aws

でのエージェント AI のパターンとワークフロー 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 のパターンとワークフロー AWS

Table of Contents

でのエージェント AI のパターンとワークフロー AWS

Aaron Sempf と Andrew Hooker、Amazon Web Services

2025 年 7 月 (ドキュメント履歴)

組織は、エージェントパターンと呼ばれる新しいアーキテクチャ分野を使用して、動的でマルチドメインの問題を解決するために、大規模言語モデル (LLMs) とソフトウェアエージェントを採用しています。エージェントパターンは、さまざまなコンテキストで目標指向の AI エージェントを設計およびオーケストレーションするために使用される基本的な設計図とモジュール構造です。

対象者

このガイドは、静的ロジック、シンボリックロジック、決定論的オートメーションを超えるインテリジェントなアプリケーションを構築したいと考えているアーキテクト、デベロッパー、製品リーダーを対象としています。

目的

このガイドでは、制御可能で目標に沿った状態で自律的に動作する AI エージェントシステムの設計フレームワークと実装アプローチを提供します。イベント駆動型のアーキテクチャパターンをさまざまなエージェント代替案と接続し、クラウドネイティブアーキテクチャを使用して本稼働グレードのエージェントシステムを構築する方法を示します。このガイドでは、以下のトピックについて説明します。

- エージェントパターン – エージェントパターンは、個々のエージェントの構造と動作を記述する再利用可能な設計テンプレートです。これには、推論エージェント、検索拡張エージェント、コーディングエージェント、音声インターフェイス、ワークフローオーケストレーター、共同マルチエージェントシステムが含まれます。各パターンは、エージェントが認識、理由、行動、学習し、マッピングする方法を示しています AWS のサービス。
- LLM ワークフロー – ワークフローは、エージェントが推論のために LLMs を使用する方法に焦点を当てています。プロンプト戦略と計画メカニズムを検討し、LLMs を使用してテキストを生成するだけでなく、エージェントループ内で構造化され、解釈可能で、信頼性の高い動作を促進する方法について概説します。
- エージェントワークフローパターン – ワークフローパターンは、複数のエージェント、ツール、環境がどのように相互作用して自律システムを形成するかを記述します。これには、タスクオーケ

ストレーション、サブエージェントの委任、イベントベースの調整、オブザーバビリティ、コントロールのパターンが含まれます。これらの側面は、スケーラブル、構成可能、監査可能な AI アーキテクチャを促進します。

このコンテンツシリーズについて

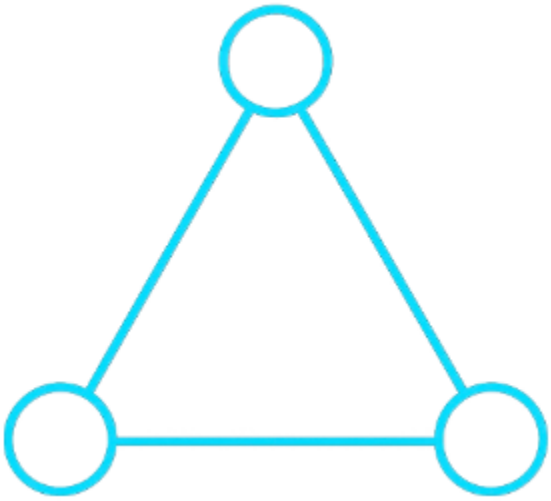
このガイドは、 でのエージェント AI に関するシリーズの一部です AWS。詳細およびこのシリーズの他のガイドについては、 AWS 「 規範ガイダンス」ウェブサイトの「エージェント AI」を参照してください。

エージェントパターン

エージェントパターンは、特定のドメイン、ユースケース、複雑さのレベルに合わせて調整できる、再利用可能な構成可能な構成要素です。ただし、エージェントシステムは従来のアプリケーションとは異なります。すべての AI エージェント設計の中心となるのは、次の 3 つの基本原則に基づく概念モデルです。

- 非同期 – エージェントは疎結合のイベントが多い環境で動作します。
- 自律性 – エージェントは人間または外部の制御なしで独立して動作します
- エージェンシー – エージェントは、特定の目標に向けて、ユーザーまたはシステムに代わって目的を持って行動します。

次の図の三角形は、ソフトウェアエージェントの中核的な構成要素である認識、理由、アクションを表しています。これにより、エージェントシステムは環境内で監視、意思決定、行動できるようになります。



設計上、エージェントパターンは AI システムを構築するためのモジュラー設計言語を提供します。つまり、アクセス可能、運用可能、拡張可能、本番稼働準備ができています。これらのシステムを設計するには、次の 3 つの相互に関連するディメンションに細心の注意が必要です。詳細については、このガイドの後半で説明します。

このセクションの内容

- 基本的な推論エージェント
- 関数を呼び出すためのツールベースのエージェント

- サーバー用のツールベースのエージェント
- コンピュータ使用エージェント
- コーディングエージェント
- 音声エージェント
- ワークフローオーケストレーションエージェント
- メモリ拡張エージェント
- シミュレーションエージェントとテスト床エージェント
- オブザーバーエージェントとモニタリングエージェント
- マルチエージェントコラボレーション

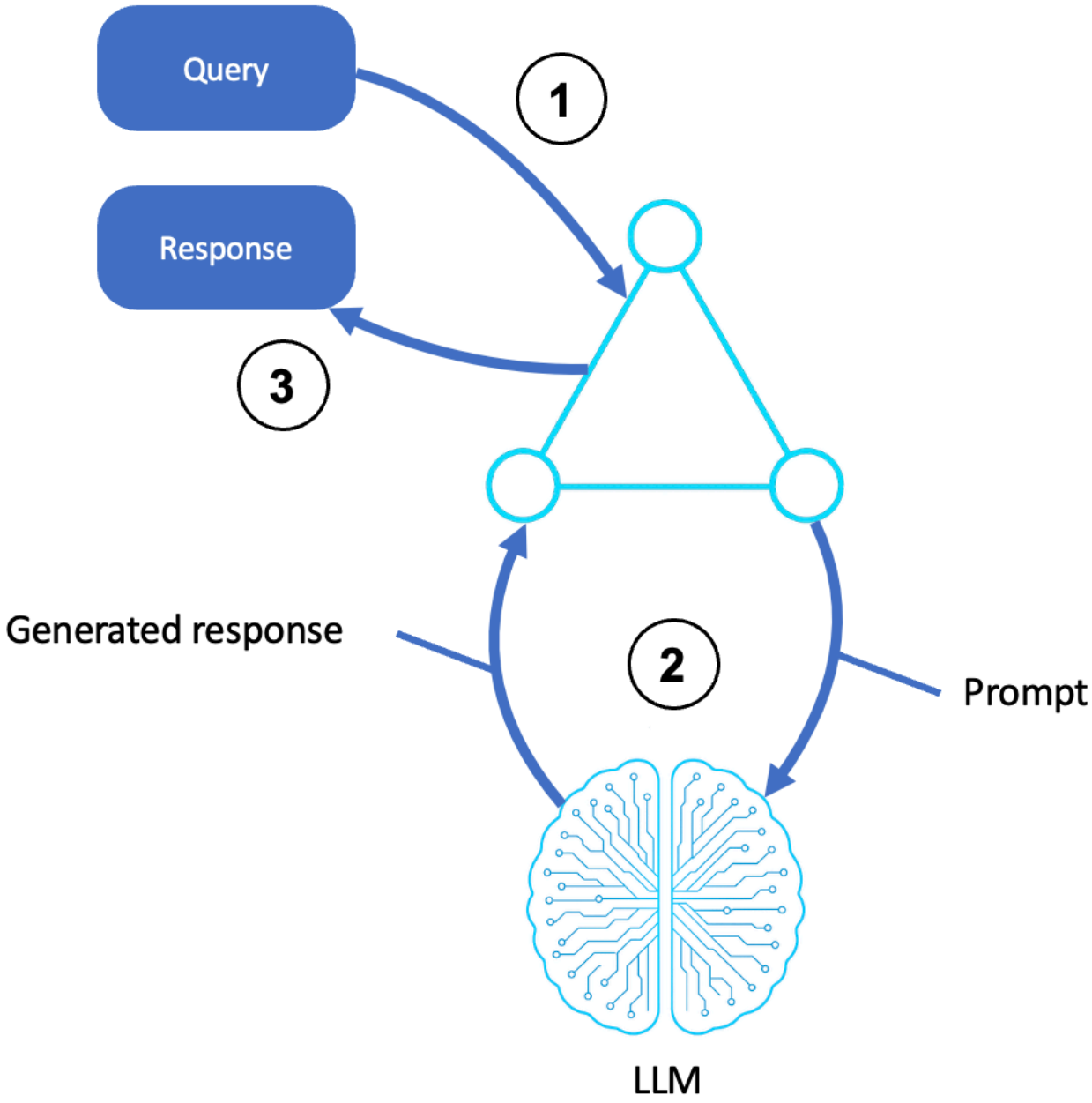
基本的な推論エージェント

基本的な推論エージェントは、クエリに応答して論理推論または意思決定を実行するエージェント AI の最もシンプルな形式です。ユーザーまたはシステムからの入力を受け入れ、クエリを処理し、構造化プロンプトを使用してレスポンスを生成します。

このパターンは、特定のコンテキストに基づいて単一ステップの推論、分類、または要約を必要とするタスクに役立ちます。メモリ、ツール、または状態管理を使用しないため、ステートレスで軽量で、大規模なワークフローで高度に構成できます。

アーキテクチャ

基本的な推論エージェントのフローを次の図に示します。



説明

1. 入力を受け取る
   - ユーザー、システム、またはアップストリームエージェントがクエリまたは命令を送信します。
   - 入力はエージェントシェルまたはオーケストレーションレイヤーに引き渡されます。
   - このステップには、前処理、プロンプトテンプレート作成、目標の特定が含まれます。

2. LLM を呼び出します
   - エージェントはクエリを構造化されたプロンプトに変換し、LLM に送信します (Amazon Bedrock 経由など)。
   - LLM は、事前トレーニング済みの知識とコンテキストを使用して、プロンプトに基づいてレスポンスを生成します。
   - 生成された出力には、推論ステップ (chain-of-thought)、最終回答、またはランク付けされたオプションが含まれます。
3. レスポンスを返します。
   - 生成された出力は、エージェントのインターフェイスに中継されます。
   - これには、フォーマット、後処理、または API レスポンスが含まれます。

機能

- 自然言語または構造化された入力をサポート
- プロンプトエンジニアリングを使用して動作をガイドします
- ステートレスでスケーラブル
- UI、CLI、APIs、パイプラインに埋め込むことができます

制限

- メモリまたは履歴認識がない
- 外部ツールやデータソースとやり取りしない
- 推論時に LLM が知っているものに限定

一般的なユースケース

- 会話の質問と回答
- ポリシーの説明と概要
- 意思決定のためのガイダンス
- 軽量で自動化されたチャットボットフロー
- 分類、ラベル付け、スコアリング

実装のガイダンス

次のツールとサービスを使用して、基本的な推論エージェントを作成できます。

- Amazon Bedrock for LLM 呼び出し (Anthropic、AI21、Meta)
- Amazon API Gateway または AWS Lambda を使用してステートレスマイクロサービスとして公開する
- Parameter Store に保存されているプロンプトテンプレート AWS Secrets Manager、またはコードとして保存されているプロンプトテンプレート

概要

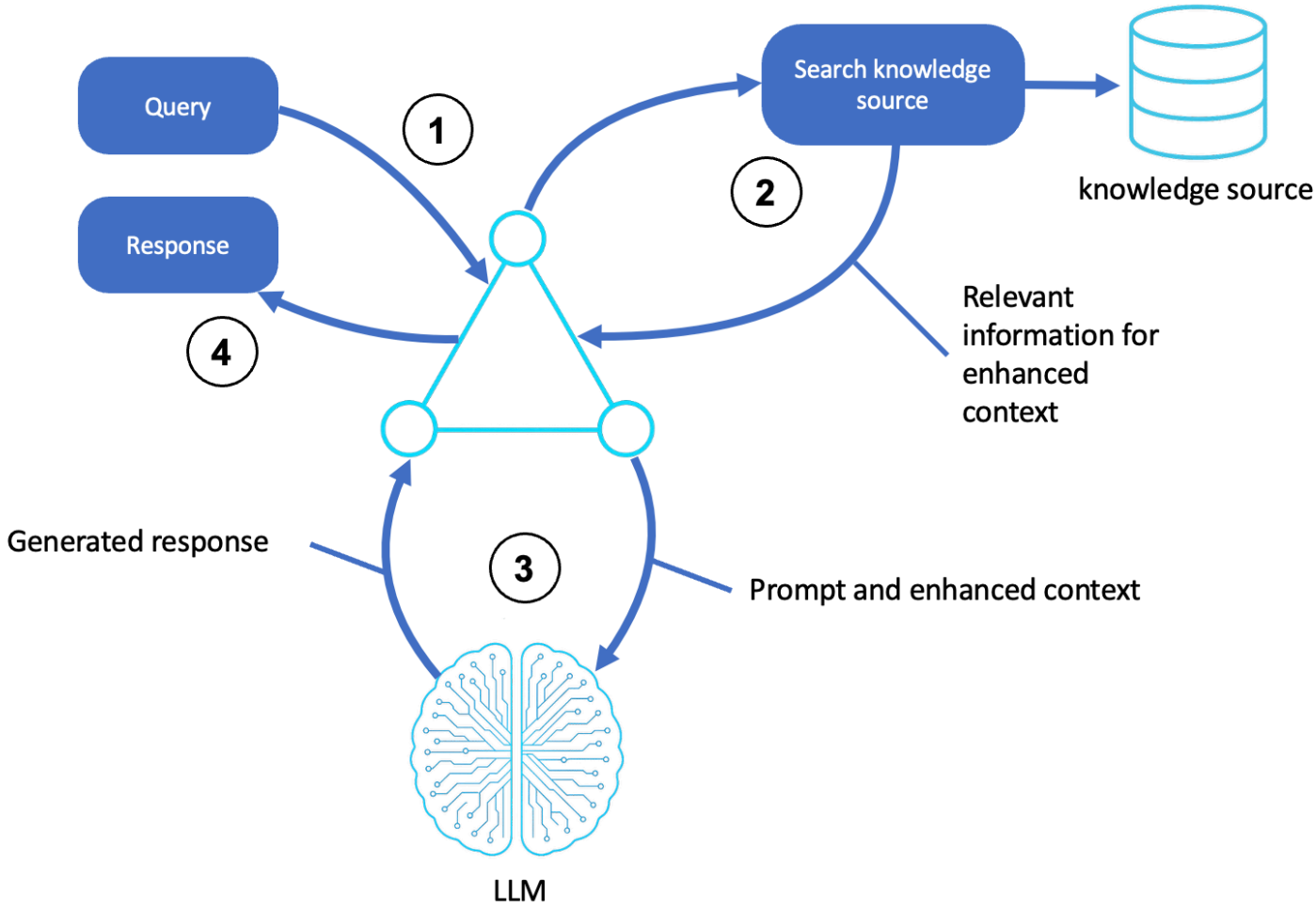
基本的な推論エージェントは、シンプルな構造のため、基本的なものです。目標はインテリジェントな出力につながる推論パスに変換するコア機能を備えています。このパターンは、多くの場合、ツールベースのエージェントや取得拡張生成 (RAG) を使用するエージェントなど、高度なパターンの出発点です。また、大規模なワークフローの信頼性の高いモジュールコンポーネントでもあります。

エージェント RAG

Retrieval-augmented generation (RAG) は、情報の取得とテキスト生成を組み合わせて、正確でコンテキストに応じたレスポンスを作成する手法です。RAG を使用すると、エージェントは LLM をエンゲージする前に関連する外部情報を取得できます。エージェントの決定をup-to-date、事実、またはドメイン固有の情報に基づいて判断することで、エージェントの有効なメモリと推論の精度を拡張します。事前トレーニング済みの重みのみに依存するステートレス LLMs とは対照的に、RAG にはコンテキストを使用してプロンプトを動的に強化する外部ナレッジ検索レイヤーがあります。

アーキテクチャ

RAG パターンのロジックを次の図に示します。



説明

1. クエリを受信する
   - ユーザーまたはアップストリームシステムは、クエリまたは目標をエージェントに送信します。
   - エージェントシェルはリクエストを受け入れ、推論のプロンプトとしてフォーマットします。
2. 外部ソースを検索します。
   - エージェントは、クエリから概念とインテントを識別します。
   - セマンティック検索またはキーワードマッチングを使用して、ベクトルストア、データベース、ドキュメントインデックスなどのナレッジソースをクエリします。
   - 最も関連性の高いパッセージ、ドキュメント、またはエンティティは、次のステップで使用するために取得されます。
3. コンテキストレスポンスを生成します。

- エージェントは、取得した情報でプロンプトを強化し、LLM のコンテキスト拡張入力を形成します。
- LLM は、生成推論 (chain-of-thoughtやリフレクションなど) を使用して入力を処理し、正確なレスポンスを生成します。

4. 最終出力を返します。
   - エージェントは、出力を任意の通信ヘッダーまたは必要な形式にラップして準備し、ユーザーまたは呼び出しシステムに返します。
   - (オプション) 取得したドキュメントと LLM 出力は、今後のクエリのためにログ記録、スコアリング、メモリへの保存が可能です。

機能

- ロングテールドメインやエンタープライズ固有のドメインでも事実に基づく出力
- モデルを微調整しないメモリ拡張
- 各クエリとユーザー状態に基づく動的コンテキスト
- ベクトルデータベース、セマンティックインデックス、メタデータフィルタリングとの完全な互換性

一般的なユースケース

- エンタープライズナレッジアシスタント
- 規制コンプライアンスボット
- カスタマーサポート副操縦士
- 検索が強化されたチャットボット
- 開発者ドキュメントエージェント

実装のガイダンス

次のツールとサービスを使用して、RAG を使用するエージェントを作成します。

- Amazon Bedrock for LLM 呼び出し
- ドキュメントまたは構造化データ検索用の Amazon Kendra、OpenSearch、または Amazon Aurora

- ドキュメントストレージ用の Amazon Simple Storage Service (Amazon S3)
- AWS Lambda 検索、プロンプト、LLM 推論をオーケストレーションする
- エージェントとのナレッジベースの統合 (メモリプラグイン、セマンティックリトリーバー、または Amazon Bedrock を使用)

概要

エージェント RAG は、静的モデル推論を動的な実世界のインテリジェンスに接続します。これにより、エージェントは知らないものを検索し、取得した知識から回答を合成し、信頼性の高い監査可能なレスポンスを生成できます。

RAG パターンは、再トレーニングなしでナレッジアクセスをスケールするインテリジェントなエージェントを構築するための基盤です。多くの場合、ツールの使用、計画、長期メモリを含む、より複雑なオーケストレーションパターンの先駆けとなります。

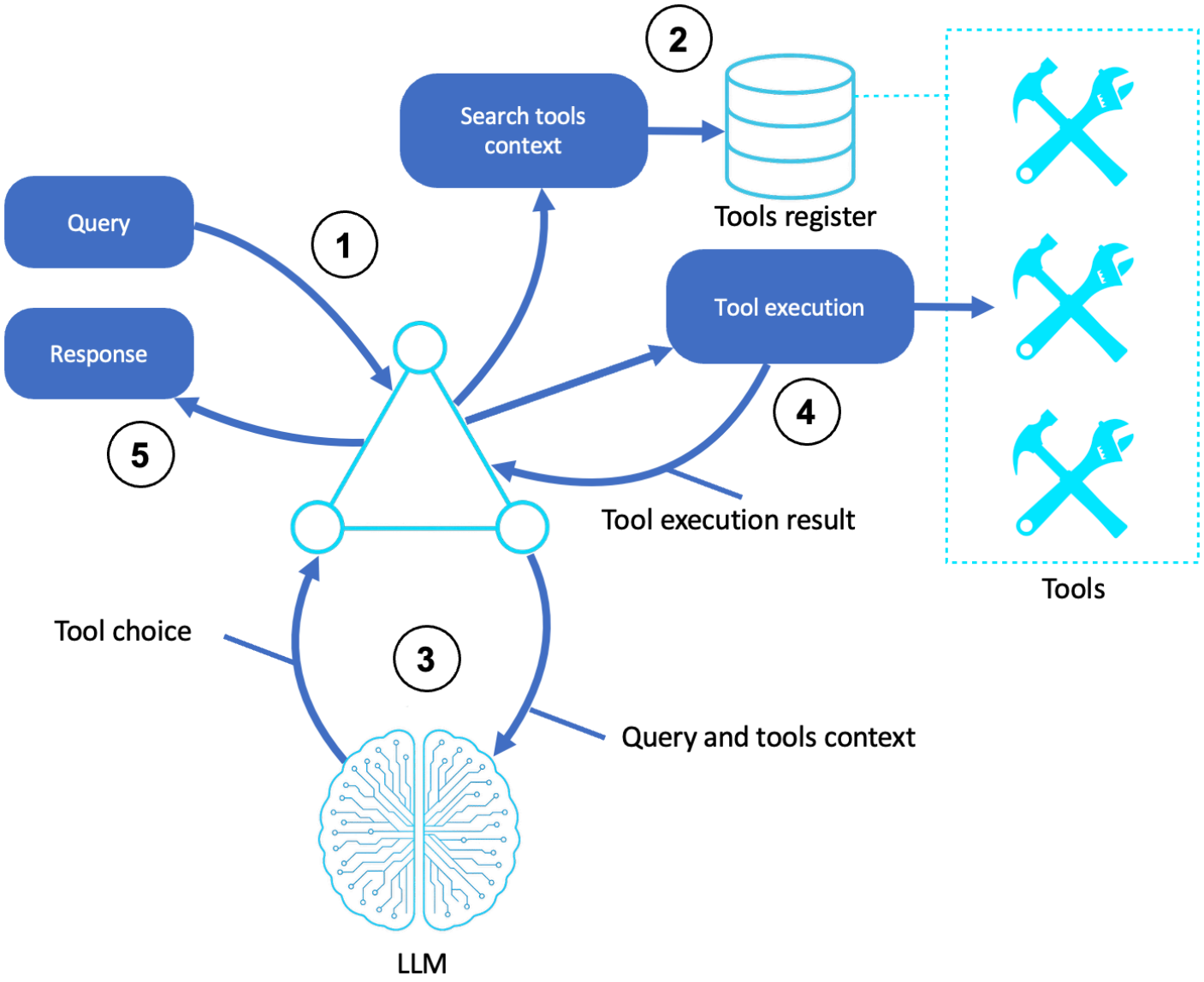
関数を呼び出すためのツールベースのエージェント

ツールベースのエージェントは、外部関数または APIsて、言語のみの推論を超えるタスクを完了することで、推論エージェントの機能を拡張します。このパターンでは、LLM を使用して使用するツールを決定し、呼び出し引数を生成し、ツールの出力を推論ループに組み込みます。

このパターンにより、エージェントは単にレスポンスを提供するのではなく、行動できるようになります。ツールインターフェイスは、算術計算やデータベース検索から外部 APIs やクラウドサービスまで、呼び出し可能な機能を表します。

アーキテクチャ

関数を呼び出すためのツールベースのエージェントを次の図に示します。



説明

1. クエリを受信する
   - エージェントは、ユーザーまたは呼び出しシステムから自然言語クエリまたはタスクを受け取ります。
2. ツールの検索
   - エージェントは内部メタデータまたはツールレジストリを使用して、使用可能なツール、スキーマ、および関連機能を検索します。
3. ツールを選択して呼び出す

- LLM は、クエリとツールのメタデータ (関数名、入力タイプ、説明など) をプロンプトで受け取ります。
- 最も関連性の高いツールを選択し、入力引数を構築し、構造化された関数呼び出しを返します。

4. 選択したツールを実行します。
   - エージェントシェルまたはツールランナーは、選択した関数を実行し、結果 (API 出力、データベース値、計算など) を返します。
5. レスポンスを返します。
   - LLM は、結果をエージェントに直接渡すか、更新されたプロンプトの一部として渡します。次に、自然言語の結果を返します。

機能

- タスクコンテキストに基づく動的ツールの選択
- スキーマベースのプロンプト (OpenAPI、JSON スキーマ、 AWS 関数インターフェイス)
- 結果の解釈と出力の推論への連鎖
- ステートレスまたはセッション対応オペレーション

一般的なユースケース

- 外部データアクセスを持つ仮想アシスタント
- 財務計算ツールと推定ツール
- API ベースのナレッジワーカー
- を呼び出す LLMs AWS Lambda、Amazon SageMaker エンドポイント、SaaS サービス

実装のガイダンス

関数を呼び出すためのツールベースのエージェントを作成するには、以下を使用します。

- 関数呼び出しをサポートする Amazon Bedrock (Anthropic Claude)
- AWS Lambda ツール実行バックエンドとしての
- Amazon API Gateway またはツールオーケストレーション AWS Step Functions 用

- コンテキスト対応ツールメタデータ用の Amazon DynamoDB または Amazon Relational Database Service (Amazon RDS)
- 状態をマッピングして出力をルーティング AWS Step Functions する Amazon EventBridge パイプラインまたは

