベント認識、セマンティック解釈が含まれる場合があり、

認識、理由、アクションループの重要な最初のステップを形成します。効果的な認識により、ダウンストリームの推論と意思決定は、関連するup-to-date状況認識に基づいて行われます。

認知モジュール

コグニティブモジュールは、ソフトウェアエージェントの議論の中核として機能します。目標主導の計画と意思決定を通じて、認識を解釈し、インテントを形成し、目的を持った行動を導く責任があります。このモジュールは、入力を構造化された推論プロセスに変換します。これにより、エージェントはリアクティブではなく意図的に動作できます。これらのプロセスは、目標、計画、意思決定の 3 つの主要なサブモジュールによって管理されます。

目標サブモジュール

目標サブモジュールは、エージェントのインテントと方向を定義します。目標は、明示的 (たとえば、「場所に移動する」または「レポートを送信する」) または暗黙的 (たとえば、「ユーザーエンゲージメントを最大化する」または「レイテンシーを最小化する」) にすることができます。これらはエージェントの推論サイクルの中心であり、計画と決定のターゲット状態を提供します。

エージェントは目標に対する進捗状況を継続的に評価し、新しい認識や学習に基づいて目標の優先順位を変更または再生成する場合があります。この目標意識により、エージェントは動的な環境で適応できます。

サブモジュールの計画

計画サブモジュールは、エージェントの現在の目標を達成するための戦略を構築します。アクションシーケンスを生成し、タスクを階層的に分解し、事前定義されたプランまたは動的に生成されたプランから選択します。

非決定的または変化する環境で効果的に運用するには、計画は静的ではありません。最新のエージェントは、chain-of-thoughtシーケンスを生成し、サブ目標を中間ステップとして導入し、条件が変化したときにリアルタイムで計画を改訂できます。

このサブモジュールはメモリや学習と密接に接続し、エージェントは過去の結果に基づいて時間の経過とともに計画を絞り込むことができます。

意思決定サブモジュール

意思決定サブモジュールは、利用可能な計画とアクションを評価して、最も適切な次のステップを選択します。認識からのインプット、現在の計画、エージェントの目標、環境コンテキストを統合します。

意思決定は以下を考慮します。

- 競合する目標間のトレードオフ
- 信頼度のしきい値 (認識の不確実性など)
- アクションの結果
- エージェントの学習経験

アーキテクチャによっては、エージェントはシンボリック推論、ヒューリスティック、強化学習、または言語モデル (LLMs) に依存して、情報に基づいた意思決定を行う場合があります。このプロセスにより、エージェントの動作がコンテキスト対応、目標整合、適応性を維持できます。

アクションモジュール

アクションモジュールは、エージェントが選択した決定を実行し、外部世界または内部システムとやり取りして有意義な効果を生み出す責任があります。これは、インテントが動作に変換される認識、理由、アクションループの Act フェーズを表します。

コグニティブモジュールがアクションを選択すると、アクションモジュールは特殊なサブモジュールを通じて実行を調整します。ここで、各サブモジュールはエージェントの統合環境と一致します。

- 物理的な介入: ロボットシステムまたは IoT デバイスに埋め込まれたエージェントの場合、このサブモジュールは決定を実際の物理的な動きまたはハードウェアレベルの指示に変換します。

  例: ロボットのステアリング、バルブのトリガー、センサーのオン。
- 統合インタラクション: このサブモジュールは、ソフトウェアシステム、プラットフォーム、APIs。

  例: クラウドサービスへのコマンドの送信、データベースの更新、API の呼び出しによるレポートの送信。
- ツール呼び出し: エージェントは、特殊なツールを使用して次のようなサブタスクを実行することで、多くの場合機能を拡張します。
  - 検索: 構造化ナレッジソースまたは非構造化ナレッジソースのクエリ
  - 要約: 大きなテキスト入力を大まかな概要に圧縮する
  - 計算: 論理計算、数値計算、またはシンボリック計算の実行

  ツール呼び出しは、モジュール式の呼び出し可能なスキルを通じて複雑な動作構成を可能にします。

学習モジュール

学習モジュールを使用すると、エージェントは経験に基づいて時間の経過とともに適応、一般化、改善できます。認識とアクションからのフィードバックを使用して、エージェントの内部モデル、戦略、決定ポリシーを継続的に改善することで、推論プロセスをサポートします。

このモジュールは、短期メモリと長期メモリの両方と連携して動作します。

- 短期メモリ: ダイアログの状態、現在のタスク情報、最近の観測値などの一時的なコンテキストを保存します。これは、エージェントがインタラクションやタスク内で継続性を維持するのに役立ちます。
- 長期記憶: 以前に遭遇した目標、アクションの結果、環境状態など、過去の経験からの永続的な知識をエンコードします。長期メモリにより、エージェントはパターンを認識し、戦略を再利用し、ミスの繰り返しを回避できます。

学習モード

学習モジュールは、さまざまな環境とエージェントロールをサポートする、教師あり学習、教師なし学習、強化学習など、さまざまなパラダイムをサポートしています。

- 教師あり学習: ラベル付きの例に基づいて内部モデルを更新し、多くの場合、人間のフィードバックやトレーニングデータセットから更新します。

  例: 以前の会話に基づいてユーザーインテントを分類する方法。
- 教師なし学習: 明示的なラベルなしでデータの非表示パターンまたは構造を識別します。

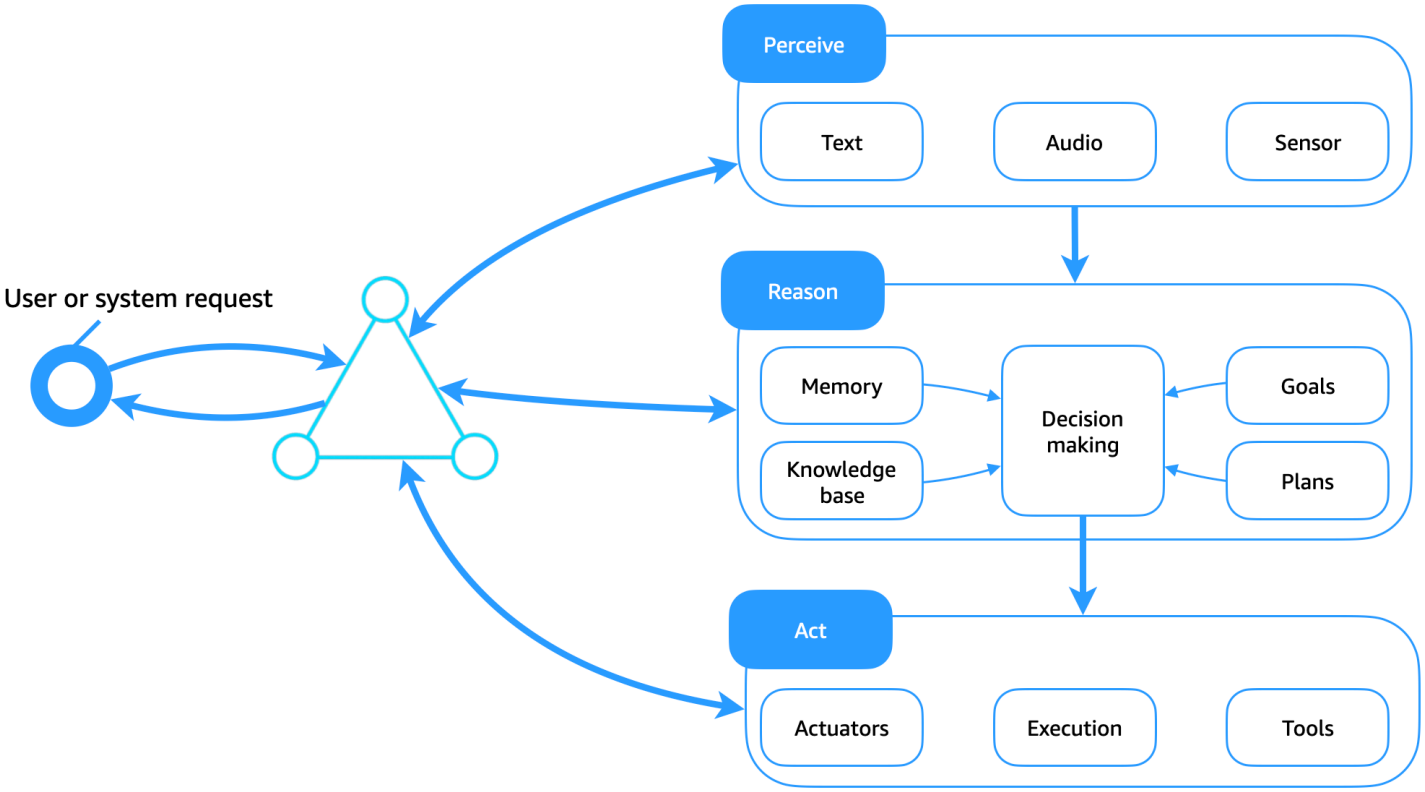
  例: 異常を検出するための環境シグナルのクラスター化。
- 強化学習: インタラクティブ環境で累積報酬を最大化することで、試行錯誤を通じて動作を最適化します。

