のエージェント AI フレームワーク、プラットフォーム、プロトコル、ツール
AWS

AWS 規範ガイダンス

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 規範ガイダンス: のエージェント AI フレームワーク、プラットフォーム、プロトコル、ツール AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

序章 1
対象者 2
目的 2
このコンテンツシリーズについて 2
フレームワーク 3
Strands Agents 4
の主な機能 Strands Agents 4
Strands Agents を使用する場合 5
の実装アプローチ Strands Agents 5
の実例 Strands Agents 6
LangChain および LangGraph 6
LangChain および の主な機能 LangGraph 6
LangChain と を使用するタイミング LangGraph 7
LangChain と の実装アプローチ LangGraph 7
と の実際の例 LangChain LangGraph 8
CrewAI 8
の主な機能 CrewAI 8
CrewAI を使用する場合 9
の実装アプローチ CrewAI 9
の実例 CrewAI 10
AutoGen 10
の主な機能 AutoGen 11
AutoGen を使用する場合 11
の実装アプローチ AutoGen 12
の実例 AutoGen 12
LlamaIndex 12
の主な機能 LlamaIndex 13
LlamaIndex を使用する場合 14
の実装アプローチ LlamaIndex 14
の実例 LlamaIndex 14
エージェント AI フレームワークの比較 15
エージェント AI フレームワークの選択に関する考慮事項 16
[Platforms] (プラットフォーム) 18
プラットフォームが重要な理由 18

エージェント AI プラットフォームのタイプ 19
プラットフォーム選択に関する考慮事項 19
Amazon Bedrock エージェント 20
Amazon Bedrock エージェントの主な機能 20
Amazon Bedrock エージェントを使用するタイミング 21
Amazon Bedrock エージェントの実装アプローチ 21
Amazon Bedrock エージェントの実例 21
Amazon Bedrock AgentCore 22
AgentCore の主な機能 23
AgentCore を使用するタイミング 24
AgentCore の実装アプローチ 24
AgentCore の実際の例 25
プロトコル 26
プロトコルの選択が重要な理由 26
オープンプロトコルの利点 27
Agent-to-agentプロトコル 27
プロトコルオプション間の決定 28
エージェントプロトコルの選択 29
エージェントプロトコルの選択に関する考慮事項 29
エージェントプロトコルの実装戦略 30
MCP の開始方法 30
A2A の開始方法 31
ツール 33
ツールカテゴリ 33
プロトコルベースのツール 33
フレームワークネイティブツール 34
メタツール 34
プロトコルベースのツール 34
MCP ツールのセキュリティ機能 35
MCP ツールの開始方法 35
AgentCore Gateway の詳細 36
フレームワークネイティブツール 36
メタツール 37
ワークフローメタツール 37
エージェントグラフのメタツール 37
メモリメタツール 38

ツール統合戦略 38
ツール統合のセキュリティのベストプラクティス 39
認証と認可 39
データ保護 39
モニタリングと監査 40
結論 41
リソース 42
AWS ブログ 42
AWS 規範ガイダンス 42
AWS リソース 43
その他のリソース 43
ドキュメント履歴 44
..... xlv

のエージェント AI フレームワーク、プラットフォーム、プロトコル、ツール AWS

Aaron Sempf、Ansley Verzosa、Joshua Samuel、Amazon Web Services (AWS)

2026 年 1 月 (ドキュメント履歴)

エージェント AI は、AI、分散システム、ソフトウェアエンジニアリングの共通部分における強力なパラダイムです。AI モデルを使用し、ツールやリソースと統合する、自律型の非同期ソフトウェアエージェントで構成されるインテリジェントシステムのクラスです。エージェントは代理人として行動し、コンテキストや目標に対する理由を把握し、意思決定を行い、ユーザーやシステムに代わって意図的なアクションを実行できます。これらのエージェントは、分散環境内で独立して、多くの場合共同で動作し、インテリジェンス、メモリ、インテントが組み込まれた委任された目標を追求するように設計されています。

では AWS、エージェント AI を活用して複雑なワークフローを自動化し、意思決定プロセスを強化し、より応答性の高いシステムを作成できます。このガイドでは、効果的なエージェント AI ソリューションを構築するために必要な主要なコンポーネントについて説明します。

- フレームワークは、利点とユースケースのレビューなど、現在のエージェント AI フレームワークをプロファイリングします。これらのフレームワークが、パターン、プロトコル、ツール間の差別化されていない重労働をどのように軽減するかについて説明します。主要な選択基準を理解し、要件に適したフレームワークを選択します。
- プラットフォームは、エージェント AI プラットフォーム (マネージドエージェント、オープンソースオーケストレーション、ハイブリッド) の概要と、選択または設計に関する考慮事項を提供します。
- プロトコルは、エージェントとのやり取りに不可欠な標準化された通信プロトコルを探索します。Agent-to-agent プロトコルは、オープンソースの Model Context Protocol (MCP) や Agent2Agent (A2A) などの独自の実装とともに登場しています。一般的なプロトコルで、さまざまなプロトコルがシームレスにやり取りできるようにする方法について説明します。
- ツールには、プロトコルベースのツール (MCP など) 、フレームワークネイティブツール、メタツールに関する情報が用意されています。組織は、ワークフロー内のキーシステムと統合するツールキットを構築し、エンドユーザーとサーバーベースのエージェントワークフローの両方を実現できます。

対象者

このガイドは、最新のクラウドネイティブアプリケーション内で AI 駆動型ソフトウェアエージェントの能力を活用しようとしているアーキテクト、デベロッパー、テクノロジーリーダーを対象としています。

目的

このガイドは以下を行う際に役立ちます。

- さまざまなエージェント AI フレームワークを比較して、ユースケースに最適なものを選択します。
- 個々のエージェントを調整されたアダプティブシステムに変換する機能を提供するエージェント AI プラットフォームについて説明します。
- 持続可能なエージェント AI アーキテクチャを構築するためのオープンプロトコルの利点を理解します。
- エージェントシステムを構築するときに、適切なツール統合戦略を作成します。

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。

フレームワーク

のエージェント AI の基礎 AWSは、自律的で目標指向の動作を可能にするコアパターンとワークフローを調べます。これらのパターンを実装する中心にあるのは、フレームワークの選択です。フレームワークは、事前記述されたコードのソフトウェア基盤であり、構造化された環境と、本番環境対応の自律型 AI エージェントの構築に必要な構築と管理、ツール、オーケストレーション機能のための共通の機能を提供します。

効果的なエージェント AI フレームワークは、raw large language model (LLM) インタラクションを推論、コラボレーション、アクションが可能な調整されたインテリジェントシステムに変換するいくつかの重要な機能を提供します。