概要

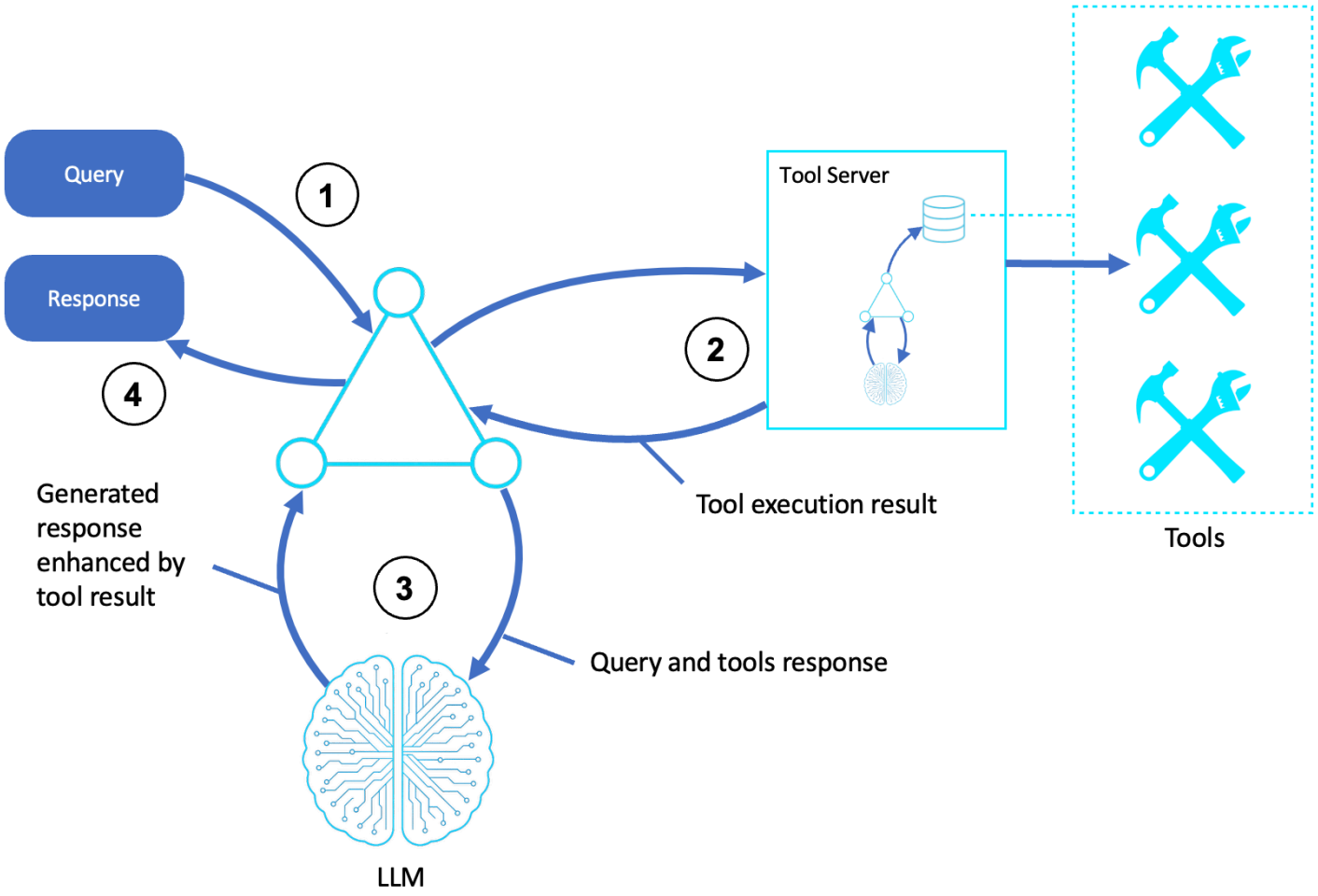
ツールベースの関数呼び出しエージェントは、言語の理解からアクションの実行への移行を表します。これらのエージェントは、LLM 推論を維持し、パッシブアシスタントをタスクを完了し、サービスにアクセスし、ビジネスオペレーションを統合するシステムに変換しながら、動的なコンテキスト対応ツールを呼び出します。このパターンは、特に宣言型スキーマ、認可フレームワーク、マルチエージェントシステムと組み合わせる場合、エンタープライズ設定におけるエージェント AI の重要なコンポーネントです。

サーバー用のツールベースのエージェント

サーバー用のツールベースのエージェントは、ツールの実行を、ツール、スクリプト、複合エージェント専用のランタイム環境を持つ外部サーバーに委任することで、関数呼び出しエージェントを強化します。エージェントループが選択および呼び出すインライン関数呼び出しとは異なり、サーバーベースのエージェントはロジックと実行パイプラインを他のエージェントまたはシステムにアウトソースします。これにより、マルチツールの連鎖、分離された実行、特殊な推論などの高度な機能が提供されます。ツールサーバーは、ツール自体に個別の AI モデル、ビジネスルール、または環境が含まれる可能性がある、複雑でステートフル、またはリソース集約型のアクションに最適です。

アーキテクチャ

サーバー用のツールベースのエージェントのパターンを次に示します。



説明

1. クエリを受信する
   - ユーザーまたはシステムがエージェントシェルにリクエストを送信します。
   - エージェントはクエリを解釈し、ツールサーバーにディスパッチする準備をします。
2. ツールサーバープロセスを実行します。
   - エージェントは、構造化パラメータとともにタスクをツールサーバーに送信します。
   - その後、ツールサーバーは以下を行うことができます。
     - 専用コンピューティングシステム (コンテナ AWS Lambda、Amazon SageMaker など) でスクリプトまたはロジックを実行する
     - LLM 推論で独自のサブエージェントを使用してツールを選択して実行する
     - 依存関係、再試行、またはマルチステップ実行フローを管理する
     - タスクの完了時に結果をプライマリエージェントに出力する

3. ツール出力で LLM 推論を使用する
   - エージェントは LLM を呼び出し、プロンプトの一部として元のクエリとツールサーバーの結果を渡します。
   - LLM は、新しく取得した情報を組み込んだレスポンスを合成します。
4. レスポンスを返します。
   - エージェントは、自然言語または構造化されたレスポンスをユーザーまたは呼び出しシステムに返します。
   - (オプション) 結果はメモリまたは監査ログに保存できます。

機能

- ツールはプライマリエージェント実行ループの外部で呼び出されます
- ツールの実行には、LLM 呼び出し、ロジックチェーン、またはサブエージェントが含まれる場合があります
- エージェントはツールラッパーだけでなく、コントローラーまたはディスパッチャーとして機能します
- ロジックのコンポジビリティ、スケーラビリティ、分離を有効にする

一般的なユースケース

- モデルチェーンのオーケストレーション (LLM、ビジョン、コードの組み合わせなど)
- AI 駆動型オートメーションパイプライン
- スクリプトランナーを使用する DevOps アシスタントエージェント
- 複雑な財務計算、シミュレーション、または最適化エージェント
- マルチモーダルツール (オーディオ、ドキュメント、アクションを組み合わせるなど)

実装のガイダンス

このパターンは、以下を使用して構築できます AWS のサービス。

- Amazon Bedrock (エージェントホストと LLM 推論)
- AWS Lambdaツールサーバーのランタイムとしての 、Amazon ECS AWS Fargate、または Amazon SageMaker エンドポイント

- Amazon API Gateway または AWS App Runner を使用してツールサーバー APIs
- デカップリングされたagent-to-toolメッセージング用の Amazon EventBridge
- AWS Step Functions ツールサーバーでマルチエージェントロジックを作成 AWS AppFabric するための または

概要

サーバーを使用するツールベースのエージェントは、高度にモジュール化され、スケーラブルです。決定ロジックを実行から切り離します。これにより、複雑なアクションや機密性の高いアクションを他のシステムにオフロードしながら、プライマリエージェントを軽量に保つことができます。これは、エンタープライズグレードのエージェント AI、特にガバナンス、オブザーバビリティ、分離、動的構成、またはそれらの任意の組み合わせを必要とする環境で重要です。

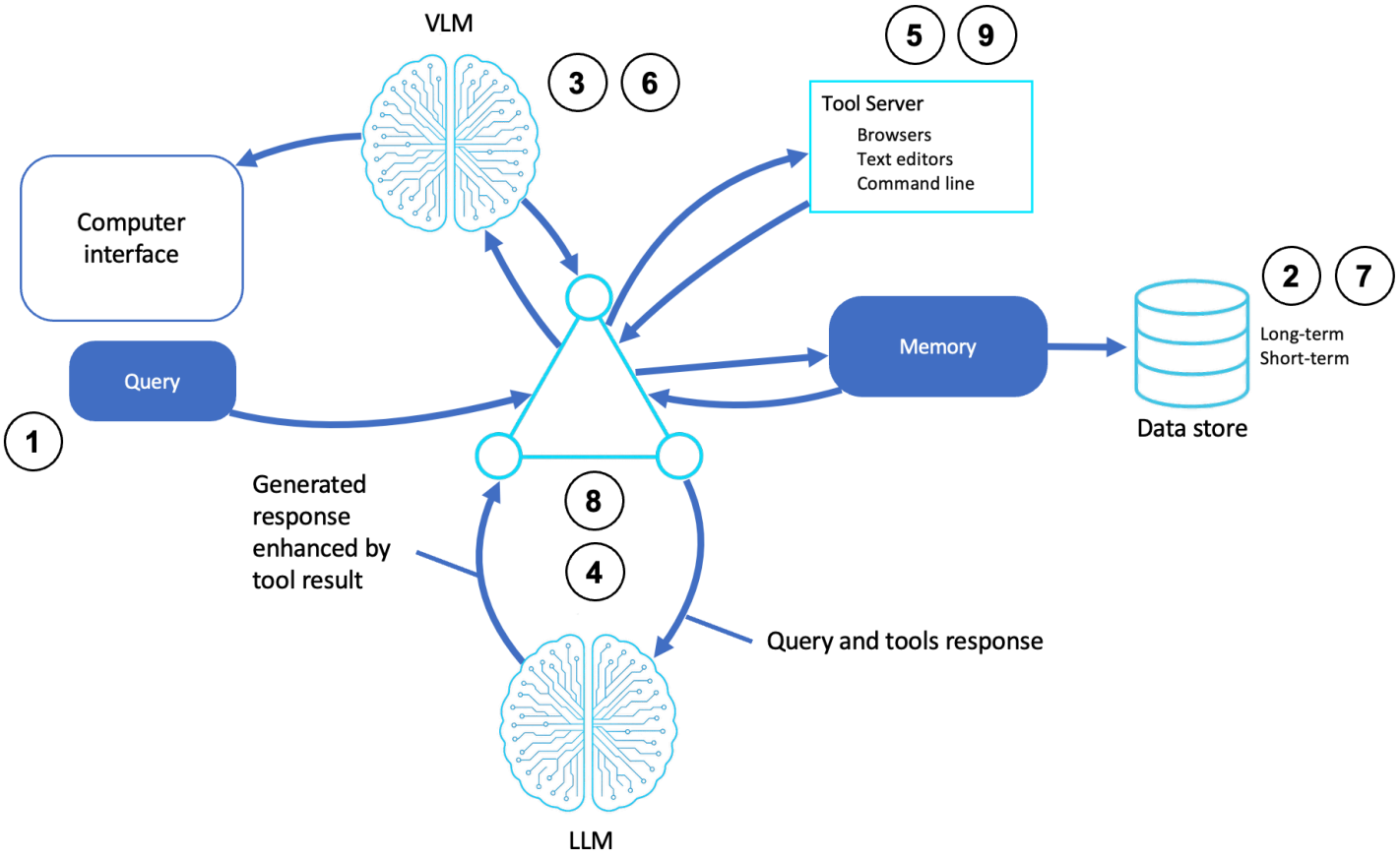
コンピュータ使用エージェント

コンピュータを使用するエージェントは、ブラウザ、ターミナル、ファイルシステム、アプリケーションなどのデジタル環境をシミュレートまたは制御できます。これらのエージェントは、LLM 推論、ビジュアル言語モデル (VLMs)、およびコマンドを実行したり入力イベントをシミュレートするツールサーバーを組み合わせて、ユーザーのインテントを解釈し、ビジュアルインターフェイスとテキストインターフェイスを操作し、目標指向のアクションを実行します。

このパターンは、エージェントがアシスタントとしてだけでなく、人間と同じようにアクションを実行するプロキシとしても機能する実用的な AI オートメーションにとって重要です。多くの場合、同じツールや環境を使用します。

アーキテクチャ

コンピュータ使用のエージェントパターンを次の図に示します。



説明

1. クエリを受信する
   - タスクまたはリクエストは、UI、API、または自然言語インターフェイスを介して提供されます。
2. メモリにアクセスします
   - エージェントは短期および長期のメモリを取得して、過去のコマンド、目標、システムの状態を再現します。
3. ビジュアルコンテキストを分析する
   - VLM は、コンピュータ画面、システム状態、または UI 要素を観察して、特定のコンテキストを理解し、実用的な項目を特定します。
4. LLM 経由の理由
   - LLM は、クエリ、メモリ状態、ツール、サーバーのレスポンスを組み合わせて、次のアクションを決定します。
5. ツールサーバーを操作する

- エージェントは、サーバーでホストされているツールを呼び出します。これには、以下が含まれる場合があります。
  - ブラウザ (ヘッドレス Chrome など) とシェル環境
  - テキストエディタとコードエディタ
  - カスタムスクリプトインターフェイス

6. ビジュアル入力を更新します。
   - システム UI が変更されたり、さらに監視が必要な場合、VLM は画面の状態またはテキストバッファを再分析することがあります。
7. メモリの更新
   - 新しいインサイト、システム状態、またはユーザーフィードバックは、短期および長期のメモリに書き込まれます。
8. 最終的な決定と説明を策定します
   - LLM は、クエリとツールの出力に基づいて結果を合成するか、アクションを推奨します。
9. レスポンスを返します。
   - エージェントは、インターフェイスに結果 (完了したタスク、確認、生成されたコンテンツなど) を返します。

機能

- ビジュアル入力とテキスト入力によるマルチモーダル推論
- シミュレートされた入力または API 駆動型入力によるアプリケーションの制御
- 永続状態のメモリ管理
- シーケンス実行の自律性 (複数ステップフロー)

一般的なユースケース

- IDEs でコードを記述して実行する AI 開発者
- 反復的なデジタルワークフロー用のコンピュータ使用エージェント
- ソフトウェアテストと品質保証のためのシミュレートされたユーザー
- 音声または高レベルの手順で UIs を移動するためのアクセシビリティエージェント
- 推論で強化されたスマートロボットプロセスオートメーション (RPA)

実装のガイダンス

- このパターンは、以下を使用して構築できます AWS のサービス。
- LLM ベースの計画と推論のための Amazon Bedrock
- シミュレートされた UI 環境でツールサーバーを実行する Amazon Elastic Compute Cloud (Amazon EC2) AWS Lambda、または Amazon SageMaker ノートブック
- メモリ永続化のための Amazon Simple Storage Service (Amazon S3) または Amazon DynamoDB
- ハイブリッドシナリオでの UI イメージ分析用の Amazon Rekognition (またはカスタムモデル)
- オブザーバビリティと監査証跡 AWS X-Ray のための Amazon CloudWatch Logs または

概要

コンピュータを使用するエージェントは自律的なデジタルオペレーターとして機能し、人間とコンピュータのやり取りと AI 主導のアクションの間のギャップを埋めます。メモリ、ツールオーケストレーション、VLMs を組み込むことで、これらのエージェントは人間向けに設計されたシステムと適応的にやり取りしたり、アクションを実行したり、ファイルを更新したり、メニューを操作したり、レスポンスを生成したりできます。

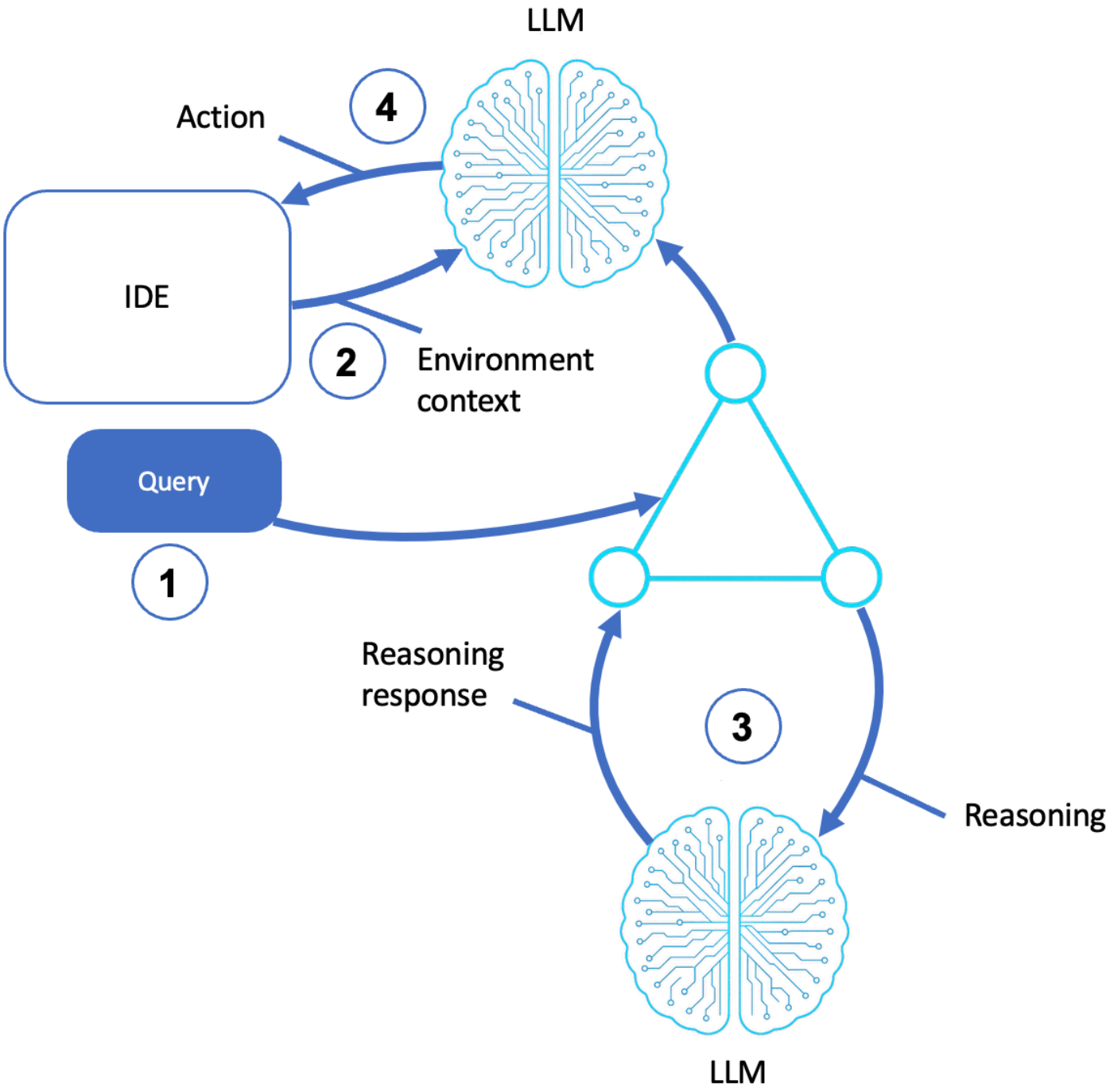
コーディングエージェント

コーディングエージェントは、タスクのプログラミング、コードの生成または変更、IDEs や CLIs。これらのエージェントは、自然言語の理解と構造化された推論を組み合わせて、関数の生成からバグ修正、テストオーサリングまで、ソフトウェア開発を支援、強化、自動化します。

オートコンプリートツールとは異なり、コーディングエージェントはユーザーの目標を積極的に解釈し、開発環境のコンテキストをクエリし (ファイルを開いてエラーを追跡するなど) 、要件を特定し、アクションを提案して実行します。

アーキテクチャ

コーディングエージェントのパターンを次の図に示します。



説明

1. クエリを受信する

- ユーザーは、コマンドパレット、チャットウィンドウ、または CLI を通じて自然言語の手順を提供します (たとえば、「この関数にログを追加する」または「読みやすいようにリファクタリングする」) 。

2. 環境コンテキストを抽出します
   - エージェントは、アクティブなファイル、カーソルの位置、コードスニペット、シンボルテーブルなど、IDE からコンテキストを収集します。
   - エラーメッセージ、テスト結果、および他のエージェントからの出力を出力します。
3. LLM の推論
   - エージェントは、クエリや環境コンテキストを含むプロンプトを LLM に送信します。
     - LLM は推論パスを実行して以下を決定します。
     - 変更する必要があるもの
     - ソリューションを生成する方法
     - リファクタリング、書き換え、またはコーディングのステップ
4. アクションを実行します
   - LLM は出力をエージェントに返し、IDE またはランタイム環境にインポートします。
   - これには、コードの挿入や変更、コメントやドキュメントの生成、ダウンストリームのビルド、テスト、リンティングタスクのトリガーなどが含まれます。

機能

- 高コンテキスト認識 (IDE 状態、カーソル、構文ツリーなど)
- 目標とフィードバックの反復的な推論
- オプションのコード計画とアクションの分離 (例: 最初の理由とアクション)
- 同期デベロッパーワークフローまたは非同期デベロッパーワークフローで動作

一般的なユースケース

- タスクの説明からのコード生成
- コードのリファクタリングと最適化
- テストケースの生成と検証
- エラーの説明とデバッグ
- ドキュメントアシスタント
- ペアプログラミングコパイロット

実装のガイダンス

- このパターンは、次のツールと を使用して構築できます AWS のサービス。
- LLM 駆動型の生成と推論のための Amazon Bedrock
- コーディングの提案と完了のための Amazon Q Developer
- AWS Lambda サンドボックス環境を実行およびテストするための または Amazon Elastic Container Service (Amazon ECS)
- AWS Cloud9、VS Code 拡張機能、またはコンテキストをホストおよび評価するためのカスタム IDE 統合
- 中間プロンプト、レスポンス、リビジョン履歴を保存するための Amazon Simple Storage Service (Amazon S3)

概要

コーディングエージェントは、自然言語の解釈、コンテキストの分析、複数ステップのコード変更の生成、ソフトウェア開発ライフサイクルとの統合が可能な新しい AI を活用した開発ツールです。

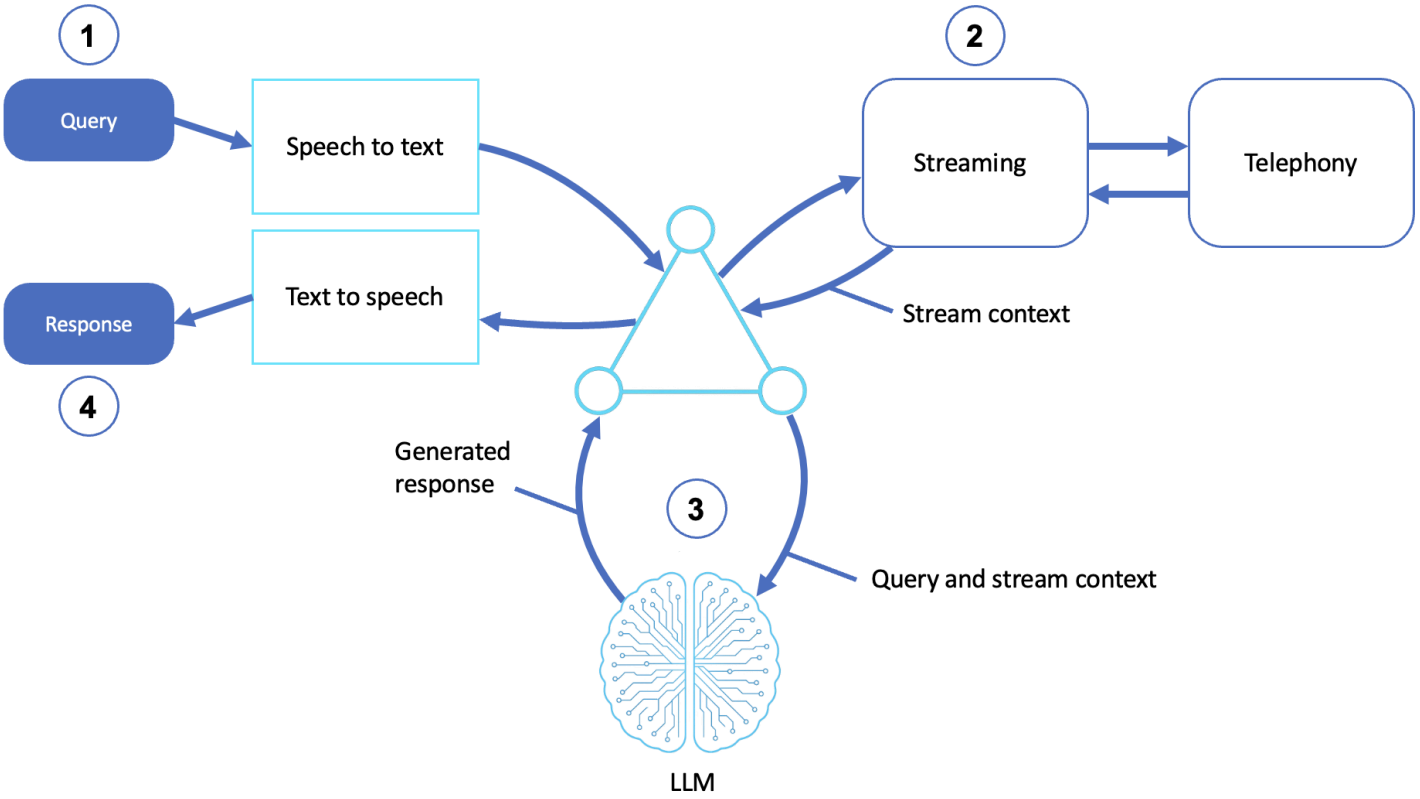
音声エージェント

音声エージェントと音声エージェントは、会話を通じてユーザーとやり取りします。これらのエージェントは、音声認識、自然言語理解、音声合成を統合して、テレフォニー、モバイル、ウェブ、埋め込みプラットフォーム全体で会話 AI を可能にします。

音声エージェントは、ハンドフリー、リアルタイム、またはアクセシビリティ駆動型の環境で特に効果的です。ストリーミングインターフェイスと LLM を活用した推論を組み合わせることで、ユーザーにとって自然なリッチで動的なインタラクションが容易になります。