  例: どの戦略が最速のタスク完了につながるかを知る。

学習は、エージェントの認知モジュールと緊密に統合されます。過去の成果に基づいて計画戦略を改良し、過去の成功の評価を通じて意思決定を強化し、認識と行動のマッピングを継続的に改善します。このクローズドラーニングとフィードバックループを通じて、エージェントは事後対応的な実行を超えて進化し、時間の経過とともに新しい目標、条件、コンテキストに適応できる自己改善システムになります。

従来のエージェントアーキテクチャ: 認識、理由、行動

次の図は、前のセクションで説明した構成要素が、認識、理由、行動のサイクルでどのように動作するかを示しています。



Perceive モジュール

認識モジュールは、外部ワールドとのエージェントの知覚インターフェイスとして機能します。生の環境入力を、推論を通知する構造化表現に変換します。これには、テキスト、オーディオ、センサー信号などのマルチモーダルデータの処理が含まれます。

- テキスト入力は、ユーザーコマンド、ドキュメント、またはダイアログから取得できます。
- 音声入力には、話し方や環境音が含まれます。
- センサー入力は、モーション、ビジュアルフィード、GPS などの実際の信号をキャプチャします。

raw 入力が取り込まれると、認識プロセスは特徴抽出を実行し、次にオブジェクトまたはイベントの認識とセマンティック解釈を実行して、現在の状況の意味のあるモデルを作成します。これらの出力

は、ダウンストリームの意思決定のための構造化されたコンテキストを提供し、実際の観測でエージェントの推論を固定します。

理由モジュール

理由モジュールは、エージェントのコグニティブコアです。コンテキストを評価し、インテントを策定し、適切なアクションを決定します。このモジュールは、学習した知識と推論の両方を使用して、目標主導型の動作を調整します。

理由モジュールは、緊密に統合されたサブモジュールで構成されます。

- メモリ: ダイアログの状態、タスクコンテキスト、エピソード履歴を短期形式と長期形式の両方で維持します。
- ナレッジベース: シンボリックルール、オントロジー、学習モデル (埋め込み、ファクト、ポリシーなど) へのアクセスを提供します。
- 目標と計画: 望ましい成果を定義し、それを達成するためのアクション戦略を構築します。目標は動的に更新でき、計画はフィードバックに基づいて適応的に変更できます。
- 意思決定: オプションを比較検討し、トレードオフを評価し、次のアクションを選択することで、中央の調停エンジンとして機能します。このサブモジュールでは、信頼度しきい値、目標の調整、コンテキスト制約の要因が示されます。

これらのコンポーネントを組み合わせることで、エージェントは環境について推論し、信条を更新し、パスを選択し、一貫性のある適応的な方法で動作できます。理由モジュールは、認識と動作のギャップを埋めます。

Act モジュール

act モジュールは、エージェントが選択した決定を実行するために、デジタル環境または物理環境のいずれかとやり取りしてタスクを実行します。ここで意図がアクションになります。

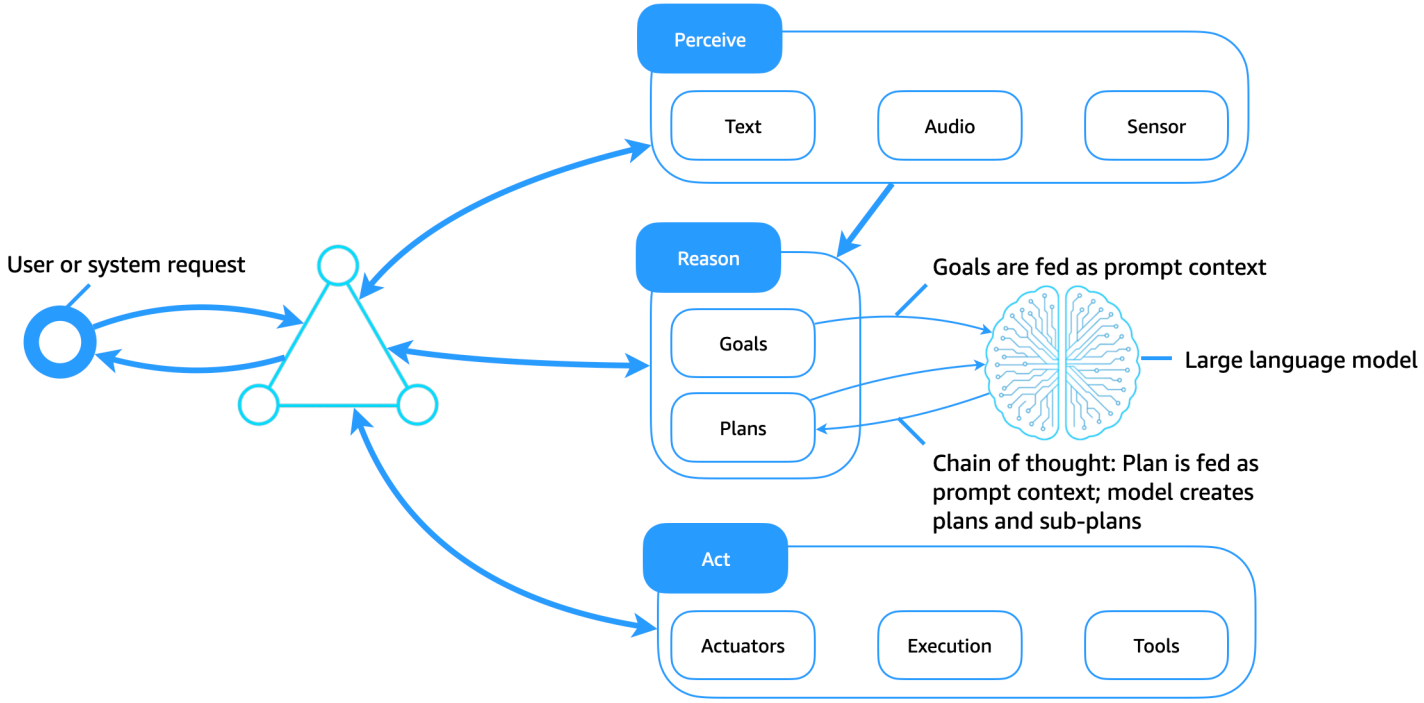
このモジュールには、次の 3 つの機能チャネルが含まれています。

- アクチュエータ: 物理的に存在するエージェント (ロボットや IoT デバイスなど) の場合、 は移動、操作、シグナリングなどのハードウェアレベルのインタラクションを制御します。
- 実行: APIs、システムの更新など、ソフトウェアベースのアクションを処理します。
- ツール: 検索、要約、コード実行、計算、ドキュメント処理などの機能を有効にします。これらのツールは、多くの場合動的でコンテキスト対応であり、エージェントのユーティリティを拡張します。

act モジュールの出力は環境にフィードバックし、ループを閉じます。これらの結果は、エージェントによって再び認識されます。エージェントの内部状態を更新し、将来の決定を通知するため、認識、理由、行動のサイクルを完了します。

生成 AI エージェント: シンボリックロジックを LLMs

次の図は、大規模言語モデル (LLMs) がソフトウェアエージェントの柔軟でインテリジェントなコグニティブコアとしてどのように機能するかを示しています。静的プランライブラリとハンドコードされたルールに依存する従来のシンボリックロジックシステムとは異なり、LLMs適応的推論、コンテキストプランニング、動的ツールの使用を可能にし、エージェントが認識、理由、行動する方法を変換します。



主な機能強化

このアーキテクチャは、従来のエージェントアーキテクチャを次のように強化します。

- 認識エンジンとしての LLMs: 目標、計画、クエリは プロンプトコンテキストとしてモデルに渡されます。LLM は推論パス (思考の連鎖など) を生成し、タスクをサブ目標に分解して、次のアクションを決定します。
- プロンプトによるツールの使用: LLMs は、ツールの使用エージェント、または APIs を呼び出し、出力を検索、クエリ、計算、解釈するための推論とアクション (ReAct) プロンプトを通じて指示できます。

