aws

でのモデルコンテキストプロトコル戦略 AWS

AWS 規範ガイダンス

AWS 規範ガイダンス: でのモデルコンテキストプロトコル戦略 AWS

Table of Contents

でのモデルコンテキストプロトコル戦略 AWS

Amazon Web Services (寄稿者)

2026 年 3 月 (ドキュメント履歴)

このガイドは、エージェント AI ジャーニーをサポートするために、組織全体でモデルコンテキストプロトコル (MCP) 戦略を開発および実装するのに役立ちます。エージェントと言語モデルがビジネスオペレーションの中心になるにつれて、エージェントソリューションを成功させるには MCP 戦略を確立することが重要です。

このガイドでは、MCP 戦略を構築するための 3 つの基本的な柱である MCP ツール設計、MCP サーバーホスティング、および MCP ガバナンスについて説明します。これらの相互接続されたコンポーネントに対処することで、組織は AI 実装全体でモデルコンテキストを管理するためのスケーラブルで安全で効果的なシステムを作成できます。このガイダンスは、最初の実験からフルスケールの本番デプロイまで、組織の AI ジャーニーのあらゆる段階で、組織に実用的なインサイトと戦略的ガイダンスを提供します。これにより、特定のニーズや目的に合ったカスタマイズされた MCP ソリューションを開発できます。

これらのベストプラクティスは、企業規模で MCP をデプロイする組織の実際の実装、現在の MCP 仕様標準の分析、本番環境のカスタム大規模言語モデル (LLM) アプリケーションから学んだ教訓に基づいています。

AI システムは、さまざまなユースケースでますます洗練され、堅牢な LLMs を使用します。LLMs、自然言語の理解、人間のようなレスポンスの生成、複雑な情報に対する推論に優れています。ただし、LLMs を会話型インターフェイスから複雑なタスクを自律的に実行できるシステムに変換するために、組織は エージェント AI アーキテクチャ、環境を認識できる AI システム、目標の理由、自律的な意思決定を行い、複数のステップにわたって調整し、ユーザーに代わって目標を達成するためのアクションを実行しています。このエージェント的アプローチは、組織が自然言語を通じてユーザーのインテントを理解し、複数のデータソースとツール間で自律的に調整し、従来のリクエスト/レスポンスパターンでは不可能な規模でパーソナライズされたエクスペリエンスを提供できる AI システムを構築するのに役立ちます。これらのエージェントをより有能にするには、組織は既存のツールとデータへのアクセスを提供して、エージェントのコンテキスト理解を強化し、ユーザーに代わって行動できるようにする必要があります。

MCP は AI ツール統合の標準化プロトコルを提供し、エージェントと外部リソース間の一貫した通信を可能にします。MCP 自体が通信標準を定義していますが、効果的に実装するには、スケーラブル、安全、保守可能なソリューションを実現するために、アーキテクチャパターン、セキュリティモデル、運用プラクティス、パフォーマンス最適化戦略を慎重に検討する必要があります。

このガイドでは、エンタープライズ MCP デプロイから学んだ教訓を合成し、 AWS Well-Architected フレームワークに沿った実用的な推奨事項を提供します。独自の MCP ソリューションの構築に不可欠な MCP ツール設計、MCP サーバーホスティング、および MCP ガバナンスの戦略について説明します。このガイドの推奨事項は、 AWS Well-Architected フレームワークの次の 5 つの柱にマッピングされています。

- セキュリティ – トークンの分離、スコープダウンされた認証情報、個別の読み取り/書き込み認可
- 運用上の優秀性 – ツール選択精度メトリクス、回帰テスト用のゴールデンデータセット
- 信頼性 – ユーザーごとおよびツールごとのレート制限、負荷削減
- パフォーマンス効率 – ワークフロースコープのツール、ツールフィルタリング、コンテキストウィンドウの使用量を減らすセマンティック検索
- コスト最適化 – チーム間で再利用可能な MCP サーバー、ツールフィルタリングによるリクエストごとのトークンコストを削減

対象者

このガイドは、組織にエージェント AI ソリューションを実装し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。このガイドの概念を理解するには、LLMs の仕組みを理解し、MCP、ツール、プロンプトエンジニアリングに関する基本的な知識を持っている必要があります。

目的

本番環境に対応した エージェント AI システムを構築するということは、ガバナンス、最適化、セキュリティをまとめて解決し、組織のポリシーをサポートすることを意味します。以下に、このガイドがこれらの目的にどのように対処するかを説明します。

- ガバナンス – 一元化されたガバナンスがない場合、どのエージェントがどのデータにアクセスしたか、どのアクセス許可を持つか、いつアクセスしたかなど、AI ワークロードに関する監査の質問に回答することはできません。また、バージョニングを適用することもできません。このガイドの MCP ホスティング戦略セクションでは、体系的な適用がないために、既知の脆弱性を持つ古いローカル MCP サーバーをユーザーが実行する方法について説明します。

  規制された業界にとって、ガバナンスは重要です。監査者は、すべてのエージェントでポリシーの適用とツールの使用状況の追跡を 1 つのペインから確認したいと考えています。MCP ガバナンスはこれを提供します。

このガイドの推奨事項に従うことで、ピアレビュー済みのベンチマークでタスクの精度を 28 ～32% 向上させることができます。詳細については、「MARCO: Multi-Agent Real-time Chat Orchestration」(ACL Anthology ウェブサイト) を参照してください。ガバナンスはコンプライアンスだけでなく、エージェント AI システムのパフォーマンスも向上します。

- 最適化 – チームは同じ統合を複数回構築する場合があります。例えば、5 つの異なるチームが AI アプリケーションがデータベースと通信するための独自のデータベースクエリスクリプトを記述すると、開発コストの 5 倍、維持するバグリストの 5 セットになります。MCP を使用すると、一度構築し、エンジニアリングコミュニティ全体で共有できます。エージェント数が増えるにつれて節約が複雑になります。

  また、ほとんどのチームが最初に気付かないリクエストごとのコスト問題もあります。すべてのツール定義はコンテキストウィンドウトークンを消費します。20 のツールでは、ユーザーの問い合わせとともに、説明のみに呼び出しごとに 5,000 ～ 10,000 トークンを費やしています。これにより、モデルが利用可能なツールのリストから適切なツールを選択するのに苦労するため、レイテンシーと LLM 推論コストが増加し、精度が低下します。

  構造化ツールラッパーを使用するエージェントは、APIs に直接アクセスするエージェントよりもデータベースタスクの精度が約 3 倍高くなります (詳細については、「Middleware for LLMs: Tools Are Instrumental for Language Agents in Complex Environments」を参照してください)。AI モデルにツールを設計して提示する方法は重要です。このガイドでは、ツールに明確なスキーマを付与し、未加工のエンドポイントではなく実際のワークフローにスコープし、コンテキストウィンドウで情報を制限することを推奨しています。このガイドの MCP ツール設計戦略セクションでは、これらの側面について詳しく説明します。
- セキュリティとコンプライアンス – クリーンアップステップをハルシネーションし、本番稼働用データベースを削除しようとするエージェント AI システムを想像してみてください。エージェントがユーザーの完全な管理者認証情報を継承した場合、削除が実行される可能性があります。読み取りアクセスと作成アクセスのみを許可するトークン分離とスコープダウンされた認証情報では、安全に失敗します。

  規制されたワークフローは、これをさらにシャープにします。このガイドでは、例 (患者データを処理する前に HIPAA 検証と個人を特定できる情報の匿名化を必要とする医療パイプライン) を示します。このようなロジックを MCP ツールに埋め込むと、コンプライアンスは毎回決定的に行われます。

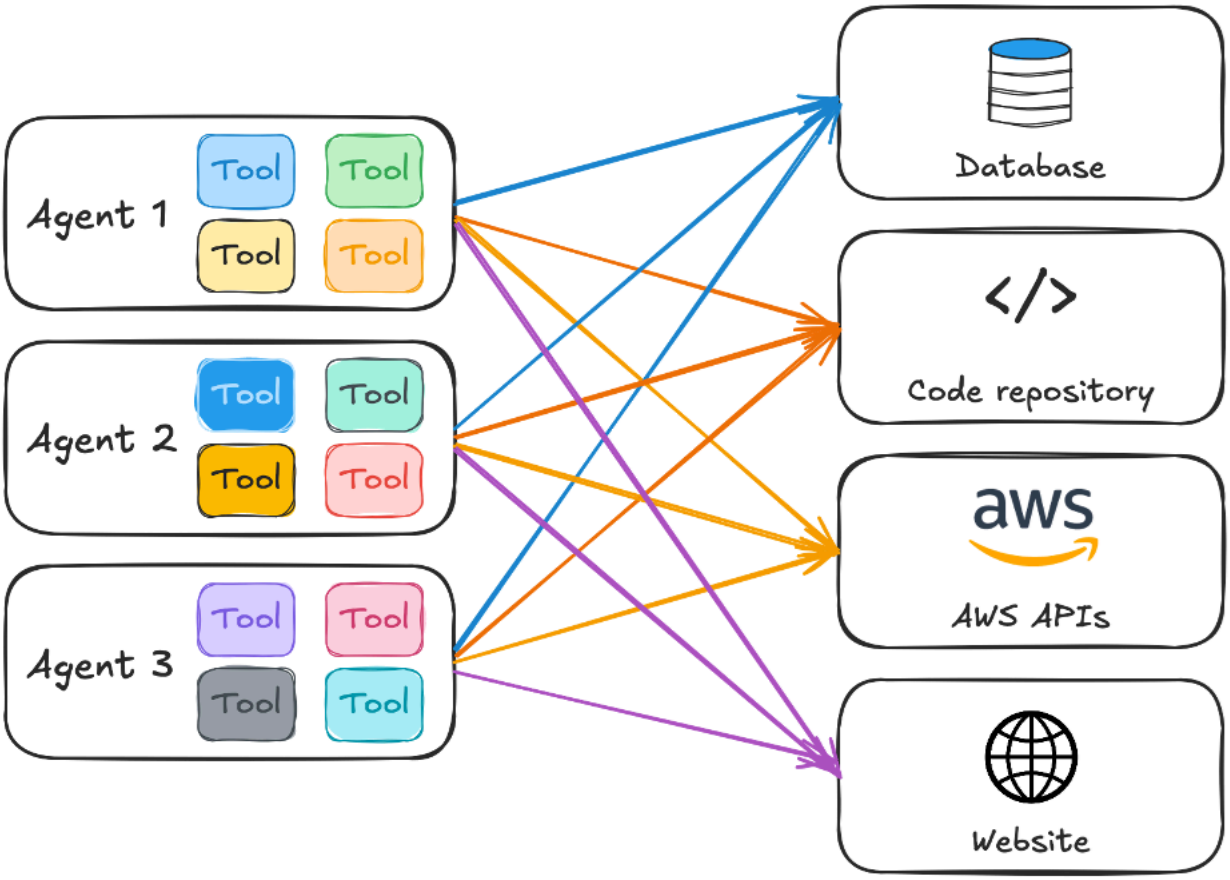
MCP とは

LLMs、トレーニングデータに基づいてプロンプトへの回答を予測することによって機能します。つまり、LLM は、すでに表示したデータとイベントに関する回答のみを提供できます。取得拡張生成 (RAG) やナレッジベースなどの方法を使用すると、コンテキストデータを含めることができます。ただし、明日の天気予報がどのようなものになるか、データベースに何人の顧客がいるかを LLM に尋ねると、これらは LLM の事前トレーニング済みの知識の範囲外であるため、ハルシネーションになるか、回答を提供できない可能性があります。これらの種類の質問に回答できるようにするには、エージェントは LLM のネイティブコンテキスト外の外部機能、データ、APIs にアクセスする必要があります。