- エージェントオーケストレーションは、単一または複数のエージェント間の情報の流れと意思決定を調整し、人間の介入なしに複雑な目標を達成します。
- ツール統合により、エージェントは外部システム、APIs、データソースとやり取りして、言語処理を超えて機能を拡張できます。詳細については、 Strands Agentsドキュメントの「ツールの概要」を参照してください。
- メモリ管理は、永続的またはセッションベースの状態を提供し、インタラクション全体のコンテキストを維持します。これは、長時間実行されるタスクやアダプティブタスクに不可欠です。より高度なフレームワークには、概要とユーザー設定を保存する長期メモリが組み込まれているため、パーソナライズされたコンテキストに応じたエージェントエクスペリエンスが可能になります。詳細については、 LangChain ブログの「エージェントフレームワークについて考える方法」を参照してください。
- ワークフロー定義は、高度な自律推論を可能にするチェーン、ルーティング、並列化、リフレクションループなどの構造化パターンをサポートします。
- デプロイとモニタリングにより、自動システムのオブザーバビリティを備えた開発から本番環境への移行が容易になります。詳細については、Amazon Bedrock AgentCore の一般提供に関するお知らせを参照してください。

これらの機能は、フレームワーク環境全体でさまざまなアプローチと重点を置いて実装され、それぞれがさまざまな自律型エージェントのユースケースと組織コンテキストに異なる利点を提供します。

このセクションでは、エージェント AI ソリューションを構築するための主要なフレームワークをプロファイリングして比較し、その長所、制限、自律運用の理想的なユースケースに焦点を当てます。

- ストランドエージェント

- LangChain と LangGraph
- CrewAI
- AutoGen
- ???
- エージェント AI フレームワークの比較

> **Note**
>
> このセクションでは、AI の機関を特にサポートするフレームワークについて説明し、機関のないフロントエンドインターフェイスや生成 AI については説明しません。

Strands Agents

Strands Agents は、オープンソースAWS ブログで説明されているように AWS、 によって最初にリリースされたオープンソース SDK です。 Strands Agentsは、モデルファーストのアプローチで自律 AI エージェントを構築するように設計されています。 とシームレスに連携するように設計された柔軟で拡張可能なフレームワークを提供し AWS のサービス 、サードパーティーのコンポーネントとの統合にオープンなままです。Strands エージェントは、完全自律型ソリューションの構築に最適です。

の主な機能 Strands Agents

Strands Agents には、次の主要な機能が含まれています。

- モデルファースト設計 – 基盤モデルがエージェントインテリジェンスの中核であるという概念に基づいて構築され、高度な自律型推論を可能にします。詳細については、 Strands Agentsドキュメントの「エージェントループ」を参照してください。
- マルチエージェントコラボレーションパターン – 分散エージェントネットワーク全体でスケーラブルなコラボレーションとガバナンスを可能にする Swarm、Graph、Workflow パターンなどの組み込み調整モデル。詳細については、「 Strands Agents ドキュメント」の「Multi-agent Patterns」を参照してください。
- MCP 統合 – Model Context Protocol (MCP) のネイティブサポートにより、LLMs、一貫した自律オペレーションが可能になります。

- AWS のサービス 統合 – Amazon Bedrock、 など AWS のサービス へのシームレスな接続 AWS Step Functionsにより AWS Lambda、包括的な自律型ワークフローを実現します。詳細については、AWS 「週次ラウンドアップ (ブログ)」を参照してください。AWS
- 基盤モデルの選択 – Amazon Bedrock の Anthropic Claude、Amazon Nova (Premier、Pro、Lite、Micro) など、さまざまな基盤モデルをサポートして、さまざまな自律推論機能を最適化します。詳細については、 Strands Agentsドキュメントの「Amazon Bedrock」を参照してください。
- LLM API 統合 – Amazon Bedrock、OpenAI など、本番デプロイ用のさまざまな LLM サービスインターフェイスとの柔軟な統合。詳細については、 Strands Agentsドキュメントの「Amazon Bedrock の基本的な使用方法」を参照してください。
- マルチモーダル機能 – テキスト、音声、画像処理などの複数のモダリティをサポートし、包括的な自律型エージェントインタラクションを実現します。詳細については、 Strands Agentsドキュメントの「Amazon Bedrock マルチモーダルサポート」を参照してください。
- ツールエコシステム – AWS のサービス 対話のための豊富なツールセットと、自律機能を拡張するカスタムツールの拡張性。詳細については、 Strands Agentsドキュメントの「ツールの概要」を参照してください。

Strands Agents を使用する場合

Strands Agents は、特に次のような自律型エージェントのシナリオに適しています。

- 自動ワークフロー AWS のサービス のために とのネイティブ統合を必要とする AWS インフラストラクチャ上に構築される組織
- 本番稼働用自律システムのエンタープライズグレードのセキュリティ、スケーラビリティ、コンプライアンス機能を必要とするチーム
- 特殊な自律タスクのために異なるプロバイダー間でモデル選択に柔軟性を必要とするプロジェクト
- エンドツーエンドの自律プロセスのために既存の AWS ワークフローやリソースとの緊密な統合を必要とするユースケース

の実装アプローチ Strands Agents

Strands Agents は、 のクイックスタートガイドと のPythonクイックスタートガイドTypescriptで説明されているように、ビジネスステークホルダーに簡単な実装アプローチを提供します。このフレームワークにより、組織は次のことが可能になります。

- 特定のビジネス要件に基づいて、Amazon Bedrock の Amazon Nova (Premier、Pro、Lite、Micro) などの基盤モデルを選択します。
- エンタープライズシステムとデータソースに接続するカスタムツールを定義します。
- テキスト、画像、音声など、複数のモダリティを処理します。
- ビジネスクエリに自律的に応答し、タスクを実行できるエージェントをデプロイします。

この実装アプローチにより、ビジネスチームは AI モデル開発に関する深い技術的専門知識を必要とせずに、自律型エージェントを迅速に開発およびデプロイできます。

の実例 Strands Agents

AWS Transform for .NET Strands Agentsは、AWS Transform 「 for .NET」で説明されているように、 を使用してアプリケーションのモダナイゼーション機能を強化します。これは、.NET アプリケーションを大規模にモダナイズするための最初のエージェント AI サービスです (AWS ブログ)。この本稼働サービスは、複数の特殊な自律型エージェントを採用しています。エージェントは連携して、レガシー .NET アプリケーションを分析し、モダナイゼーション戦略を計画し、人間の介入なしにクラウドネイティブアーキテクチャへのコード変換を実行します。 AWS Transform for .NET は、エンタープライズ自律システムの Strands Agentsの本番稼働準備状況を示しています。

LangChain および LangGraph

LangChain は、エージェント AI エコシステムで最も確立されたフレームワークの 1 つです。 は、LangChain ブログで説明されているように、その機能LangGraphを拡張して、複雑でステートフルなエージェントワークフローをサポートします。これらを組み合わせることで、独立した運用のための豊富なオーケストレーション機能を備えた高度な自律型 AI エージェントを構築するための包括的なソリューションを提供します。