- コンテキスト対応計画: エージェントは、ハードコードされた計画ライブラリを必要とせずに、エージェントの現在の目標、入力環境、フィードバックに基づいて計画を動的に生成または修正します。
- プロンプトコンテキストをメモリとして: シンボリックナレッジベースを使用する代わりに、エージェントはメモリ、計画、目標をモデルに渡されるプロンプトトークンとしてエンコードします。
- 数ショットのコンテキスト内学習による学習: LLMsプロンプトエンジニアリングを通じて動作を適応させるため、明示的な再トレーニングやリジッドプランライブラリの必要性が軽減されます。

LLM ベースのエージェントの長期メモリの達成

構造化ナレッジベースに長期メモリを保存していた従来のエージェントとは異なり、生成 AI エージェントは LLMs のコンテキストウィンドウの制限内で動作する必要があります。メモリを拡張し、永続的なインテリジェンスをサポートするために、生成 AI エージェントは、エージェントストア、検索拡張生成 (RAG)、コンテキスト内学習とプロンプト連鎖、事前トレーニングなど、いくつかの補完的な手法を使用します。

エージェントストア: 外部長期メモリ

エージェントの状態、ユーザー履歴、決定、結果は、長期エージェントメモリストア (ベクトルデータベース、オブジェクトストア、ドキュメントストアなど) に保存されます。関連するメモリはオンデマンドで取得され、実行時に LLM プロンプトコンテキストに挿入されます。これにより、永続的なメモリループが作成され、エージェントはセッション、タスク、またはインタラクション間で継続性を保持します。

RAG

RAG は、取得した知識を生成機能と組み合わせることで LLM のパフォーマンスを向上させます。目標またはクエリが発行されると、エージェントは検索インデックスを検索します (たとえば、ドキュメントのセマンティック検索、以前の会話、構造化された知識など) 。取得した結果は LLM プロンプトに追加され、外部ファクトまたはパーソナライズされたコンテキストに基づいて生成されます。この方法では、エージェントの有効なメモリが拡張され、信頼性と事実の正確性が向上します。

コンテキスト内学習とプロンプト連鎖

エージェントは、セッション内のトークンコンテキストと構造化されたプロンプトチェイニングを使用して、短期的なメモリを維持します。現在の計画、以前のアクション結果、エージェントステータスなどのコンテキスト要素は、動作をガイドするために呼び出し間で渡されます。

継続的な事前トレーニングと微調整

ドメイン固有のエージェントの場合、LLMs はログ、エンタープライズデータ、製品ドキュメントなどのカスタムコレクションで引き続き事前トレーニングできます。または、ヒューマンフィードバック (RLHF) からの命令ファインチューニングまたは強化学習は、エージェントのような動作をモデルに直接埋め込むことができます。これにより、推論パターンがプロンプト時間ロジックからモデルの内部表現に移行し、プロンプトの長さが短縮され、効率が向上します。

エージェント AI の利点の組み合わせ

これらの手法を併用すると、生成 AI エージェントは次のことが可能になります。

- 経時的なコンテキスト認識を維持します。
- ユーザー履歴または設定に基づいて動作を適応させます。
- up-to-date、事実、またはプライベートな知識を使用して意思決定を行います。
- 永続的、準拠、説明可能な動作でエンタープライズユースケースにスケールします。

外部メモリ、取り出しレイヤー、継続的なトレーニングで LLMs を強化することで、エージェントはシンボリックシステムだけでは達成できなかったレベルの認知継続性と目的を達成できます。

従来の AI とソフトウェアエージェントおよびエージェント AI の比較

次の表は、従来の AI、ソフトウェアエージェント、エージェント AI の詳細な比較を示しています。

特性	従来の AI	ソフトウェアエージェント	エージェント AI
例	スパムフィルター、イメージ分類子、レコメンデーションエンジン	チャットボット、タスクスケジューラ、モニタリングエージェント	AI アシスタント、自律型デベロッパーエージェント、マルチエージェント LLM オーケストレーション
実行モデル	バッチまたは同期	イベント駆動型またはスケジュール型	非同期、イベント駆動型、目標駆動型

特性	従来の AI	ソフトウェアエージェント	エージェント AI
自律性	制限あり。多くの場合、人間または外部のオーケストレーションが必要です	中、事前定義された境界内で独立して動作する	高い。適応戦略で独立して行動する
応答性	入力データに反応する	環境とイベントに反応する	リアクティブでプロアクティブ。アクションを予測して開始します
プロアクティブ	まれ	一部のシステムに存在する	コア属性。目標指向の動作を促進
コミュニケーション	最小、通常はスタンドアロンまたは API バインド	エージェント間またはエージェントヒューマンメッセージング	豊富なマルチエージェントとヒューman-in-the-loopインタラクション
意思決定	モデル推論のみ (分類、予測など)	シンボリック推論、ルールベースまたはスクリプトによる決定	コンテキスト、目標ベース、動的推論 (多くの場合 LLM 拡張)
委任されたインテント	いいえ。ユーザーが直接定義したタスクを実行します	部分的。範囲が限定されているユーザーまたはシステムに代わって動作します。	はい。多くの場合、サービス、ユーザー、またはシステム間で、委任された目標に沿って行動します
学習と適応	多くの場合、モデル中心 (ML トレーニングなど)	ときに適応的	埋め込み学習、メモリ、推論 (フィードバック、自己修正など)

特性	従来の AI	ソフトウェアエージェント	エージェント AI
エージェンシー	なし、人間用のツール	暗黙的または基本的	明示的。目的、目標、自己指示で動作する
コンテキスト認識	低、ステートレスまたはスナップショットベース	中程度、一部の状態追跡	高。メモリ、状況コンテキスト、環境モデルを使用する
インフラストラクチャロール	アプリケーションまたは分析パイプラインに埋め込む	ミドルウェアまたはサービスレイヤーコンポーネント	クラウド、サーバーレス、またはエッジシステムと統合された構成可能なエージェントメッシュ

要約は、以下のとおりです。

- 従来の AI はツール中心で機能的に狭くなっています。予測または分類に焦点を当てています。
- 従来のソフトウェアエージェントは、自律性と基本的な通信を導入しますが、多くの場合、ルールバインドまたは静的です。
- エージェント AI は、自律性、非同期性、エージェンシーを結び付けます。これにより、複雑なシステム内で推論、行動、適応できるインテリジェントな目標駆動型エンティティが可能になります。これにより、エージェント AI はクラウドネイティブで AI 主導の未来に最適です。

次のステップ

このガイドでは、従来のソフトウェアエージェントの生成 AI を活用した自律的でインテリジェントなシステムへの進化を表すエージェント AI の履歴と基盤について説明しました。初期のソフトウェアエージェントが事前定義されたルールとロジックに従って固定境界内でタスクを自動化する方法を説明し、エージェントがオープンエンド環境で動的に推論、学習、適応できるようにする大規模な言語モデルを組み込むことで、エージェント AI がこの基盤をどのように構築するかを説明しました。

このシリーズの次の出版物を確認することで、エージェント AI を詳しく調べることができます。

- でのエージェント AI の運用 AWSは、エージェント AI を独立した実験からエンタープライズ規模の価値を生み出すインフラストラクチャに変換する組織戦略を提供します。
- のエージェント AI パターンとワークフロー AWSでは、目標指向の AI エージェントの設計、構成、オーケストレーションに使用される基本的な設計図とモジュール構造について説明します。
- のエージェント AI フレームワーク、プロトコル、ツールは AWS、エージェント AI ソリューションを構築する際に考慮すべきソフトウェア基盤、ツールキット、プロトコルをカバーしています。
- でエージェント AI 用のサーバーレスアーキテクチャを構築する AWSと、最新の AI ワークロードの自然な基盤としてサーバーレスアーキテクチャについて説明し、 で AI ネイティブサーバーレスアーキテクチャを構築する方法について説明します AWS クラウド。
- でエージェント AI 用のマルチテナントアーキテクチャを構築するには AWS、ホスティングに関する考慮事項、デプロイモデル、コントロールプレーンなど、マルチテナント設定での AI エージェントの使用について説明します。