ツールについて

ツールを使用して、LLM に追加のシステムやコンテキストへのアクセスを許可できます。 ツールは、明確な目標を達成するために LLM に提供される関数です。ツールは、API の呼び出し、データベースのクエリ、計算ツールオペレーションの実行、コードサンドボックスの操作、ウェブ検索の実行、さらには別の AI システムやagent-as-a-tool呼び出すこともできます。各ツールには、ツールの動作、使用タイミング、受け入れるパラメータを LLM に伝える説明を含める必要があります。これにより、LLM はユーザーの入力に基づいて呼び出すツールまたはツールの組み合わせについて微妙な決定を行うことができます。LLM は、エージェントが使用できるツールについて説明され、ツールを呼び出すようにエージェントに指示するレスポンスを生成できます。たとえば、データベース内の顧客数を LLM に尋ねると、LLM はエージェントにレスポンスを送り返し、特定の入力パラメータを使用して`query_database`ツールを実行するようにリクエストします。LLM は、呼び出すツールとツール呼び出しの入力を決定します。エージェントはツールを実行し、自然言語入力を構文的に正しい関数呼び出しに変換してクエリを実行します。エージェントは LLM からの命令に基づいてツールを呼び出し、その結果は LLM に返されます。これにより、LLM はテキストベースの入力を推論し、ジョブに適したツールを選択できます。

次の図は、各エージェントが各ターゲットに対して独自のツールセットを管理する方法を示しています。



スケーリングツールへのアクセスは、エージェント AI ソリューションにとって課題となる可能性があります。

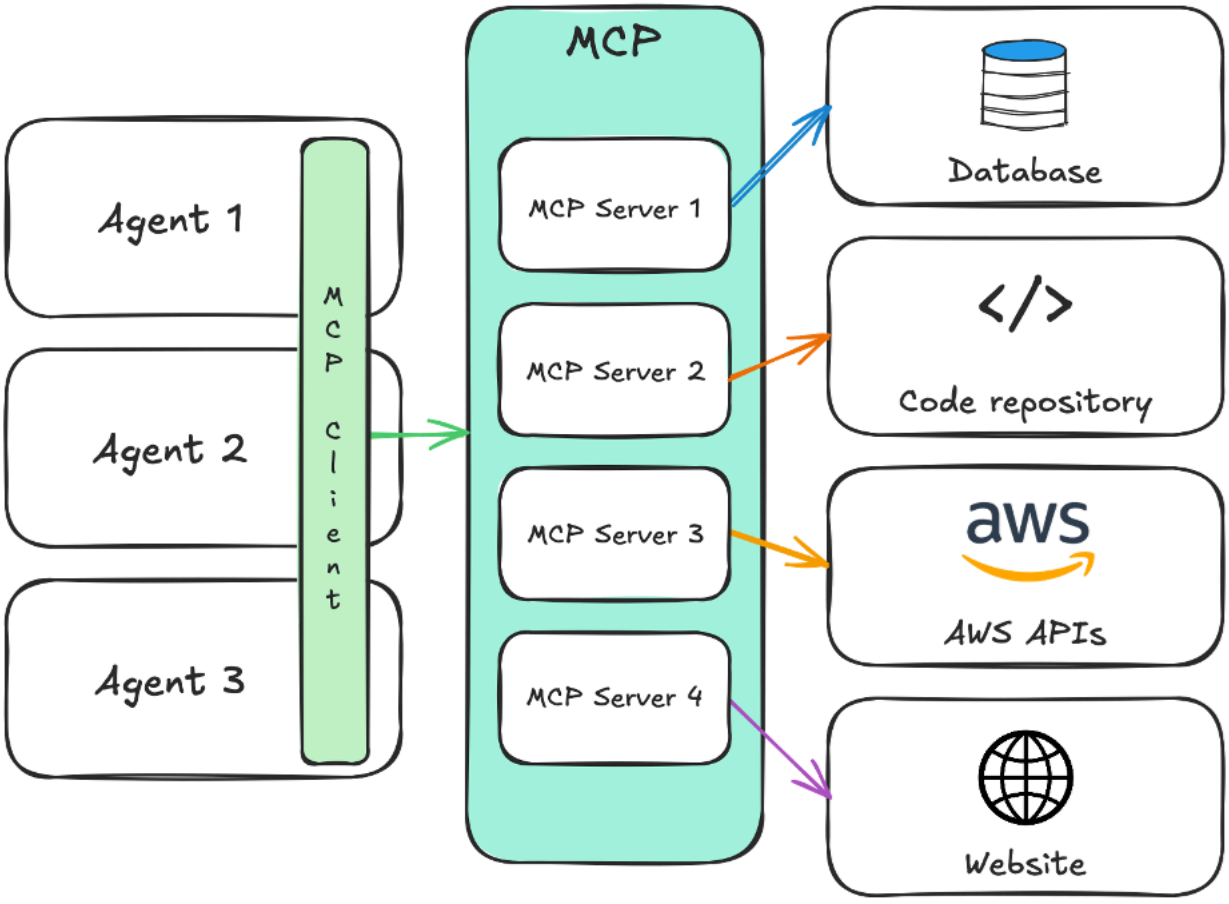
- すべての開発者が同じ外部機能用に独自のツールを作成している場合、これらの外部機能を操作するための重複した労力と標準化されていない方法が多数あります。これにより、エージェント間で実装に一貫性がなくなります。ライブラリで標準ツールを開発して配布することで、この問題を解決できますが、一元化されたガバナンスはありません。これにより、セキュリティポリシーの適用、ツールの使用状況の追跡、チーム間のバージョニングの管理、組織の標準への準拠の確保が困難になります。さらに、ツールをエージェントに直接埋め込む場合は、新しいツールが作成されるか、既存のツールが更新されるたびにエージェントを再デプロイする必要があります。
- LLM にツールを提供すると、そのコンテキストウィンドウが消費されます。コンテキストウィンドウは、モデルが一度に考慮できるトークンの数 (LLMs が処理するテキストの単位。通常は単語、単語の一部、または句読点を表します) です。LLMsにはコンテキストウィンドウの制限があります。ツールとそのドキュメントは、システムプロンプトとユーザープロンプトとともに、その有限コンテキストウィンドウを使用します。コンテキストウィンドウがいっぱいになると、LLMs、関連情報の識別が困難になり、処理の複雑さが増し、推論の容量が減るため、パフォーマンスが低下する可能性があります。ツール定義、システムプロンプト、会話履歴が各

LLM 呼び出しで提供されるため、制限されたコンテキストウィンドウスペースで競合すると、チャレンジが複雑になります。

したがって、ツールの数と文書化方法は、応答時間や精度など、LLM のパフォーマンスに直接影響します。

MCP は、エージェントを外部機能に接続するためのユニバーサルスタンダードを確立します。一般的に「AI アプリケーション用 USB-C」と呼ばれます。MCP サーバーは、ツールをエージェントに直接登録する代わりに、JSON-RPC 2.0 で検出および呼び出されるホスティングツールの仲介として機能します。MCP では、エージェントに数十または数百のさまざまなツールを追加し、それらを経時的に維持する代わりに、エージェントがアクセスできるツールをカプセル化する MCP サーバーを登録できます。このアプローチは、ツールのパッケージ化、提示、呼び出し方法を標準化します。これにより、エージェント内でのツールの使用に伴うスケールとガバナンスの課題に対処できます。また、外部機能に使用するツールからエージェント開発とオペレーションを切り離します。

次の図は、MCP を使用して外部リソースにアクセスするエージェントを示しています。



ただし、MCP 標準では、スケーリングとガバナンスに関するすべての課題が解決されるわけではありません。MCP サーバーの実装は、効果的なツール設計、ホスティング、エンタープライズガバナンス戦略と組み合わせる必要があります。このガイドでは、エージェント AI ソリューションの一部として MCP を構築して使用するのに役立つ各戦略のベストプラクティスを提供します。

MCP を使用するタイミング

MCP は、エージェント AI イニシアチブをスケーリングするための戦略的インフラストラクチャを提供します。ツールの管理とガバナンスを一元化することで、MCP サーバーは複数のエージェント間でカスタム統合を構築および維持するための累積コストを削減します。これにより、エージェントのエコシステムが拡大するにつれてリターンが増加します。

MCP は、次の場合に戦略の一部になる可能性があります。

- データベース、APIs、内部ツール、サードパーティー統合などのエンタープライズシステムやサービスにエージェントがアクセスする方法を一元管理する必要があります。
- 開発者は、実装間で一貫性のないカスタム統合の作成にあまりにも多くの時間を費やしています。
- 一般的な機能を提供できるツールが重複している。
- 標準化され管理された MCP インターフェイスを通じて、独自のツールやデータを外部のコンシューマーやサードパーティーのエージェントシステムに提供し、セキュリティと制御を維持しながら新しい収益ストリームを解放したいと考えています。

MCP サーバーが戦略の一部になると判断したら、既存のオープンソース MCP サーバーの実装がニーズを満たしているかどうか、拡張が必要かどうか、カスタムサーバーを構築する必要があるかどうかを評価します。構築済みの MCP サーバーの実装の多くはパブリックリポジトリで利用でき、ファイルシステムアクセス、ウェブブラウジング、コードサンドボックス、データベースアクセス、API 統合などの一般的な機能を対象としています。

多くの場合、既存の MCP サーバーで十分です。たとえば、 はAWS MCP Server、AI アシスタントとエージェントに自然言語インタラクション AWS のサービス を通じて への安全で認証されたアクセスを提供するマネージドリモート MCP サーバーである AWS を提供します。を使用して、 AWS ドキュメントへのリアルタイムアクセス、構文的に正しい API コール、ベストプラクティスに従う AWS エージェント SOPs と呼ばれる構築済みのワークフローを組み合わせることで、複雑な複数ステップの AWS タスク AWS MCP Server を実行できます。 AWS は を継続的にテスト AWS MCP Server して、カスタマーエージェントがそれらを正常に使用できることを確認します。

これらの既存の MCP サーバーをエージェントでテストして、ユースケースを満たしているかどうかを判断する必要があります。エージェントがワークフローを完了できなかったり、正しくないレスポンスや最適でないレスポンスを生成したり、複雑なマルチステッププロセスをナビゲートできなかったり、重要なドメイン固有のベストプラクティスやセキュリティ上の考慮事項を見逃したりした場合は、いくつかの側面で強化を検討する必要があります。