LangChain および の主な機能 LangGraph

LangChain および には、次の主要な機能LangGraphが含まれています。

- コンポーネントエコシステム – さまざまな自律型エージェント機能用の事前構築済みコンポーネントの広範なライブラリ。特殊なエージェントの迅速な開発を可能にします。詳細については、LangChainドキュメントの「クイックスタート」を参照してください。
- 基盤モデルの選択 – Anthropic Claude、Amazon Bedrock の Amazon Nova モデル (Premier、Pro、Lite、Micro) など、さまざまな推論機能のためのさまざまな基盤モデルをサポートします。詳細については、 LangChainドキュメントの「入出力」を参照してください。

- LLM API 統合 – Amazon Bedrock、 など、柔軟なデプロイOpenAIのための複数の大規模言語モデル (LLM) サービスプロバイダー向けの標準化されたインターフェイス。詳細については、LangChainドキュメントのLLMs」を参照してください。
- マルチモーダル処理 – テキスト、画像、オーディオ処理のサポートが組み込まれており、リッチなマルチモーダル自律型エージェントのインタラクションを可能にします。詳細については、LangChainドキュメントの「マルチモダリティ」を参照してください。
- グラフベースのワークフロー – 複雑な自律型エージェントの動作をステートマシンとして定義LangGraphし、高度な決定ロジックをサポートできます。詳細については、LangGraph「プラットフォーム GA の発表」を参照してください。
- メモリ抽象化 – 短期および長期のメモリ管理のための複数のオプション。これは、時間の経過とともにコンテキストを維持する自律型エージェントに不可欠です。詳細については、 LangChainドキュメントの「チャットボットにメモリを追加する方法」を参照してください。
- ツール統合 – さまざまなサービスと APIs、自律型エージェント機能を拡張します。詳細については、 LangChainドキュメントの「ツール」を参照してください。
- LangGraph プラットフォーム – 長時間稼働の自律型エージェントをサポートする、本番環境向けのマネージドデプロイおよびモニタリングソリューション。詳細については、LangGraph「プラットフォーム GA の発表」を参照してください。

LangChain と を使用するタイミング LangGraph

LangChain と LangGraphは、次のような自律型エージェントのシナリオに特に適しています。

- 自律的な意思決定のために高度なオーケストレーションを必要とする複雑な複数ステップの推論ワークフロー
- さまざまな自律機能のための構築済みコンポーネントと統合の大規模なエコシステムへのアクセスを必要とするプロジェクト
- 自動システムの構築を希望する既存の Pythonベースの機械学習 (ML) インフラストラクチャと専門知識を持つチーム
- 長時間実行される自律型エージェントセッション全体で複雑な状態管理を必要とするユースケース

LangChain と の実装アプローチ LangGraph

LangChain と は、 LangGraphドキュメントに詳述されているように、ビジネスステークホルダーに構造化された実装アプローチLangGraphを提供します。このフレームワークにより、組織は次のことが可能になります。

- ビジネスプロセスを表す高度なワークフローグラフを定義します。
- 決定ポイントと条件付きロジックを使用して、複数ステップの推論パターンを作成します。
- マルチモーダル処理機能を統合して、さまざまなデータ型を処理します。
- 組み込みのレビューおよび検証メカニズムを使用して、品質管理を実装します。

このグラフベースのアプローチにより、ビジネスチームは複雑な意思決定プロセスを自律型ワークフローとしてモデル化できます。チームは、推論プロセスの各ステップと、決定パスを監査する能力を明確に把握できます。

と の実際の例 LangChain LangGraph

Vodafone は、LangChainエンタープライズのケーススタディで説明されているように、 LangChain (および LangGraph) を使用して自動エージェントを実装し、データエンジニアリングとオペレーションのワークフローを強化しています。パフォーマンスメトリクスを自律的にモニタリングし、ドキュメントシステムから情報を取得し、自然言語インタラクションを通じて実用的なインサイトを提示する内部 AI アシスタントを構築しました。

このVodafone実装では、LangChainモジュラードキュメントローダー、ベクトル統合、および複数の LLMs (OpenAI、LLaMA#3、および Gemini) のサポートを使用して、これらのパイプラインを迅速にプロトタイプ化およびベンチマークします。次に、モジュール式のサブエージェントLangGraphをデプロイして、マルチエージェントオーケストレーションを構造化するために使用されます。これらのエージェントは、収集、処理、要約、推論タスクを実行します。 APIs を通じてこれらのエージェントをクラウドシステムLangGraphに統合しました。

CrewAI

CrewAI は、特に GitHub で利用可能な自律型マルチエージェントオーケストレーションに焦点を当てたオープンソースフレームワークです。人間の介入なしに複雑なタスクを解決するために協力する、特殊な自律型エージェントのチームを作成するための構造化されたアプローチを提供します。CrewAI は、ロールベースの調整とタスクの委任を強調します。

の主な機能 CrewAI

CrewAI には、次の主要な機能があります。

- ロールベースのエージェント設計 – 自律エージェントは、特殊な専門知識を実現するために、特定のロール、目標、バックストーリーで定義されます。詳細については、 CrewAIドキュメントの「Crafting Effective Agents」を参照してください。

- タスクの委任 – 機能に基づいて適切なエージェントにタスクを自動的に割り当てるための組み込みメカニズム。詳細については、 CrewAIドキュメントの「タスク」を参照してください。
- エージェントコラボレーション – 人間による仲介のない、エージェント間の自律的なコミュニケーションと知識共有のためのフレームワーク。詳細については、 CrewAIドキュメントの「コラボレーション」を参照してください。
- プロセス管理 – シーケンシャルおよび並列の自律タスク実行のための構造化ワークフロー。詳細については、 CrewAIドキュメントの「プロセス」を参照してください。
- 基盤モデルの選択 – Amazon Bedrock の Anthropic Claude、Amazon Nova モデル (Premier、Pro、Lite、Micro) など、さまざまな基盤モデルをサポートして、さまざまな自律推論タスクを最適化します。詳細については、 CrewAIドキュメントのLLMs」を参照してください。
- LLM API 統合 – Amazon Bedrock、、ローカルモデルのデプロイなどOpenAI、複数の LLM サービスインターフェイスとの柔軟な統合。詳細については、 CrewAIドキュメントの「プロバイダー設定の例」を参照してください。
- マルチモーダルサポート – テキスト、画像、その他のモダリティを処理し、包括的な自律型エージェントインタラクションを実現するための新しい機能。詳細については、 CrewAIドキュメントの「マルチモーダルエージェントの使用」を参照してください。