リソース

このガイドで説明されている概念の詳細については、以下のガイドと記事を参照してください。

AWS リファレンス

- Amazon Bedrock Agents
- Amazon Q Developer
- ストランドエージェント SDK

その他のリファレンス

- Hewitt、Carl、Peter Bishop、Richard Steiger。「人工知能のためのユニバーサルモジュラー ACTOR 形式」 第 3 回人工知能国際共同会議 (1973): 235-245。https://www.ijcai.org/Proceedings/73/Papers/027B.pdf
- Lesser、Victor R.、関連出版物 (全リストを参照):
  - Lesser、Victor R. と Daniel D. Corkill。「機能的に正確、協調的な分散システム」 Systems, Man, and Cybernetics 11, no. 1 (1981): 81-96。https://ieeexplore.ieee.org/abstract/document/4308581
  - Decker、Keith S.、Victor R. Lesser。「調整サービスにおける通信」 AAAI Workshop on Planning for Interagent Communication (1994)。https://www.researchgate.net/profile/Victor-Lesser/publication/2768884_Communication_in_the_Service_of_Coordination/links/00b7d51cc2a0750cb4000000/Communication-in-the-Service-of-Coordination.pdf
  - Durfee、Edmund H.、Victor R. Lesser、Daniel D. Corkill。「協調分散問題解決の傾向」 IEEE Transactions on knowledge and data Engineering (1989)。http://mas.cs.umass.edu/Documents/ieee-tkde89.pdf
  - Durfee、Edmund H.、V.R. Lesser、D.D. Corkill、「Distributed Artificial Intelligence」。分散型問題解決ネットワークでの通信による協力 (1987): 29-58。https://www.academia.edu/download/79885643/durf94_1.pdf
  - Lâasri、Brigitte、Hassan Lâasri、Susan Lander、Victor Lesser。「インテリジェントネゴシエートエージェントの汎用モデル」。International Journal of Cooperative Information Systems 01, no. 02 (1992): 291-317。https://doi.org/10.1142/S0218215792000210

  - Lander、Susan E.、Victor R. Lesser。「異種エージェント間の分散検索における交渉の役割を理解する」 IJCAI'93: 人工知能に関する第 13 回国際共同会議の議事録 (1993): 438-444。https://www.ijcai.org/Proceedings/93-1/Papers/062.pdf
  - Lander、Susan、Victor R. Lesser、Margaret E. Connell。「エキスパートエージェント協力のための競合解決戦略」 CKBS'90: ナレッジベースシステムの協力に関する国際作業会議 (1990 年 10 月): 183 ~ 200。https://doi.org/10.1007/978-1-4471-1831-2_10
  - Prasad、M.V. Nagendra、Victor Lesser、Susan E. Lander。「異種マルチエージェントシステムでの実験の学習」。IJCAI-95 Workshop on Adaptation and Learning in Multi-agent Systems (1995): 59-64。https://www.researchgate.net/publication/2784280_Learning_Experiments_in_a_Heterogeneous_Multi-agent_System
- Nwana、Hyacinth S。「ソフトウェアエージェント: 概要」。ナレッジエンジニアリングレビュー 11、いいえ。3 (1996 年 10 月/11 月): 205 ~ 244。https://teaching.shu.ac.uk/aces/rh1/elearning/multiagents/introduction/nwana.pdf
- Selfridge、Oliver G。「Pandemonium: A Paradigm for Learning」 思考プロセスのメカニズム: 国立物理学研究所 1 (1959): 511 ~ 529。https://aitopics.org/download/classics:504E1BAC
- Vaswani、Ashish、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Lukasz Kaiser、Illia"sukhin。「注意のみ」 第 31 回ニューラル情報処理システム会議 (NIPS) の議事録。ニューラル情報処理システム 30 の進歩 (2017): 5998-6008. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- "dridge, Michael and Nicholas R. Jennings。「インテリジェントエージェント: 理論と実践」 ナレッジエンジニアリングレビュー 10、いいえ。2 (1995 年 1 月): 115 ~ 152。https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/woodridge_intelligent_agents.pdf

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。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	—	2025 年 7 月 14 日

AWS 規範ガイダンスの用語集

以下は、 AWS 規範ガイダンスによって提供される戦略、ガイド、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。

数字

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します。
- 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 の Oracle 用 Amazon Relational Database Service (Amazon RDS) に移行します AWS クラウド。
- 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: カスタマーリレーションシップ管理 (CRM) システムを Salesforce.com に移行します。
- 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 の EC2 インスタンス上の Oracle に移行します AWS クラウド。
- 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を購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例: Microsoft Hyper-Vアプリケーションを に移行します 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。

A

ABAC

「属性ベースのアクセスコントロール」を参照してください。

抽象化されたサービス

「 マネージドサービス」を参照してください。

ACID

アトミック性、一貫性、分離性、耐久性を参照してください。

アクティブ - アクティブ移行

(双方向レプリケーションツールまたは二重書き込み操作を使用して) ソースデータベースとターゲットデータベースを同期させ、移行中に両方のデータベースが接続アプリケーションからのトランザクションを処理するデータベース移行方法。この方法では、1 回限りのカットオーバーの必要がなく、管理された小規模なバッチで移行できます。より柔軟ですが、アクティブ/パッシブ移行よりも多くの作業が必要です。

アクティブ - パッシブ移行

ソースデータベースとターゲットデータベースが同期されるデータベース移行方法。ただし、データがターゲットデータベースにレプリケートされている間、ソースデータベースのみが接続アプリケーションからのトランザクションを処理します。移行中、ターゲットデータベースはトランザクションを受け付けません。

集計関数

行のグループで動作し、グループの単一の戻り値を計算する SQL 関数。集計関数の例としては、 `SUM`や などがありま`MAX`す。

AI

「人工知能」を参照してください。

AIOps

「人工知能オペレーション」を参照してください。

匿名化

データセット内の個人情報を完全に削除するプロセス。匿名化は個人のプライバシー保護に役立ちます。匿名化されたデータは、もはや個人データとは見なされません。

アンチパターン

繰り返し起こる問題に対して頻繁に用いられる解決策で、その解決策が逆効果であったり、効果がなかったり、代替案よりも効果が低かったりするもの。

アプリケーションコントロール

マルウェアからシステムを保護するために、承認されたアプリケーションのみを使用できるようにするセキュリティアプローチ。

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。

人工知能 (AI)

コンピューティングテクノロジーを使用し、学習、問題の解決、パターンの認識など、通常は人間に関連づけられる認知機能の実行に特化したコンピュータサイエンスの分野。詳細については、「人工知能 (AI) とは何ですか?」を参照してください。

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。

非対称暗号化

暗号化用のパブリックキーと復号用のプライベートキーから成る 1 組のキーを使用した、暗号化のアルゴリズム。パブリックキーは復号には使用されないため共有しても問題ありませんが、プライベートキーの利用は厳しく制限する必要があります。

原子性、一貫性、分離性、耐久性 (ACID)

エラー、停電、その他の問題が発生した場合でも、データベースのデータ有効性と運用上の信頼性を保証する一連のソフトウェアプロパティ。

属性ベースのアクセス制御 (ABAC)

部署、役職、チーム名など、ユーザーの属性に基づいてアクセス許可をきめ細かく設定する方法。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「 の ABAC AWS」を参照してください。

信頼できるデータソース

最も信頼性のある情報源とされるデータのプライマリーバージョンを保存する場所。匿名化、編集、仮名化など、データを処理または変更する目的で、信頼できるデータソースから他の場所にデータをコピーすることができます。

アベイラビリティーゾーン

他のアベイラビリティーゾーンの障害から AWS リージョン 隔離され、同じリージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供する 内の別の場所。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 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です。 AWS CAF 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、運用という 6 つの重点分野にガイダンスを整理しています。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、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションのための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 ワークロード認定フレームワーク (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、作業見積もりを提供するツール。 AWS WQF は AWS Schema Conversion Tool (AWS SCT) に含まれています。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。

B

不正なボット

個人や組織を混乱させたり、損害を与えたりすることを意図したボット。

BCP

「事業継続計画」を参照してください。

動作グラフ

リソースの動作とインタラクションを経時的に示した、一元的なインタラクティブビュー。Amazon Detective の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントのData in a behavior graphを参照してください。

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。エンディアン性も参照してください。

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。

ブルームフィルター

要素がセットのメンバーであるかどうかをテストするために使用される、確率的でメモリ効率の高いデータ構造。

ブルー/グリーンデプロイ

2 つの異なる同一の環境を作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (青) で実行し、新しいアプリケーションバージョンを別の環境 (緑) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティややり取りをシミュレートするソフトウェアアプリケーション。インターネット上の情報のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボットと呼ばれる他のボットの中には、個人や組織を混乱させたり、損害を与えたりすることを意図したものもあります。