既存の MCP サーバーがニーズを完全に満たしておらず、既存のツールを正しく使用したり、正確なレスポンスを生成したりするのに苦労する場合は、カスタムサーバーを構築する前に、次の拡張アプローチを検討してください。

- エージェントコンテキストの強化 – エージェントが既存の MCP サーバーでツールを適切または効率的に使用できない場合は、これらのツール定義に追加のドキュメントや例を追加することを検討してください。これにより、LLM に追加のコンテキストが提供されます。
- 補完ツールの追加 – エージェントがワークフローを正常に完了するために必要な追加の組織データまたはコンテキストにアクセスするツールを使用して、既存の MCP サーバーを拡張します。
- 基盤となる APIs – パラメータの複雑さを軽減し、より明確なエラーメッセージを提供し、エージェントが使用できる適切なデフォルトを提供することで、サービス APIs を簡素化して LLM フレンドリにします。

既存の MCP サーバーの実装を使用すると、一般的な機能の開発が加速されますが、ユースケースで特殊な機能が必要な場合は、カスタム MCP サーバーを構築することが不可欠です。カスタム MCP サーバーは、ドメインの専門知識をカプセル化し、組織標準を適用し、複雑なワークフローのエージェントの信頼性を向上させ、セキュリティ要件への準拠をサポートします。次の状況では、カスタム MCP サーバーの構築を検討してください。

- ドメイン固有のワークフロー – 必要な知識が API ドキュメントに記録されていない場合は、ドメインの専門知識を必要とするマルチステップワークフローをカスタム MCP ツールにカプセル化する必要があります。 例えば、医療保険の相互運用性と説明責任に関する法律 (HIPAA) のコンプライアンスを検証し、PII を匿名化し、HL7 FHIR 形式に変換する必要がある複雑な医療データパイプラインをエージェントにオーケストレーションさせる代わりに、ドメインの専門知識を直接埋め込む`process_patient_data`ツールを提供します。これにより、ワークフローステップを正しく調整して実行するための LLM への依存関係が削除され、一貫性とコンプライアンスが向上します。
- ゴールデンパス抽象化 – エージェントは、組織のコンテキストが不足し、組織のベストプラクティスではなく基本的なパターンがデフォルトであるため、最適なアプローチの実装に苦労する可能性があります。これらのシナリオでは、これらのゴールデンパスをカスタム MCP ツールにカプセル化することで、コスト、パフォーマンス、またはセキュリティに関する規範的な標準

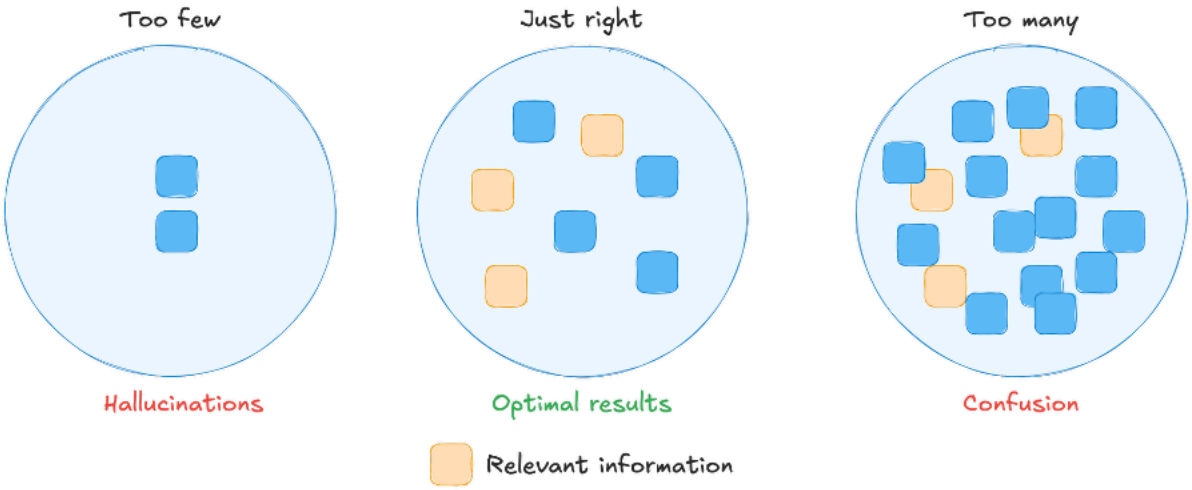
を適用できます。たとえば、エージェントが安全でない、または非効率なデフォルト設定でインフラストラクチャをデプロイするのではなく、組織の標準を直接埋め込むツールを提供します `deploy_secure_infrastructure`。

- 複雑なマルチサービスオーケストレーション – 各ステップで使用する正しいシーケンスとサービスのセットを推測することで、エージェントに複雑なワークフローをオーケストレーションさせる代わりに、MCP ツール内でその口ジックを決定的に構築できます。また、エージェントが認識していない最適なサービス統合パターンに関する専門知識を提供することもできます。 これにより、エージェントの精度と効率を向上させることもできます。
- サービス固有のベストプラクティス – これは、エージェントがエージェントツールを介してアクセスされるサービスに固有の暗号化ポリシー、アクセスコントロール、コンプライアンスパターンを実装するのに役立つセキュリティに焦点を当てたツールに共通です。さらに、サービス固有の運用上のベストプラクティスが明確でない場合、MCP サーバーを使用すると、それらが実装され、理由がエージェントに委ねられないようにすることができます。

MCP ツール設計戦略

MCP クライアントとサーバーの主な仕事は、LLM がそれらを使用してレスポンスを改善できるように、LLM にツールを検出して提示することです。これにより、MCP ツール設計は、効果的な MCP ソリューションを構築するための最も重要な戦略の 1 つです。モデルの観点から見ると、ツールは、より正確で完全なレスポンスを提供するために必要に応じて呼び出すことができる関数です。関数インターフェイスは、ツールの基盤となる実装を抽象化します。この実装は、単一の API コールのラッパーから複雑なワークフローロジックまで多岐にわたります。

ただし、LLM に提供されるツールの数とのバランスを取る必要があります。ツールが少なすぎる場合、LLM は適切なコンテキストと情報を収集できない可能性があるため、モデル内で利用可能な情報で最良の推測を行います。ツールが多すぎると、LLM が正しいツールの選択とシーケンスについて混乱し、ハルシネーションにつながる可能性があります。目標は、ツールの数を正しく把握することです。次の図は、ツールが少なすぎる、または多すぎるという課題を示しています。



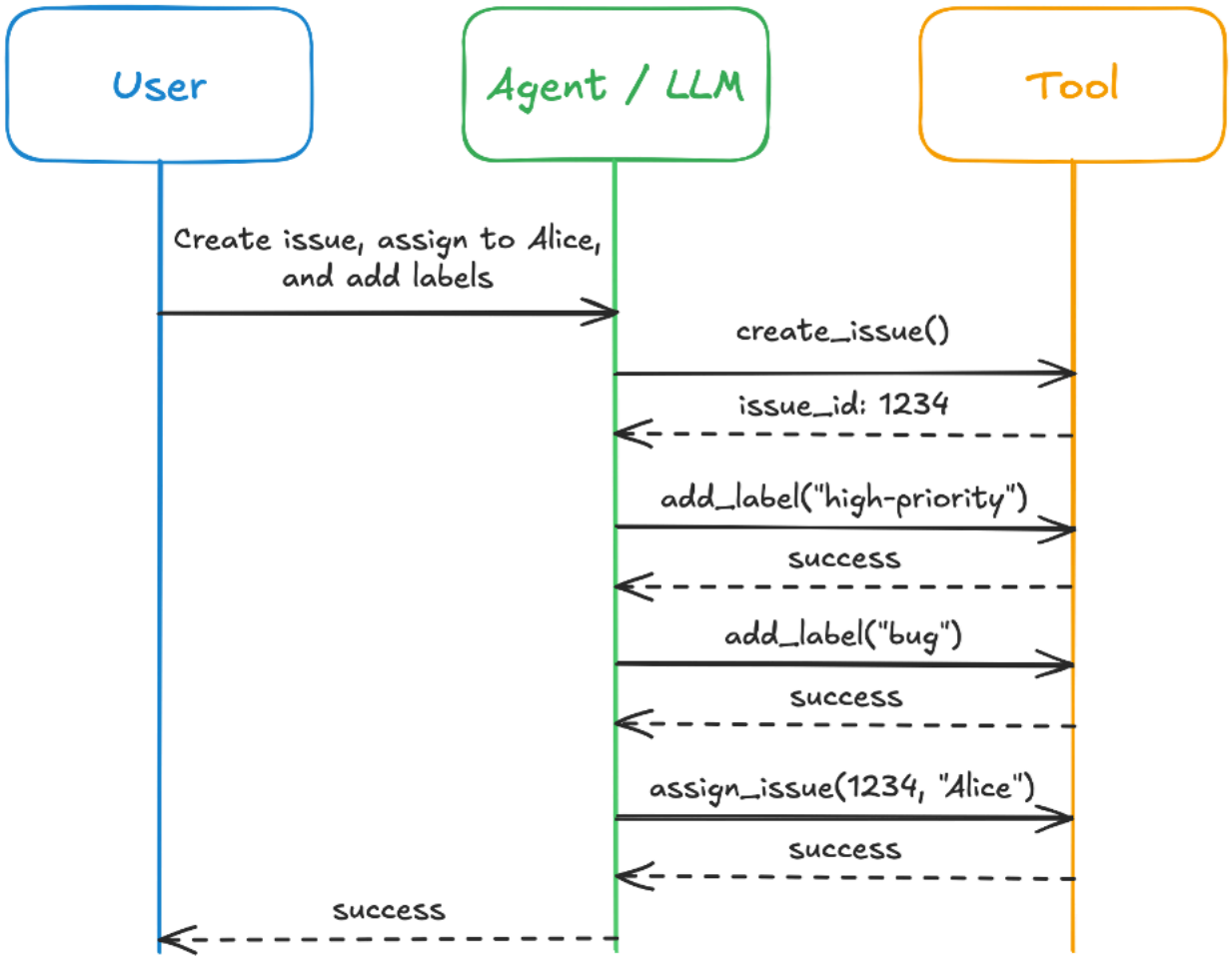
このソリューションでは、提供する ツールの数と、各ツールの範囲 設定方法を理解する必要があります。ツールの詳細度は、個々の API コールにマッピングするかワークフローを完了するかにかかわらず、エージェントが必要とするツールの総数と、ツールをどの程度効果的に使用できるかに直接影響します。このセクションでは、MCP ツールの範囲設定、ツール定義の作成、ツールの検出、および整理のベストプラクティスについて説明します。