アーキテクチャ

音声エージェントと音声エージェントを次の図に示します。



説明

1. 音声クエリを受信する
   - ユーザーは電話、マイク、または埋め込みシステムにリクエストを発声します。
   - speech-to-text (STT) モジュールは、音声をテキストに変換します。
2. ストリーミングコンテキストとテレフォニーコンテキストを統合する
   - エージェントはストリーミングインターフェイスを使用して、オーディオ I/O をリアルタイムで管理します。
   - コンタクトセンターまたはテレコムコンテキストにデプロイされている場合、テレフォニー統合はセッションルーティング、デュアルトーンマルチ周波数 (DTMF) 入力、およびメディアトランスポートを処理します。

注: DTMF は、電話キーパッドのボタンを押すと生成されるトーンを指します。音声エージェント内でのストリーミングおよびテレフォニーコンテキスト統合のコンテキストでは、DTMF は、特にインタラクティブ音声応答 (IVR) システムで、通話中の信号入力メカニズムとして使用されます。DTMF 入力により、エージェントは次のことを実行できます。

- メニューの選択を認識します (たとえば、「請求の場合は 1 を押します。 サポートするには 2 を押します。」)
- 数値入力 (アカウント番号、PINs収集する
- コールフローのワークフローまたは状態遷移をトリガーする
- 必要に応じて音声からタッチトーンに戻す

1. LLM ストリームコンテキスト経由の理由
   - クエリはエージェントに送信され、セッションメタデータ (発信者 ID、以前のコンテキストなど) とともに LLM に渡されます。
   - LLM は、インタラクションが進行中の場合は、chain-of-thought戦略またはマルチターンメモリを使用してレスポンスを生成します。
2. 音声レスポンスを返します。
   - エージェントは、テキスト読み上げ (TTS) を使用してtext-to-speechに変換します。
   - 音声チャネルを介してユーザーに音声を返します。

機能

- リアルタイムの音声の理解と生成
- STT と TTS をサポートする多言語 I/O
- テレフォニーまたはストリーミング APIsとの統合
- ターン間のセッション認識とメモリのハンドオフ

一般的なユースケース

- 対話型 IVR システム
- 仮想受付担当者と予約スケジューラ
- 音声駆動型ヘルプデスクエージェント
- ウェアラブル音声アシスタント
- スマートホームとアクセシビリティツール用の音声インターフェイス

実装のガイダンス

このパターンは、次のツールと を使用して構築できます AWS のサービス。

- STT 用の Amazon Lex V2 または Amazon Transcribe
- TTS 用の Amazon Polly
- ストリーミングとテレフォニー用の Amazon Chime SDK、Amazon Connect、または Amazon Interactive Video Service (Amazon IVS)
- Anthropic、AI21、またはその他の基盤モデルによる推論のための Amazon Bedrock
- AWS Lambda STT、LLM、TTS、セッションコンテキストを接続するには

(オプション) 追加の機能強化には以下が含まれます。

- Amazon Kendra または OpenSearch for コンテキスト対応 RAG
- セッションメモリ用の Amazon DynamoDB
- トレーサビリティ AWS X-Ray のための Amazon CloudWatch Logs と

概要

音声エージェントは、自然な会話を通じてやり取りするインテリジェントなシステムです。音声インターフェイスを LLM 推論およびリアルタイムストリーミングインフラストラクチャと統合することで、音声エージェントはシームレスでアクセス可能でスケーラブルなインタラクションを可能にします。

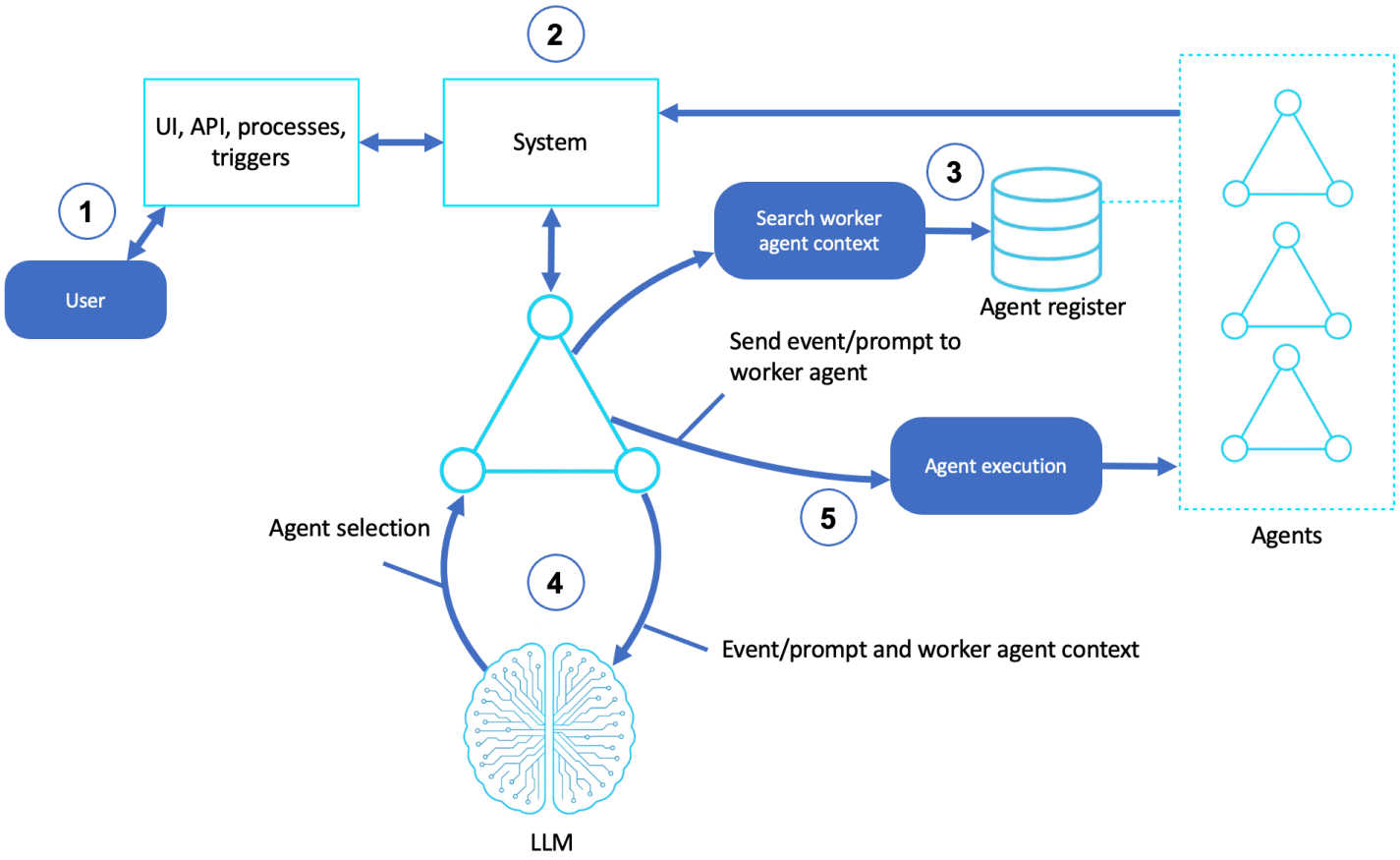
ワークフローオーケストレーションエージェント

ワークフローオーケストレーションエージェントは、分散システム全体の複数ステップのタスク、プロセス、サービスを管理および調整します。これらのエージェントは、推論や単独での行動ではなく、作業をサブエージェントや他のシステムに委任し、実行コンテキストを維持し、中間結果に基づいて適応します。

これらのエージェントは自動化フローの基本的な部分です。これらは、長時間実行されるタスク、マルチエージェント構成、クロスドメイン統合を処理する場合に特に便利です。この場合、さまざまなエージェントとツールを順番に呼び出すか、条件付きで呼び出す必要があります。

アーキテクチャ

ワークフローオーケストレーションエージェントを次の図に示します。



説明

1. ユーザー入力を受信します
   - ユーザー (または外部トリガー) は、UI、API、またはシステムイベントを通じてタスクを開始します。
2. システムイベントを処理します
   - システムコンポーネントはリクエストを受け取り、オーケストレーションを必要とするイベントまたはコマンドを出力します。
3. コンテキストを取得します。
   - ワークフローエージェントはナレッジベースとエージェントレジストリをクエリして、メタデータ、ドメイン、以前の成功率に基づいてタスクに適したワーカーエージェントを見つけます。
4. LLM エージェントを選択します。

- LLM は、タスクの説明と使用可能なオプションを分析することで、最適なエージェントまたはワークフロープランを選択するのに役立ちます。
- また、タスク固有のプロンプトを作成して、選択したエージェントに送信することもできます。

5. 委任と実行
   - 選択したワーカーエージェントはイベントまたはプロンプトを受け取り、コマンドの実行を開始します。
   - 実行状態を追跡し、失敗時に再試行し、シーケンス内の次のエージェントに中間結果を渡すことができます。

機能

- エージェント構成 (スーパーバイザー、共同作業者エージェント、ツールなど)
- イベント駆動型またはスケジュールされた実行
- 時間の経過に伴うメモリと状態の追跡
- 階層型または並列タスクオーケストレーション (非同期ワークフローと比較した同期)
- 動的なエージェントの選択と連鎖

一般的なユースケース

- マルチステップオートメーション (データの取り込みやレポートなど)
- カスタマーサービスのルーティングとエスカレーション (agent-as-coordinatorなど)
- AI エージェントは、同じループ内の人間やボットと連携します。
- LLM を活用したロジックを使用してエンタープライズプロセスを自動化する
- ハイブリッドシステムは AI エージェントと従来のオーケストレーションツールを組み合わせる

実装のガイダンス

このパターンは、次のツールと を使用して構築できます AWS のサービス。

- 推論とエージェント選択のための Amazon Bedrock
- AWS Step Functions ワークフロー構成用の または Amazon EventBridge
- AWS Lambda 実行ユニットまたはタスクランナーとして

- 状態と結果を追跡するための Amazon DynamoDB、Amazon Simple Storage Service (Amazon S3)、または Amazon RDS
- AWS AppFabric または Amazon AppFlow によるクロスシステム調整
- (オプション) Amazon SageMaker 実行エージェントを使用してドメイン固有のワーカーエージェントをホストする

概要

ワークフローエージェントは、マルチエージェント環境で目標を調整、適応、調整します。つまり、AI エージェントは、モジュール式の説明可能なワークフローを通じて、コラボレーション、ランタイム条件への適応、複雑な結果の提供を行うことができます。

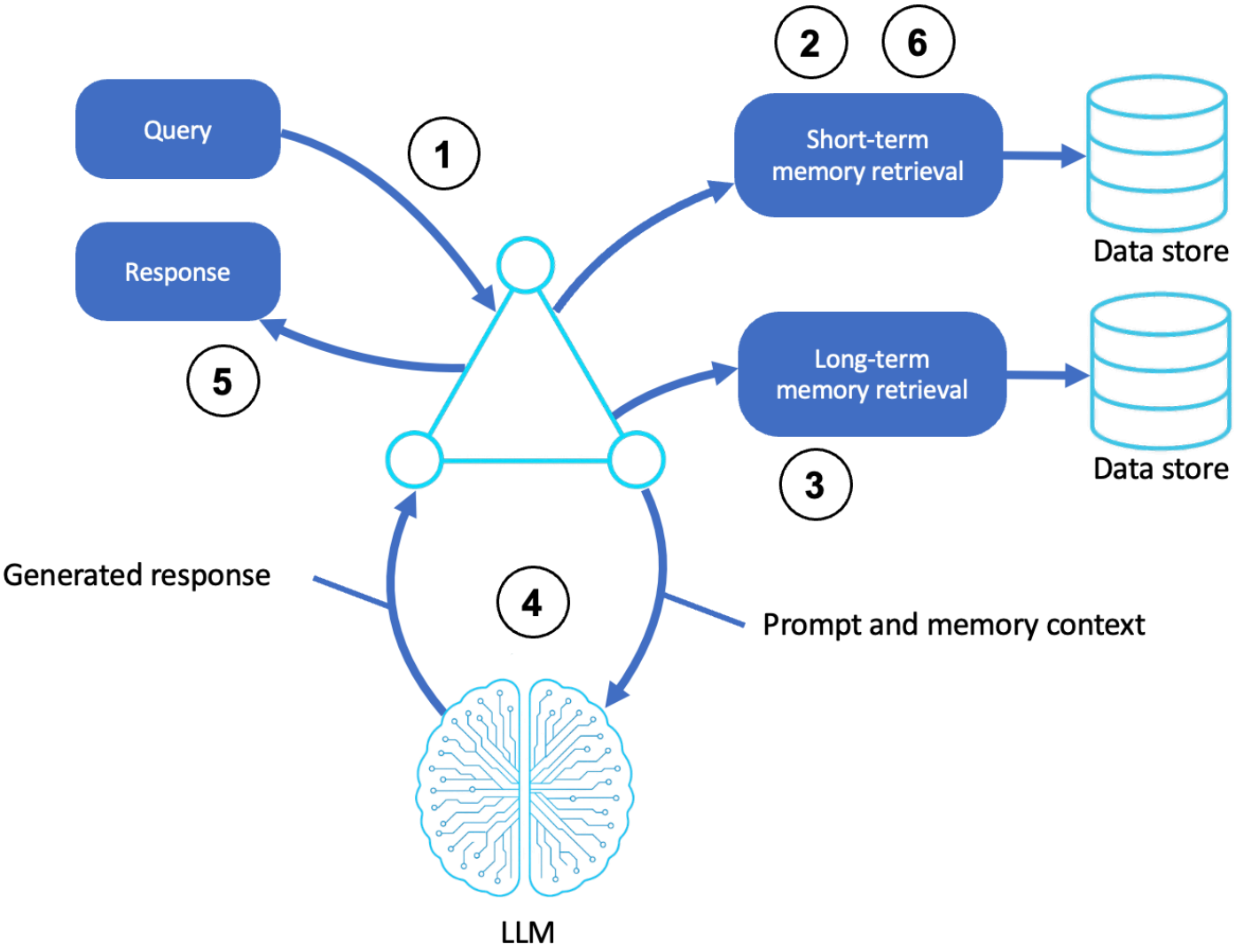
メモリ拡張エージェント

メモリ拡張エージェントは、短期メモリと長期メモリを使用して、保存、取得、および理由の機能が強化されています。これにより、複数のタスク、セッション、インタラクションにわたってコンテキストを維持できるため、より一貫性があり、パーソナライズされ、戦略的に対応することができます。

ステートレスエージェントとは異なり、メモリ拡張エージェントは履歴データを参照して適応し、以前の結果から学び、ユーザーの目標、好み、環境に合った意思決定を行います。

アーキテクチャ

メモリ拡張エージェントを次の図に示します。



説明

1. 入力またはイベントを受信します
   - エージェントはユーザークエリまたはシステムイベントを受け取ります。これは、テキスト、API トリガー、または環境変更です。
2. 短期メモリを取得します。
   - エージェントは、セッションまたはワークフローに関連する最近の会話履歴、タスクコンテキスト、またはシステム状態を取得します。
3. 長期メモリを取得します
   - エージェントは、次のような履歴インサイトについて長期メモリ (ベクトルデータベースやキーバリューストアなど) をクエリします。

- ユーザー設定
- 過去の決定と結果
- 学習した概念、概要、経験

4. LLM 経由の理由
   - メモリコンテキストは LLM プロンプトに埋め込まれているため、エージェントは現在の入力と以前の知識の両方に基づいて推論できます。
5. 出力を生成します
   - エージェントは、タスク履歴とユーザーの入力に応じてパーソナライズされたコンテキスト対応、計画、またはアクションを生成します。
6. メモリの更新
   - 更新された目標、成功と失敗のシグナル、構造化されたレスポンスなどの新しい情報は、将来のタスクのために保存されます。

機能

- 会話またはイベント間のセッション継続性
- 時間の経過に伴う目標の永続性
- 進化する状態に基づくコンテキスト認識
- 以前の成功と失敗から得られる適応性
- ユーザー設定と履歴に沿ったパーソナライゼーション

一般的なユースケース

- ユーザー設定を記憶する会話型副操縦士
- コードベースの変更を追跡するコーディングエージェント
- タスク履歴に応じて適応するワークフローエージェント
- システム知識から進化するデジタルツイン
- 冗長な取り出しを回避する調査エージェント

メモリ拡張エージェントの実装

メモリ拡張エージェント AWS のサービス には、次のツールと を使用します。

メモリレイヤー	AWS のサービス	目的
短期	Amazon DynamoDB、Redis、Amazon Bedrock コンテキスト	最近のインタラクション状態の迅速な取得
長期 (構造化)	Amazon Aurora、Amazon DynamoDB、Amazon Neptune	事実、関係、ログ
長期 (セマンティック)	OpenSearch、Postgre SQL、Pinecone	埋め込みベースの取得 (RAG)
[Storage (ストレージ)]	Amazon S3	トランスクリプト、構造化メモリ、ファイルの保存
オーケストレーション	AWS Lambda or AWS Step Functions	メモリインジェクションと更新ライフサイクルの管理
理由	Amazon Bedrock	Anthropic Claude または Mistral とメモリプロンプト

メモリ注入プロンプトの実装

メモリをエージェントの推論に統合するには、構造化状態と取得拡張コンテキストインジェクションの組み合わせを使用します。

- 言語モデルのプロンプトを構築するときは、最新のエージェント状態と最近の対話履歴を構造化された入力として含めます。これにより、完全なコンテキストで推論できます。
- 検索拡張生成 (RAG) を使用して、関連するドキュメントや事実を長期メモリから取得します。
- 圧縮と関連性に関する以前の計画、コンテキスト、インタラクションを要約します。
- 推論中にベクトルストアや構造化ログなどの外部メモリモジュールを挿入して、意思決定をガイドします。

概要

メモリ拡張エージェントは、経験から学び、ユーザーコンテキストを記憶することで、思考の継続性を維持します。これらのエージェントは、長期的なコラボレーション、パーソナライゼーション、戦略的推論を使用することで、リアクティブインテリジェンスを上回ります。エージェント AI に関して、メモリを使用すると、エージェントは適応型デジタルツールのように動作し、ステートレスツールのように動作しなくなります。

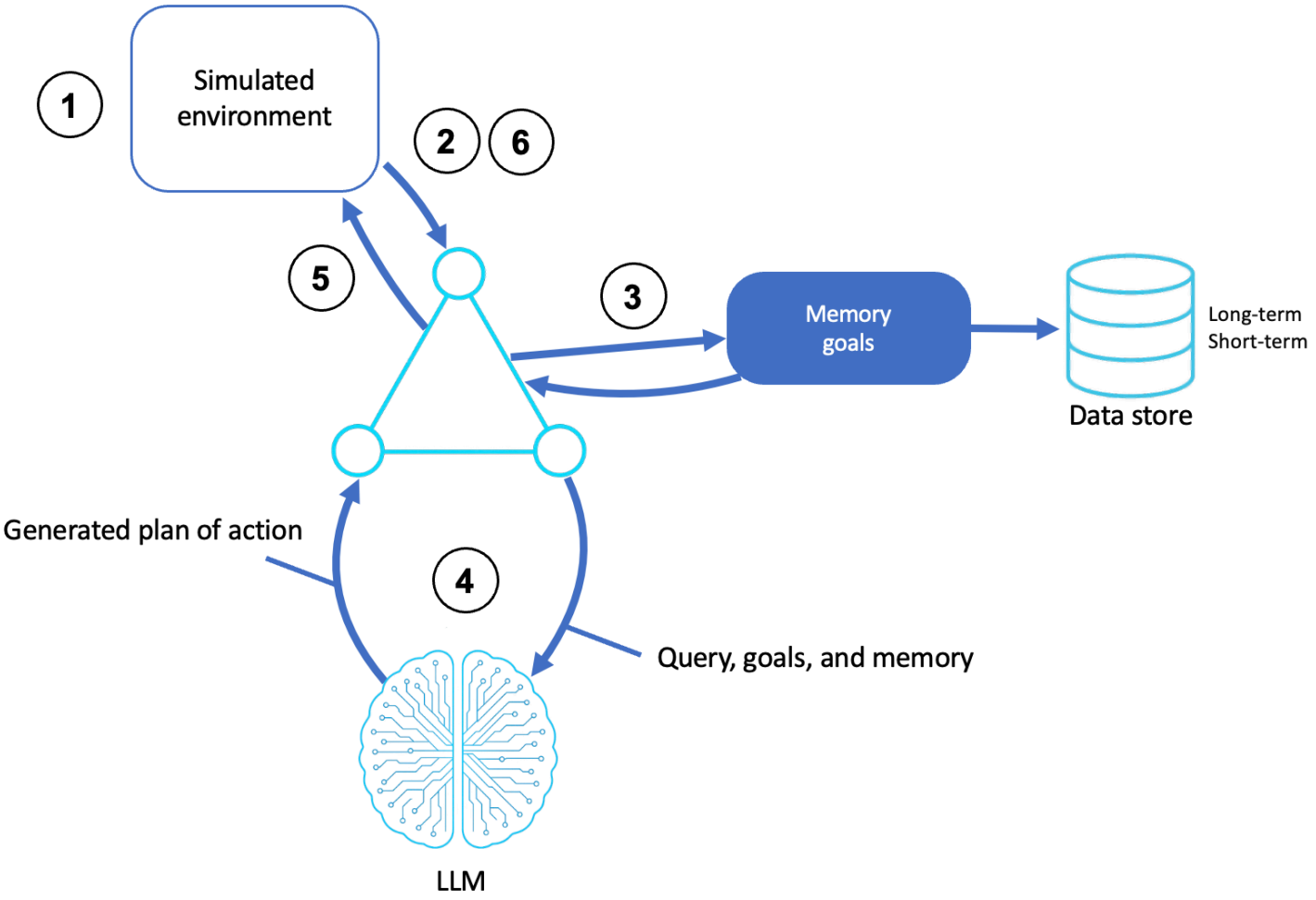
シミュレーションエージェントとテスト床エージェント

シミュレーションエージェントとテスト床エージェントは、理由、行動、学習を行う仮想化環境または制御された環境で動作します。これらのエージェントは、実際の環境に適用する前に、動作、モデルの結果をシミュレートし、繰り返し可能な設定で戦略をトレーニングします。

このパターンは、反復開発、強化学習 (RL)、自律的な意思決定評価、および緊急行動テストに役立ちます。シミュレーションエージェントは多くの場合、クローズドループで動作し、環境からフィードバックを受け取り、それに応じて動作を調整するため、空間推論、リアルタイム制御、複雑なシステムダイナミクスを含むタスクにとって不可欠です。

アーキテクチャ

次の図は、シミュレーションエージェントまたはテスト床エージェントを示しています。



説明

1. 環境を開始する
   - エージェントはシミュレートされた環境 (3D ワールド、物理エンジン、CLI サンドボックス、合成データストリームなど) を開始します。
   - エージェントは、初期タスク、目標、またはポリシーを使用して環境にロードされます。
2. エージェントを認識する
   - エージェントはシミュレーションテレメトリ (センサーエミュレーション、仮想カメラ、構造化ログなど) を通じて現在の状態を認識します。
3. 目標とメモリを取得します。
   - エージェントは、割り当てられた目標、シナリオ手順、またはコンテキスト目標を取得します。
   - また、以下を含む以前のメモリを取得することもできます。

    - 長期戦略またはポリシー
    - 環境マップまたは既知の制約
    - 同様のシミュレーションの過去の成功または失敗
4. 理由と計画
    - LLM は、シミュレートされた状態、タスク目標、学習した知識を解釈します。
    - アクションプランまたはコントロールコマンドを生成します。
5. シミュレートされたアクションを実行します
    - エージェントは、プランの実行、状態の変更、スペースのナビゲート、仮想エンティティとのやり取りを行います。
6. 学習
    - エージェントはアクションの結果を評価します
    - エージェントの設定によっては、以下を行う場合があります。
        - RL を実行する
        - 将来のファインチューニングのログ結果
        - リアルタイムで戦略を適応させる

機能

- 合成環境または仮想環境内で動作
- trial-and-error学習、ポリシーの改良、システムモデリングをサポート
- 動作、障害処理、エッジケースの低リスクテスト
- マルチエージェント設定で緊急エージェント動作分析を有効にする
- クローズドループ制御とhuman-in-the-loop探索の両方をサポート

一般的なユースケース

- ロボット、ドローン、ゲームの強化学習
- 仮想道路での自動車両トレーニング
- DevOps およびテスト床シナリオ用のシミュレートされた UIs または CLIs
- ソーシャルシミュレーションにおける緊急の動作実験
- 本番稼働前の決定ロジックの安全性検証

実装のガイダンス

次のツールと を使用して、シミュレーションエージェントとテスト床エージェントを構築できます AWS のサービス。

コンポーネント	AWS のサービス	目的
環境	Amazon SageMaker スタジオラボの Amazon ECS、Amazon Amazon EC2、またはカスタムシミュレーター	仮想ワールド (Gazebo、Unity、Unreal) またはサンドボックス CLIs を実行する
エージェントロジック	Amazon Bedrock、Amazon SageMaker、または AWS Lambda	LLM ベースのプランナーまたは RL エージェント
フィードバックループ	Amazon SageMaker 強化学習、Amazon CloudWatch、またはカスタムログ	報酬の追跡、結果のスコアリング、動作のログ記録
メモリとリプレイ	Amazon S3、Amazon DynamoDB、または Amazon RDS	永続的な状態、エピソード履歴、またはシナリオデータ
視覚的表現	Amazon CloudWatch ダッシュボードまたは Amazon SageMaker ノートブック	ポリシーの変更、結果、トレーニングメトリクスを観察する

追加のアプリケーションは次のとおりです。

- AWS SimSpace Weaver 大規模な空間シミュレーション用
- AWS IoT Core シャドウデバイスをテストするための
- エージェントの評価とベンチマークのための Amazon SageMaker Experiments