ボットネット

マルウェアに感染し、ボットハーダーまたはボットオペレーターとして知られる単一の当事者によって制御されているボットのネットワーク。ボットは、ボットとその影響をスケールするための最もよく知られているメカニズムです。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発したり、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

例外的な状況では、承認されたプロセスを通じて、ユーザーが AWS アカウント 通常アクセス許可を持たない にすばやくアクセスできるようにします。詳細については、 Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータ AWS を参照してください。

ブラウンフィールド戦略

環境の既存インフラストラクチャ。システムアーキテクチャにブラウンフィールド戦略を導入する場合、現在のシステムとインフラストラクチャの制約に基づいてアーキテクチャを設計します。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

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。

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。

C

CAF

AWS 「クラウド導入フレームワーク」を参照してください。

Canary デプロイ

エンドユーザーへのバージョンのスローリリースと増分リリース。確信できたら、新しいバージョンをデプロイし、現在のバージョン全体を置き換えます。

CCoE

「Cloud Center of Excellence」を参照してください。

CDC

「データキャプチャの変更」を参照してください。

変更データキャプチャ (CDC)

データソース (データベーステーブルなど) の変更を追跡し、その変更に関するメタデータを記録するプロセス。CDC は、ターゲットシステムでの変更を監査またはレプリケートして同期を維持するなど、さまざまな目的に使用できます。

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストします。AWS Fault Injection Service (AWS FIS) を使用して、 AWS ワークロードにストレスを与え、その応答を評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

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。

クライアント側の暗号化

ターゲットがデータ AWS のサービス を受信する前のローカルでのデータの暗号化。

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 投稿を参照してください。

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的にエッジコンピューティングテクノロジーに接続されています。

クラウド運用モデル

IT 組織において、1 つ以上のクラウド環境を構築、成熟、最適化するために使用される運用モデル。詳細については、「クラウド運用モデルの構築」 を参照してください。

導入のクラウドステージ

組織が に移行するときに通常実行する 4 つのフェーズ AWS クラウド:

- 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、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。AWS 移行戦略とどのように関連しているかについては、「移行準備ガイド」を参照してください。

CMDB

「設定管理データベース」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、 GitHubまたは が含まれますBitbucket Cloud。コードの各バージョンはブランチと呼ばれます。マイクロサービス

の構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

空である、または、かなり空きがある、もしくは、古いデータや無関係なデータが含まれているバッファキャッシュ。データベースインスタンスはメインメモリまたはディスクから読み取る必要があり、バッファキャッシュから読み取るよりも時間がかかるため、パフォーマンスに影響します。

コールドデータ

めったにアクセスされず、通常は過去のデータです。この種類のデータをクエリする場合、通常は低速なクエリでも問題ありません。このデータを低パフォーマンスで安価なストレージ階層またはクラスに移動すると、コストを削減することができます。

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオなどのビジュアル形式から情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI は CV 用の画像処理アルゴリズムを提供します。

設定ドリフト

ワークロードの場合、設定が想定状態から変化します。これにより、ワークロードが非準拠になる可能性があり、通常は段階的かつ意図的ではありません。

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。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。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンの単一のエンティティとしてデプロイすることも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性

の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

ネットワーク内のデータを重要度と機密性に基づいて識別、分類するプロセス。データに適した保護および保持のコントロールを判断する際に役立つため、あらゆるサイバーセキュリティのリスク管理戦略において重要な要素です。データ分類は、 AWS Well-Architected フレームワークのセキュリティの柱のコンポーネントです。詳細については、データ分類を参照してください。

データドリフト

実稼働データと ML モデルのトレーニングに使用されたデータとの間に有意な差異が生じたり、入力データが時間の経過と共に有意に変化したりすることです。データドリフトは、ML モデル予測の全体的な品質、精度、公平性を低下させる可能性があります。

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

一元管理とガバナンスを備えた分散型の分散型データ所有権を提供するアーキテクチャフレームワーク。

データ最小化

厳密に必要なデータのみを収集し、処理するという原則。でデータ最小化を実践 AWS クラウドすることで、プライバシーリスク、コスト、分析のカーボンフットプリントを削減できます。

データ境界

AWS 環境内の一連の予防ガードレール。信頼された ID のみが、期待されるネットワークから信頼されたリソースにアクセスできるようにします。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

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。

データ出所

データの生成、送信、保存の方法など、データのライフサイクル全体を通じてデータの出所と履歴を追跡するプロセス。

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには通常、大量の履歴データが含まれており、通常はクエリや分析に使用されます。

データベース定義言語 (DDL)

データベース内のテーブルやオブジェクトの構造を作成または変更するためのステートメントまたはコマンド。

データベース操作言語 (DML)

データベース内の情報を変更 (挿入、更新、削除) するためのステートメントまたはコマンド。

DDL

「データベース定義言語」を参照してください。

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。

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を採用するときは AWS、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加して、リソースの安全性を確保します。たとえば、多層防御アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS Organizations ドキュメントのAWS Organizationsで使用できるサービスを参照してください。

デプロイ

アプリケーション、新機能、コードの修正をターゲットの環境で利用できるようにするプロセス。デプロイでは、コードベースに変更を施した後、アプリケーションの環境でそのコードベースを構築して実行します。

開発環境

「環境」を参照してください。

検出管理

イベントが発生したときに、検出、ログ記録、警告を行うように設計されたセキュリティコントロール。これらのコントロールは副次的な防衛手段であり、実行中の予防的コントロールをすり抜けたセキュリティイベントをユーザーに警告します。詳細については、Implementing security controls on AWSのDetective controlsを参照してください。

開発バリューストリームマッピング (DVSM)

ソフトウェア開発ライフサイクルのスピードと品質に悪影響を及ぼす制約を特定し、優先順位を付けるために使用されるプロセス。DVSM は、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

建物、工場、産業機器、生産ラインなど、現実世界のシステムを仮想的に表現したものです。デジタルツインは、予知保全、リモートモニタリング、生産最適化をサポートします。

ディメンションテーブル

スタースキーマでは、ファクトテーブル内の量的データに関するデータ属性を含む小さなテーブル。ディメンションテーブル属性は通常、テキストフィールドまたはテキストのように動作する離散数値です。これらの属性は、クエリの制約、フィルタリング、結果セットのラベル付けに一般的に使用されます。

ディザスタ

ワークロードまたはシステムが、導入されている主要な場所でのビジネス目標の達成を妨げるイベント。これらのイベントは、自然災害、技術的障害、または意図しない設定ミスやマルウェア攻撃などの人間の行動の結果である場合があります。

ディザスタリカバリ (DR)

災害によるダウンタイムとデータ損失を最小限に抑えるために使用する戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ」を参照してください。

DML

「データベース操作言語」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。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件への準拠に影響するランディングゾーンの変更を検出したりできます。

DVSM

「開発値ストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

IoT ネットワークのエッジにあるスマートデバイスの計算能力を高めるテクノロジー。クラウドコンピューティングと比較すると、エッジコンピューティングは通信レイテンシーを短縮し、応答時間を短縮できます。

電子データ交換 (EDI)

組織間のビジネスドキュメントの自動交換。詳細については、「電子データ交換とは」を参照してください。

暗号化

人間が読み取り可能なプレーンテキストデータを暗号文に変換するコンピューティングプロセス。

暗号化キー

暗号化アルゴリズムが生成した、ランダム化されたビットからなる暗号文字列。キーの長さは決まっておらず、各キーは予測できないように、一意になるように設計されています。

エンディアン

コンピュータメモリにバイトが格納される順序。ビッグエンディアンシステムでは、最上位バイトが最初に格納されます。リトルエンディアンシステムでは、最下位バイトが最初に格納されます。

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS

Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (会計、MES、プロジェクト管理など) を自動化および管理するシステム。

エンベロープ暗号化

暗号化キーを、別の暗号化キーを使用して暗号化するプロセス。詳細については、 AWS Key Management Service (AWS KMS) ドキュメントの「エンベロープ暗号化」を参照してください。