CrewAI を使用する場合

CrewAI は、次のような自律型エージェントのシナリオに特に適しています。

- 専門的なロールベースの専門知識を自律的に活用できる複雑な問題
- 複数の自律型エージェント間の明示的なコラボレーションを必要とするプロジェクト
- チームベースの問題分解が自律的な問題解決を改善するユースケース
- さまざまな自律エージェントロール間で懸念を明確に分離する必要があるシナリオ

の実装アプローチ CrewAI

CrewAI は、 CrewAIドキュメントの「開始方法」で説明されているように、ビジネスステークホルダー向けの AI エージェントアプローチのチームのロールベースの実装を提供します。このフレームワークにより、組織は次のことが可能になります。

- 特定のロール、目標、専門知識領域を持つ特殊な自律型エージェントを定義します。
- 特殊な機能に基づいてエージェントにタスクを割り当てます。
- タスク間に明確な依存関係を確立して、構造化ワークフローを作成します。

- 複数のエージェント間のコラボレーションを調整して、複雑な問題を解決します。

このロールベースのアプローチは人間のチーム構造を反映しており、ビジネスリーダーは直感的に理解して実装できます。組織は、ヒューマンチームの運営方法と同様に、ビジネス目標を達成するために協力する専門の専門知識領域を持つ自律型チームを作成できます。ただし、自律型チームは人間の介入なしに継続的に作業できます。

の実例 CrewAI

AWS は、CrewAI公開されたケーススタディで説明されているように、Amazon Bedrock と統合された CrewAI を使用して自律型マルチエージェントシステムを実装 AWS し、ベンダーに依存しない安全なフレームワークCrewAIを開発しました。CrewAI オープンソースの「フローとクルー」アーキテクチャは、Amazon Bedrock 基盤モデル、メモリシステム、コンプライアンスガードレールとシームレスに統合されます。

実装の主な要素は次のとおりです。

- 設計図とオープンソース – AWS および は、CrewAIエージェントを Amazon Bedrock モデルとオブザーバビリティツールにマッピングするリファレンスデザインをCrewAIリリースしました。https://aws.amazon.com/blogs/machine-learning/build-agentic-systems-with-crewai-and-amazon-bedrock/また、マルチエージェント AWS セキュリティ監査クルー、コードモダナイゼーションフロー、コンシューマーパッケージ製品 (CPG) バックオフィスオートメーションなどのサンプルシステムもリリースしました。
- オブザーバビリティスタックの統合 – このソリューションは、Amazon CloudWatch、AgentOps、および によるモニタリングを埋め込みLangFuse、概念実証から本番稼働までのトレーサビリティとデバッグを可能にします。
- 実証済みの投資利益率 (ROI) – 初期のパイロットでは、大規模なコードモダナイゼーションプロジェクトの実行が 70#% 高速化され、CPG バックオフィスフローの処理時間が約 90#% 短縮されました。

AutoGen

AutoGen は、 によって最初にリリースされたオープンソースフレームワークですMicrosoft。 は、会話型および協調型の自律型 AI エージェントを有効にすることにAutoGen重点を置いています。これは、複雑な自律ワークフローのためにエージェント間の非同期のイベント駆動型インタラクションに重点を置いたマルチエージェントシステムを構築するための柔軟なアーキテクチャを提供します。

の主な機能 AutoGen

AutoGen には、次の主要な機能があります。

- 会話エージェント – 自律型エージェント間の自然言語会話を中心に構築され、対話を通じて高度な推論を可能にします。詳細については、 AutoGenドキュメントの「マルチエージェント会話フレームワーク」を参照してください。
- 非同期アーキテクチャ – ノンブロッキングの自律型エージェントインタラクションのためのイベント駆動型設計で、複雑な並列ワークフローをサポートします。詳細については、 AutoGenドキュメントの「非同期チャットのシーケンスでの複数のタスクの解決」を参照してください。
- Human-in-the-loop – 必要に応じて、他の自律型エージェントワークフローへのオプションの人間の参加を強力にサポートします。詳細については、 AutoGenドキュメントの「 エージェントでのヒューマンフィードバックの許可」を参照してください。
- コードの生成と実行 — コードを記述して実行できるコードに焦点を当てた自律型エージェント専用の機能。詳細については、 AutoGenドキュメントの「コード実行」を参照してください。
- カスタマイズ可能な動作 – さまざまなユースケースに対応する柔軟な自律型エージェント設定と会話制御。詳細については、 AutoGenドキュメントの agentchat.conversable_agent を参照してください。
- 基盤モデルの選択 – Amazon Bedrock の Anthropic Claude、Amazon Nova モデル (Premier、Pro、Lite、Micro) など、さまざまな基盤モデルをサポートし、さまざまな自律推論機能を提供します。詳細については、 AutoGenドキュメントの「LLM 設定」を参照してください。
- LLM API 統合 – Amazon Bedrock、、 などOpenAI、複数の LLM サービスインターフェイスの標準化された設定Azure OpenAI。詳細については、 AutoGen API リファレンスの「oai.openai_utils」を参照してください。
- マルチモーダル処理 – リッチなマルチモーダル自律型エージェントインタラクションを可能にするテキストおよび画像処理をサポートします。詳細については、 AutoGenドキュメントの「マルチモーダルモデルの使用: GPT-4VAutoGen」を参照してください。

AutoGen を使用する場合

AutoGen は、次のような自律型エージェントシナリオに特に適しています。

- 複雑な推論のために自律型エージェント間で自然な会話フローを必要とするアプリケーション
- 完全自律型オペレーションとオプションの人的監視機能の両方を必要とするプロジェクト
- 人間の介入なしでの自律的なコード生成、実行、デバッグを含むユースケース

- 柔軟で非同期の自律型エージェントの通信パターンを必要とするシナリオ

の実装アプローチ AutoGen

AutoGen は、「 AutoGenドキュメント」の「開始方法https://microsoft.github.io/autogen/docs/Getting-Started」で説明されているように、ビジネスステークホルダー向けの会話型実装アプローチを提供します。このフレームワークにより、組織は次のことが可能になります。

- 自然言語の会話を通じて通信する自律型エージェントを作成します。
- 複数のエージェント間でイベント駆動型の非同期インタラクションを実装します。
- 必要に応じて、完全自律型オペレーションとオプションの人間による監視を組み合わせます。
- 対話を通じてコラボレーションするさまざまなビジネス機能に特化したエージェントを開発します。

この会話型アプローチは、自律システムの推論をビジネスユーザーが透過的に利用できるようにします。意思決定者は、エージェント間の対話を観察して結論がどのように達するかを理解し、必要に応じて人間の判断が必要なときに会話に参加できます。