概要

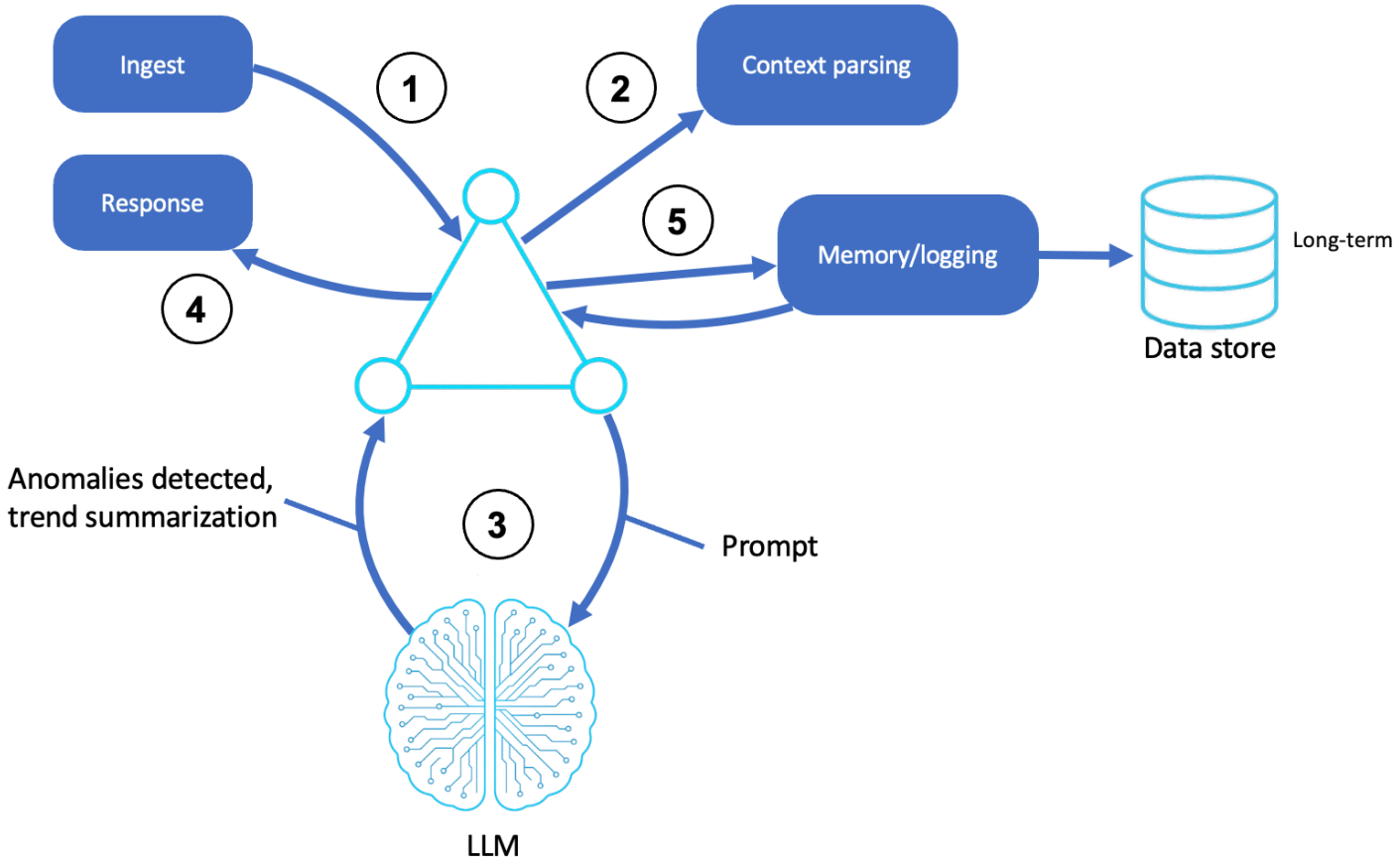
シミュレーションエージェントとテスト床エージェントは、実稼働システムにデプロイされる前に構造化された探索用です。これらのエージェントを使用して、自律ナビゲーションポリシーのトレーニング、合成環境でのビジネスプロセスのテスト、調整パターンのスワームの評価を行います。

オブザーバーエージェントとモニタリングエージェント

オブザーバーエージェントとモニタリングエージェントは、システム、環境、インタラクションをパッシブに監視して、パターンの検出、インサイトの生成、アクションのトリガーを行います。インテリジェントな監視者として、動作を直接開始することなく、アラート、診断、監査を強化します。

これらのエージェントは、従来のモニタリングに適応性や推論がない場合、特に AI-in-the-loopモニタリング、異常検出、コンプライアンス監視、セキュリティインテリジェンスに優れています。オブザーバーエージェントは、システムテレメトリとユーザーインタラクションを継続的にモニタリングするイベントリスナーです。エージェントは、認識、解釈、および条件付きエスカレーションまたは報告に依存します。

アーキテクチャ



説明

1. テレメトリの取り込み
   - エージェントは、次のような 1 つ以上のシステムソースから入力を受け取ります。
     - ログ (アプリケーション、インフラストラクチャ、セキュリティ)
     - メトリクス (パフォーマンス、レイテンシー、使用状況)
     - イベント (API コール、ユーザーアクション、センサーデータ)
2. コンテキストを解析する
   - Raw 入力は、タイムスタンプ、アクターアイデンティティ、システム状態、トレース ID などのメタデータで解析、構造化、強化されます。
3. LLM を使用する理由
   - エージェントは LLM またはロジックモジュールを使用して、異常を特定し、傾向を要約し、分散トレースまたは時間枠間で相関させることで、解析された入力を解釈します。
4. 分類またはアラート

- エージェントは、観察された動作が以下を必要とするかどうかを判断します。
  - アラートまたはエスカレーション
  - レポートまたはダッシュボードの更新
  - レスポンストリガー (自動修復やポリシーの適用など)

5. ログメモリまたはフィードバックループ
   - エージェントは、長期学習、監査、または他のエージェントの将来の参照に関するイベントと決定を保存します。

機能

- パッシブおよび非侵襲的 (エージェントは直接動作しません)
- 高度にスケーラブルで非同期
- ノイズの多いシグナルまたは分散シグナル間の AI 駆動型の相関関係
- 監査、コンプライアンス、リアルタイムインサイトをサポート
- ダウンストリームエージェントまたは人間のワークフローにフィードできる

一般的なユースケース

- マイクロサービスと APIs の AI 拡張オブザーバビリティ
- モデルドリフト、ポリシー違反、またはout-of-band動作のモニタリング
- 顧客アクティビティ分析またはインタラクションの概要
- コミットまたはデプロイをモニタリングするコードレビューエージェント
- LLM 推論を使用したセキュリティまたはコンプライアンスログのモニタリング

実装のガイダンス

オブザーバーとモニタリングエージェントは、次のツールと を使用して構築できます AWS のサービス。

コンポーネント	AWS のサービス	目的
イベントの取り込み	Amazon EventBridge、Amazon CloudWatch	構造化テレメトリと非構造化テレメトリを取り込む

	Logs、Amazon Kinesis、Amazon S3	
前処理	AWS Lambda, AWS Glue, AWS Step Functions	raw データを構造化プロンプトに変換する
推論エンジン	Amazon Bedrock、Amazon SageMaker、 AWS Lambda	イベントの分析、動作の分類、インサイトの生成
ストレージとメモリ	Amazon S3、Amazon DynamoDB、OpenSearch	永続的な観測値、概要、出力
アラートとエスカレーション	Amazon SNS、 AWS AppFabric、Amazon EventBridge	ダウンストリームシステムまたはエージェントのトリガー

追加のアプリケーションは次のとおりです。

- AWS Security Hub CSPM セキュリティログモニタリング用
- エージェント出力を視覚化するための Amazon Quick Suite

概要

オブザーバーエージェントとモニタリングエージェントは、システムや動作をリアルタイムで追跡します。人間やルールが見逃す可能性のあるパターンを特定することで、異常を検出し、セキュリティを監査し、運用インテリジェンスを収集します。この機能は、変化する条件に適応し、包括的なデータ分析に基づいて意思決定できるシステムを作成するのに役立ちます。

マルチエージェントコラボレーション

マルチエージェントコラボレーションとは、それぞれが異なるロール、専門分野、または目標を持つ複数の自律型エージェントが、複雑なタスクを解決するために交渉するパターンを指します。これらのエージェントは、情報を共有し、責任を分割し、目標に向かって集合的に推論することで、独立して、または他のエージェントと運用できます。

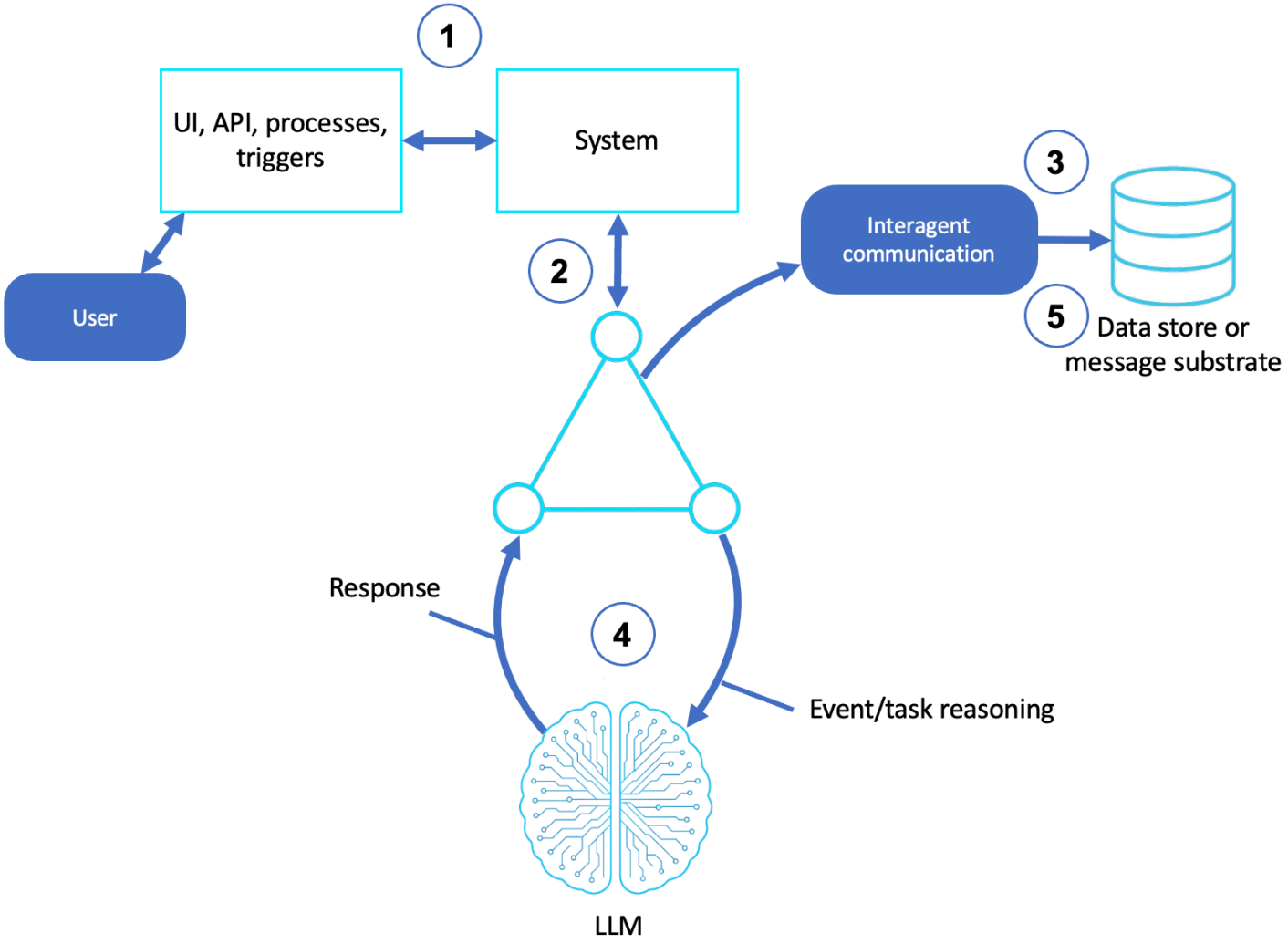
このパターンは、構造化フローの下位エージェントにタスクを一元的に調整して委任するワークフローエージェントとは異なります。対照的に、マルチエージェントコラボレーションは、適応性、並

列性、認識の分割を可能にすることで、peer-to-peerまたは緊急の調整を強調します。次の表は、マルチエージェントコラボレーションとワークフローエージェントを比較したものです。

機能	ワークフローエージェント	目的
コントロール	一元化されたコーディネーター	分散型、分散型、またはロールベースのピア
インタラクション	1 人のエージェントが実行を委任して追跡する	複数のエージェントが交渉、共有、適応する
設計	タスクの事前定義されたシーケンス	エマージェントで柔軟なタスクディストリビューション
調整	手続き型オーケストレーション	協力的または競争的なやり取り
ユースケース	エンタープライズプロセスの自動化	複雑な推論、探索、緊急戦略

アーキテクチャ

次の図は、マルチエージェントコラボレーションを示しています。



説明

1. タスクを開始する
   - ユーザーまたはシステムは、高レベルの目標または問題を発行します。
   - 「マネージャー」エージェントまたは開始コンテキストは、目標を定義します。
2. ロールの割り当てまたは検出
   - エージェントは、プランナー、研究者、エグゼキューター、批評家、説明者などの他のロールに自己割り当て (シンボリックロジックまたは推論) または委任 (イベントブローカー) されます。
3. 他のエージェントと通信する
   - エージェントは、共有メモリ、メッセージングキュー、またはプロンプトチェーンを介して通信します。
   - 相互にサブタスクの議論、クエリ、提案を行う場合があります。

4. 特殊な推論を使用する
    - 各エージェントは、独自のモデルまたはドメインロジックを使用して、問題の一部を解決します。
    - エージェントは、ロール固有のプロンプトとメモリで LLMs を使用できます。
5. 出力または目標を調整する
    - エージェントは、コントリビューションを最終的な回答、計画、またはアクションに合成します。
    - (オプション) 監視エージェントは、合成された出力を検証または要約できます。

機能

- 特殊なロールまたはスキルを持つピアレベルのエージェント
- コミュニケーションまたはネゴシエーションによる緊急の動作
- 複雑または多面的な問題の並列処理
- 熟考、自己修正、リフレクションの反復をサポート
- ソーシャルダイナミクス、科学コラボレーション、またはエンタープライズチームの役割をモデル化する

一般的なユースケース

- 自律型研究チーム (検索エージェント、要約者、検証者)
- ソフトウェア開発 (プランナー、コーダー、テスター)
- ビジネスシナリオモデリング (財務、ポリシー、コンプライアンス)
- 交渉、入札、またはマルチパーティーの推論
- マルチモーダルタスク (イメージ、テキスト、ロジック)

実装のガイダンス

次のツールと を使用して、マルチエージェントシステムを構築できます AWS のサービス。

コンポーネント	AWS のサービス	目的

エージェントホスティング	Amazon Bedrock、Amazon SageMaker、 AWS Lambda	個々の LLM 駆動型エージェントをホストする
通信レイヤー	Amazon SQS、Amazon EventBridge、 AWS AppFabric	エージェント間のメッセージングと調整
共有メモリ	Amazon DynamoDB、Amazon S3、または OpenSearch	マルチエージェントメモリまたはブラックボードシステム
オーケストレーションレイヤー	AWS Step Functions、 AWS Lambda パイプライン	キックオフ、タイムアウト、フォールバック、再試行ロジック
エージェント ID	Amazon Bedrock エージェント (ロール定義) AWS AppConfig 、および Amazon Bedrock converse API (Amazon Bedrock 外のエージェント)	ロールベースのツールまたはエージェントの呼び出しと境界の適用
緊急時のやり取り	Amazon EventBridge パイプラインまたはエージェントレジストリ	動的タスクルーティングまたはエスカレーションを有効にする

概要

マルチエージェントコラボレーションは、問題解決タスクをモジュール式のロール駆動型エージェントに分散します。ワークフローオーケストレーションとは異なり、コラボレーションパターンは、人間が問題を解決する方法を反映する緊急のインテリジェンス、レジリエンス、スケーラビリティを使用します。これは、オープンエンドドメイン、クリエイティブタスク、マルチモーダル推論、および多様な視点から恩恵を受ける環境にとって特に重要です。

結論

前述のパターンは、エージェント AI の実際の実装に対する基本的なアプローチを示しています。基本的な推論からメモリ拡張インテリジェンスまで、各パターンは、自律性、非同期性、および機関に基づく認識、認識、およびアクションに対して一意に設定されます。

これらのパターンは、インテリジェントで目標指向のシステムを構築するための語彙と技術的な設計図を共有します。パターンがユーザーインターフェイスに埋め込まれているか、クラウドサービスを介してオーケストレーションされているか、エージェントのチーム間で調整されているかにかかわらず、各パターンは適応可能でモジュール式です。

重要事項

- エージェントパターンは構成可能 – ほとんどの実世界のエージェントは、2 つ以上のパターン (たとえば、ツールベースの推論とメモリを持つ音声エージェント) をブレンドします。
- エージェント設計はコンテキストに基づく – インタラクションサーフェス、タスクの複雑さ、レイテンシー耐性、ドメイン固有の制約に基づいてパターンを選択します。
- AWS ネイティブ実装を実現可能 – Amazon Bedrock、Amazon SageMaker AWS Lambda AWS Step Functions、およびイベント駆動型アーキテクチャでは、すべてのエージェントパターンを大規模に配信できます。

LLM ワークフロー

エージェントパターンでは、認識、アクション、学習、認識の一連のモジュラー機能を中心に構築された一般的な AI エージェントパターンを調べました。多くのエージェントパターンの認知モジュールの中心には、推論、計画、意思決定が可能な大規模言語モデル (LLM) があります。ただし、LLM のみを呼び出すだけでは、インテリジェントな目標指向の動作を生成するには不十分です。

複雑なタスクを確実に実行するには、エージェントは構造化ワークフロー内に LLM を埋め込む必要があります。構造化ワークフローでは、モデルの機能がツール、メモリ、計画ループ、調整ロジックで強化されます。これらの LLM ワークフローにより、エージェントは目標を分割し、サブタスクをルーティングし、外部サービスを呼び出し、結果を反映し、他のエージェントと調整できます。

この章では、再利用可能なワークフローを中心に整理された、堅牢で拡張可能なインテリジェントな LLM 駆動型コグニティブモジュールを構築するためのコア設計パターンについて説明します。

このセクションの内容

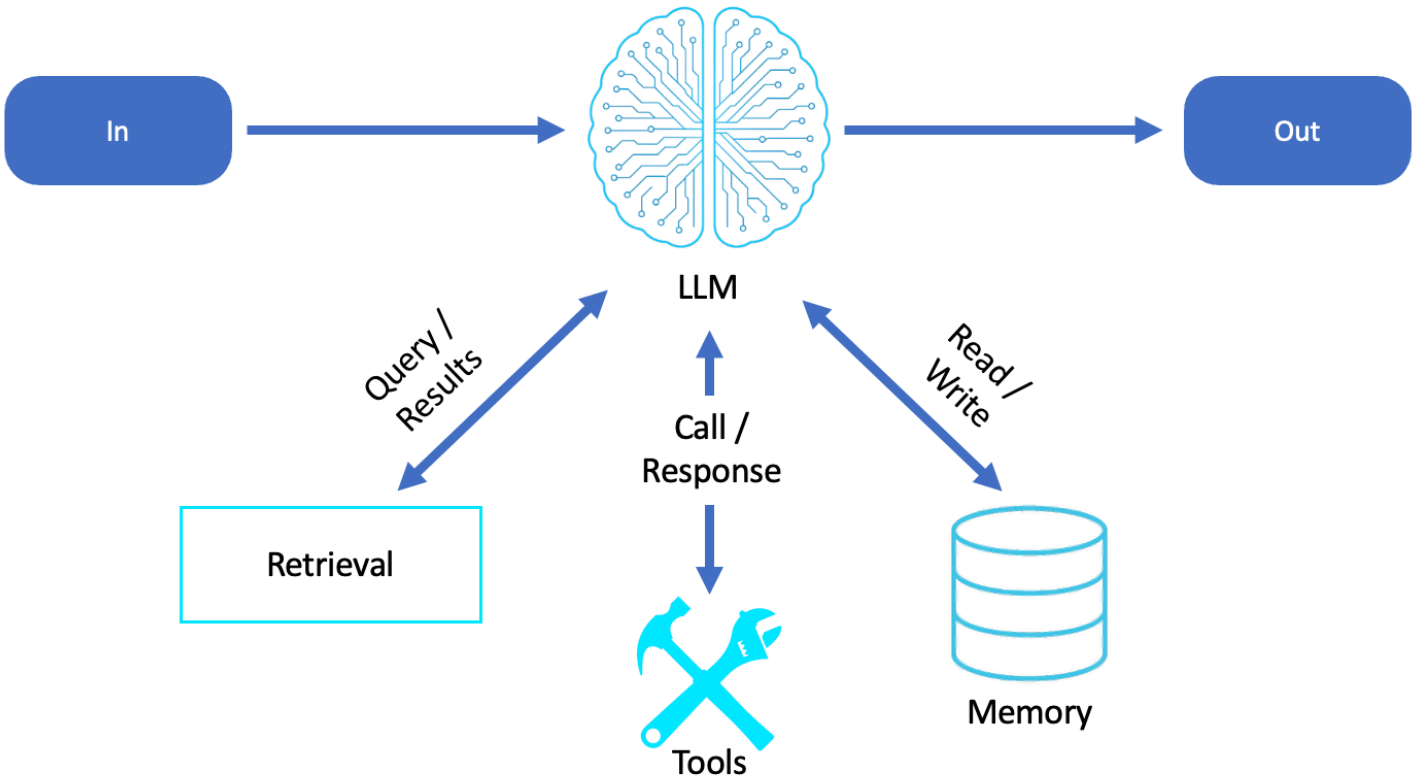
LLM で拡張された認識の概要

その中核として、ソフトウェアエージェントのコグニティブモジュールは、拡張でラップされた LLM として見ることができます。エージェントは、次の構成要素を使用して、環境内で効果的に推論できます。

- プロンプト — コンテキスト、手順、例、メモリを使用した入力のフレーミング
- 取得 – ベクトル検索やセマンティックメモリなど、取得拡張生成 (RAG) を通じて、LLM プロンプトup-to-date知識やドメイン固有の知識を提供します。
- ツールの使用 – LLM が APIsか、関数を呼び出して情報を取得または処理できるようにする

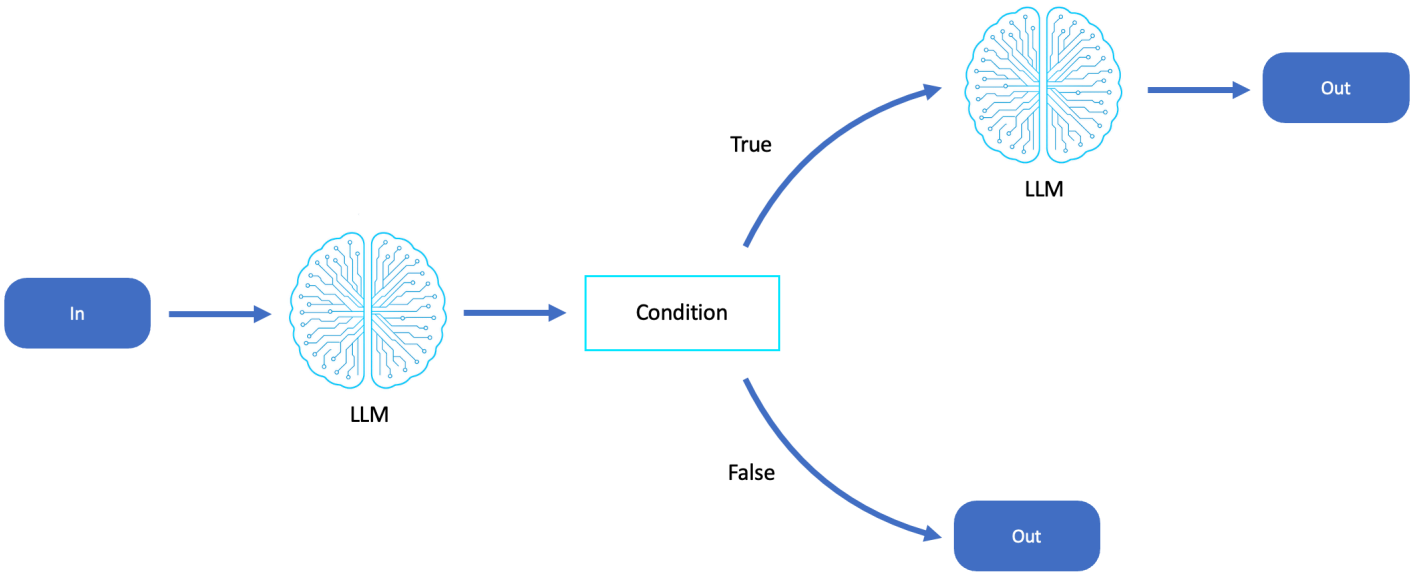
- メモリ – 構造化データベースまたはコンテキスト概要を使用して、永続的またはセッションベースの状態を推論ループに組み込む

これらの拡張は、LLM が時間の経過とともに、タスク間でどのように使用されるかを定義するワークフローで構成され、ステートレスエンジンから動的推論エージェントに変換されます。



プロンプト連鎖のワークフロー

プロンプト連鎖は、複雑なタスクを一連のステップに分解します。各ステップは、前のステップの出力を処理または構築する個別の LLM 呼び出しです。



プロンプト連鎖ワークフローは、タスクを論理的にシーケンシャル推論ステップに分割でき、中間出力が次のステージを通知するシナリオに適しています。ドキュメントレビュー、コード生成、ナレッジ抽出、コンテンツ絞り込みなど、構造化された思考、プログレッシブトランスフォーメーション、またはレイヤード分析を必要とするワークフローに優れています。