環境

実行中のアプリケーションのインスタンス。クラウドコンピューティングにおける一般的な環境の種類は以下のとおりです。

- 開発環境 — アプリケーションのメンテナンスを担当するコアチームのみが使用できる、実行中のアプリケーションのインスタンス。開発環境は、上位の環境に昇格させる変更をテストするときに使用します。このタイプの環境は、テスト環境と呼ばれることもあります。
- 下位環境 — 初期ビルドやテストに使用される環境など、アプリケーションのすべての開発環境。
- 本番環境 — エンドユーザーがアクセスできる、実行中のアプリケーションのインスタンス。CI/CD パイプラインでは、本番環境が最後のデプロイ環境になります。
- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

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、ID とアクセスの管理、検出コントロール、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。

ERP

「エンタープライズリソース計画」を参照してください。

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。

F

ファクトテーブル

星スキーマの中央テーブル。事業運営に関する量的データを保存します。通常、ファクトテーブルには、メジャーを含む列とディメンションテーブルへの外部キーを含む列の 2 つのタイプの列が含まれます。

フェイルファスト

開発ライフサイクルを短縮するために頻繁で段階的なテストを使用する哲学。これはアジャイルアプローチの重要な部分です。

障害分離の境界

では AWS クラウド、障害の影響を制限し、ワークロードの耐障害性を高めるのに役立つアベイラビリティーゾーン AWS リージョン、コントロールプレーン、データプレーンなどの境界。詳細については、AWS 「障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

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、「 を使用した機械学習モデルの解釈可能性 AWS」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。

数ショットプロンプト

同様のタスクの実行を求める前に、タスクと必要な出力を示す少数の例を LLM に提供します。この手法は、プロンプトに埋め込まれた例 (ショット) からモデルが学習するコンテキスト内学習のアプリケーションです。少数ショットプロンプトは、特定のフォーマット、推論、またはドメインの知識を必要とするタスクに効果的です。「ゼロショットプロンプト」も参照してください。

FGAC

「きめ細かなアクセスコントロール」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

段階的なアプローチを使用する代わりに、変更データキャプチャによる継続的なデータレプリケーションを使用して、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

一般化およびラベル付けされていないデータの大規模なデータセットでトレーニングされている大規模な深層学習ニューラルネットワーク。FMs は、言語の理解、テキストと画像の生成、自然言語の会話など、さまざまな一般的なタスクを実行できます。詳細については、「基盤モデルとは」を参照してください。

G

生成 AI

大量のデータでトレーニングされ、シンプルなテキストプロンプトを使用してイメージ、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できる AI モデルのサブセット。詳細については、「生成 AI とは」を参照してください。

ジオブロッキング

地理的制限を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リストを使って指定します。詳細については、CloudFront ドキュメントのコンテンツの地理的ディストリビューションの制限を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローはレガシーと見なされ、トランクベースのワークフローはモダンで推奨されるアプローチです。

ゴールデンイメージ

そのシステムまたはソフトウェアの新しいインスタンスをデプロイするためのテンプレートとして使用されるシステムまたはソフトウェアのスナップショット。例えば、製造では、ゴールデンイメージを使用して複数のデバイスにソフトウェアをプロビジョニングし、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

新しい環境に既存のインフラストラクチャが存在しないこと。システムアーキテクチャにグリーンフィールド戦略を導入する場合、既存のインフラストラクチャ (別名ブラウンフィールド) との互換性の制約を受けることなく、あらゆる新しいテクノロジーを選択できます。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

ガードレール

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは AWS Config、、Amazon GuardDuty AWS Security Hub CSPM、、AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

H

HA

「高可用性」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。

ハイアベイラビリティ (HA)

課題や災害が発生した場合に、介入なしにワークロードを継続的に運用できること。HA システムは、自動的にフェイルオーバーし、一貫して高品質のパフォーマンスを提供し、パフォーマンスへの影響を最小限に抑えながらさまざまな負荷や障害を処理するように設計されています。

ヒストリアンのモダナイゼーション

製造業のニーズによりよく応えるために、オペレーションテクノロジー (OT) システムをモダナイズし、アップグレードするためのアプローチ。ヒストリアンは、工場内のさまざまなソースからデータを収集して保存するために使用されるデータベースの一種です。

ホールドアウトデータ

機械学習モデルのトレーニングに使用されるデータセットから保留される、ラベル付きの履歴データの一部。モデル予測をホールドアウトデータと比較することで、ホールドアウトデータを使用してモデルのパフォーマンスを評価できます。

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。

ホットデータ

リアルタイムデータや最近の翻訳データなど、頻繁にアクセスされるデータ。通常、このデータには高速なクエリ応答を提供する高性能なストレージ階層またはクラスが必要です。

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性が高いため、通常の DevOps のリリースワークフローからは外れた形で実施されます。

ハイパーケア期間

カットオーバー直後、移行したアプリケーションを移行チームがクラウドで管理、監視して問題に対処する期間。通常、この期間は 1～4 日です。ハイパーケア期間が終了すると、アプリケーションに対する責任は一般的に移行チームからクラウドオペレーションチームに移ります。

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

AWS クラウド 環境内のアクセス許可を定義する 1 つ以上の IAM プリンシパルにアタッチされたポリシー。

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。

IIoT

「産業用モノのインターネット」を参照してください。

イミュータブルインフラストラクチャ

既存のインフラストラクチャを更新、パッチ適用、または変更する代わりに、本番環境のワークロード用に新しいインフラストラクチャをデプロイするモデル。イミュータブルインフラストラクチャは、本質的にミュータブルインフラストラクチャよりも一貫性、信頼性、予測性が高くなります。詳細については、 AWS 「 Well-Architected フレームワーク」の「イミュータブルインフラストラクチャを使用したデプロイ」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリ

ケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

アプリケーションを 1 回ですべてカットオーバーするのではなく、小さい要素に分けて移行するカットオーバー戦略。例えば、最初は少数のマイクロサービスまたはユーザーのみを新しいシステムに移行する場合があります。すべてが正常に機能することを確認できたら、残りのマイクロサービスやユーザーを段階的に移行し、レガシーシステムを廃止できるようにします。この戦略により、大規模な移行に伴うリスクが軽減されます。

インダストリー 4.0

2016 年に Klaus Schwab によって導入された用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩によるビジネスプロセスのモダナイゼーションを指します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

アプリケーションのインフラストラクチャを一連の設定ファイルを使用してプロビジョニングし、管理するプロセス。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。

産業分野における IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「Building an industrial Internet of Things (IIoT) digital transformation strategy」を参照してください。

インスペクション VPC

AWS マルチアカウントアーキテクチャでは、VPC (同一または異なる 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

インターネットまたはローカル通信ネットワークを介して他のデバイスやシステムと通信する、センサーまたはプロセッサが組み込まれた接続済み物理オブジェクトのネットワーク。詳細については、「IoT とは」を参照してください。

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、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「モノのインターネット」を参照してください。

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。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

強制アクセス制御 (MAC) の実装で、ユーザーとデータ自体にそれぞれセキュリティラベル値が明示的に割り当てられます。ユーザーセキュリティラベルとデータセキュリティラベルが交差する部分によって、ユーザーに表示される行と列が決まります。

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント 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。

大規模言語モデル (LLM)

大量のデータに対して事前トレーニングされた深層学習 AI モデル。LLM は、質問への回答、ドキュメントの要約、テキストの他の言語への翻訳、文の完了など、複数のタスクを実行できます。詳細については、LLMs」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。

LBAC

「ラベルベースのアクセスコントロール」を参照してください。

最小特権

タスクの実行には必要最低限の権限を付与するという、セキュリティのベストプラクティス。詳細については、IAM ドキュメントの最小特権アクセス許可を適用するを参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。エンディアン性も参照してください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

パターン認識と学習にアルゴリズムと手法を使用する人工知能の一種。ML は、モノのインターネット (IoT) データなどの記録されたデータを分析して学習し、パターンに基づく統計モデルを生成します。詳細については、「機械学習」を参照してください。

メインブランチ

「ブランチ」を参照してください。

マルウェア

コンピュータのセキュリティまたはプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスにつながる可能性があります。マルウェアの例としては、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス はインフラストラクチャレイヤー、オペレーティングシステム、プラットフォーム AWS を運用し、エンドポイントにアクセスしてデータを保存および取得します。Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB は、マネージドサービスの例です。これらは抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するためのソフトウェアシステムで、原材料を工場の完成製品に変換します。

MAP

「移行促進プログラム」を参照してください。

メカニズム

ツールを作成し、ツールの導入を推進し、調整を行うために結果を検査する完全なプロセス。メカニズムは、動作中にそれ自体を強化して改善するサイクルです。詳細については、 AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