ツールスコープ

ツールの開発には、細粒度と粗粒度の 2 つのアプローチがあります。

粒度

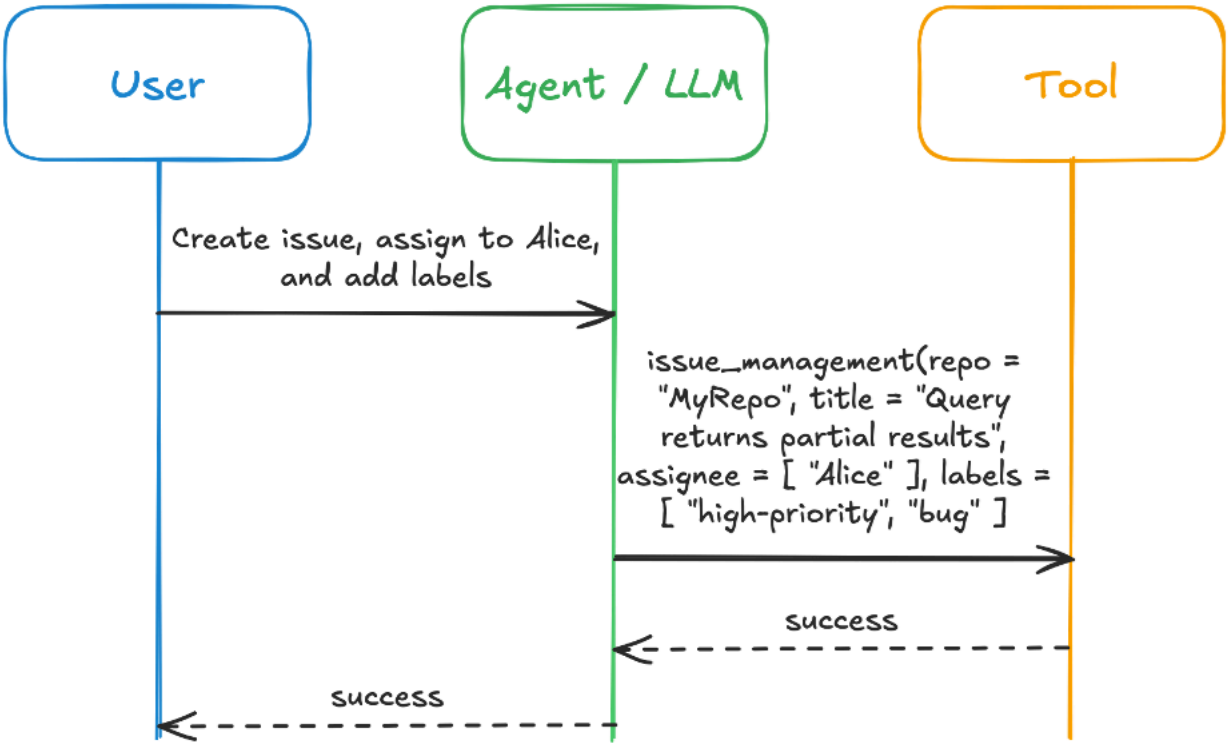
きめ細かなアプローチでは、API、アクション、またはクエリごと
にツールを作成します。たとえば、Git リポジトリclose_issueの
create_issue、get_issueadd_label、assign_issue、、 ツールを作成できます。これにより、LLM は各 API をきめ細かく呼び出し、必要に応じて各 API をオーケストレーションできます。次のプロンプトを考慮してください。「クエリは部分的な結果のみを返します」という製品サービスの問題を作成し、バグと高優先度としてラベル付けして、Alice に割り当てます。」 次の図は、tool-per-APIアプローチがこのプロンプトにどのように応答するかを示しています。



この方法では、システムプロンプトと登録されたすべてのツール定義が、呼び出しごとに LLM に提供されます。これにより、追加のコンテキストが消費され、各ツール呼び出しは LLM への個々の呼び出しを表すため、レイテンシーペナルティが発生します。また、ワークフロー内でのエラー処理の複雑さも増します。

粗粒度

粗粒度またはワークフロー駆動型のアプローチは、ワークフロー指向のツールです。このツールは、API 構造よりもend-to-endユーザーインテントに焦点を当てています。tool-per-APIの代わりに、多くの APIs1 つあります。前の Git リポジトリの例を使用して、エージェントによって 1 回呼び出されるcreate_and_setup_issueツールを作成できます。ツール実装は、ツールに提供されたパラメータに基づいて、問題を作成し、ラベルを追加し、ユーザーに割り当てます。次の図は、粗粒度のアプローチが同じプロンプトをどのように処理するかを示しています。



このアプローチは、すべての複雑さが LLM レイヤーからどのように隠されているかを示しています。ツール実装内にオーケストレーションロジックが埋め込まれている場合、シーケンシャルステップ、ログ記録、再試行ロジック、サーキットブレーカー、レート制限はすべて、ツール内で決定的に実行されます。ワークフロー駆動型のアプローチにより、LLM は適切なパラメータを使用して正しいツールを簡単に呼び出すことができます。Amazon EC2 APIs など、一部の RunInstances API がすでにワークフローインテントを提供している可能性があることに注意してください。このような場合、tool-per-APIがワークフロー指向の設計を提供する場合があります。

ただし、ツールも粗すぎる可能性があります。単一のワークフローツールがあまりにも多くのことをしようとし、可能なパラメータが多数ある場合、LLM はツールを正しく使用する方法についての推論に苦労する可能性があります。また、パラメータの選択とエラー処理でチャレンジを作成することもできます。したがって、ツール開発は、ユーザーのインテントに沿ったバランスを取り、1 つ

のツールで機能が少なすぎたり多すぎたりしないようにする必要があります。一般的に発生するオペレーション (3 つ以上の API コールなど) をバンドルする完全なユーザーワークフローを中心にツールを設計することをお勧めします。また、8 つ以上のパラメータを超えるツールを分解するか、複数の個別のユーザーインテントを処理することをお勧めします。実際のプロンプトで テストして、エージェントが各ツールを正しく使用できることを確認します。

決定論的なツールとして簡単にカプセル化できない複雑で動的なワークフローがある場合は、agent-as-toolパターンの使用を検討してください。プライマリエージェントがワークフロー内の複雑なタスクをオーケストレーションしようとする代わりに、専門エージェントはツールとして機能します。これらのタイプのツールは、高度な意思決定と分岐を実装でき、決定論的なコードでは簡単に管理できないエラーを処理してロジックを再試行できます。これは Agent2Agent (A2A) プロトコルと似ていますが、異なります。A2A プロトコルは補完的であり、エージェントフレームワーク内のエージェント間の相互運用性とコラボレーションを提供します。

まず、最も一般的なユーザーワークフローをマッピングしてワークフロー分析を開始し、各エージェントが必要とするコア機能を特定することをお勧めします。これにより、実行可能な最小限のツールセットが確立されます。大規模な MCP サーバーの開発経験に基づいて、以下のプラクティスをお勧めします。これらのプラクティスが競合する場合は、ユーザーのインテントとワークフローに優先順位を付けます。

MCP ツールスコープのベストプラクティス

- ユーザーストーリーを考え、一般的なオペレーションをバンドルする – ツールは、複数のオペレーションのオーケストレーションを必要とせずに、ユーザーとのやり取りを完了するように直接マッピングする必要があります。ワークフローに通常 3 つ以上の個別の呼び出しが必要な場合は、それらを 1 つのツールに結合します。これにより、LLM の認知負荷を軽減し、ツール呼び出しの数を最小限に抑え、タスクの完了に必要なコンテキストの消費とレイテンシーを減らし、精度とレイテンシーを向上させます。
- パラメータを 8 以下に制限する – ツールが 8 つのパラメータを超える場合は、複数のツールに分解します。LLMs複雑さが増すにつれてパラメータの選択に苦労します。

> **Note**
>
> バンドルオペレーションで 8 つ以上のパラメータが必要な場合は、厳密なパラメータ制限よりもワークフローの簡素化の方が価値があるため、パラメータ数よりもバンドルを優先します。

- 読み取りオペレーションと書き込みオペレーションを分離する – データを読み取って変更するためのさまざまなツールを提供します。この分離により、エージェントが破壊的になり得るオペレーションを実行しているときが明確になり、さまざまな認可ポリシーが有効になり、情報収集中の意図しない変更のリスクが軽減されます。
- 適切なデフォルトを提供する – LLM が個々のリクエストに固有のパラメータのみを指定する必要があるようにツールを設計します。デフォルトは、LLM が説明する必要がある情報を最小限に抑えることで、パラメータの複雑さを軽減し、ツール選択の精度を向上させます。
- 決定論的な実行を優先する – 可能な場合は、ツールの実行と出力を決定します。決定論的ツールは信頼性が高く、テストが容易です。インテリジェントなオーケストレーション、分岐ロジック、高度なエラー処理を必要とする複雑なワークフローで、決定論的なコードでは簡単に管理できない場合は、特殊なエージェントをツールとして使用することを検討してください。ただし、このパターンは複雑になるため、選択的に使用してください。

ツール定義

LLM は、直接処理できないリクエストを受け取ると、リクエストの完了に役立つ利用可能なツールを確認します。LLM は、提供されたツールの名前と説明、およびプロンプトで提供された手順のセマンティック理解に基づいてツールを選択します。次に、定義された入力スキーマに基づいて入力を作成し、出力スキーマに基づいて出力を期待します。したがって、LLM がツールを効果的に選択できるように、わかりやすいツール定義と検証済みの入出力スキーマを作成することが重要です。通常、このドキュメントの作成にはツール仕様アプローチとドキュメント作成アプローチの 2 つのアプローチがあります。

ツール仕様のアプローチ

推奨されるアプローチは、ツールを定義するときに MCP ツールの仕様に直接従うことです。次の例は、Strands Agent ツールデコレータを使用して示されています。

```
@tool(
  name = "search_website",
  description = "This tool searches the provided website for semantic matches to the
 query provided",
  inputSchema = {
    "json": {
      "type": "object",
      "properties": {
        "url": {
          "type": "string",
```

```
            "description": "The url of the website to load and search."
          },
          "query": {
            "type": "string",
            "description": "The content you want to try and match in the website."
        }
      },
      "required": ["url", "query"]
  },
  outputSchema = {
    "json": {
      "type": "object",
      "properties": {
      "results": {
          "type": "array",
          "items": {
            "type": "string"
          }
        }
      }
    }
  }
)
def search_website:
  …
```

、、name、 descriptionなどの標準フィールドを使用するとinputSchema、LLM と人間の両方が理解できる一貫したドキュメントがすべてのツールにoutputSchema用意されます。すべてのツールは、少なくともこれらのフィールドを定義し、オプションでタイトルと注釈を提供する必要があります。これは、ツールの動作に関するオプションのヒントです。可能であれば、LLM が正しいオプションを簡単に選択できるように、パラメータ値に列挙型を使用します。列挙型は、ステータスや優先度の値などの有限セットに最適ですが、自由形式のテキスト、動的値、任意の数値、リソース識別子には適していません。このような場合は、代わりに明確な説明と例を提供します。また、可能な場合はデフォルト値を含めます。これにより、LLM は正しいオプションを推測する必要はありません。ツール定義は、各呼び出しの LLM プロンプトに含まれ、システム命令と会話履歴とともにコンテキストウィンドウ領域を消費することに注意してください。