説明

- タスクの複雑さが、1 回の LLM 呼び出しのコンテキストウィンドウまたは推論深度を超えています。
- 1 つのステップ (分析、要約、計画など) からの出力は、フォローアップの決定または生成フェーズの入力になります。
- 推論段階 (監査可能な中間結果など) 全体で透明性と制御が必要です。
- ステップ間で外部検証、フィルタリング、またはエンリッチメントロジックをプラグインする場合。
- これは、研究エージェント、編集アシスタント、計画システム、マルチステージ副操縦士など、パイプラインスタイルの推論ループで動作するエージェントに最適です。

機能

- LLM 呼び出しの線形チェーンまたは分岐チェーン
- 構造化入力として渡されるか、フォローアッププロンプトに埋め込まれた中間結果

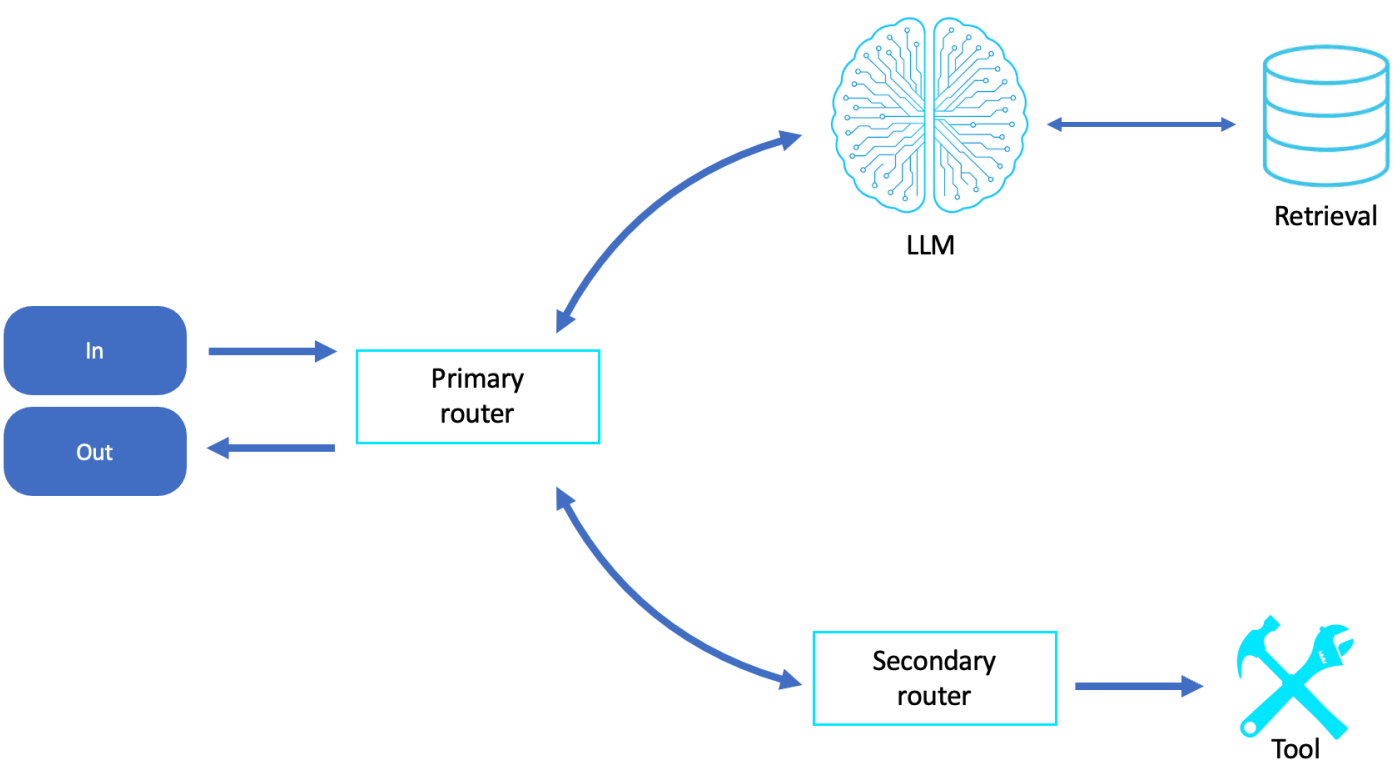
- AWS Step Functions AWS Lambda、、またはエージェント固有のランナーとオーケストレーション可能

一般的なユースケース

- マルチステップ推論タスク (「批評書き換えの概要」など)
- 調査アシスタントがレイヤード出力を合成する (「検索抽出ファクトの回答の質問」など)
- コード生成パイプライン (「計画書き込みコードテストコードの説明出力の生成」)

ルーティングのワークフロー

ルーティングパターンでは、分類子またはルーターエージェントは LLM を使用してクエリのインテントまたはカテゴリを解釈し、入力を特殊なダウンストリームタスクまたはエージェントにルーティングします。



ルーティングワークフローは、エージェントが入力インテント、タスクタイプ、またはドメインをすばやく分類し、リクエストを特殊なサブエージェント、ツール、またはワークフローに委任する必要があるシナリオで使用されます。これは、一般アシスタント、エンタープライズ機能へのフロントド

ア、ドメインにまたがるユーザー向け AI インターフェイスなどの機能エージェントに特に役立ちます。

ルーティングは、次の場合に特に効果的です。

- さまざまなタスク (検索、要約、予約、計算など) にわたるリクエストのトリアージ。
- 入力は、より特殊なワークフローに入る前に前処理または正規化する必要があります。
- 入力タイプ (画像とテキスト、構造化クエリと非構造化クエリなど) が異なる場合は、カスタム処理が必要です。
- エージェントは会話型スイッチボードとして機能し、タスクを特殊なエージェントまたはマイクロサービスに委任します。
- このワークフローは、インテリジェントなディスパッチによってエージェントの動作の品質と効率の両方が決定される、ドメイン固有の副操縦士、カスタマーサポートボット、エンタープライズサービスルーター、マルチモーダルエージェントで一般的です。

機能

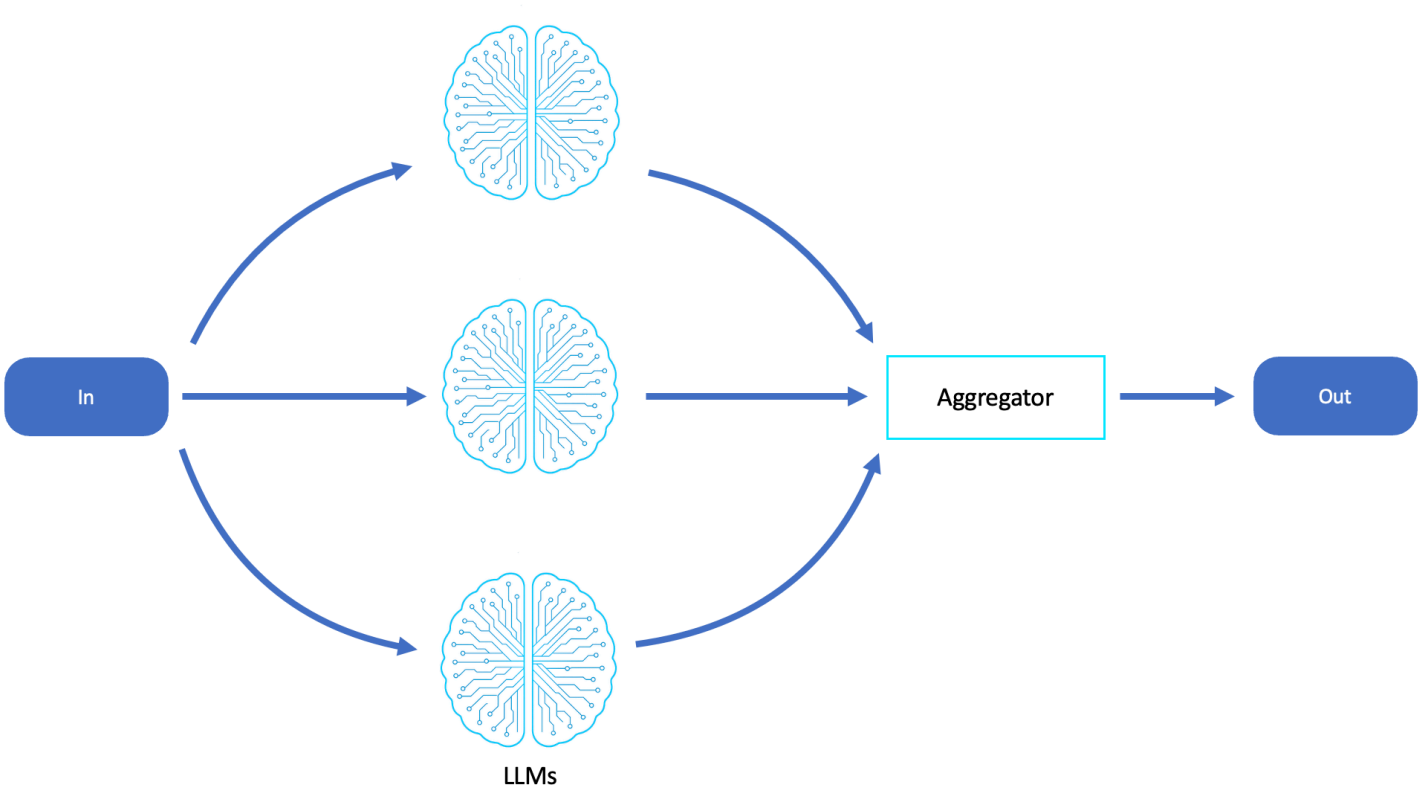
- ファーストパス LLM はディスパッチャーとして機能します
- ルートは個別のワークフローや他のエージェントパターンを呼び出すことができます
- 機能のモジュラー拡張をサポート

一般的なユースケース

- マルチドメインアシスタント (「これは法的、医療的、または財務的な質問ですか？」)
- LLM 推論で強化された決定木
- 動的ツールの選択 (検索とコード生成など)

並列化のワークフロー

このワークフローでは、複数の LLM 呼び出しまたはエージェントによって同時に処理できる独立したサブタスクにタスクを分割します。その後、出力はプログラムで集約され、結果に合成されます。



並列化ワークフローは、タスクを独立した非連続サブタスクに分割して同時に処理できる場合に使用します。これにより、効率、スループット、スケーラビリティが大幅に向上します。これは、エージェントが複数の入力にわたってコンテンツを分析または生成する必要がある、データ量の多い、バッチ指向、またはマルチパースペクティブな問題スペースで特に強力です。

並列化は、次の場合に特に効果的です。

- サブタスクは相互の中間結果に依存しないため、調整せずに並行して実行できます。
- タスクでは、同じ推論プロセスを多くの項目で繰り返します (たとえば、複数のドキュメントを要約したり、オプションのリストを評価したりします)。
- 多様性、創造性、堅牢性を促進するために、複数の仮説や視点が並行して検討されます。
- LLM の同時実行により、高ボリュームまたは高頻度のリクエストのレイテンシーを減らす必要があります。
- このワークフローは、ドキュメント処理エージェント、調査エンジンまたは比較エンジン、バッチサマリ、マルチエージェントブレインストーミング、スケーラブルな分類またはラベル付けタスクで一般的に使用されます。特に、迅速で並行した推論がパフォーマンス上の利点である場合です。

機能

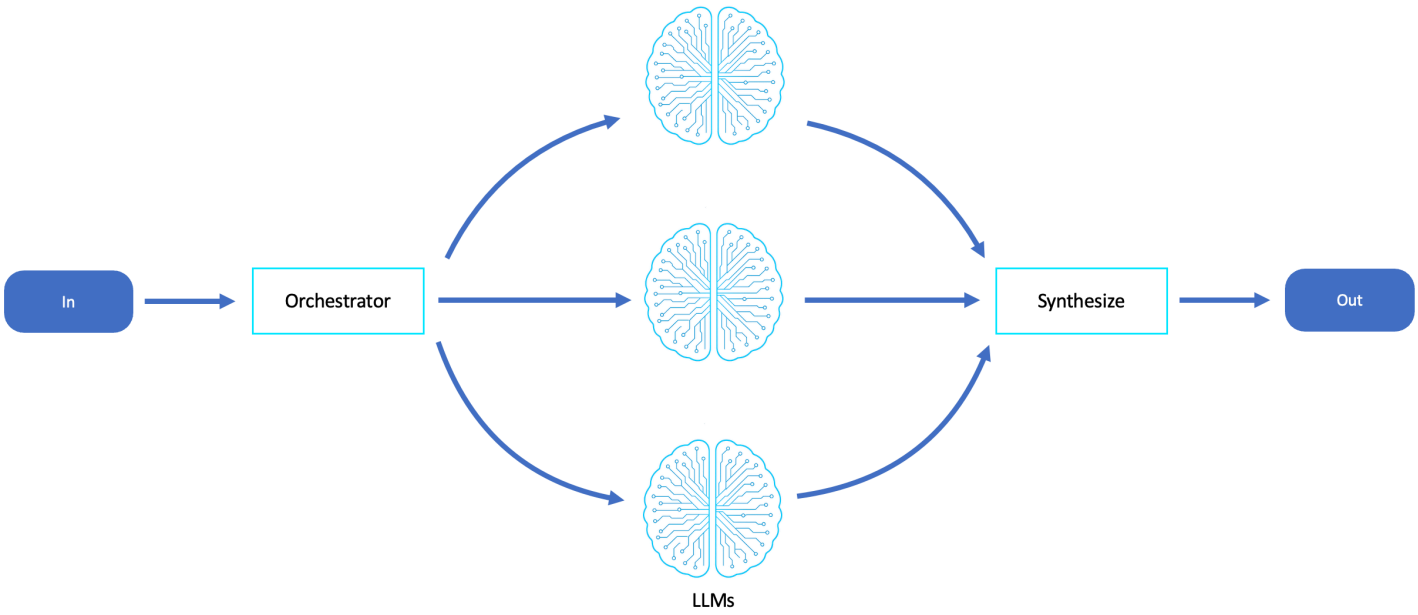
- LLM タスクの並列実行 (AWS Lambda、 AWS Fargate、または AWS Step Functions マップ状態を使用)
- 合成段階で結果の整列、検証、または重複排除が必要です
- ステートレスエージェントループに最適

一般的なユースケース

- 複数のドキュメントまたはパースペクティブを並行して分析する
- 多様なドラフト、概要、または計画の生成
- バッチジョブ間のスループットの高速化

オーケストレーションのワークフロー

中央オーケストレーターエージェントは LLM を使用して、特定のロールまたはドメインの専門知識を持つ特殊なワーカーエージェントまたはモデルにサブタスクを計画、分解、委任します。これは人間のチーム構造を反映し、複数のエージェントにわたる緊急の動作をサポートします。



オーケストレーションワークフローは、構造化された分解と特殊な実行を必要とする、複雑、階層的、または学際的なシナリオに最適です。これは、作業の分割を必要とするタスクに特に適してお

り、タスクのさまざまなサブコンポーネントは、異なる機能、知識、またはツールセットを持つエージェントによって最も適切に処理されます。

このワークフローは、次の場合に特に効果的です。

- タスクは、範囲、タイプ、推論 (計画、調査、実装、テストなど) が異なるサブタスクに分割できます。
- LLM またはメタエージェントは、他のエージェントを調整し、進行状況をモニタリングし、結果を合成する必要があります。
- エージェントの責任をモジュール化して、スケーラビリティ、再利用、特殊なチューニングを実現したいと考えています。
- システムには、人間のチーム (プロジェクトマネージャー、開発者、レビュー担当者など) がコラボレーションでどのように動作するかを模倣した、ロールベースの動作が必要です。

オーケストレーションは、マルチターンプランニングエージェント、ソフトウェア開発副操縦士、エンタープライズプロセスエージェント、自律型プロジェクトエグゼキューターに最適です。これは、一元化されたタスクの内訳を必要とするが、実行ロジックが分散されているマルチエージェントシステムを実装する場合に特に便利です。これにより、エージェントレイヤー間で拡張性とより説明しやすい動作が可能になります。

機能

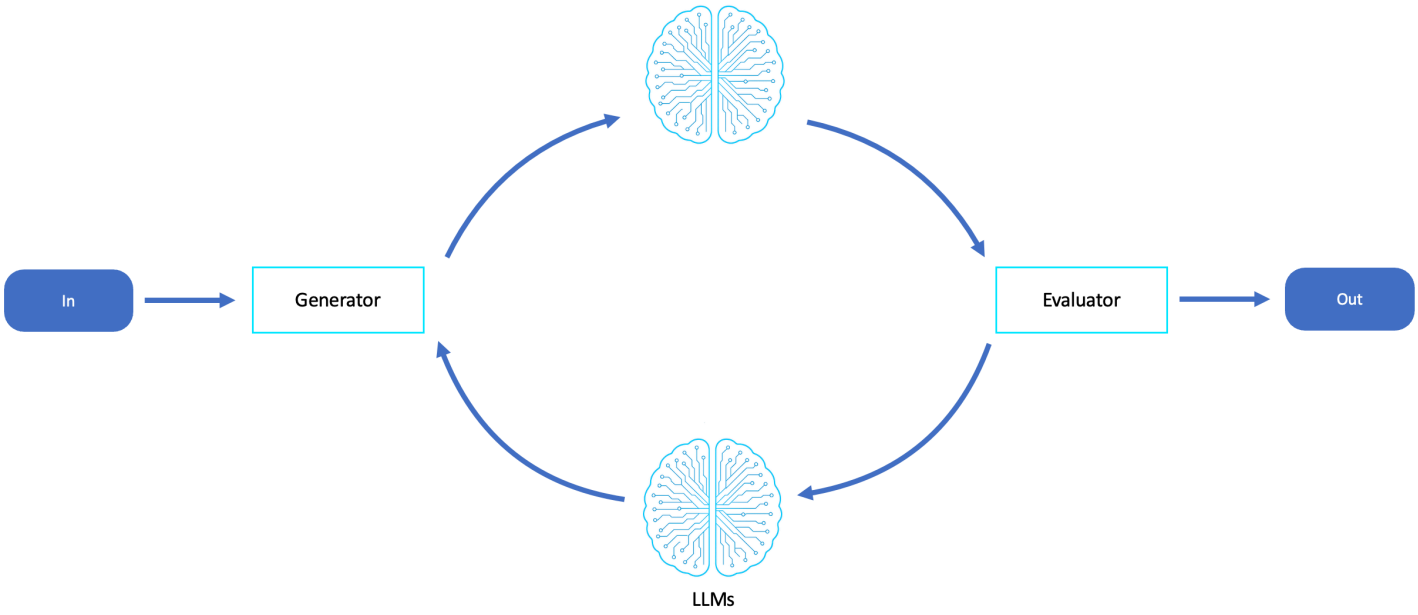
- オーケストレーターが目標メタ推論を実行する
- ワーカーエージェントには、ツールアクセス、メモリ、またはドメイン固有のプロンプトが含まれる場合があります
- 階層 (複数レベルのタスク委任) にすることができます

一般的なユースケース

- プロジェクトマネージャー、調整研究者、ライター、品質保証エージェント
- 計画、実行、テストを組み合わせたコーディング副操縦士
- ツールチェーンまたは API アクセスパターンを監督するエージェント

評価者とリフレクション/絞り込みループのワークフロー

このワークフローは、1 つの LLM が結果を生成し、別の が結果を評価または批評するフィードバックループを提供します。これにより、自己リフレクション、最適化、反復的な改善が促進されます。



評価者ワークフローは、出力の品質、精度、調整が重要で、単一パス生成の信頼性が低い、または不十分なシナリオに最適です。このワークフローは、より高い正確性基準を満たすため、またはフィードバックに基づいて改善された代替案を検討するために、エージェントが出力を自己批評、反復、改良する必要がある場合に優れています。

このワークフローは、次の場合に特に効果的です。

- 出力には、主観的な品質メトリクス (スタイル、トーン、読みやすさなど) または目標基準 (正確性、安全性、パフォーマンスなど) が含まれます。
- エージェントは、トレードオフを通じて推論し、制約を評価し、目標に向けて最適化する必要があります。
- 特に規制対象ドメイン、顧客向けドメイン、またはクリエイティブドメインでは、組み込みの冗長性と品質保証が必要です。
- ヒューHuman-in-the-loopレビューは高価または利用できないため、自律的な検証が必要です。

このワークフローは、コンテンツ生成、コード合成とレビュー、ポリシーの適用、アライメントチェック、指示調整、RAG 後処理に使用されます。また、継続的なフィードバックが時間の経過と

ともにより良い対応を形成し、信頼できる自律的な決定ループを構築するのに役立つ自己改善エージェントにも役立ちます。

一般的なユースケース

- ブルーチームエージェントと比較したレッドチームエージェント
- コードまたは計画を生成、評価、および改訂するエージェント
- 品質保証、幻覚検出、スタイルの適用

機能

- さまざまなモデルを使用した分離された生成と評価をサポート (生成には Claude、評価には Mistral など)
- フィードバックは構造化され、改訂された出力を促すために使用されます。
- 複数の反復または収束のしきい値をサポート

結論

LLMs最新のソフトウェアエージェントのコグニティブコアを提供しますが、raw モデルの呼び出しでは、目的意識があり、堅牢で、制御可能なインテリジェンスを実現するには不十分です。出力生成から構造化された推論と目標に沿った動作に移行するには、モデルが入力を処理し、コンテキストを管理し、アクションを調整する方法を定義する意図的なワークフローパターンに LLMs を埋め込む必要があります。

LLM ワークフローでは、エージェントのコグニティブモジュールを構築するための基盤が導入されています。

- プロンプトの連鎖は、複雑な推論をモジュール式の監査可能なステップに分割します。
- ルーティングにより、インテリジェントなタスク分類とターゲットを絞った委任が可能になります。
- 並列化はスループットを加速し、多様な推論を促進します。
- エージェントオーケストレーションは、タスク分解とロールベースの実行を通じてマルチエージェントコラボレーションを構築します。
- 評価者 (リフレクトリファインループ) は、自己改善、品質管理、およびアライメントチェックを可能にします。

各ワークフローは、エージェントのニーズ、タスクの複雑さ、ユーザーの期待に適応できる構成可能なパターンを表します。これらのワークフローは相互に排他的ではありません。これらは、多くの場合、動的推論、マルチエージェント調整、エンタープライズグレードの信頼性をサポートするハイブリッドアーキテクチャに結合されるブロックを構築しています。

エージェントワークフローパターンに関する次の章に移行すると、これらの LLM ワークフローはより大きなシステム内に埋め込み構造として再び表示され、目標の委任、ツールオーケストレーション、決定ループ、ライフサイクルの自律性をサポートします。これらの LLM ワークフローをマスターすることは、テキストを予測するだけでなく、理由、適応、意図的な行動を行うソフトウェアエージェントを設計するために不可欠です。

エージェントワークフローパターン

エージェントワークフローパターンは、モジュラーソフトウェアエージェントを構造化大規模言語モデル (LLM) ワークフローと統合し、自律的な推論とアクションを可能にします。従来のサーバーレスアーキテクチャとイベント駆動型アーキテクチャから着想を得たこれらのパターンは、コアロジックを静的コードから LLM 拡張エージェントに移行し、適応性とコンテキストに応じた意思決定を強化します。この進化により、従来のクラウドアーキテクチャは、スケーラビリティと応答性の基本的な原則を維持しながら、決定論的なシステムから動的な解釈とインテリジェントな拡張が可能なシステムに変換されます。

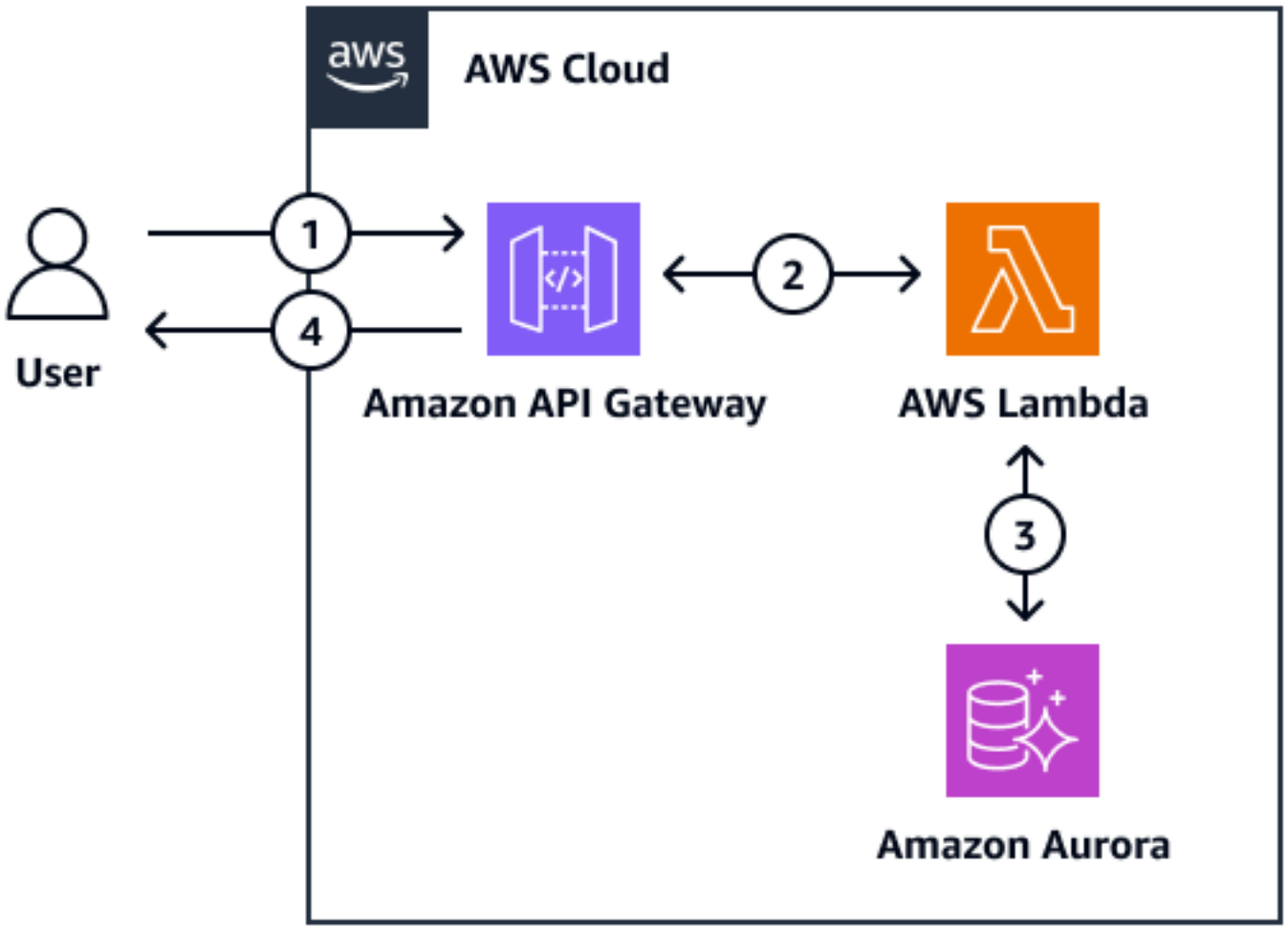
このセクションの内容

イベント駆動型システムから認識拡張型システムへ

最新のクラウドアーキテクチャ、特にサーバーレスおよびイベント駆動型の原則に基づいて構築されたものは、従来、応答性が高くスケーラブルなシステムを作成するために、ルーティング、ファンアウト、エンリッチメントなどのパターンに依存してきました。エージェント AI システムは、LLM で強化された推論と認知の柔軟性を中心に再構築しながら、これらの基盤の上に構築されます。このアプローチにより、より高度な問題解決機能と自動化機能が可能になり、クラウド環境での複雑なタスクの処理方法に変革をもたらす可能性があります。