の実例 AutoGen

Magentic-One は、Microsoft AI Frontiers ブログで説明されているように、さまざまな環境で複雑なマルチステップタスクを自律的に解決するように設計されたオープンソースのゼネラリストマルチエージェントシステムです。その中核となるのはオーケストレーターエージェントです。オーケストレーターエージェントは、高レベルの目標を分解し、構造化台帳を使用して進捗状況を追跡します。このエージェントは、サブタスクを特殊なエージェント (WebSurfer、、FileSurfer、 など ComputerTerminal) に委任しCoder、必要に応じて再計画することで動的に適応します。

システムはAutoGenフレームワーク上に構築され、モデルに依存しず、デフォルトで GPT-4o になります。GAIA、、 などのベンチマーク全体で、タスク固有の調整AssistantBenchWebArenaなしで最先端のパフォーマンスを実現します。さらに、AutoGenBench提案によるモジュール式の拡張性と厳格な評価もサポートしています。

LlamaIndex

LlamaIndex は、大規模言語モデル (LLMs) を外部データソースに接続して、高度な検索拡張生成 (RAG) およびエージェント AI アプリケーションを可能にするために特別に設計されたデータフレー

ムワークです。このフレームワークは、エージェントシステム、カスタムオーケストレーションパターン、システム統合のための抽象化と高速開発ワークフローを提供し、ナレッジ駆動型 AI ソリューションのtime-to-productionします。

の主な機能 LlamaIndex

LlamaIndex は、エンタープライズエージェント AI アプリケーションに特に適した包括的な機能を提供します。

- データ中心のアーキテクチャ – PDFs、MicrosoftWord ドキュメント、スプレッドシートなど、100 を超えるデータ形式からの情報の取り込み、インデックス作成、取得に優れています。このフレームワークは、エンタープライズデータを AI エージェント用に最適化されたクエリ可能なナレッジベースに変換します。詳細については、LlamaIndex のドキュメントを参照してください。
- 本番稼働対応デプロイ – LlamaIndexは、 を通じてオープンソースフレームワークとマネージドサービスの両方を提供しLlamaCloud、セキュリティコントロール、スケーラビリティ、オブザーバビリティ統合、デプロイの柔軟性などのエンタープライズグレードの機能を提供します。詳細については、LlamaIndexフレームワークドキュメントを参照してください。
- 高度なドキュメント処理 – 複雑なレイアウト、ネストされたテーブル、マルチモーダルコンテンツ、さらには手書きのメモを処理するドキュメント解析、抽出、インデックス作成、検索機能LlamaCloudを提供します。この高度な解析により、エージェントはグラフ、図、複雑なフォーマットを含む実際のエンタープライズドキュメントを効果的に操作できます。詳細については、LlamaCloud のドキュメントを参照してください。
- ワークフローオーケストレーション – 複数ステップのエージェントシステムを構築するためのイベント駆動型の非同期ファーストオーケストレーションエンジンLlamaAgentsを提供します。ワークフローは、ループ、並列実行、条件分岐、ステートフル再開などの複雑なパターンをサポートしているため、高度なエージェントインタラクションに最適です。詳細については、LlamaIndex ワークフローのドキュメントを参照してください。
- エージェント取り出し機能 – ハイブリッド検索、セマンティック検索、自動ルーティングなどの高度な取り出しモード。各クエリに最適な取り出し戦略をインテリジェントに決定します。このフレームワークは、精度を高めるために再ランク付けして、複数のナレッジベースにわたる複合取得をサポートします。詳細については、LlamaIndexRAG ドキュメントを参照してください。
- オブザーバビリティと評価 – さまざまなオブザーバビリティと評価ツールとLlamaIndex統合されます。この統合機能は、アプリケーションのトレースとデバッグ、パフォーマンスの評価、コストのモニタリングに役立ちます。詳細については、「トレースとデバッグと評価」のLlamaIndexドキュメントを参照してください。 https://developers.llamaindex.ai/python/framework/module_guides/evaluating

LlamaIndex を使用する場合

LlamaIndex は、データ集約型のワークフローとナレッジ管理を重視するエージェント AI シナリオに特に適しています。

- エージェントが契約、レポート、マニュアル、規制申請などの大量のエンタープライズドキュメントからインサイトを処理、分析、抽出する必要があるドキュメントの多いアプリケーション
- 組織が大規模なインフラストラクチャ管理オーバーヘッドなしでドキュメント中心のエージェントを迅速に構築およびデプロイしたい実稼働シナリオへの迅速なプロトタイプ
- 取得精度とコンテキストの関連性を優先する RAG ファーストアーキテクチャ。特にテーブル、イメージ、構造化データを含む複雑でマルチモーダルなドキュメントを操作する場合
- 解析、分析、要約、コンプライアンスチェックなど、ドキュメント処理のさまざまな側面に特化したエージェントを必要とするマルチエージェントドキュメントワークフロー

の実装アプローチ LlamaIndex

LlamaIndex は、さまざまな実装アプローチに対応する低レベルの構成要素と高レベルの抽象化の両方を提供します。

- LlamaIndex 高レベルの APIs。このアプローチにより、エージェント AI を初めて使用するビジネスチームや開発者がLlamaIndexアクセスできるようになります。
- SharePoint、Amazon Simple Storage Service (Amazon S3)、データベース、APIs などの一般的なエンタープライズシステム向けの LlamaHub を介したエンタープライズ統合。このアプローチにより、既存のデータインフラストラクチャとシームレスに統合できます。
- 最大制御のためのオープンソースのセルフホストデプロイと、運用オーバーヘッドとエンタープライズ機能を削減するためのLlamaCloudマネージドサービス間の柔軟なデプロイオプション。
- アプリケーションはシンプルなクエリエンジンから開始し、要件の進化に応じてエージェント機能、マルチエージェントオーケストレーション、複雑なワークフローを段階的に追加できます。

の実例 LlamaIndex

この例では、航空ナビゲーションおよび運用ソリューションを専門とする航空会社の子会社に焦点を当てています。調整されていない AI チャットボットトライアルのパイロット化に伴う課題の増大に対処する必要があります。このトライアルにより、組織全体で作業が繰り返され、開発サイクルが長くなり、コンプライアンスの障害が発生し、実装が分離されました。

エージェント作成をはるかに効率的にするLlamaIndexオープンソースフレームワーク上に構築された再利用可能なテンプレートベースのソリューションである統合エージェントフレームワークを開発しました。チェーン指向とグラフベースの競合するフレームワークをいくつか比較しました。最終的に、柔軟な設計、モジュラーコンポーネント、本番稼働対応のオーケストレーションコントロールという 3 つの重要な利点を選択LlamaIndexしました。

このプラットフォームは、エージェントの開発とデプロイの時間を 512 時間から 64 時間に 87% 短縮します。この削減は、チームが約 50 行のコードと JSON 設定ファイルを使用してエージェントを構築できるようにすることで実現されました。チームは、セキュリティ、コンプライアンス、特権システムアクセスが組み込まれた統合フレームワークを活用しました。詳細については、LlamaIndex「 カスタマーのケーススタディ」を参照してください。