Docstring アプローチ

Python でツールを記述する場合のもう 1 つの方法は、docstrings を使用してツールの説明、使用状況、出力を提供することです。このアプローチの例を次に示します。

```
def search_website(url: str, query: str) -> list:

    """
    This tool loads the specified website and then attempts to find content that
 matches the provided query through semantic search. It provides back a list of strings
 that are the sentences that match the query.
    Args:
      url: the website url to load
      query: the content you want to semantically match in the website
    """
```

Docstrings はスキーマや標準化された形式を適用しません。このアプローチを使用すると、ツール開発者が各ツールをどのようにドキュメント化するかに基づいて、一貫性のない結果が得られる可能性があります。このアプローチに従うには、組織全体の標準を定義して適用することが不可欠です。

MCP ツール定義のベストプラクティス

- MCP ツール仕様に従う – すべてのツールに `name`、`description`、`inputSchema`、および `outputSchema`フィールドを指定します。Python 実装では、Pydantic モデルを使用して、フィールドの説明、自動型検証、列挙型による制約値を通じてインラインドキュメントを提供します。これにより、スキーマが自己文書化され、有効なパラメータオプションに対する LLM の理解が向上します。
- プロンプトとして記述する – ツールの説明は、LLM の意思決定をガイドする手順です。ツールの目的の必須コンポーネント (ツールの動作) 、ツールを使用するタイミング (ユーザーのインテントパターンまたはシナリオ) 、出力のコンテキスト (出力が使用される内容) 、パラメータ、エラー条件を含めます。
- 具体的な例を提供する – 実際の値を含むワークフロー例を含めることは、正しいツールの使用について LLMsガイドする最も効果的な方法です。
- 依存関係を明示的に文書化する – 前提条件、番号付きシーケンス、状態の変更、フォローアップアクションを含めます。

ツール検出

エージェント内のツールを検出して MCP サーバーに登録するには、静的定義、動的検出、検索関数の 3 つのアプローチがあります。

静的定義

まず、エージェントコードで使用可能なツールを静的に直接定義できます。このアプローチでは、リモートツール (Strands Agent SDK などのフレームワーク内のクライアント側の参照オブジェクト) を、MCP クライアントによってアクセスされる MCP サーバーによって提供される各ツールに定義します。次の例では、ストリーミング可能な HTTP トランスポートを使用しています。

```
from mcp.client.streamable_http import streamablehttp_client
from strands import Agent
from strands.tools.mcp import MCPClient

streamable_http_mcp_client = MCPClient(
    lambda: streamablehttp_client("https://mcp1:8000/mcp")
)

reverse_text = RemoteTool(
    name="reverseText",
    client=streamable_http_mcp_client
)

agent = Agent(tools=[reverse_text])
```

個々のツール登録により、LLM で使用できるツールを非常に選択的に選択できるため、使用するコンテキストウィンドウの量を最小限に抑えることができます。トレードオフは、使用可能なツールの名前を知る必要があり、使用可能なツールが MCP サーバーで変更されると脆弱になる可能性があることです。

動的検出

次のアプローチは、動的検出を使用し、使用可能なすべてのツールをエージェントに登録することです。このアプローチでは、より多くのツールが MCP サーバーに追加されるにつれて、コンテキストを直線的に消費します。このアプローチの例を次に示します。

```
from mcp.client.streamable_http import streamablehttp_client
from strands import Agent
from strands.tools.mcp import MCPClient

streamable_http_mcp_client = MCPClient(
    lambda: streamablehttp_client("https://mcp1:8000/mcp")
)
```

```
with streamable_http_mcp_client:
    tools = streamable_http_mcp_client.list_tools_sync()
    agent = Agent(tools=tools)
```

一般的なツール定義が約 250～500 トークン (名前、説明、スキーマを含む) を消費するシナリオを考えてみましょう。20 個の ツールを登録すると、コンテキストウィンドウの トークンが 5,000～10,000 個消費されます。少数の MCP サーバーがあり、ツールの数を制御できる場合、このオプションは実装が最も簡単です。ただし、ツールのリストが増加することが予想される場合、エージェントにサイレントコンテキスト管理の問題が発生する可能性があります。このアプローチの代替バリエーションは`list_tools`、Strands Agents SDK が提供するものなど、 を呼び出すときにツールフィルターパラメータを使用して、エージェントに登録されているツールの数を減らすことです。

検索関数

3 番目のオプションは、検索関数を使用して、ランタイム中に関連するツールを見つけることです。MCP サーバーから使用可能なすべてのツールを一覧表示し、ユーザープロンプトに基づいてそれらのツールに対してセマンティック検索を実行します。次に、結果のツールがエージェントに登録されます。Amazon Bedrock AgentCore Gateway は、この種のソリューションの実装を容易にするネイティブセマンティック検索機能を提供します。

MCP ツール検出のベストプラクティス

- コンテキストウィンドウの保存 – コンテキストウィンドウをできるだけ保存するツール検出および登録アプローチを選択します。
- ツールフィルタリングまたはセマンティック検索機能を使用する – LLM にスコープダウンされたツールセットを動的に提供し、適切なツールを選択する際の精度と有効性を向上させます。ツールフィルタリングは、ツール名 (完全一致またはパターン) 、ツールの説明 (セマンティック一致) 、またはドメインタグまたはカテゴリタグで動作できます。セマンティック検索は、ユーザーのインテントをツールの説明と照合する場合に特に効果的です。どちらのアプローチもコンテキストウィンドウの使用を減らします。

ツールの整理

適切なツールを発見し、LLM がそれらを効果的に使用できることを確認することは、効果的なツール開発の最も重要な部分の 1 つです。MCP サーバーの開発を開始するには、以下を決定する戦略が必要です。

- 1 つの MCP サーバーに入るツールの数

- 同じ MCP サーバーに配置するべきではないツール
- ツールに名前を付けて検索可能にし、名前の衝突を防ぐ方法 (同じ名前の異なるツール)
- ツールと MCP サーバーをドキュメント化して LLM で使いやすくする方法

名前空間組織は、ツール名の衝突を防ぎ、関連機能をグループ化し、LLMs。このパターンは、非構造化蓄積ではなく、整理されたストレージシステムに類似した構造化分類を確立します。ツールの命名には domain-noun-verb パターンをお勧めします。たとえば、、`github_issue_create`、`github_issue_list`、`github_issue_update`、`github_pullrequ` たは ですgithub_pullrequest_merge。このパターンの利点は、アルファベット順のソート動作を調べる際に明らかです。ツールがアルファベット順に表示されると、すべての問題関連のオペレーションがクラスター化され (`create`、`list`、`update`)、プルリクエストオペレーションが続きます (`create`、`list`、)`merge`。名詞 (リソースタイプ) は組織の境界として機能します。この構造により、関連する機能が自然にグループ化されるため、LLM ツールのスキャンとヒューマンドキュメントのナビゲーションの両方が容易になります。

MCP サーバーはドメインレベルで制限する必要がありますが、提供する機能の職務の分離に基づいて分割できます。たとえば、データベースへの書き込みオペレーションと読み取りオペレーション用に個別の MCP サーバーがあるとします。この分離を適用するには、ユーザーのインテントとアクセス許可に基づいてアクセスできる MCP サーバーを制限するガードレールをエージェントレベルで実装することをお勧めします。これは、次の組み合わせによって実現できます。

- 条件付きサーバーのロード – エージェントがユーザー入力で読み取りオペレーションを検出した場合にのみ、読み取り専用 MCP サーバーをロードします。
- アクセス許可ベースのフィルタリング – ユーザー認可を使用して、適切な MCP サーバーのみへのアクセスを許可します。

最後に、MCP サーバーによって提供されるツールの数の上限を作成します。エージェントが MCP サーバーをどのように使用するかについて仮定しないでください。使用可能なすべてのツールをナイーブに一覧表示し、すべてを LLM に提供する場合があります。1 つのサーバーに 50 を超えるツールがある場合は、複数のサーバーに分割することを検討する必要があります。

MPC ツール組織のベストプラクティス

- ツールに domain-noun-verb 命名基準を使用する – MCP サーバーとエージェントの両方で名前の衝突を防ぐための戦略を実装します。
- 上限を設定する – 1 つの MCP サーバー内のツールの数を制限します。

- MCP サーバーの分割 – 職務の分離を使用して MCP サーバーを論理グループに分割します。

MCP ホスティング戦略

使用可能なツールを MCP サーバーに抽象化すると、エージェント開発と使用可能なツールが切り離されます。これにより、MCP サーバーをホストする場所と、それらのサーバー内でツールがどのように整理されるかという課題が生じます。

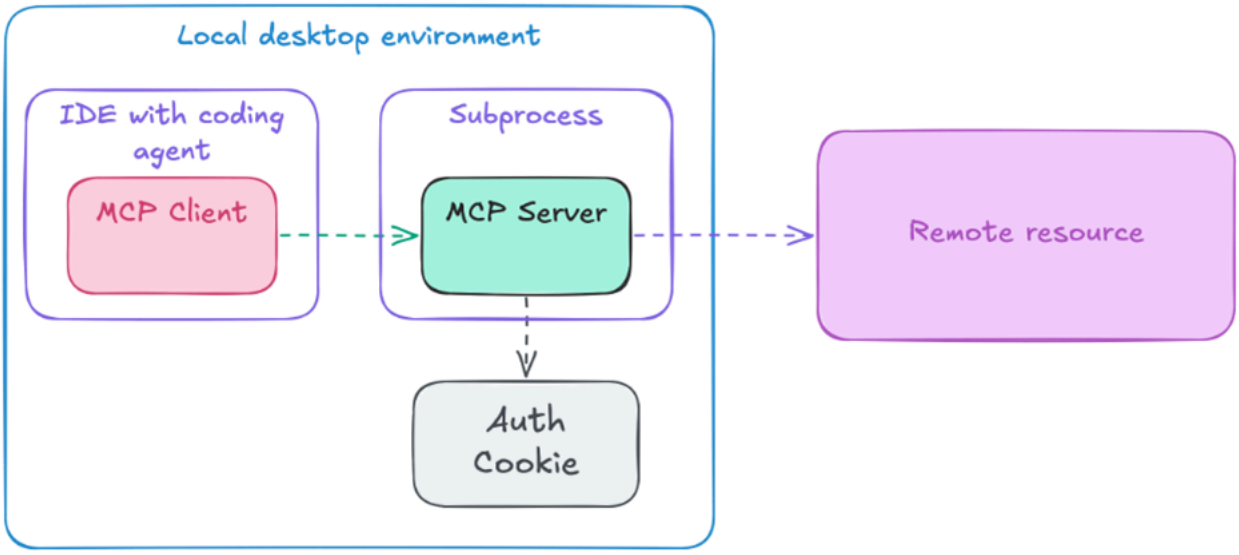
ホスティングアプローチ

MCP サーバーをホストするには、エンドユーザーマシンでローカルに実行する、リモートでホストする、または MCP ゲートウェイを介してホストする 3 つのオプションがあります。各オプションには利点とトレードオフがあります。