組織の一部である管理アカウント AWS アカウント 以外のすべて AWS Organizations。 アカウントが 組織のメンバーになることができるのは、一度に 1 つのみです。

MES

「製造実行システム」を参照してください。

メッセージキューイングテレメトリトランスポート (MQTT)

リソースに制約のある IoT デバイス用の、パブリッシュ/サブスクライブパターンに基づく軽量 machine-to-machine (M2M) 通信プロトコル。

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たマイクロサービスの統合」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、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

組織がクラウドに移行する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。

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、およびスプリントで作業する DevOps プロフェッショナルが含まれます。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、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

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は、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

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: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

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 AWS クラウド。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 AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、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。

移行戦略

ワークロードを に移行するために使用するアプローチ AWS クラウド。詳細については、この用語集の「7 Rs エントリ」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。

ML

???「機械学習」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、「」の「アプリケーションをモダナイズするための戦略 AWS クラウド」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、『』の「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 クラウド」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。

MPA

「移行ポートフォリオ評価」を参照してください。

MQTT

「Message Queuing Telemetry Transport」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。

ミュータブルインフラストラクチャ

本番ワークロードの既存のインフラストラクチャを更新および変更するモデル。Well-Architected AWS フレームワークでは、一貫性、信頼性、予測可能性を向上させるために、イミュータブルインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセスコントロール」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

移行プロセス中にソースワークロードを停止させる移行方法。この方法はダウンタイムが長くなるため、通常は重要ではない小規模なワークロードに使用されます。

OI

「 オペレーションの統合」を参照してください。

OLA

「運用レベルの契約」を参照してください。

オンライン移行

ソースワークロードをオフラインにせずにターゲットシステムにコピーする移行方法。ワークロードに接続されているアプリケーションは、移行中も動作し続けることができます。この方法はダウンタイムがゼロから最小限で済むため、通常は重要な本番稼働環境のワークロードに使用されます。

OPC-UA

「Open Process Communications - Unified Architecture」を参照してください。

オープンプロセス通信 - 統合アーキテクチャ (OPC-UA)

産業用オートメーション用のmachine-to-machine (M2M) 通信プロトコル。OPC-UA は、データの暗号化、認証、認可スキームを備えた相互運用性標準を提供します。

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。

運用準備状況レビュー (ORR)

インシデントや潜在的な障害の理解、評価、防止、または範囲の縮小に役立つ質問とそれに関連するベストプラクティスのチェックリスト。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携するハードウェアおよびソフトウェアシステム。製造では、OT と情報技術 (IT) システムの統合が、Industry 4.0 変換の主な焦点です。

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。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail をログに記録する、 によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの組織の証跡の作成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化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイド を参照してください。

オリジンアクセスコントロール (OAC)

Amazon Simple Storage Service (Amazon S3) コンテンツを保護するための、CloudFront のアクセス制限の強化オプション。OAC は AWS リージョン、すべての S3 バケット、 AWS KMS (SSE-KMS) によるサーバー側の暗号化、S3 バケットへの動的 PUT および DELETEリクエストをサポートします。

オリジンアクセスアイデンティティ (OAI)

CloudFront の、Amazon S3 コンテンツを保護するためのアクセス制限オプション。OAI を使用すると、CloudFront が、Amazon S3 に認証可能なプリンシパルを作成します。認証されたプリンシパルは、S3 バケット内のコンテンツに、特定の CloudFront ディストリビューションを介してのみアクセスできます。OACも併せて参照してください。OAC では、より詳細な、強化されたアクセスコントロールが可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション内から開始されたネットワーク接続を処理する VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

P

アクセス許可の境界

ユーザーまたはロールが使用できるアクセス許可の上限を設定する、IAM プリンシパルにアタッチされる IAM 管理ポリシー。詳細については、IAM ドキュメントのアクセス許可の境界を参照してください。

個人を特定できる情報 (PII)

直接閲覧した場合、または他の関連データと組み合わせた場合に、個人の身元を合理的に推測するために使用できる情報。PII の例には、氏名、住所、連絡先情報などがあります。

PII

個人を特定できる情報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。

PLC

「プログラム可能なロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

アクセス許可を定義 (アイデンティティベースのポリシーを参照)、アクセス条件を指定 (リソースベースのポリシーを参照)、または の組織内のすべてのアカウントに対する最大アクセス許可を定義 AWS Organizations (サービスコントロールポリシーを参照) できるオブジェクト。

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。

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。

述語

`true` または を返すクエリ条件。一般的に`false`は `WHERE`句にあります。

述語プッシュダウン

転送前にクエリ内のデータをフィルタリングするデータベースクエリ最適化手法。これにより、リレーショナルデータベースから取得して処理する必要があるデータの量が減少し、クエリのパフォーマンスが向上します。

予防的コントロール

イベントの発生を防ぐように設計されたセキュリティコントロール。このコントロールは、ネットワークへの不正アクセスや好ましくない変更を防ぐ最前線の防御です。詳細については、Implementing security controls on AWSのPreventative controlsを参照してください。

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM AWS アカウントロール、または ユーザーのルートユーザーです。詳細については、IAM ドキュメントのロールに関する用語と概念内にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通じてプライバシーを考慮するシステムエンジニアリングアプローチ。

プライベートホストゾーン

1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナ。詳細については、Route 53 ドキュメントの「プライベートホストゾーンの使用」を参照してください。

プロアクティブコントロール

非準拠リソースのデプロイを防ぐように設計されたセキュリティコントロール。これらのコントロールは、プロビジョニング前にリソースをスキャンします。リソースがコントロールに準拠していない場合、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「 コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

設計、開発、発売から成長と成熟、拒否と削除まで、ライフサイクル全体にわたる製品のデータとプロセスの管理。

本番環境

「環境」を参照してください。

プログラム可能なロジックコントローラー (PLC)

製造では、マシンをモニタリングし、製造プロセスを自動化する、信頼性の高い適応可能なコンピュータです。

プロンプトの連鎖

1 つの LLM プロンプトの出力を次のプロンプトの入力として使用して、より良いレスポンスを生成します。この手法は、複雑なタスクをサブタスクに分割したり、事前レスポンスを繰り返し改善または拡張したりするために使用されます。これにより、モデルのレスポンスの精度と関連性が向上し、より詳細でパーソナライズされた結果が得られます。

仮名化

データセット内の個人識別子をプレースホルダー値に置き換えるプロセス。仮名化は個人のプライバシー保護に役立ちます。仮名化されたデータは、依然として個人データとみなされます。

パブリッシュ/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。スケーラビリティと応答性を向上させます。たとえば、マイクロサービスベースの MES では、マイクロサービスは他のマイクロサー

ビスがサブスクライブできるチャネルにイベントメッセージを発行できます。システムは、公開サービスを変更せずに新しいマイクロサービスを追加できます。

Q

クエリプラン

SQL リレーショナルデータベースシステムのデータにアクセスするために使用される手順などの一連のステップ。

クエリプランのリグレッション

データベースサービスのオプティマイザーが、データベース環境に特定の変更が加えられる前に選択されたプランよりも最適性の低いプランを選択すること。これは、統計、制限事項、環境設定、クエリパラメータのバインディングの変更、およびデータベースエンジンの更新などが原因である可能性があります。

R

RACI マトリックス

責任、説明責任、相談、情報 (RACI) を参照してください。

RAG

「取得拡張生成」を参照してください。

ランサムウェア

決済が完了するまでコンピュータシステムまたはデータへのアクセスをブロックするように設計された、悪意のあるソフトウェア。

RASCI マトリックス

責任、説明責任、相談、情報 (RACI) を参照してください。

RCAC

「行と列のアクセスコントロール」を参照してください。

リードレプリカ

読み取り専用に使用されるデータベースのコピー。クエリをリードレプリカにルーティングして、プライマリデータベースへの負荷を軽減できます。

再設計

「7 Rs」を参照してください。

目標復旧時点 (RPO)

最後のデータリカバリポイントからの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の とは独立しています。詳細については、「アカウントで使用できる を指定する AWS リージョン」を参照してください。

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。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

プラットフォーム変更

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または回復するアプリケーションの機能。高可用性とディザスタリカバリは、 で回復性を計画する際の一般的な考慮事項です AWS クラウド。詳細については、AWS クラウド「レジリエンス」を参照してください。

リソースベースのポリシー

Amazon S3 バケット、エンドポイント、暗号化キーなどのリソースにアタッチされたポリシー。このタイプのポリシーは、アクセスが許可されているプリンシパル、サポートされているアクション、その他の満たすべき条件を指定します。