エージェント AI フレームワークの比較

自律型エージェント開発用のエージェント AI フレームワークを選択するときは、各オプションが特定の要件にどのように適合するかを検討してください。技術的な能力だけでなく、チームの専門知識、既存のインフラストラクチャ、長期的なメンテナンス要件など、組織の適合性も考慮してください。多くの組織は、自律型 AI エコシステムのさまざまなコンポーネントに複数のフレームワークを活用して、ハイブリッドアプローチの恩恵を受ける可能性があります。

次の表は、主要な技術的側面における各フレームワークの成熟度レベル (最強、強、適切、弱) を比較したものです。この表には、フレームワークごとに、本番環境のデプロイオプションと学習曲線の複雑さに関する情報も含まれています。

Framework	AWS 統合	自動マルチエージェントサポート	自律ワークフローの複雑さ	マルチモーダル機能	基盤モデルの選択	LLM API 統合	本番稼働用デプロイ	学習曲線
AutoGen	弱い	強力	強力	適切	適切	強力	自分で行う (DIY)	急勾配
CrewAI	弱い	強力	適切	弱い	適切	適切	DIY	中

LangChain / LangGraph	適切	強力	最も強い	最も強い	最も強い	最も強い	プラットフォームまたは DIY	急勾配
LlamaIndex	適切	適切	強力	適切	強力	強力	プラットフォームまたは DIY	中
Strands Agents	最も強い	強力	最も強い	強力	強力	最も強い	DIY	中

エージェント AI フレームワークの選択に関する考慮事項

自律型エージェントを開発するときは、次の主要な要因を考慮してください。

- AWS インフラストラクチャ統合 – に大きく投資されている組織は AWS 、自律型ワークフロー AWS のサービス のために Strands Agentsと のネイティブ統合から最も恩恵を受けます。詳細については、AWS 「週次ラウンドアップ (ブログ)」を参照してください。AWS
- 基盤モデルの選択 – 自律型エージェントの推論要件に基づいて、優先基盤モデル (Amazon Bedrock の Amazon Nova モデルや Anthropic Claude など) に最適なサポートを提供するフレームワークを検討します。詳細については、 Anthropicウェブサイトの「効果的なエージェントの構築」を参照してください。
- LLM API 統合 – 本番デプロイ用の任意の大規模言語モデル (LLM) サービスインターフェイス (Amazon Bedrock や などOpenAI) との統合に基づいてフレームワークを評価します。詳細については、 ドキュメントの Strands Agents「モデルインターフェイス」を参照してください。
- マルチモーダル要件 – テキスト、画像、音声を処理する必要がある自律型エージェントの場合は、各フレームワークのマルチモーダル機能を検討してください。詳細については、 LangChain ドキュメントの「マルチモダリティ」を参照してください。
- 自律型ワークフローの複雑さ — 高度な状態管理を備えたより複雑な自律型ワークフローは、 の高度なステートマシン機能を優先する可能性がありますLangGraph。

- 自律型チームコラボレーション – 専門エージェント間の明示的なロールベースの自律型コラボレーションを必要とするプロジェクトは、 のチーム指向アーキテクチャからメリットを得ることができますCrewAI。
- 自律型開発パラダイム – 自律型エージェントの会話型非同期パターンを好むチームは、 のイベント駆動型アーキテクチャを好むかもしれませんAutoGen。
- マネージド型またはコードベースのアプローチ – 最小限のコーディングでフルマネージド型のエクスペリエンスを希望する組織は、Amazon Bedrock エージェントを検討する必要があります。より詳細なカスタマイズを必要とする組織は、特定の自律型エージェントの要件により適した特殊な機能を持つ Strands Agentsやその他のフレームワークを優先する場合があります。
- 自律システムの本番稼働準備 — 本番稼働用自律エージェントのデプロイオプション、モニタリング機能、エンタープライズ機能を検討します。

[Platforms] (プラットフォーム)

エージェント AI プラットフォームは、本稼働グレードのエージェントシステムのデプロイ、スケーリング、管理に必要な基本的なランタイム、オーケストレーション、統合レイヤーを提供します。フレームワークは、エージェントの構築方法を定義し、プロトコルはエージェントの通信方法を管理します。プラットフォームは、これらのエージェントが大規模に安全に運用、コラボレーション、進化する環境を提供します。

エージェントプラットフォームは、モデル実行、コンテキスト管理、ツール統合、オブザーバビリティ、ガバナンス機能を統合環境に組み合わせます。これらのプラットフォームにより、組織は実験からエンタープライズ規模のデプロイに移行できます。

このセクションでは、次の操作を行います。

- プラットフォームが重要な理由
- エージェント AI プラットフォームのタイプ
- プラットフォーム選択に関する考慮事項
- Amazon Bedrock エージェント
- Amazon Bedrock AgentCore

プラットフォームが重要な理由

エージェント AI プラットフォームは、本番環境で自律システムを運用しようとする組織にとって重要です。以下の機能を提供します。

- エージェントをホスト、スケーリング、調整するためのランタイムオーケストレーションを提供します。
- マルチエージェントワークフロー全体で状態、コンテキスト、メモリを管理します。
- エンタープライズ標準に沿ったセキュリティ、アイデンティティ、ガバナンスのコントロールを提供します。
- 標準 APIsまたはプロトコルを使用して、ツールエコシステムや外部システムと統合します。
- エージェントインタラクションとイベントフロー全体でオブザーバビリティと監査可能性を有効にします。
- クロスモデル相互運用性をサポートし、エージェントが 1 つの環境で複数の基盤モデルを使用できるようにします。

これらの機能により、個々のエージェントは、エンタープライズおよび規制の境界内で確実に動作できる調整された適応型システムになります。

エージェント AI プラットフォームのタイプ

エージェント AI プラットフォームは通常、次の 1 つ以上のカテゴリに分類されます。

- マネージドエージェント – フルマネージドプラットフォームは、組み込みのインフラストラクチャ、メモリ、オーケストレーション機能を提供します。これにより、運用上のオーバーヘッドが軽減され、本番稼働までの時間を短縮できます。
- オープンソースオーケストレーション – オープンソースのエージェントプラットフォームは、カスタマイズ可能な環境またはオンプレミスデプロイを好む組織に柔軟性と透明性を提供します。
- ハイブリッドエンタープライズ – ハイブリッドプラットフォームは、マネージドコンポーネントとセルフホストコンポーネントを統合し、クラウドマネージドサービスのスケーラビリティとエンタープライズシステムの制御を組み合わせます。