イベント駆動型アーキテクチャ

次の図は、一般的な分散システムを示しています。

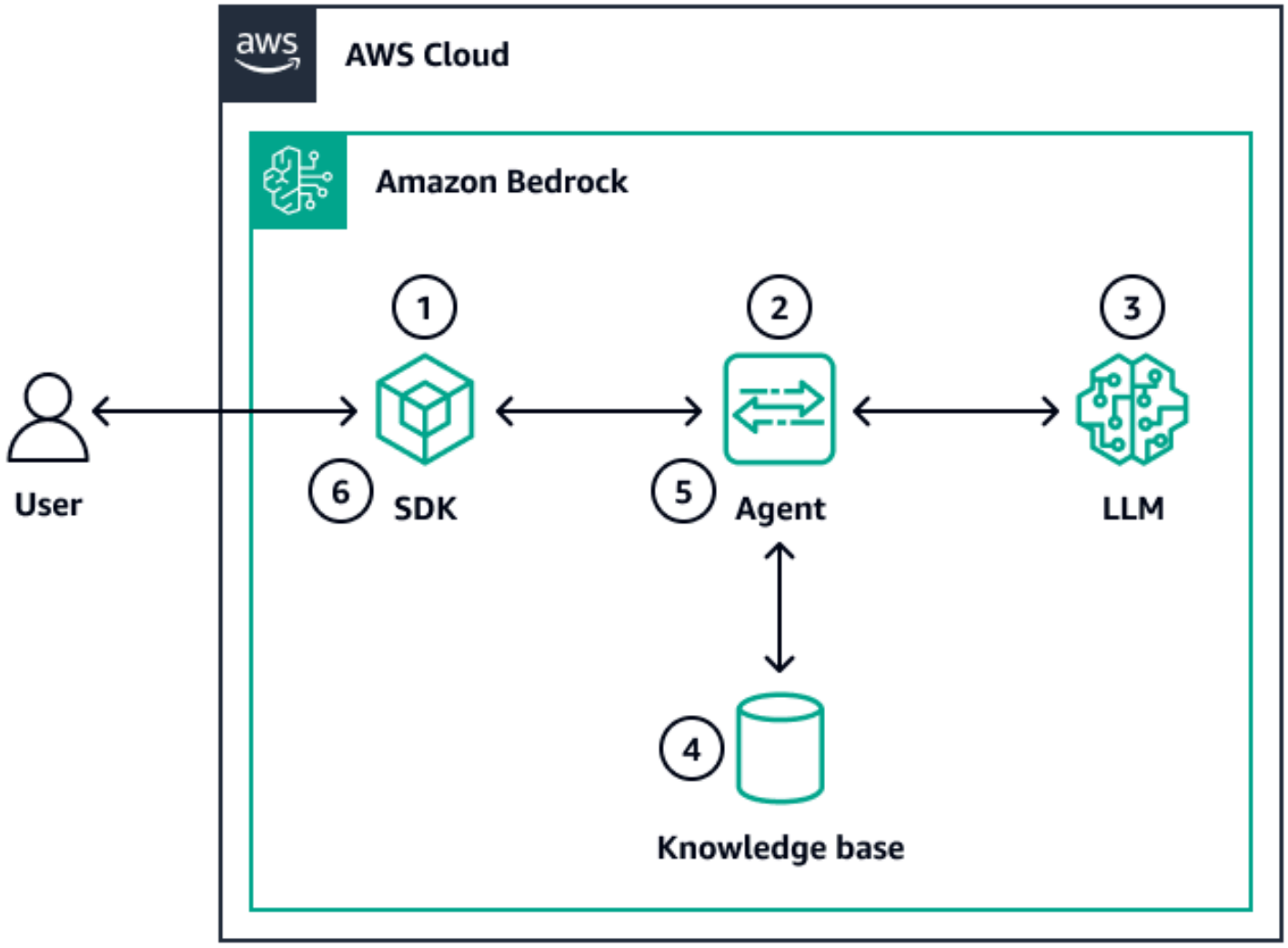


1. ユーザーが Amazon API Gateway にリクエストを送信します。
2. Amazon API Gateway は、リクエストを AWS Lambda 関数にルーティングします。
3. AWS Lambda Amazon Aurora データベースをクエリしてデータエンリッチメントを実行する
4. Amazon API Gateway は、エンリッチされたペイロードを呼び出し元に返します。

この構造は信頼性が高くスケーラブルですが、基本的には静的です。ビジネスルールとロジックパスは明示的にコーディングする必要があり、変化するコンテキストや不完全な情報への適応は制限されます。

認知拡張ワークフロー

エージェントアーキテクチャは、イベント駆動型システムに認知拡張を追加します。次の図は、同等のエージェントを示しています。



1. ユーザーは SDK または API コールを通じてクエリを送信します。
2. Amazon Bedrock エージェントはクエリを受け取ります。
3. エージェントは LLM を呼び出してクエリを解釈します
4. エージェントは、Amazon Bedrock ナレッジベースまたは他の外部データソースを検索してセマンティックエンリッチメントを実行します。
5. LLM は、コンテキストが豊富な目標に沿ったレスポンスを合成します。
6. システムは合成されたレスポンスをユーザーに返します。

このフローでは、LLM はロジックを使用し、インテントを理解し、関連するコンテキストを取得して組み合わせ、最善の対応方法を決定します。このパターンは、メッセージが外部データで拡張されてからさらにルーティングされる従来のエンリッチメントパターンを反映しています。ただし、エー

ジェントシステムでは、このエンリッチメントは静的ルックアップではありません。代わりに、エンリッチメントは動的で、意味的にガイドされ、目的によって駆動されます。

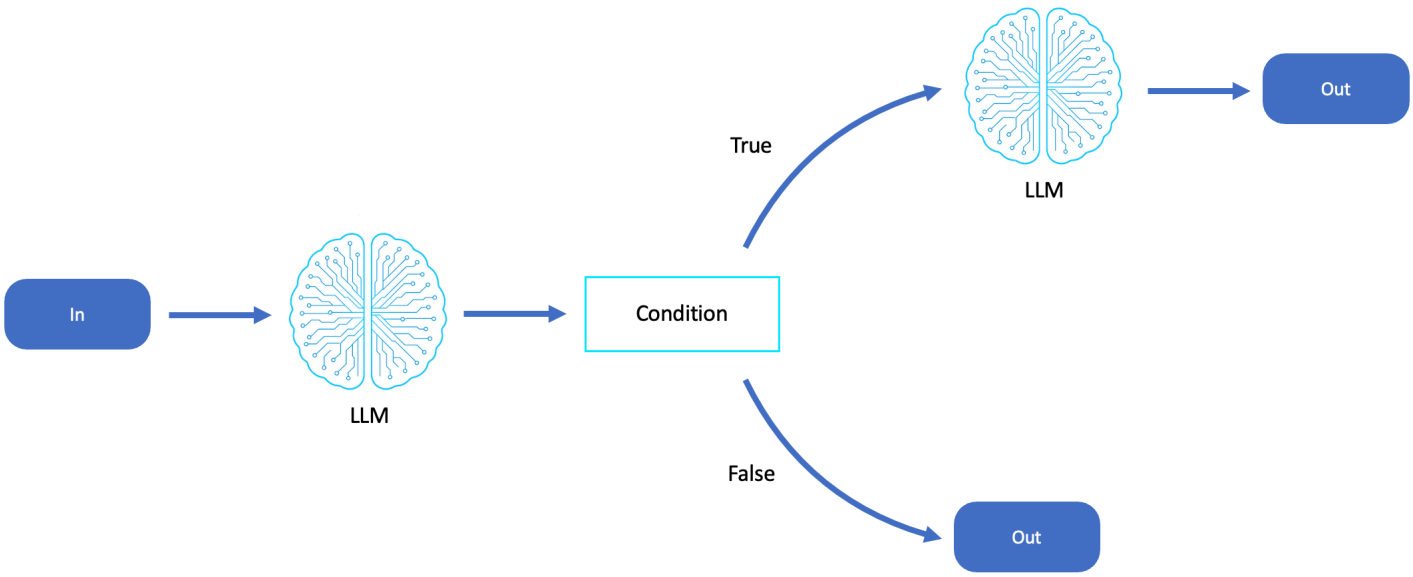
コアインサイト

各 LLM ワークフローは、従来のイベント駆動型アーキテクチャスタイルをミラーリングして進化させるエージェントワークフローパターンにマッピングできます。エージェントワークフローの基本的な構成要素は、LLM のコンテキストをデータ、ツール、メモリで拡張できることです。これにより、通知され、適応性があり、ユーザーのインテントに沿った推論ループが作成されます。従来のシステムがルックアップデータでメッセージを強化する場合、エージェントシステムはソフトウェアがスクリプトのように動作したり、インテリジェントな共同作業者のように動作したりすることを可能にします。

Saga パターンのプロンプト連鎖

LLM プロンプト連鎖をイベント駆動型サガとして再考することで、ワークフローが分散され、回復可能になり、自律型エージェント間で意味的に調整される、新しい運用モデルが生まれます。各プロンプト/レスポンスステップは、アトミックタスクとして再フレームされ、イベントとして出力され、専用エージェントによって消費され、コンテキストメタデータで強化されます。

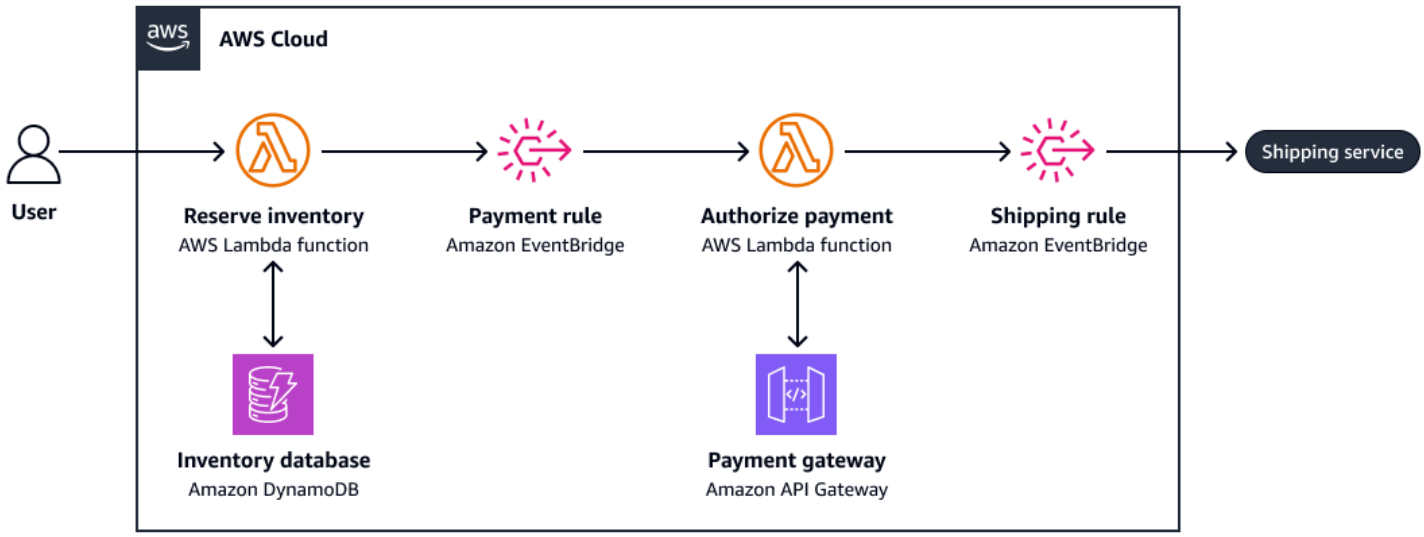
次の図は、LLM プロンプトの連鎖の例です。



Saga コレオグラフィ

Saga コレオグラフィパターンは、中央コーディネーターを持たない分散システムにおける実装アプローチです。代わりに、各サービスまたはコンポーネントは、次のワークフローアクションをトリガーするイベントを発行します。このパターンは、複数のサービス間でトランザクションを管理するための分散システムで広く使用されています。Saga では、システムは一連の調整されたローカルトランザクションを実行します。失敗した場合、システムは整合性を維持するために補償アクションをトリガーします。

次の図は、Saga コレオグラフィの例です。



1. インベントリを予約する
2. 支払いを承認する
3. 配送注文を作成する

ステップ 3 が失敗すると、システムは補償アクション (支払いのキャンセルやインベントリのリリースなど) を呼び出します。

このパターンは、サービスが疎結合され、部分的な障害がある場合でも、時間の経過とともに状態を一貫して解決する必要があるイベント駆動型アーキテクチャで特に役立ちます。

プロンプトの連鎖パターン

プロンプトの連鎖は、構造と目的の両方で saga パターンに似ています。コンテキストを保持し、ロールバックとリビジョンを許可しながら、順番に構築する一連の推論ステップを実行します。

エージェントコレオグラフィ

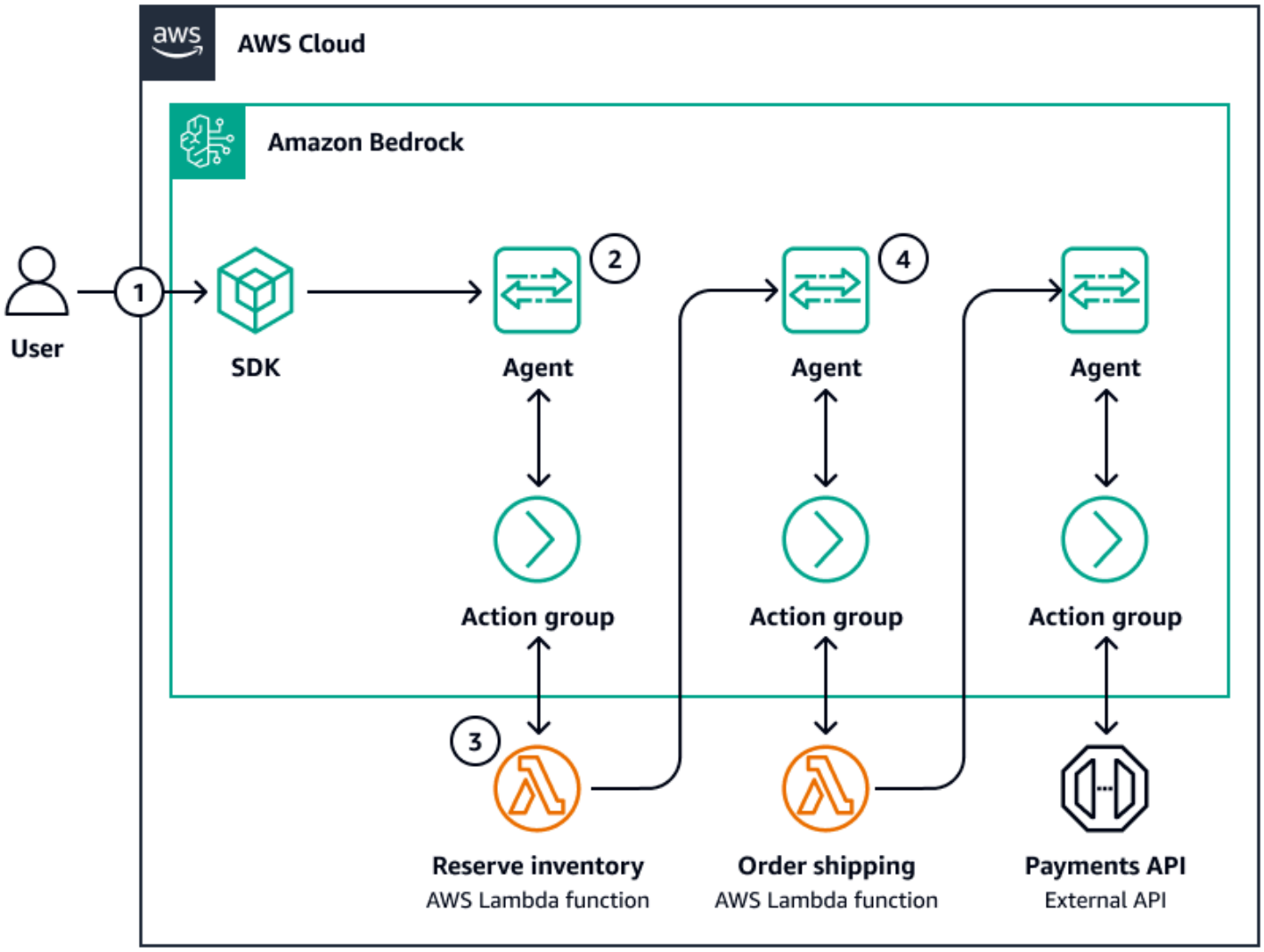
1. LLM は複雑なユーザークエリを解釈し、仮説を生成します
2. LLM がタスクを解決するための計画を策定する
3. LLM がサブタスクを実行する (ツール呼び出しの使用や知識の取得など)
4. LLM は、結果が不十分であると判断した場合、出力を絞り込むか、前のステップを再検討します。

中間結果に欠陥がある場合、システムは次のいずれかを実行できます。

- 別のアプローチを使用してステップを再試行する
- 前のプロンプトに戻って再計画する
- 評価者ループ (評価者最適化パターンなど) を使用して障害を検出して修正する

Saga パターンと同様に、プロンプトの連鎖により、部分的な進行とロールバックのメカニズムが可能になります。これは、データベーストランザクションを補償するのではなく、反復的な絞り込みと LLM 指向の修正によって行われます。

次の図は、エージェントのコレオグラフィの例です。



1. ユーザーは SDK を介してクエリを送信します。
2. Amazon Bedrock エージェントは、次のように推論を調整します。
   - 解釈 (LLM)
   - 計画 (LLM)
   - ツールまたはナレッジベースによる実行
   - レスポンスの構築
3. ツールが失敗するか、不十分なデータを返す場合、エージェントはタスクを動的に再計画または再フレーズできます。
4. メモリ (短期ベクトルストアなど) はステップ間で状態を保持できます

重要事項

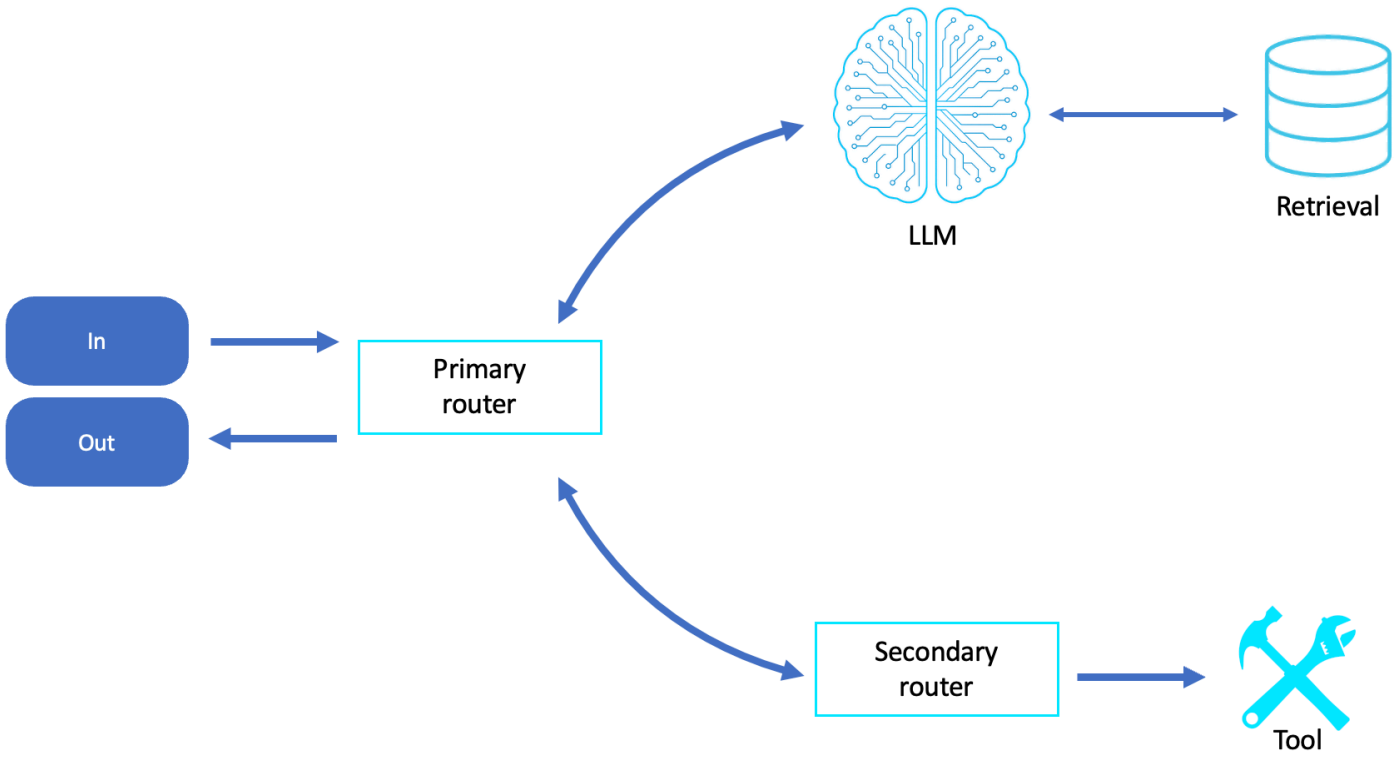
Saga パターンが補償ロジックを使用して分散サービス呼び出しを管理する場合、プロンプトチェーンはリフレクティブシーケンスと適応再計画を使用して推論タスクを管理します。どちらのシステムも、段階的な進行、分散された決定ポイント、障害復旧を可能にし、厳格なロールバックではなく、情報に基づいた推論を通じてこれらをすべて実行します。

プロンプト連鎖では、トランザクション推論が導入されます。これは、Sagas と認識的に同等です。つまり、より広範な目標指向の対話の一環として、各「思考」が再評価、改訂、または中止されます。

動的ディスパッチパターンのルーティング

タスクがドキュメント解析から自律的なソフトウェア生成まで多岐にわたる最新のエージェントシステムでは、最も能力の高い大規模言語モデル (LLM) またはエージェントにリクエストを動的にルーティングする能力が重要になります。静的ルーティングロジックは、多くの場合、オーケストレーションスクリプトまたは API レイヤーに埋め込まれており、リアルタイム、マルチモデル、マルチ機能環境に必要な適応性がありません。これに対処するために、LLM ルーティングワークフローを動的ディスパッチパターンを活用するイベント駆動型アーキテクチャに変換し、LLM 呼び出しをインテリジェントにルーティングされたコンテキスト対応イベントに変換できます。