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

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任 (A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートを含めると、そのマトリックスは RASCI マトリックスと呼ばれ、サポートを除外すると RACI マトリックスと呼ばれます。

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、Implementing security controls on AWSのResponsive controlsを参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

取得拡張生成 (RAG)

LLM がレスポンスを生成する前にトレーニングデータソースの外部にある信頼できるデータソースを参照する生成 AI テクノロジー。例えば、RAG モデルは組織のナレッジベースまたはカスタムデータのセマンティック検索を実行する場合があります。詳細については、「RAG とは」を参照してください。

ローテーション

攻撃者が認証情報にアクセスすることをより困難にするために、シークレットを定期的に更新するプロセス。

行と列のアクセス制御 (RCAC)

アクセスルールが定義された、基本的で柔軟な SQL 表現の使用。RCAC は行権限と列マスクで構成されています。

RPO

「目標復旧時点」を参照してください。

RTO

目標復旧時間を参照してください。

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。

S

SAML 2.0

多くの ID プロバイダー (IdP) が使用しているオープンスタンダード。この機能を使用すると、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 にログイン AWS マネジメントコンソール したり AWS 、API オペレーションを呼び出すことができます。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントのSAML 2.0 ベースのフェデレーションについてを参照してください。

SCADA

「監視コントロールとデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存する AWS Secrets Managerパスワードやユーザー認証情報などの機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値は、バイナリ、1 つの文字列、または複数の文字列にすることができます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの内容」を参照してください。

設計によるセキュリティ

開発プロセス全体でセキュリティを考慮するシステムエンジニアリングアプローチ。

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、予防的、検出的、応答的、プロアクティブの 4 つの主なタイプがあります。

セキュリティ強化

アタックサーフェスを狭めて攻撃への耐性を高めるプロセス。このプロセスには、不要になったリソースの削除、最小特権を付与するセキュリティのベストプラクティスの実装、設定ファイル内の不要な機能の無効化、といったアクションが含まれています。

Security Information and Event Management (SIEM) システム

セキュリティ情報管理 (SIM) とセキュリティイベント管理 (SEM) のシステムを組み合わせたツールとサービス。SIEM システムは、サーバー、ネットワーク、デバイス、その他ソースからデータを収集、モニタリング、分析して、脅威やセキュリティ違反を検出し、アラートを発信します。

セキュリティレスポンスの自動化

セキュリティイベントに自動的に応答または修復するように設計された、事前定義されたプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例としては、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータの、それ AWS のサービス を受け取る による暗号化。

サービスコントロールポリシー (SCP)

AWS Organizationsの組織内の、すべてのアカウントのアクセス許可を一元的に管理するポリシー。SCP は、管理者がユーザーまたはロールに委任するアクションに、ガードレールを定義したり、アクションの制限を設定したりします。SCP は、許可リストまたは拒否リストとして、許可または禁止するサービスやアクションを指定する際に使用できます。詳細については、 AWS Organizations ドキュメントの「サービスコントロールポリシー」を参照してください。

サービスエンドポイント

のエントリポイントの URL AWS のサービス。ターゲットサービスにプログラムで接続するには、エンドポイントを使用します。詳細については、AWS 全般のリファレンスの「AWS のサービス エンドポイント」を参照してください。

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。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットなど、サービスのパフォーマンス側面の測定。

サービスレベルの目標 (SLO)

サービスレベルのインジケータによって測定される、サービスの状態を表すターゲットメトリクス。

責任共有モデル

クラウドのセキュリティとコンプライアンス AWS について と共有する責任を説明するモデル。AWS はクラウドのセキュリティを担当しますが、お客様はクラウドのセキュリティを担当します。詳細については、責任共有モデルを参照してください。

SIEM

セキュリティ情報とイベント管理システムを参照してください。

単一障害点 (SPOF)

システムを中断する可能性のあるアプリケーションの 1 つの重要なコンポーネントの障害。

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

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、『』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」を参照してください。

SPOF

単一障害点を参照してください。

スタースキーマ

1 つの大きなファクトテーブルを使用してトランザクションデータまたは測定データを保存し、1 つ以上の小さなディメンションテーブルを使用してデータ属性を保存するデータベース組織構造。この構造は、データウェアハウスまたはビジネスインテリジェンスの目的で使用するように設計されています。

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。

サブネット

VPC 内の IP アドレスの範囲。サブネットは、1 つのアベイラビリティーゾーンに存在する必要があります。

監視制御とデータ収集 (SCADA)

製造では、ハードウェアとソフトウェアを使用して物理アセットと本番稼働をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして潜在的な問題を検出したり、パフォーマンスをモニタリングしたりする方法でシステムをテストします。Amazon CloudWatch Synthetics を使用して、これらのテストを作成できます。

システムプロンプト

LLM にコンテキスト、指示、またはガイドラインを提供して動作を指示する手法。システムプロンプトは、コンテキストを設定し、ユーザーとのやり取りのルールを確立するのに役立ちます。

T

tags

AWS リソースを整理するためのメタデータとして機能するキーと値のペア。タグは、 リソースの管理、識別、整理、検索、フィルタリングに役立ちます。詳細については、「AWS リソースのタグ付け」を参照してください。

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。

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。

テスト環境

「環境」を参照してください。

トレーニング

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。

トランジットゲートウェイ

VPC とオンプレミスネットワークを相互接続するために使用できる、ネットワークの中継ハブ。詳細については、 AWS Transit Gateway ドキュメントの「トランジットゲートウェイとは」を参照してください。

トランクベースのワークフロー

デベロッパーが機能ブランチで機能をローカルにビルドしてテストし、その変更をメインブランチにマージするアプローチ。メインブランチはその後、開発環境、本番前環境、本番環境に合わせて順次構築されます。

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定したサービスにアクセス許可を付与します。信頼されたサービスは、サービスにリンクされたロールを必要なときに各アカウントに作成し、ユーザーに代わって管理タスクを実行します。詳細については、 ドキュメントの「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

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。

ツーピザチーム

2 枚のピザで養うことができるくらい小さな DevOps 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。

U

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、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。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。

未分化なタスク

ヘビーリフティングとも呼ばれ、アプリケーションの作成と運用には必要だが、エンドユーザーに直接的な価値をもたらさなかったり、競争上の優位性をもたらしたりしない作業です。未分化なタスクの例としては、調達、メンテナンス、キャパシティプランニングなどがあります。

上位環境

???「環境」を参照してください。

V

バキューミング

ストレージを再利用してパフォーマンスを向上させるために、増分更新後にクリーンアップを行うデータベースのメンテナンス操作。

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

プライベート IP アドレスを使用してトラフィックをルーティングできる、2 つの VPC 間の接続。詳細については、Amazon VPC ドキュメントの「VPC ピア機能とは」を参照してください。

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

頻繁にアクセスされる最新の関連データを含むバッファキャッシュ。データベースインスタンスはバッファキャッシュから、メインメモリまたはディスクからよりも短い時間で読み取りを行うことができます。

ウォームデータ

アクセス頻度の低いデータ。この種類のデータをクエリする場合、通常は適度に遅いクエリでも問題ありません。

ウィンドウ関数

現在のレコードに関連する行のグループに対して計算を実行する SQL 関数。ウィンドウ関数は、移動平均の計算や、現在の行の相対位置に基づく行の値へのアクセスなどのタスクの処理に役立ちます。

ワークロード

ビジネス価値をもたらすリソースとコード (顧客向けアプリケーションやバックエンドプロセスなど) の総称。

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。

WORM

「書き込み 1 回」、「読み取り多数」を参照してください。

WQF

AWS 「ワークロード認定フレームワーク」を参照してください。

Write Once, Read Many (WORM)

データを 1 回書き込み、データの削除や変更を防ぐストレージモデル。承認されたユーザーは、必要な回数だけデータを読み取ることができますが、変更することはできません。このデータストレージインフラストラクチャはイミュータブルと見なされます。

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を利用する攻撃、通常はマルウェア。

ゼロデイ脆弱性

実稼働システムにおける未解決の欠陥または脆弱性。脅威アクターは、このような脆弱性を利用してシステムを攻撃する可能性があります。開発者は、よく攻撃の結果で脆弱性に気付きます。

ゼロショットプロンプト

LLM にタスクを実行する手順を提供しますが、タスクのガイドに役立つ例 (ショット) はありません。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

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。

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。