ローカルホスティング

ローカルホスティングは、標準入出力ストリームで JSON-RPC を使用してサーバーと通信するエージェントとともに、ローカルマシンのサブプロセスとして MCP サーバーを実行します。このアプローチでは、クライアントとサーバー間の認証は必要ありません。ツールは、ローカルアプリケーションやファイルとやり取りしたり、ローカルに保存されている認証情報を使用したり、ユーザーのローカルマシンのネットワークアクセスを継承したりできます。これは最もシンプルなホスティングパターンであり、いくつかの利点があります。

多くのお客様は、ローカルサーバーを使用して MCP の使用を開始します。これにより、エンジニアはローカル環境からさまざまな問題を迅速に反復して解決できます。エンジニアのコーディングアシスタントが使用する Git リポジトリに接続する MCP サーバーを考えてみましょう。MCP サーバーをローカルに保持することは、エンジニアの一意の認証情報を使用してリポジトリにアクセスでき、リモート MCP サーバーにネットワーク呼び出しを追加しないため、非常に理にかなっています。次の図は、IDE のコーディングエージェントで使用されているローカルでホストされた MCP サーバーを示しています。



これらのタイプのデプロイでは、MCP サーバーの開発方法と分散方法を考慮する必要があります。ほとんどのお客様は、エンドユーザーがサーバーを登録およびダウンロードできる MCP レジストリを開発しています。これは、ユーザーが特定の機能を検索し、ニーズに適した MCP サーバーを見つけることができるコンテナレジストリと非常によく似ています。

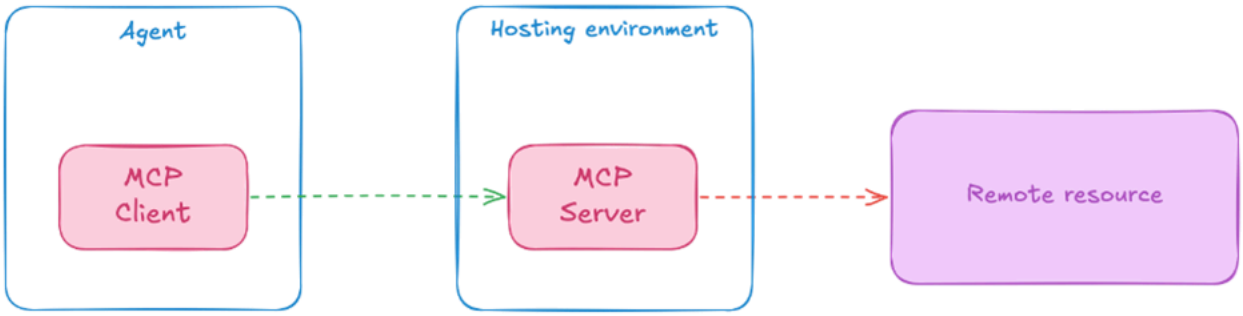
公式 MCP レジストリなどのパブリック MCP レジストリがあり、プライベートにホストされたレジストリがあります。通常、組織は MCP レジストリ戦略を、オープンソースソフトウェア配布、コンテナレジストリ、内部パッケージ管理に関する既存のポリシーと整合させます。セキュリティスキャン、承認ワークフロー、コンプライアンス要件などの要因を考慮する必要があります。

ただし、ローカルホスティングには、組織が考慮すべき運用上の課題があります。まず、エンドユーザーは MCP サーバーを個別に検出、ダウンロード、設定する必要があります。これにより、ローカルで使用する個々の MCP サーバーの使用を開始するための複雑さが増す可能性があります。次に、MCP サーバーのライフサイクルを制御できません。つまり、ユーザーはセキュリティの脆弱性や機能不足により、古いバージョンをローカルで引き続き実行できます。これにより、コンプライアンス要件を満たすことが複雑になる可能性があります。Kiro などの一部の IDEs や CLI ツールを使用すると、組織は使用可能な MCP ツールを管理および制御できるため、チーム間の一貫性とセキュリティが確保されます。

リモートホスティング

2 番目のオプションは、HTTP または HTTPS 経由でアクセスされるリモート MCP サーバーをホストすることです。これにより、ネットワーク接続されたクライアントにアクセスできます。リモートホスティングを使用すると、MCP リソースと機能へのアクセスを一元的に制御し、認証と認可を実装し、MCP サーバーロジックのバージョニングと更新を制御できます。リモートホスティングで

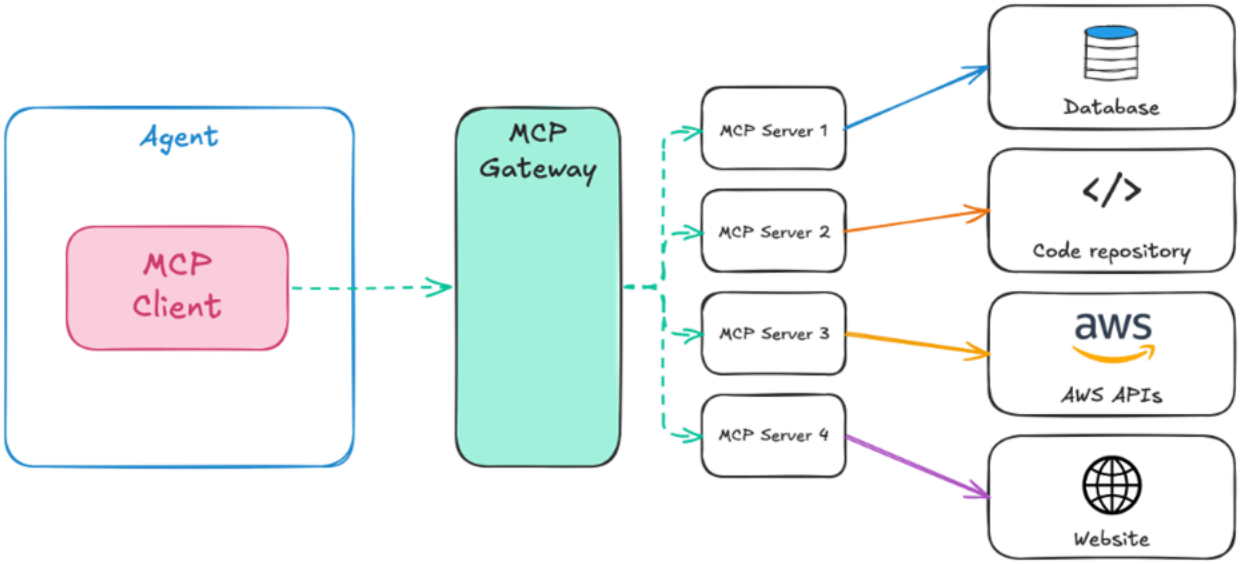
は、エンドユーザーがエージェントで使用する MCP サーバーを検出できるように、引き続き MCP レジストリを使用する必要があります。次の図は、リモートホスティングのアプローチを示しています。



エージェント開発の観点から見ると、MCP サーバーがローカルでもリモートでもエクスペリエンスは似ています。最も重要な変更は、エージェントの MCP サーバーへのアクセスとサーバーの外部リソースへのアクセスの両方を含む、認証と認可の実装です。リモート MCP サーバーの実装は、マルチテナントアクセスと特権管理を検討するために慎重に計画する必要があります。MCP ガバナンス戦略の章には、認証と認可に関する考慮事項に関する詳細情報が含まれています。

MCP ゲートウェイ

最後のオプションは MCP ゲートウェイを使用することです。MCP ゲートウェイは、MCP クライアントとサーバー間の一元化されたプロキシとして機能し、登録された MCP サーバーへのアクセスをオーケストレーションします。ゲートウェイがない場合、各エージェントは使用するすべてのリモート MCP サーバーを登録する必要があります。ゲートウェイを使用すると、エージェントは認証、認可、ルーティング、プロトコル変換を管理する単一のエンドポイントに接続できます。新しい MCP サーバーとツールは動的に追加でき、エージェントはすぐに使用できるようになります。次の図は、MCP ゲートウェイのアプローチを示しています。



Docker MCP Gateway などの一部のゲートウェイソリューションも MCP サーバーのライフサイクルを管理し、必要に応じてサーバーをオンデマンドで起動します。Amazon Bedrock AgentCore Gateway などの MCP ゲートウェイは、ネイティブセマンティック検索機能を提供することでツール検出の管理にも役立ちます。これにより、エージェントは 1 つのエンドポイントで MCP クライアントに接続でき、コンテキストウィンドウの使用を最適化できます。その結果、MCP ツールを効果的に選択して使用できるシンプルなエージェントになります。ただし、リモート MCP サーバーアプローチと同様のアイデンティティ関連の課題があります。

MCP サーバーをホストするためのベストプラクティス

- ホスティングオプションのスペクトルは、すべてに適合する 1 つのサイズではありません。現在の MCP サーバーの使用の多くはローカルです。
- リモート MCP サーバーの使用を開始する際の主な考慮事項は、MCP サーバーに対する一貫した認証と認可、および MCP サーバーがダウンストリームリソースに対する認証と認可を実行する方法です。
- MCP ゲートウェイは、複数のリモート MCP サーバーをホストするための接続と認証、認可を簡素化します。また、該当するツールを検索してコンテキストウィンドウ管理を改善する機能も提供します。

MCP ガバナンス戦略

MCP が組織に提供するもう 1 つの重要な機能は、一元化されたガバナンスのサポートです。MCP ガバナンス戦略は、MCP サーバーとそれらがアクセスするリソースの両方に対する認証と認可に対処する必要があります。また、ダウンストリームリソースを保護するためのレート制限、ツールの使用状況とパフォーマンスをモニタリングするための運用メトリクス、MCP サーバーのデプロイとディストリビューションの管理にも対処する必要があります。

認証と認可

認証および認可戦略の最も重要な部分の 1 つは、MCP サーバーからのダウンストリームリソースアクセスを管理することです。ユーザーがエージェントを呼び出すと、認証と認可が実行され、ユーザーにエージェントを呼び出すアクセス許可が付与されます。次に、エージェントは MCP サーバーの特定のツールの呼び出しを調整します。ツールごとにアクセスを許可する方法を決定する必要があります。

1 つのオプションはmachine-to-machine認可であり、ユーザーの同意や操作は必要ありません。たとえば、時間ベースのエージェント呼び出しでは、MCP サーバーを使用してアプリケーションから口グを収集し、分析します。このシナリオでは、エージェントは指定されたデータへのアクセスを事前に許可されています。2 番目のオプションは、ユーザーが委任したアクセスです。ユーザーは、ユーザー固有のデータとリソースへのアクセスに同意したことになります。