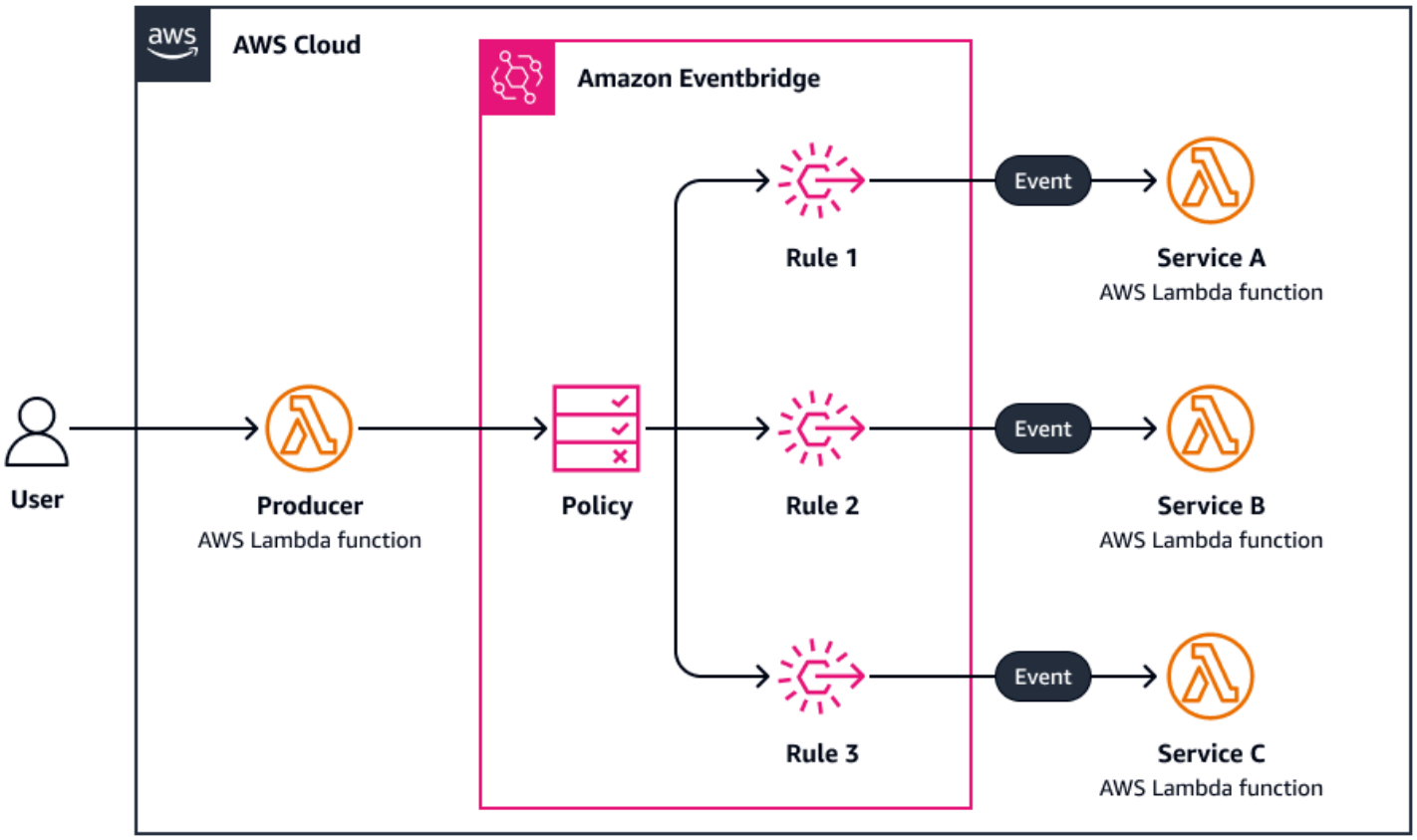
次の図は、LLM ルーティングの例です。



動的ディスパッチ

従来の分散システムでは、動的ディスパッチパターンは、イベントタイプ、ソース、ペイロードなどの受信イベント属性に基づいて、実行時に特定のサービスを選択して呼び出します。これは通常、Amazon EventBridge を使用して実装されます。Amazon EventBridge では、受信イベントを評価して適切なターゲット (AWS Lambda 関数 AWS Step Functions、Amazon Elastic Container Service タスクなど) にルーティングできます。

次の図は、動的ディスパッチの例です。



1. アプリケーションはイベントを発行します (例: {"type": "orderCreated", "priority": "high"})。
2. Amazon EventBridge は、ルーティングルールに照らしてイベントを評価します。
3. イベントの属性に基づいて、システムは次のように動的にディスパッチします。
   - `HighPriorityOrderProcessor` (サービス A)
   - `StandardOrderProcessor` (サービス B)
   - `UpdateOrderProcessor` (サービス C)

このパターンは、疎結合、ドメインベースの特殊化、ランタイムの拡張性をサポートしています。これにより、システムは変化する要件やイベントセマンティクスにインテリジェントに対応できます。

LLM ベースのルーティング

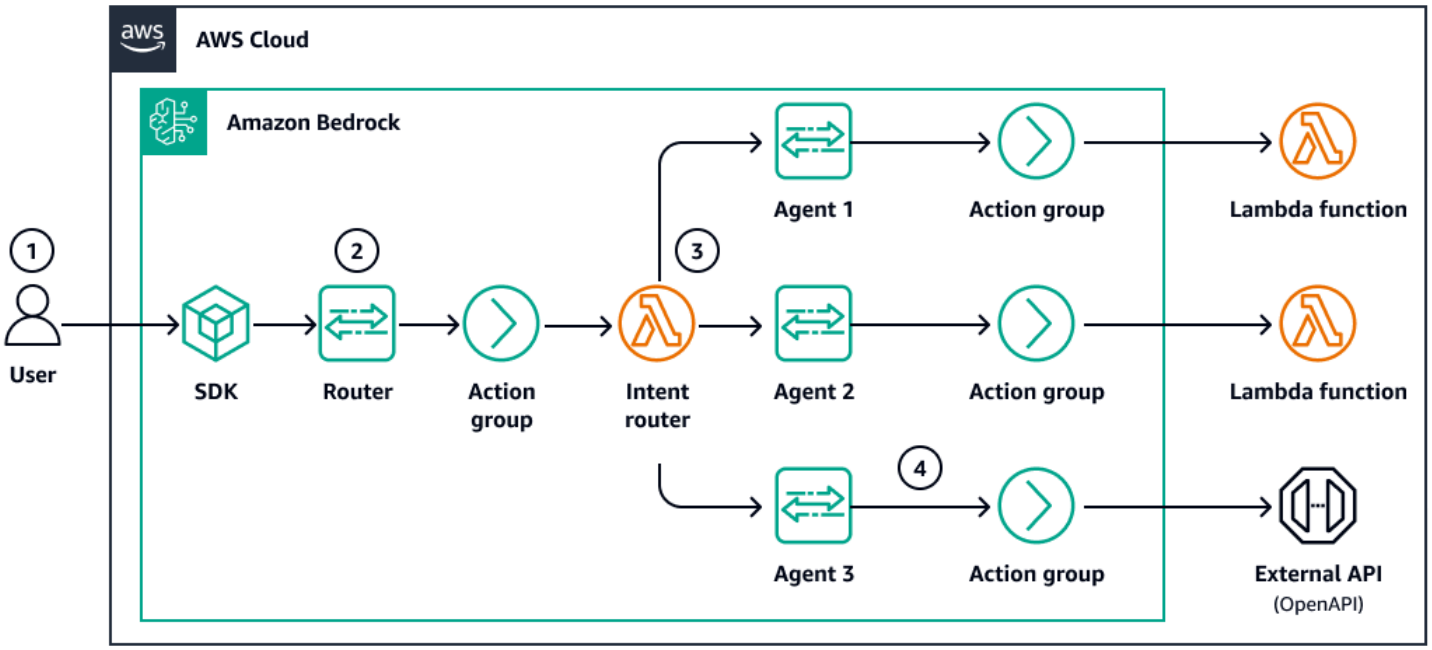
エージェントシステムでは、ルーティングは動的タスク委任も実行しますが、Amazon EventBridge ルールやメタデータフィルターの代わりに、LLM は自然言語を通じてユーザーのインテントを分類して解釈します。その結果、柔軟でセマンティック、適応型のディスパッチが作成されます。

エージェントルーター

このアーキテクチャにより、事前定義されたスキーマやイベントタイプなしで豊富なインテントベースのディスパッチが可能になります。これは、非構造化入力や複雑なクエリに最適です。

1. ユーザーがリクエストを送信します「契約条件の確認をお手伝いいただけますか？」
2. LLM はこれを法的文書タスクとして解釈します。
3. エージェントは、タスクを次のいずれかにルーティングします。
   - 契約レビュープロンプトテンプレート
   - 法的推論サブエージェント
   - ドキュメント解析ツール

次の図は、エージェントルーターの例です。



1. ユーザーは SDK を通じて自然言語リクエストを送信します。
2. Amazon Bedrock エージェントは、LLM を使用してタスク (法律、技術、スケジューリングなど) を分類します。
3. エージェントは、アクショングループを介してタスクを動的にルーティングして、必要なエージェントを呼び出します。
   - ドメイン固有のエージェント
   - 特殊なツールチェーン

- カスタムプロンプト設定

4. 選択したハンドラーはタスクを処理し、カスタマイズされたレスポンスを返します。

重要事項

従来の動的ディスパッチでは、構造化イベント属性に基づくルーティングに Amazon EventBridge ルールが使用されますが、エージェントルーティングでは LLMsを意味論的に分類およびルーティングします。これにより、以下を有効にすることでシステムの柔軟性が向上します。

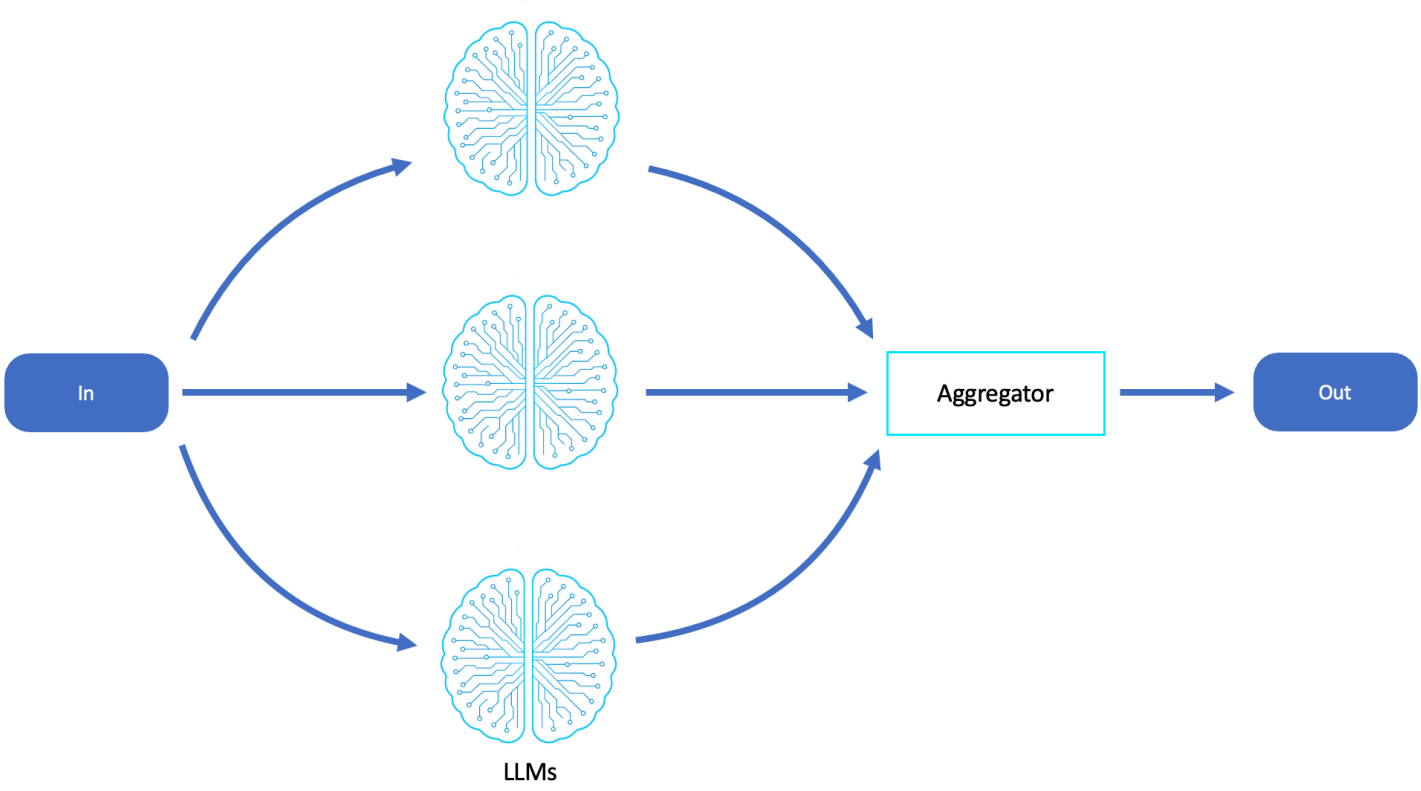
- より広範な入力の理解
- インテリジェントなフォールバックとツールの選択
- 新しいエージェントロールまたはプロンプトスタイルによる自然な拡張性

エージェントルーティングは、リジッドルールを動的コグニティブディスパッチに置き換えます。これにより、システムはコードではなく言語で進化できます。

並列化パターンと散布図パターン

大きなドキュメントの要約、複数のソリューションパスの評価、多様な視点の比較など、多くの高度な推論および生成タスクは、プロンプトの並列実行からメリットを得ます。スケーラビリティ、応答性、耐障害性が必要な場合、従来のシーケンシャルワークフローは短くなります。これを克服するために、LLM ベースの並列化は、タスクが自律エージェントに動的にファンアウトされ、結果がインテリジェントに合成されるイベント駆動型の散布図パターンを使用して再考できます。

次の図は、LLM 並列化ワークフローの例です。



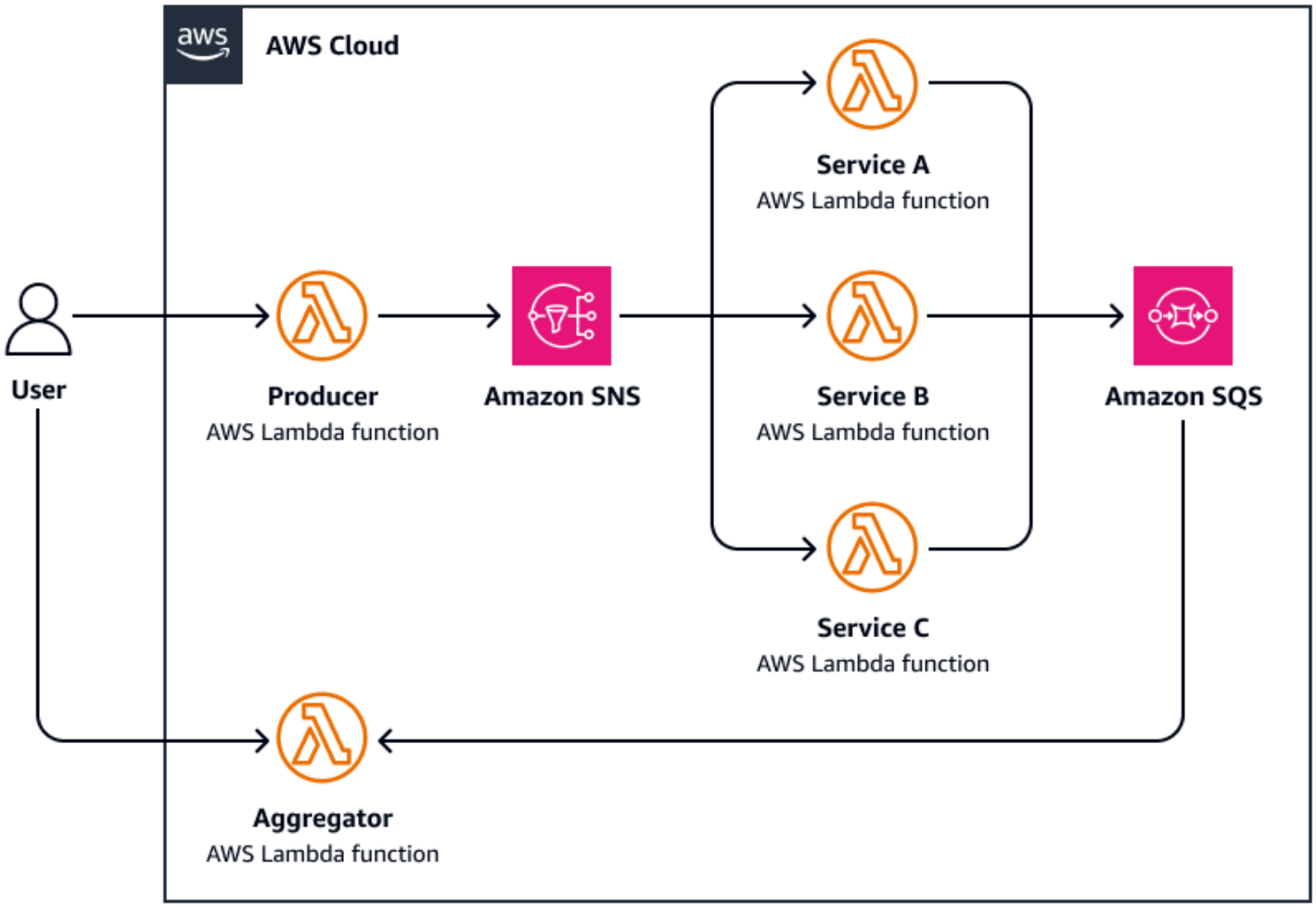
Scatter-gather

分散システムでは、散布図パターンはタスクを複数のサービスまたは処理ユニットに並行して送信し、応答を待機してから、結果を統合出力に集約します。ファンアウトとは異なり、scatter-gather はレスポンスを想定し、通常は結果を結合、比較、選択するロジックを適用するため、調整されます。

並列化と散布収集の一般的な実装は次のとおりです。

- AWS Step Functions 並列タスク実行の状態をマッピングする
- AWS Lambda 同時実行で、複数の呼び出し関数の結果を調整する
- 相関 IDsと集約ワークフローを備えた Amazon EventBridge
- Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB、またはキューを使用してファンアウトを管理し、結果を収集するカスタムコントローラーパターン

次の図は、散布図の例です。



1. ユーザーは、Amazon Simple Notification Service (Amazon SNS) トピックに並列メッセージを発行することでタスクを分散する中央コーディネーター関数にリクエストを送信します。
2. 各メッセージにはタスクメタデータが含まれ、特殊なワーカー にルーティングされます AWS Lambda。
3. 各ワーカーは、割り当てられたサブタスクを AWS Lambda 個別に処理します (外部 API のクエリ、ドキュメントの処理、データの分析など)。
4. 結果は、Amazon Simple Queue Service (Amazon SQS) などの一般的なストレージレイヤーに書き込まれます。
5. アグリゲータ関数は、すべてのレスポンスが完了するまで待機し、以下を実行します。
   - 結果を収集して集計します (概要のマージ、最適な一致の選択など)
   - 最終レスポンスを送信するか、ダウンストリームワークフローをトリガーします

散布物収集パターンの一般的なユースケースは次のとおりです。

- フェデレーティッド検索
- 価格比較エンジン
- 集計データ分析
- マルチモデル推論

LLM ベースの並列化 (散布図認識)

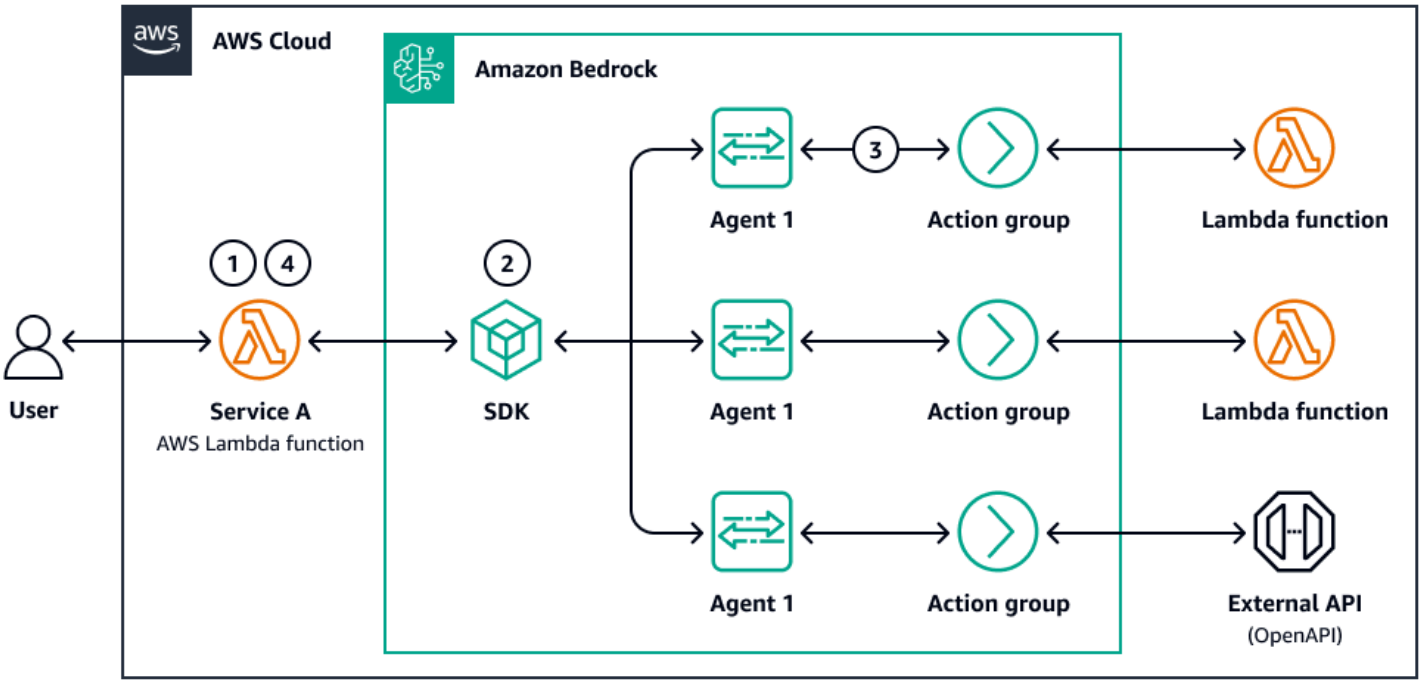
エージェントシステムでは、並列化は、複数の LLM コールまたはエージェントにサブタスクを分散することで散布図を密接にミラーリングし、それぞれが問題の一部を個別に推論します。返される結果は集約プロセスによって収集および合成されます。集約プロセスは、多くの場合、別の LLM エージェントまたはコントローラーエージェントです。

エージェント並列化

1. エージェントは、「これら 10 件のレポート全体でインサイトを要約する」リクエストを送信します。
2. レポートは 10 個の並列 LLM 要約タスクに分散されます。
3. すべての概要を返すと、エージェントは以下を実行します。
   - 概要を統一された説明に集約します
   - テーマまたは矛盾を特定する
   - 合成された出力をユーザーに送信します

このエージェントワークフローにより、スケーラブル、モジュール式、適応型の並列推論が可能になります。これは、高い認知スループットを必要とするユースケースに最適です。

次の図は、エージェントの並列化の例です。



1. ユーザーがマルチパートクエリまたはドキュメントセットを送信します。
2. コントローラー AWS Lambda またはステップ関数は、サブタスクを分散します。各タスクは、独自のプロンプトを使用して Amazon Bedrock LLM コールまたはサブエージェントを呼び出します。
3. 呼び出しとサブタスクが完了すると、結果が保存され (Amazon S3 やメモリストアなど) 、集計ステップが出力をマージ、比較、またはフィルタリングします。
4. システムは、ユーザーまたはダウンストリームエージェントに最終レスポンスを返します。

このシステムには、トレーサビリティ、耐障害性、オプションの結果の重み付けまたは選択ロジックを備えた分散推論ループがあります。

重要事項

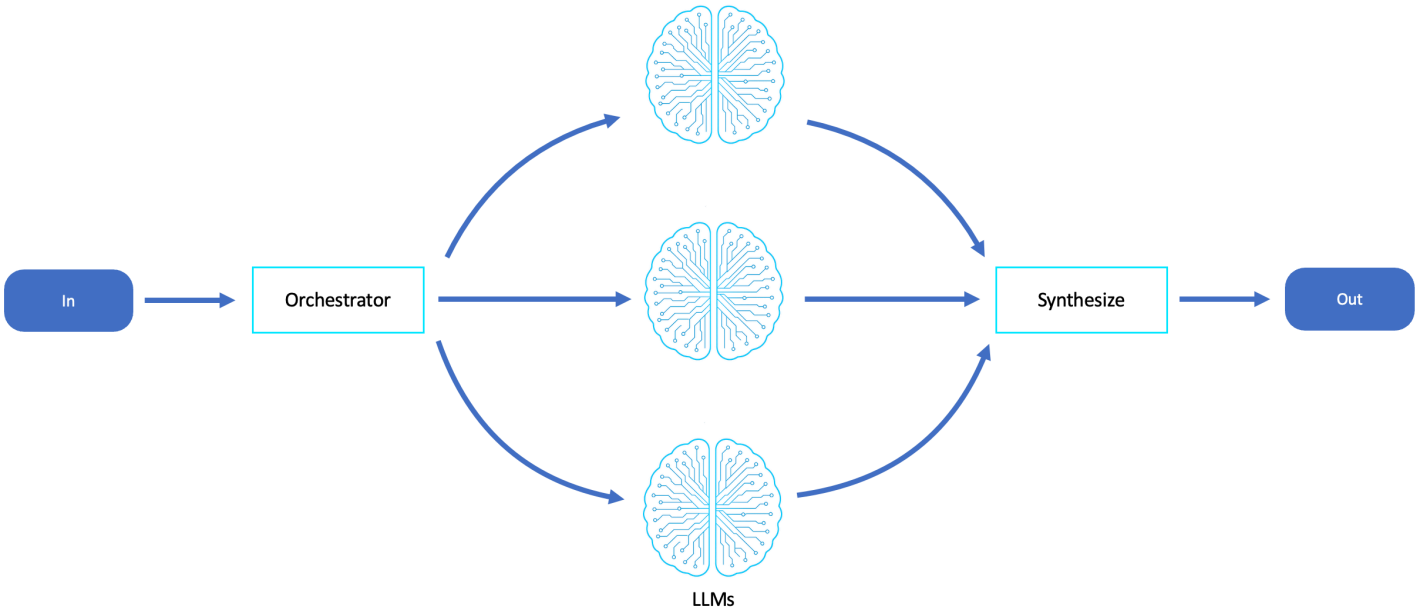
エージェント並列化は、散乱集合パターンを使用して LLM タスクを分散し、並列処理とインテリジェントな結果合成を可能にします。