プラットフォーム選択に関する考慮事項

エージェント AI プラットフォームを選択または設計する場合、組織は次の点を考慮する必要があります。

- 統合の深さ — プラットフォームが既存のデータソース、ツール、プロトコルとどの程度統合されているかを評価します。
- スケーラビリティ – プラットフォームが自律的なワークロードとマルチエージェントコラボレーションをサポートするように動的にスケーリングできるようにします。
- セキュリティとコンプライアンス — 組織およびリージョンの要件に照らして、データのプライバシー、暗号化、ガバナンスの機能を評価します。
- 拡張性 – 新しいツール、モデル、またはエージェントを時間の経過とともに追加できるモジュールアーキテクチャのプラットフォームを選択します。
- オブザーバビリティ – エージェントインタラクションの詳細なテレメトリ、トレーサビリティ、監査ログを提供するプラットフォームを優先します。
- コスト効率 – サーバーレスモデルまたは使用量ベースのモデルを検討して、可変ワークロードのコストを最適化します。

Amazon Bedrock エージェント

Amazon Bedrock エージェントは、アプリケーションで自律型エージェントを構築および設定できるフルマネージドサービスです。基盤モデル、データソース、ソフトウェアアプリケーション、ユーザーとの会話間のやり取りをオーケストレーションできます。エージェントの作成に対する合理化されたアプローチでは、容量のプロビジョニング、インフラストラクチャの管理、カスタムコードの記述を行う必要はありません。

Amazon Bedrock エージェントの主な機能

Amazon Bedrock エージェントには、次の主要な機能があります。

- フルマネージドサービス – 容量をプロビジョニングしたり、基盤となるシステムを管理したりすることなく、インフラストラクチャ管理を完了します。詳細については、Amazon Bedrock ドキュメントの「AI エージェントを使用してアプリケーションのタスクを自動化する」を参照してください。
- API 駆動型開発 – モデル、手順、ツール、設定パラメータを指定して、シンプルな API コールを通じてエージェントを定義して実行します。詳細については、Amazon Bedrock ドキュメントの「エージェントを手動で作成および設定する」を参照してください。
- アクショングループ – API スキーマを使用してアクショングループを作成して、エージェントが実行できる特定のアクションを定義します。詳細については、Amazon Bedrock ドキュメントの「アクショングループを使用してエージェントが実行するアクションを定義する」を参照してください。
- ナレッジベースの統合 – Amazon Bedrock ナレッジベースにシームレスに接続して、組織のデータでエージェントのレスポンスを強化します。詳細については、Amazon Bedrock ドキュメントの「ナレッジベースを使用したエージェントの Augment レスポンス生成」を参照してください。
- 高度なプロンプトテンプレート – 前処理、オーケストレーション、ナレッジベースのレスポンス生成、後処理用のプロンプトテンプレートを使用してエージェントの動作をカスタマイズします。詳細については、Amazon Bedrock ドキュメントの「Amazon Bedrock の高度なプロンプトテンプレートを使用したエージェントの精度の向上」を参照してください。
- トレースとオブザーバビリティ – 組み込みトレース機能を使用して、エージェントのstep-by-stepの推論プロセスを追跡します。詳細については、Amazon Bedrock ドキュメントの「トレースを使用してエージェントのstep-by-step推論プロセスを追跡する」を参照してください。
- バージョニングとエイリアス – エージェントの複数のバージョンを作成し、制御されたロールアウトのエイリアスを介してデプロイします。詳細については、「Amazon Bedrock ドキュメント」

の「アプリケーションで Amazon Bedrock エージェントをデプロイして使用する」を参照してください。

Amazon Bedrock エージェントを使用するタイミング

Amazon Bedrock エージェントは、次のような自律型エージェントのシナリオに特に適しています。

- インフラストラクチャを管理せずにエージェントを構築およびデプロイするためのフルマネージド型のエクスペリエンスを必要とする組織
- コードではなく設定によるエージェントの迅速な開発とデプロイを必要とするプロジェクト
- ナレッジベースやガードレールなどの他の Amazon Bedrock 機能との緊密な統合からメリットを得るユースケース
- エージェントをゼロから構築するための社内リソースがないが、本番環境対応の自律機能が必要なチーム

Amazon Bedrock エージェントの実装アプローチ

Amazon Bedrock エージェントは、ビジネスステークホルダー向けに設定ベースの実装アプローチを提供します。このサービスにより、組織は次のことが可能になります。

- 複雑なコードを記述せずに、 AWS マネジメントコンソール または API コールを通じてエージェントを定義します。
- エージェントが実行できる APIsとオペレーションを指定するアクショングループを作成します。
- ナレッジベースを接続して、ドメイン固有の情報をエージェントに提供します。
- ビジュアルインターフェイスを使用して、エージェントの動作をテストして反復処理します。

このマネージド型アプローチにより、ビジネスチームは AI モデル開発やインフラストラクチャ管理に関する深い技術的専門知識を必要とせずに、自律型エージェントを迅速に開発およびデプロイできます。

Amazon Bedrock エージェントの実例

このAWS ブログ記事で説明されている財務オペレーション (FinOps) ソリューションは、Amazon Bedrock マルチエージェントフレームワークを使用して AI 駆動型のクラウドコスト管理アシスタントを作成します。費用対効果の高い Amazon Nova 基盤モデルは、中央の FinOps スーパーバイザーエージェントがタスクを専門エージェントに委任するソリューションを強化します。これらのエー

ジェントは、 を使用して AWS 支出データを取得および分析 AWS Cost Explorer し、 を使用してコスト削減の推奨事項を生成します AWS Trusted Advisor。

このシステムには、Amazon Cognito を介した安全なユーザーアクセス、 でホストされるフロントエンド AWS Amplify、リアルタイム分析と予測のための AWS Lambda アクショングループが含まれています。財務チームは、「2025 年 2 月のコストはいくらだったか」などの自然言語クエリを聞くことができます。システムは、詳細な内訳、最適化の提案、予測で応答します。これらはすべて、 を使用してデプロイされたスケーラブルなサーバーレスアーキテクチャ内で行われます AWS CloudFormation。

Amazon Bedrock AgentCore

Amazon Bedrock AgentCore は、あらゆるフレームワーク、モデル、プロトコルを使用して、高機能エージェントを大規模に安全に構築、デプロイ、運用するためのエージェントプラットフォームです。AgentCore を使用すると、インフラストラクチャ管理なしで、以下を実行できます。

- エージェントをより迅速に構築します。
- エージェントがツールとデータ間でアクションを実行できるようにします。
- 低レイテンシーと拡張ランタイムでエージェントを安全に実行します。
- 本番環境のエージェントをモニタリングします。

AgentCore は、特殊なエージェントインフラストラクチャの構築に伴う差別化されていない負担を排除し、エージェントの本番稼働までの時間を短縮します。そのサービスは一緒に使用することも独立して使用することもできます。また、CrewAI、、LangGraphLlamaIndex、 などのフレームワークと互換性がありますStrands Agents。AgentCore は、Amazon Bedrock の内外で利用可能な基盤モデルとも互換性があり、最高の柔軟性を提供します。