次の表は、認証と認可のパターンを示しています。

係数	ユーザー委任アクセス	Machine-to-machine
データの所有権	データに対するユーザー固有の認可	システムまたは組織全体のデータ
ユーザーとのやり取り	ユーザーが存在し、同意できる	ユーザー操作なし
オペレーションのタイミング	インタラクティブまたはリアルタイム	バックグラウンド、スケジュール済み、またはバッチ
アクセス許可のスコープ	アクセス許可はユーザーによって異なります	エージェントレベルでの一貫したアクセス許可

ユーザー委任アクセスには慎重な実装が必要であり、セキュリティチームとともに開発する必要があります。エージェントは、LLM が選択したツールと、追加の認可が必要かどうかを評価できる必要があります。MCP ツールには、認証と認可の要件とアクセストークンを取得する場所を示す説明が含まれている必要があります。クライアントアプリケーションは中間認証リクエストをサポートしている必要があり、MCP クライアントはツール呼び出しごとに取得した認証情報をエージェントに返す必要があります。

MCP ツールには常に外部機能にアクセスするための独自のトークンがあり、アクセスがログに記録され、監査されていることを確認する必要があります。ユーザー認証情報は、エージェントシステムを介して伝播しないでください。たとえば、MCP サーバーは、エージェントの呼び出しに使用されたデータにアクセスするために同じトークンを使用しないでください。ダウンストリーム呼び出しでは、明示的にスコープ設定され、目的別に生成されたトークンを使用する必要があります。これにより、追加のガードレールを提供して、アクションに代わって意図しないデータアクセスを防ぐことができます。また、幻覚が意図しない結果を生成するのを防ぐのにも役立ちます。完全な管理者権限を持つユーザーが、本番稼働前のデータベースのクローン作成をエージェントに依頼するとします。そのためには、ユーザーに必要なのは READと アクセスCREATE許可のみです。LLM がハルシネーションし、このリクエストの一部として古いデータベースをクリーンアップする必要があると考えるとします。ユーザーの認証情報を再利用する場合、ユーザーの元の認証情報には DELETE アクセス許可があるため、成功する可能性があります。代わりに、MCP サーバーが READと アクセスCREATE許可のみを持つリクエストに意図的にスコープダウンされたトークンを使用する場合、本番データベースを削除しようとすると失敗します。

Amazon Bedrock AgentCore Identity を使用して、これらのパターンを実装できます。MCP サーバーによってホストされるツールを一覧表示および呼び出すアクセス許可が、MCP サーバーが公開する外部機能に対するアクセス許可を意味するかどうかについて、意図的に選択していることを確認してください。MCP サーバーからリソースへ、およびユーザーに戻るこの ID フローは、使用する認証および認可サービスのタイプによって異なります。MCP サーバーの大規模な処理方法を決定する必要があります。

認証および認可パターンを設計するときは、アクセスするツールごとに異なるアクセストークンを取得するトークン分離メカニズムを実装します。ツールとサーバー間でトークンを再利用しないでください。AgentCore Identity は、このトークン分離機能を提供します。ワークロードトークン (machine-to-machine認証用) とユーザートークン (ユーザー委任アクセス用) の両方を自動的に管理して、適切な分離を確保し、アクセス許可のエスカレーションを防ぎます。これは、リモート MCP サーバーまたは MCP ゲートウェイを組み込む場合に特に重要です。

MCP 認証と認可のベストプラクティス

- トークン分離 – 発信者からダウンストリームサービスにベアラートークンを渡さないでください。aud (オーディエンス) フィールドがトークンを受信するサーバーと一致することを確認します。対象者クレームは、トークンがどのサービスを対象としているかを指定し、異なる MCP サーバー間での不正なトークンの再利用を防止します。
- アクセスアプローチを選択する – MCP サーバーが提供するツールごとにmachine-to-machineアクセスとユーザー委任アクセスを選択します。同じ認証パターンを使用する同じ MCP サーバーでツールをグループ化することを検討してください。

負荷の制御

他の分散システムと同様に、MCP サーバーフリートの負荷を制御する方法を検討する必要があります。まず、MCP サーバーにレート制限を実装するかどうかと、制限を実装する場所を検討します。レート制限を実装しない場合は、ダウンストリームリソースによって実行されるレート制限を渡します。多くのシステムは、ユーザーやアカウント ID などのリクエスト属性に基づいて制限をレートすることを選択します。ダウンストリームサービスに送信されるリクエストが同じ属性を実行し、複数のユーザーが別のユーザーによって駆動される負荷の影響を受けないようにします。

レート制限を実装することを選択した場合、推奨されるアプローチは MCP サーバーレベルでプライマリレート制限を実装することです。バックエンドサービスはセカンダリ保護を提供し、エージェントはレート制限のフィードバックに基づいて動作を適応させます。レート制限が MCP サーバーごとか、ツールごとかを検討します。MCP サーバーあたりのレート制限は、マルチテナント環境で MCP サーバーフリートとサービスを保護するのに役立ちます。ただし、これは非常に粗い場合があります。ツールごとのレート制限は、それ自体が十分にレート制限されない可能性のある圧倒的なダウンストリームリソースを防ぐように設計されています。ツールが複数の APIsを呼び出す場合は、それらの APIs で許可される最低レートに合わせてレート制限を設定する必要があります。

HTTP ヘッダーにレート制限情報を渡すことは、ユーザーや自動システムにとって独自のリクエストレートと再試行戦略の管理に役立つメトリクスでもあります。例えば、次の例に示すように、これらのヘッダーを MCP サーバーからエージェントに送り返すことができます。