Saga オーケストレーションパターン

LLMs によって駆動されるワークフローがますます複雑になり、プロンプトチェーン、データ処理ステップ、ツール呼び出し、エージェントのコラボレーションにまたがるようになるにつれて、インテ

リジェントなオーケストレーションの必要性が不可欠です。これらのワークフローは、緊密に結合されたスクリプトや静的な事前定義された実行フローに依存するのではなく、イベント駆動型のオーケストレーションパターンとして実装できるため、LLM ベースのシステムは自律型エージェント間でマルチステップタスクを動的に調整、モニタリング、適応できます。

次の図は、オーケストレーターの例です。



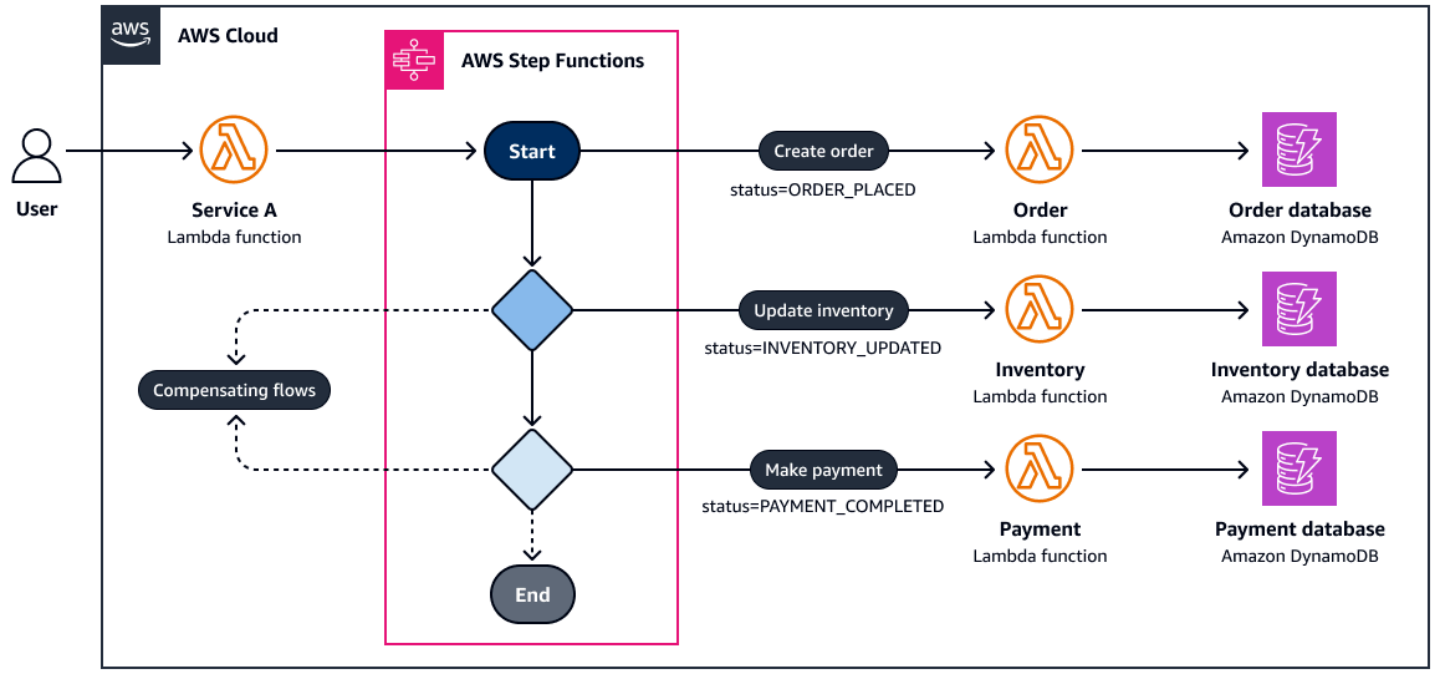
イベントオーケストレーション

従来の分散システムでは、イベントオーケストレーションとは、中央コーディネーターが複数のサービスまたはタスクにまたがって制御フローを明示的に指示することで、複雑なワークフローを管理するパターンを指します。イベントコレオグラフィ (各サービスが独立して反応する) とは異なり、オーケストレーションはプロセス全体の一元化されたロジック、可視性、制御を提供します。

これは通常、次のツールを使用して実装されます。

- AWS Step Functions – ステートフルワークフローを定義して実行する
- AWS Lambda – オーケストレーションされたフロー内で個別のタスクを実行する
- Amazon SQS または Amazon EventBridge – 非同期ステップまたはレスポンスをトリガーします

次の図は、Saga オーケストレーションの例です。



AWS Step Functions ワークフローは、顧客注文プロセスを管理します。

1. 注文の作成 (AWS Lambda)
2. インベントリの更新 (AWS Lambda)
3. 支払いを行う (AWS Lambda)

オーケストレーターは、再試行、並列ブランチ、タイムアウト、失敗を管理することで、各ステップを調整します。

ロールベースのエージェントシステム (オーケストレーター)

エージェントシステムでは、オーケストレーターパターンはイベントオーケストレーションをミラーリングしますが、それぞれが定義されたロールまたは専門分野を持つ複数の推論エージェントにロジックを分散します。中央オーケストレーターエージェントは、タスク全体を解釈し、サブタスクに分解して、ワーカーエージェントに委任します。ワーカーエージェントはそれぞれ、特定のドメイン (研究、コーディング、要約、レビューなど) 用に最適化されています。

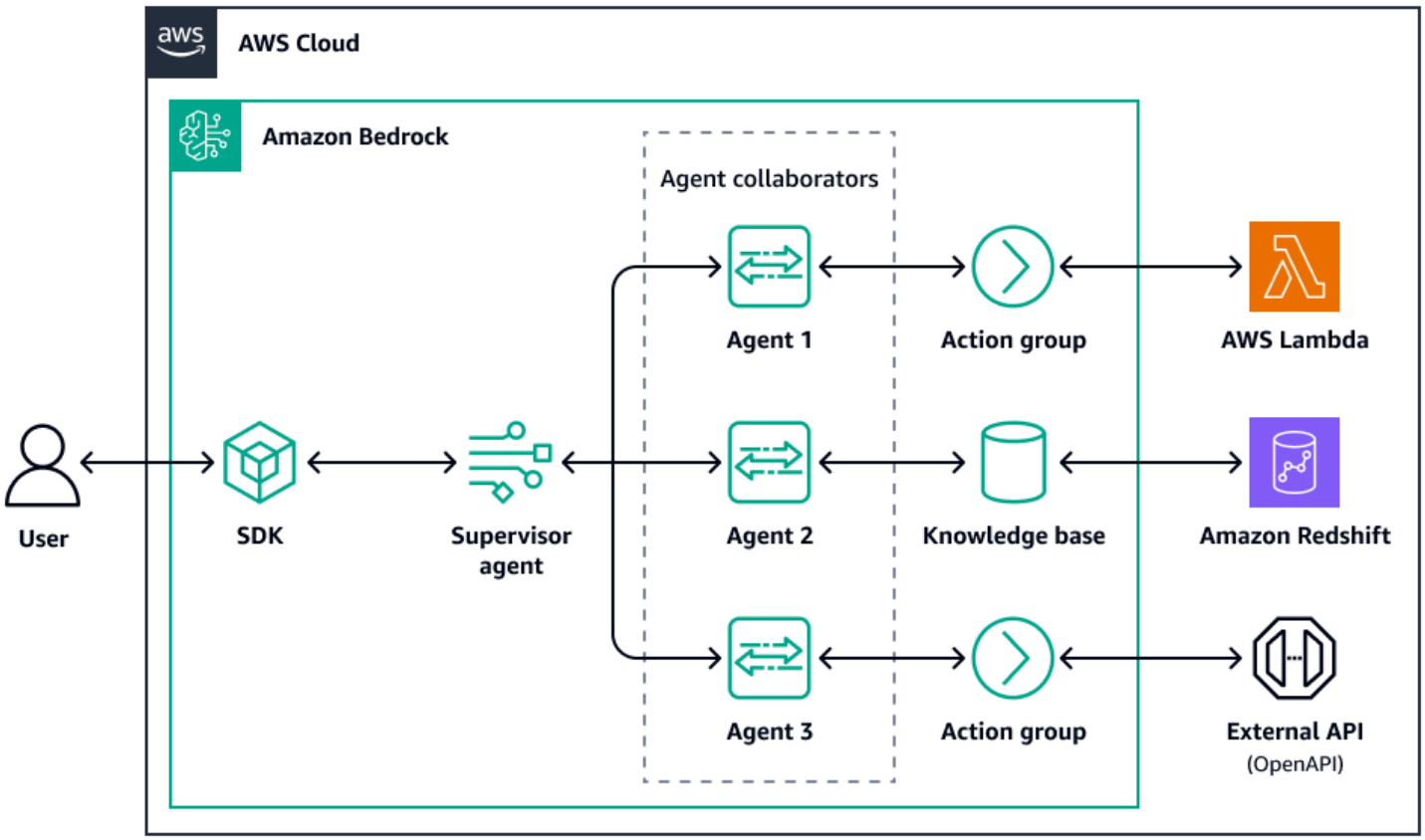
スーパーバイザー

1. ユーザーは「プロジェクト概要を作成し、上位 5 つの競合相手を要約する」というクエリを送信します。
2. オーケストレーターエージェントは以下を実行します。

- 競合データを見つけるために調査エージェントを割り当てます
- 未加工の検出結果を要約エージェントに送信します
- ショートライターエージェントに結果を渡す
- ユーザーの最終出力をコンパイルします

各エージェントは独立して動作しますが、オーケストレーターはタスクを調整します。これは、ワークフロータスクを処理する Lambda 関数のようなものです。

次の図は、スーパーバイザーの例です。



1. ユーザーは Amazon Bedrock スーパーバイザーエージェントにタスクを送信します。
2. スーパーバイザーエージェントは、エージェントコラボレーターごとにリクエストをサブタスクに解析します。
3. 各サブタスクは、ロール固有のプロンプトまたはツールチェーンを持つ共同作業者エージェントに割り当てられます。
4. ワーカーエージェントAPIs またはツールを呼び出します。
5. 各ワーカーエージェントは、構造化形式で出力を返します。

6. すべてのワーカーが結果を返すと、スーパーバイザーは評価、合成、最終レスポンスを返します。

この構造により、複雑なマルチステップエージェントワークフローでのモジュール性、適応性、イントロスペクションが可能になります。

重要事項

イベントオーケストレーションが一元管理 (例: AWS Step Functions) を使用してサービス実行を指示する場合、ロールベースのエージェントシステムは LLM 搭載のオーケストレーターエージェントを使用して、目標の理由を示し、サブタスクをワーカーエージェントに委任し、最終出力を合成します。

どちらのパラダイムでも、オーケストレーターは以下を実行します。

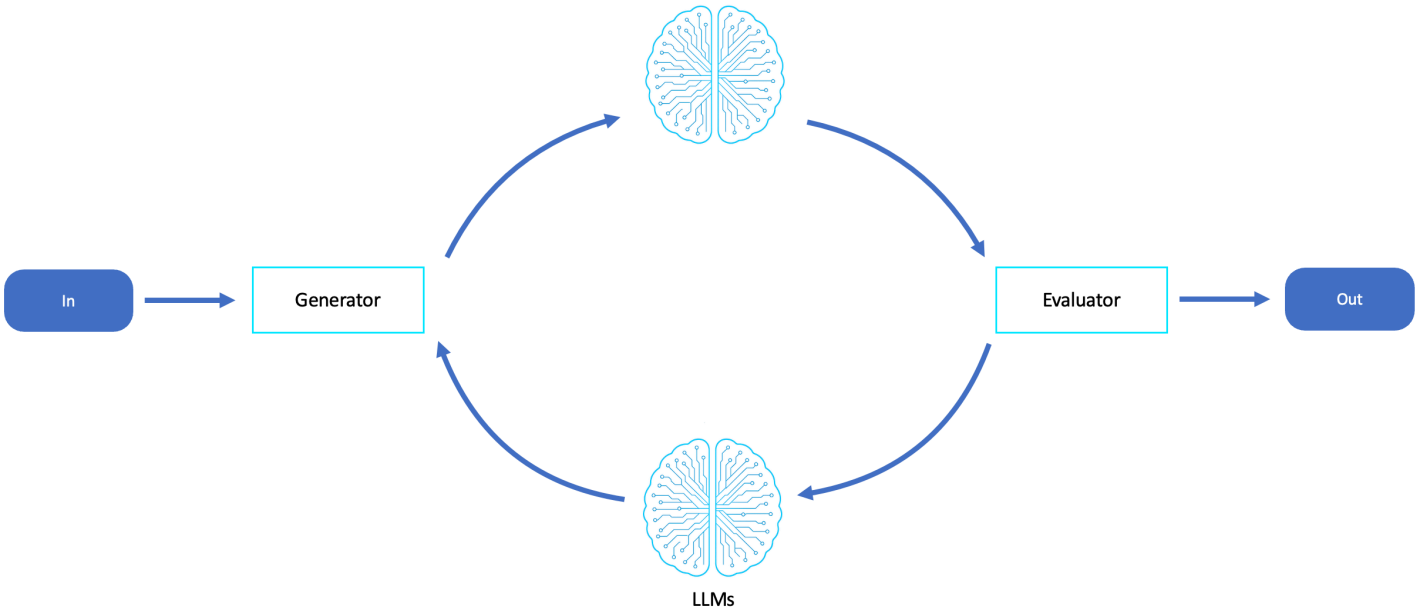
- コンテキストと実行フローを維持します
- 分岐、シーケンス、エラー処理を処理します
- 分散コンポーネントから統一された結果を生成します

ただし、エージェントオーケストレーションは推論、適応性、セマンティック委任を追加します。これにより、オープンエンド、あいまい、進化するタスクに適しています。

評価者の反射/絞り込みループパターン

コード生成、要約、自律的な意思決定などのタスクは、ランタイムフィードバックから大きなメリットを得るため、システムは監視と改良を通じて進化できます。これを運用するために、リフレクションと改良のサイクルをイベント駆動型のフィードバック制御ループとして実装できます。これは、自律的でインテリジェントなワークフローに適したシステムエンジニアリングに着想を得たパターンです。

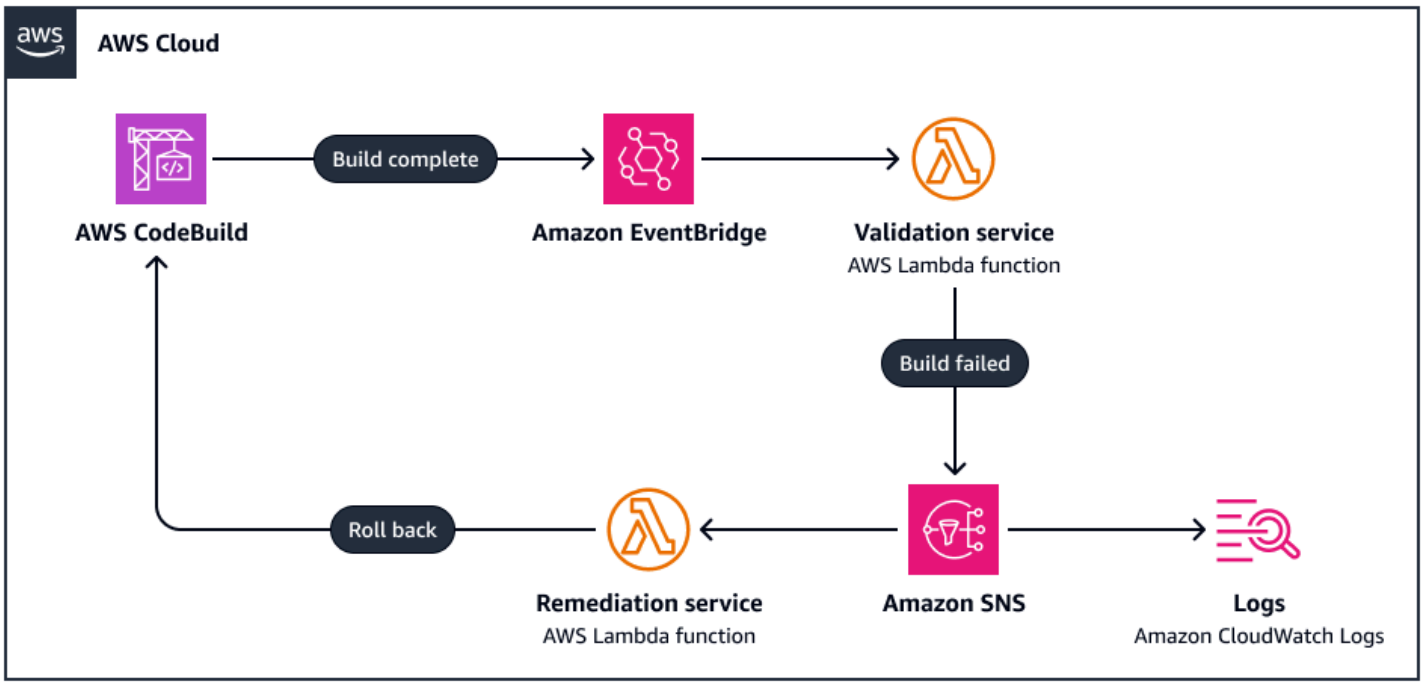
次の図は、評価者のリフレクション/絞り込みフィードバックループの例です。



フィードバックコントロールループ

フィードバックコントロールループは、独自の出力と動作をモニタリングし、定義された基準または目的の状態に照らして評価し、それに応じてアクションを調整するパターンです。このアーキテクチャは制御理論に着想を得ており、自動化、継続的インテグレーションと継続的デリバリー (CI/CD) パイプライン、機械学習オペレーションなどの分野で基盤となっています。

次の図は、フィードバックコントロールループの例です。



1. デプロイパイプラインは buildComplete イベントを出力します。
2. イベントは、ビルドを検証する自動テストジョブまたは評価ジョブをトリガーします。
3. 検証が失敗した場合 (テストの失敗、セキュリティの問題、ポリシー違反など)、システムは次の操作を行います。
   - buildComplete イベントを発行する
   - 問題をログに記録するか、通知を送信します
   - ロールバック、パッチ適用、再試行などの修復または修正アクションをトリガーします

ループは、許容可能な結果またはエスカレーションを生成するか、タイムアウトが発生するまで続行されます。このパターンは、一般的に以下に使用されます。

- 評価タスクまたは修復タスクにイベントをルーティングする Amazon EventBridge ルール
- AWS Step Functions 反復再試行ロジックと評価結果の分岐用
- フィードバックトリガーとアラートの Amazon Simple Notification Service (Amazon SNS) または Amazon CloudWatch アラーム
- AWS Lambda 是正措置を適用するための 関数またはコンテナ化されたワーカー

フィードバックコントロールループ (評価者)

評価者ワークフローは、LLMsまたは推論エージェントを搭載した認知フィードバックループです。このプロセスは、以下で構成されます。

1. ジェネレーターエージェントまたは LLM は出力 (計画、回答、ドラフトなど) を生成します。
2. 評価者エージェントは、批評プロンプトまたは評価ルーブリックを使用して結果を確認します。
3. フィードバックに基づいて、元のエージェントまたは新しいオプティマイザエージェントが出力を修正します。

ループは、結果が一連の基準を満たすか、承認されるか、再試行制限に達するまで繰り返されます。

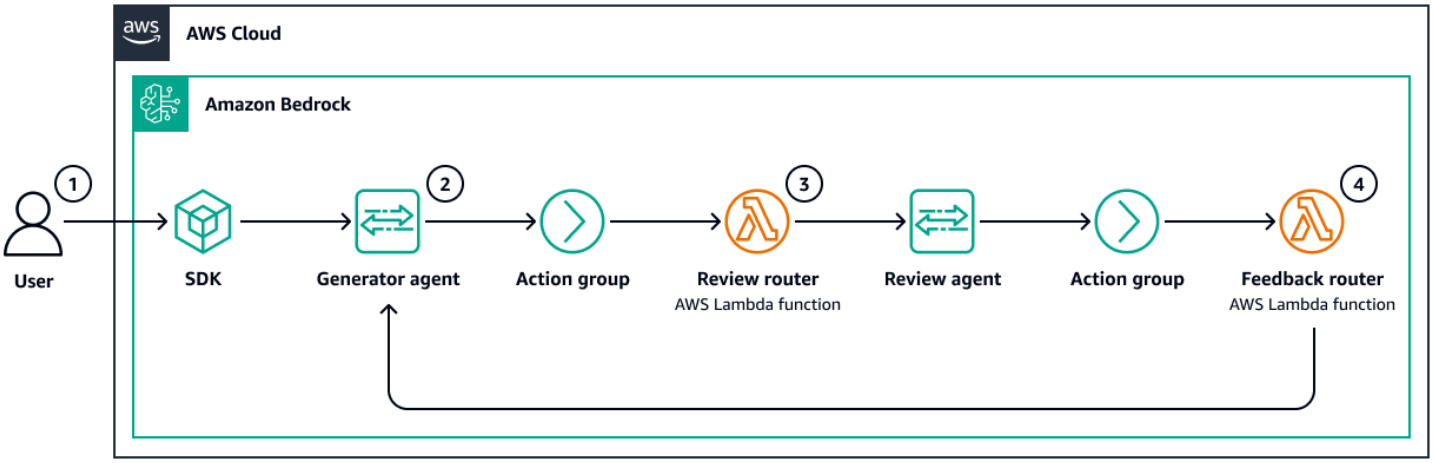
評価者

1. ユーザーは、ポリシーの概要を書き込むようにエージェントに依頼します。
2. ジェネレーターエージェントはそれをドラフトします。
3. 評価者エージェントは、カバレッジ、トーン、法的正確性をチェックします。

4. レスポンスが不十分な場合は、フィードバックループが収束するまで絞り込まれ、再送信されます。

これにより、自己評価、反復的な絞り込み、適応的な出力制御がすべて人間による入力なしで可能になります。

次の図は、フィードバックコントロールループ (評価者) の例です。



1. ユーザーがタスクを発行する (ビジネス戦略のドラフトを作成するなど) 。
2. Amazon Bedrock エージェントは、LLM を使用して最初のドラフトを生成します。
3. 2 番目のエージェント (またはフォローアッププロンプト) は構造化された評価 (たとえば、「この出力を明確さ、完全性、トーンで評価する」) を実行します。
4. 評価がしきい値を下回ると、レスポンスは次のように修正されます。
   - 批評が埋め込まれたジェネレーターの再呼び出し
   - 特殊なリファイナーエージェントにフィードバックを送信する
   - 許容可能な応答に達するまで反復する

AWS Lambda コントローラーや などのオプションコンポーネントは、フィードバックのしきい値、再試行、フォールバック戦略を管理 AWS Step Functions できます。

重要事項

従来のフィードバックコントロールループがイベント、メトリクス、修復ロジックを使用してシステム動作を検証および調整する場合、エージェント評価者ループは推論エージェントを使用して出力を動的に評価、反映、修正します。

両方のパラダイムで：

- 出力は生成後に評価されます
- 修正アクションまたは改良アクションは、フィードバックに基づいてトリガーされます
- システムが目標品質または目標に継続的に適応する

エージェントバージョンは静的検証をセマンティックリフレクションに変換し、独自の有効性を評価する自己改善エージェントを可能にします。

でのエージェントワークフローの設計 AWS

このガイドの各パターンは、 を使用して構築できます AWS のサービス。Amazon Bedrock エージェントは、オーケストレーション、データアクセス、インタラクションチャネルを提供します。

コンポーネント	AWS のサービス	目的
LLM の推論	Amazon Bedrock	エージェントロジック、計画、ツールの使用
ツールの実行	AWS Lambda、Amazon ECS、Amazon SageMaker	エージェント用の外部ツールをホストする
メモリと RAG	Amazon Bedrock ナレッジベース、Amazon S3、OpenSearch	永続メモリとセマンティックメモリ
オーケストレーション	AWS Step Functions	マルチステップタスクとエージェントの調整
イベントルーティング	Amazon EventBridge、Amazon SQS	分離されたエージェント間メッセージング
[ユーザーインターフェイス]	Amazon API Gateway、 AWS AppSync、 SDK	アプリケーションまたはシステムのエントリポイント
モニタリング	Amazon CloudWatch、 AWS X-Ray、 AWS Distro for OpenTelemetry	オブザーバビリティとエージェントのイントロスペクション

結論

エージェントワークフローパターンは、イベント駆動型アーキテクチャの次の進化段階であり、ビジネスロジックは静的に定義されず、大規模言語モデル (LLM) で強化された認識を使用して動的に推論されます。従来のクラウドネイティブプリミティブを LLM ワークフローやエージェント設計パターンと組み合わせることで、組織は目的に応じて対応し、経験から学ぶ適応型、インテリジェント、モジュラーシステムを構築できます。

これらのパターンでは、Amazon Bedrock はエージェント認識へのゲートウェイであり、LLM ベースのエージェントがイベントワークフローにアクセスし、ツールとメモリを操作し、構造化され、追跡可能で、調整された結果を提供できるようにします。

エージェントシステムを設計およびデプロイする際、これらのワークフローパターンは、自律的で構成可能な AI アーキテクチャを構築するための設計図を提供します。これらのシステムはサーバーレスのベストプラクティスに基づいており、インテリジェントな基盤モデルで強化されています。

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 規範ガイダンスが提供する戦略、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

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。

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。

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。信頼できる ID のみが、期待されるネットワークから信頼できる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「 Disaster Recovery of Workloads on AWS: Recovery in the Cloud」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。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

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、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で、障害の影響を制限し、ワークロードの耐障害性を向上させるのに役立ちます。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。

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、、 AWS Security Hub CSPM、Amazon GuardDuty、、 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。

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。

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。

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 から分離され、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、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 AWS マネジメントコンソール にログインしたり 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パスワードやユーザー認証情報などの AWS Secrets Manager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。

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。

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。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。

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。