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 45
X-RateLimit-Reset: 1640995200
```

さらに、単一の顧客がレート制限を超えていないが、負荷がシステムのパフォーマンスに影響を与えている場合に、サービス全体を保護するために負荷分散を検討してください。

負荷を制御するためのベストプラクティス

- レート制限アプローチを選択する – ダウンストリームリソースの使用、または MCP サーバーとツールの使用に基づいて、個々のユーザーをレート制限する計画を立てます。
- 負荷の排除を検討する – 1 人または少数の顧客によって駆動されない一般的な過負荷から MCP サーバーフリートを保護します。

運用メトリクス

MCP 実装でキャプチャする主要なメトリクスは、提供するカスタマーエクスペリエンスに焦点を当てる必要があります。これらのメトリクスには、通常、トークンの使用、ツール選択の精度、エージェントに登録されているツールの数、ツールのレイテンシーが含まれます。たとえば、各ツールによって返される出力トークンをモニタリングすると、ツールがコンテキストウィンドウの使用のしきい値を超えたときにアラームを設定できます。ツールがそのしきい値を超えた場合は、ツールの動作を確認することをお勧めします。これは MCP ツール設計戦略にも関連しています。ツール選択精度メトリクスは、エージェントが特定のタスクに適したツールをどの程度適切に選択しているかを示し、実行速度と成功率はパフォーマンスのボトルネックと信頼性の問題を強調します。

例えば、ツール選択とツール使用の精度メトリクスを評価するために、 AWS チームは回帰テスト用のゴールデンデータセットを作成しました。データセットは、ユーザークエリの履歴 API 呼び出しログの LLMs を使用して合成的に生成されました。事前定義されたツール選択メトリクスとツール使用メトリクス (ツール選択精度、ツールパラメータ精度、マルチターン関数呼び出し精度など) を使用すると、 AWS チームは AI エージェントの能力を客観的に評価して、適切なツールを正しく識別し、パラメータに正確な値を入力し、会話のターン全体で一貫したツール呼び出しシーケンスを維持できます。

エージェントに登録されているツールの数に関するメトリクスを測定すると、潜在的なコンテキストウィンドウ管理の課題や、MCP サーバーによって提供される利用可能なツールの変更を特定するのに役立ちます。MCP サーバーとツールのユーザーエクスペリエンスを示す運用メトリクスを定期的に確認する必要があります。

寄稿者

オーサリング

- Alex Torres、シニアソリューションアーキテクト、 AWS
- Saikat Gomes、シニアカスタマーソリューションマネージャー、 AWS
- Mike Haken、シニアプリンシパルソリューションアーキテクト、 AWS
- Sreeja Das、プリンシパルエンジニア、 AWS

レビューアー

- ソリューションアーキテクトマネージャー、Ted Swinyar AWS
- Raju Patil、シニアデータサイエンティスト、 AWS

テクニカルライター

- " AbouHarb、シニアテクニカルライター、 AWS

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	—	2026 年 3 月 16 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: お客様のオンプレミスの Oracle データベースを AWS クラウドの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。
- 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: お客様のオンプレミスの Oracle データベースを AWS クラウドの EC2 インスタンス上の Oracle に移行する。
- 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。

- 廃止 — 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

A2A (Agent-to-Agent)

タスクの委任と状態転送をサポートするagent-to-agentコラボレーション用のステートフルプロトコル。

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

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。アクティブ/パッシブ移行よりも柔軟な方法ですが、さらに多くの作業が必要となります。

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を同期させながら、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、接続しているアプリケーションからのトランザクションをソースデータベースのみで処理するデータベース移行方法。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。

[エージェント]

目標を達成するためのツールを使用して、自律的に推論、計画、アクションを実行できる AI システム。

エージェントオペレーション

AI エージェントを本番環境で大規模に構築、テスト、デプロイ、実行するための運用プラクティス。

集計関数

複数行に処理を行い、グループ全体を対象に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 MAX などがあります。

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

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。

アプリケーション制御

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 の重要な要素であり、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?」をご覧ください。

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 (AZ)

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 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

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

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「動作グラフのデータ」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」もご覧ください。

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

それぞれが独立しているが、同一の環境を 2 つ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 (ブルー) 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 (グリーン) 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ており、ボットハーダーまたはボットオペレーターと呼ば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。ボットネットは、ボットとその影響力を拡大する仕組みとして、非常によく知られています。

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、 AWS 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、AWSでのコンテナ化されたマイクロサービスの実行ホワイトペーパーの「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

カナリアデプロイ

エンドユーザーへのバージョンリリースを、時間をかけて段階的に行うこと。確信が持てたら新規バージョンをデプロイして、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

「変更データキャプチャ」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すること。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。

シチズンデベロッパー

専門的な技術スキルを持たないノーコード/ローコードプラットフォームを使用して AI アプリケーションを作成するビジネスユーザー。

クライアント側の暗号化

ターゲットが AWS のサービス 受信する前に、ローカルでデータを暗号化します。

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、 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、 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。移行戦略との関連性については、 AWS「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」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 や Bitbucket Cloud があります。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といった、ビジュアル形式の情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI では、CV 用の画像処理アルゴリズムを利用できます。

設定ドリフト

ワークロードにおいて、設定が想定した状態から変化すること。これによって、ワークロードが非準拠になる可能性がありますが、この状態は、徐々に生じ、意図的なものではありません。

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

非一元的で分散型のデータ所有権を持つとともに、一元的な管理およびガバナンスを行え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、一般的に、大量の履歴データが含まれており、多くの場合、それら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ます。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。

深層学習

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。

トラブルシューティング

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、「 AWSでのセキュリティコントロールの実装」の「検出的コントロール」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

スタースキーマにおいて、ファクトテーブルの定量データに関するデータ属性が含まれる小さいテーブル。ディメンションテーブルの属性は、通常、テキストフィールド、またはテキストのように扱える個別の数値で示されます。これらの属性は、一般的に、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)

ディザスタ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の戦略とプロセス。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

「開発バリュー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、応答時間を改善できます。

電子データ交換 (EDI)

組織間で行う、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処理。

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。たとえば、 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

スタースキーマの中央にあるテーブル。ビジネスオペレーションに関する定量的データが保存されます。一般的に、ファクトテーブルは、2 種類の列で構成されます。1 つは測定値が含まれる列、もう 1 つはディメンションテーブルへの外部キーが含まれる列です。

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、頻繁かつ段階的にテストを行う哲学であり、アジャイルアプローチでは、この考え方がきわめて重要です。

障害分離境界

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

LLM に、タスクと望ましい出力を示す例を少数提示した後に、類似のタスクを実行させること。この手法は、プロンプトに記述された例 (ショット) からモデルが学習する「インコンテキスト学習」の一種です。数ショットプロンプトは、特定のフォーマット、推論、専門知識が必要な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制御」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

変更データのキャプチャによる継続的なデータ複製を利用して、段階的なアプローチではなく、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)

大規模な深層学習ニューラルネットワークであり、一般化およびラベル付けされていないデータからなる大規模データセットでトレーニングされています。FM により、言語理解、テキストおよび画像生成、自然言語での会話といった、一般的な各種タスクを実行できます。詳細については、「基盤モデルとは何ですか?」を参照してください。

FM ゲートウェイ

基盤モデルへのアクセスを制御および正規化する一元化された仲介者。LLM ゲートウェイとも呼ばれます。

G

生成 AI

AI モデルのサブセット。大量のデータでトレーニングされており、シンプルなテキストプロンプトを使用して、画像、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ます。詳細については、「生成 AI とは何ですか?」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 ワークフローは古いと見なされている方法であり、トランクベースのワークフローは推奨されている新しい方法です。

ゴールデンイメージ

システムまたはソフトウェアのスナップショットであり、システムまたはソフトウェアの新規インスタンスをデプロイするテンプレートとして使用されます。製造の例で言えば、ゴールデンイメージを使用すると、複数のデバイスにソフトウェアをプロビジョニングして、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。

ガードレール (AI)

エージェントの入力と出力をフィルタリング、検証、制約して、責任ある安全な AI 動作を確保するのに役立つ安全メカニズム。

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。

高可用性 (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用データセットから保留される、ラベル付き履歴データの一部。ホールドアウトデータを使用すると、モデル予測をホールドアウトデータと比較して、モデルのパフォーマンスを評価できます。

ヒューhuman-in-the-loop (HitL)

エージェント???の実行が重要な決定時点で人間によるレビューと承認のために一時停止するワークフローパターン。

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

「インダストリアル IoT」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

既存インフラストラクチャの更新、パッチ適用、変更などを行わずに、本番環境ワークロードに使用する新規インフラストラクチャをデプロイするモデル。本質的に、イミュータブルインフラストラクチャは、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、予測性に優れています。詳細については、 AWS Well-Architected フレームワークにある「イミュータブルインフラストラクチャを使用してデプロイする」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 氏が提唱した用語で、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ズを意味します。

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。

インダストリアル 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、「インダストリアル IoT (IIoT) デジタルトランスフォーメーション戦略の構築」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

「IoT」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)

大量のデータで事前トレーニングされた深層学習 AI モデル。LLM では、質問への回答、ドキュメントの要約、他言語へのテキスト翻訳、文を完成させるなど、さまざまなタスクを実行できます。詳細については、「大規模言語モデル (LLM) とは何ですか?」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

「ラベルベースアクセス制御」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」もご覧ください。

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や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、不正アクセスを招く可能性があります。マルウェアの例には、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。

マネージドサービスの例として、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB が挙げられます。このサービスは、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、制御するソフトウェアシステムであり、工場では、これによって、原材料から製品を完成させます。

MAP

「Migration Acceleration Program」を参照してください。

MCP

「モデルコンテキストプロトコル」を参照してください。

モデルコンテキストプロトコル (MCP)

エージェントツーツール通信のステートレスプロトコル。

MCP サーバー

Model Context Protocol を通じて 1 つ以上のツールを公開するサービス。

メカニズム

ツールを作成してその導入を推進し、導入結果を調べて調整を行うための包括的なプロセス。メカニズムとは、運用中にそれ自体を強化し改善するサイクルを意味します。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。

Message Queuing Telemetry Transport (MQTT)

発行/サブスクライブのパターンに基づく、軽量のマシンツーマシン (M2M) 通信プロトコルであり、リソースに限りのある IoT デバイスに使用されます。

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)

オンラインツール。これによって、 AWS クラウドに移行するビジネスケースの検証に必要な情報を得られます。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 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 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、「AWS クラウドで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。

MPA

「Migration Portfolio Assessment」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。

ミュータブルなインフラストラクチャ

本番ワークロードに使用する既存のインフラストラクチャを更新および変更するためのモデル。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

「オリジンアクセス制御」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

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」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。

Open Process Communications - Unified Architecture (OPC-UA)

産業オートメーション用のマシンツーマシン (M2M) 通信プロトコル。OPC-UA により、相互運用の際に、データ暗号化、認証、認可の各スキームを標準化できます。

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)

質問と関連するベストプラクティスのチェックリスト。インシデントや起こり得る障害を理解、評価、防止したり、その範囲を縮小したりする際に役立ちます。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させるハードウェアおよびソフトウェアシステム。製造分野では、Industry 4.0 への変革を進める上で、OT と情報技術 (IT) システムの統合に焦点が当てられています。

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。

オリジンアクセス制御 (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制御が可能です。

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

「プログラマブル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

次の操作を可能にするオブジェクト: アクセス許可を定義する (ID ベースのポリシーを参照)。アクセス条件を指定する (リソースベースのポリシーを参照)。 AWS Organizations の組織における全アカウントにアクセス許可の上限を定義する (サービスコントロールポリシーを参照)。

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。

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、「移行の準備状況の評価」を参照してください。

述語

`true` または `false` を返すためのクエリ条件。一般的に、`WHERE` 句に記述されます。

述語プッシュダウン

データベースクエリを最適化する手法。これによって、転送前にクエリ内のデータをフィルタリングします。この手法を取ると、リレーショナルデータベースから取得し処理する必要のあるデータの量が減少するため、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、「AWSでのセキュリティコントロールの実装」の「予防的コントロール」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」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してプライバシーが考慮されているシステムエンジニアリングのアプローチ。

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防止を目的としたセキュリティコントロール。このコントロールにより、プロビジョニング前にリソースをスキャンします。コントロールに準拠していないリソースは、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)

製品の設計、開発、発売から、成長、成熟、衰退、廃棄に至る、製品のライフサイクル全体を通してデータとプロセスを管理すること。

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

製造分野で使用される、信頼性と適応性に優れたコンピュータであり、これによって、マシンをモニタリングするとともに、製造プロセスを自動化します。

プロンプトチェイニング

1 つの LLM プロンプトによる出力を次のプロンプトの入力に使用して、より良いレスポンスを生成します。この手法を使用すると、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良または拡張したりできます。これによって、モデルのレスポンスの精度と関連性が向上し、粒度の高いパーソナライズされた結果を得られます。

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。

発行/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。これにより、スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES の場合、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可能なチャネルにイベントメッセージを発行できます。このシステムでは、発行サービスの変更なしに、新規マイクロサービスを追加できます。

Q

クエリプラン

手順などの一連のステップであり、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ます。

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

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RAG

「検索拡張生成」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

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RCAC

「行と列のアクセス制御」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。

リアーキテクト

「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、「アカウントが使用できる AWS リージョン を指定する」を参照してください。

リグレッション

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。

リプラットフォーム

「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

中断に抵抗または中断から回復するアプリケーションの機能。 AWS クラウドでの回復力を計画する際には、一般的に、高可用性とディザスタリカバリが考慮されます。詳細については、「AWS クラウド の耐障害性」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。サポートが含まれる場合は RASCI マトリックスと呼ばれ、含まれない場合は 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、「 AWSでのセキュリティコントロールの実装」の「レスポンシブコントロール」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。

検索拡張生成 (RAG)

生成 AI の技術。これにより、LLM では、レスポンスの生成前に、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が参照されます。例えば、RAG モデルによって、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できる場合があります。細については、「RAG (検索拡張生成) とは何ですか?」を参照してください。

ローテーション

定期的にシークレット情報を更新して、攻撃者が認証情報にアクセスするのをより困難にするプロセス。

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 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したりでき、組織内のすべてのユーザーを IAM で作成する必要はありません。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

「監視制御とデータ取得」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。シークレット値には、バイナリ、1 つの文字列、複数の文字列を指定できます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの概要」を参照してください。

セキュリティバイデザイン

開発プロセス全体を通してセキュリティが考慮されているシステムエンジニアリングのアプローチ。

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、主に 4 つの種類があります。4 つとは、予防、検出、レスポンシブ、プロアクティブです。

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への自動レスポンスまたは自動修復を目的として、事前定義および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には、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を、 AWS のサービス が受信する によって暗号化します。

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、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

サービスレベルインジケータによって測定され、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。

シャドウ AI

組織内の管理対象チャネルの外部で構築または使用される認可されていない AI アプリケーション。

SIEM

「Security Information and Event Management システム」を参照してください。

単一障害点 (SPOF)

特定のアプリケーションを構成する単一の重要なコンポーネントで発生し、システム稼働に支障をきたす可能性のある障害。

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、「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。

SPOF

「単一障害点」を参照してください。

スタースキーマ

データベースの編成構造を意味し、1 つの大きいファクトテーブルにトランザクションデータまたは測定データが保存され、1 つ以上の小さいディメンションテーブルにデータ属性が保存されます。この構造は、データウェアハウスやビジネスインテリジェンスを用途と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。

監視制御とデータ取得 (SCADA)

製造分野において、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運用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、起こり得る問題を検出したり、パフォーマンスをモニタリングしたりすることで、システムをテストします。Amazon CloudWatch Synthetics を使用すると、こうしたテストを作成できます。

システムプロンプト

コンテキスト、指示、ガイドラインなどを提示して、LLM に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て、ユーザーとやり取りするルールを確立するのに有用です。

T

タグ

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。

tool

エージェントが外部システムでオペレーションを実行するために呼び出すことができる関数または API。

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 枚のピザを分け合える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。

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

現在のレコードに何らかの形で関連している行のグループに計算を実行する SQL 関数。ウィンドウ関数は、移動平均を計算したり、現在の行の相対位置に基づいて他の行の値にアクセスするといった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

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

データを 1 回のみ書き込むことで、データの削除や変更を防ぐストレージモデル。承認済みユーザーは、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

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

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 にタスク実行の手順は提示するが、実行のガイドとして役立つ例 (ショット) は提示しない方法。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。