AgentCore は、いくつかの主要なサービスで構成されています。

- Amazon Bedrock AgentCore ランタイム – AI エージェントやツールのデプロイと実行に必要なインフラストラクチャを管理することなく、エージェントをホストして実行するための安全でサーバーレスでスケーラブルな環境を提供します。
- Amazon Bedrock AgentCore Memory – マネージドメモリシステムを提供し、エージェントは即時および長期的な知識を維持することで、よりパーソナライズされた一貫性のある会話のためにインタラクションのコンテキストを保持できます。
- Amazon Bedrock AgentCore Gateway – エージェントに適したツールを作成、保護、検索するプロセスを簡素化します。AgentCore Gateway を使用すると、デベロッパーは APIs、Lambda 関数、

および既存のサービスをモデルコンテキストプロトコル (MCP) 互換ツールに変換し、エージェントに提供できます。

- Amazon Bedrock AgentCore Identity – AI エージェントの開発を加速する、安全でスケーラブルなエージェント ID とアクセス管理サービスを提供します。AgentCore Identity を使用すると、検証可能な一意の ID をエージェントに割り当てることができ、きめ細かなアクセスコントロールと、エージェントによるエンタープライズシステムとの安全なやり取りが可能になります。
- Amazon Bedrock AgentCore 組み込みツール – 組み込みツールを使用して開発とテストのワークフローを強化できます。これらのツールを使用してアプリケーションを効果的に操作し、AI エージェントがサンドボックス環境で安全にコードを記述して実行できるようにします。ブラウザツールを使用して、AI エージェントがウェブサイトを大規模に操作できるようにします。
- Amazon Bedrock AgentCore Observability – ログ記録とモニタリング機能を提供し、エージェントのパフォーマンスと動作をリアルタイムで可視化して、デバッグと最適化を容易にします。

AgentCore の主な機能

AgentCore には、次の主要な機能が含まれています。

- フルマネージド型で拡張可能 – AgentCore はフルマネージド型サービスです。つまり、 は基盤となるインフラストラクチャとメンテナンス AWS を処理します。また、拡張可能で、エージェントの機能をカスタマイズして強化できます。詳細については、AgentCore ドキュメントの「AgentCore ランタイムの開始方法」を参照してください。 AgentCore
- 長期メモリと短期メモリ – 現在の会話や長期的な知識のコンテキストを再現するメモリシステムをエージェントに装備することで、よりパーソナライズされた関連性の高いインタラクションを提供します。詳細については、「 AgentCore ドキュメント」の「AgentCore メモリの開始方法」を参照してください。 AgentCore
- ツールの開発と統合の簡素化 – エージェントが単一の安全なエンドポイントを通じてツールを検出して使用できるようにします。既存のエンタープライズリソースをわずか数行のコードでエージェント対応ツールにすばやく変換できるため、開発者は独自の機能の構築に集中できます。詳細については、AgentCore ドキュメントの「AgentCore Gateway の開始方法」を参照してください。 AgentCore
- 安全でスケーラブルなインフラストラクチャ – AgentCore は、エージェントのデプロイと運用のための安全でスケーラブルな環境を提供します。これには、ID とアクセスの管理、データ暗号化、ネットワークセキュリティの機能が含まれています。詳細については、AgentCore ドキュメントの「AgentCore Identity の開始方法」を参照してください。 AgentCore

- さまざまなツールとの統合 – エージェントを、コードインタープリタや AgentCore 組み込みツールを使用して構築できるブラウザツールなど、さまざまなツールと統合できます。詳細については、AgentCore ドキュメントの「AgentCore Code Interpreter の開始方法」およびAgentCore Browser の開始方法」を参照してください。 AgentCore
- 包括的なオブザーバビリティとモニタリング – 包括的なツールを使用してエージェントを詳細に可視化し、本番環境でのパフォーマンスをトレース、デバッグ、モニタリングします。エージェントの実行パス全体を視覚化して、推論を監査し、障害を解決します。リアルタイムのダッシュボードと標準化されたテレメトリデータを使用して、主要な運用メトリクスを追跡します。詳細については、AgentCore ドキュメントの「Amazon Bedrock AgentCore リソースにオブザーバビリティを追加する」を参照してください。 AgentCore

AgentCore を使用するタイミング

AgentCore は、次のような自律型エージェントシナリオに特に適しています。

- インフラストラクチャ、セキュリティ、組み込みツール、オブザーバビリティ、スケーリングを処理するフルマネージドサービスを使用して、開発を加速し、運用オーバーヘッドを削減したい組織
- 連携または独立して動作し、 CrewAI や などのフレームワークLangGraph、および任意のソースからの基盤モデルと互換性があるモジュラーサービスによる柔軟性を必要とするプロジェクト
- コンテキストを維持し、過去のやり取りから学び、パーソナライズされた関連する応答を提供する必要がある、ステートフルで会話型のエージェントを必要とするユースケース
- さまざまなアプリケーション、データソース、APIs と簡単に統合することで複雑なタスクを実行できるエージェント

AgentCore の実装アプローチ

AgentCore は、オープンソースまたはカスタムエージェントフレームワークを使用して構築された概念実証から本番稼働に移行する組織向けに設計されています。AgentCore を使用すると、組織は以下を実行できます。

- エージェントをサーバーレスインフラストラクチャに安全にデプロイし、エンドツーエンド end-to-end のセキュリティとコンプライアンスのためのセッション分離と組み込みの ID とアクセス管理により、あらゆるフレームワークとモデルをサポートします。スターターツールキットを使用して、主要なエージェントフレームワークの AgentCore ランタイムエージェントをすばやく作成します。

- エージェントを強化するには、永続メモリを統合してコンテキストを保持し、AgentCore Gateway を介したツールの開発と統合を簡素化します。組み込みのブラウザツールとコードインタープリタを高度なワークフローに活用します。
- Amazon CloudWatch Application Insights と を搭載したオブザーバビリティダッシュボードを使用して、本番環境の AI エージェントをトレース、デバッグ、モニタリングし OpenTelemetry、AgentCore リソース (ランタイム、メモリ、ゲートウェイ、ツール) の主要なメトリクスを追跡します。
- フルマネージドのモジュラーサービス、組み合わせ可能なブロックをまとめて、または個別に、エージェントフレームワークとモデルプロバイダーを使用して、デプロイとイノベーションを加速します。この柔軟性により、組織はプロトタイプから本番稼働への移行を迅速化できます。

このマネージド型アプローチにより、組織はあらゆる規模でエンタープライズグレードの AI エージェントとマルチエージェントシステムを迅速かつ安全に構築、デプロイ、実行できます。

AgentCore の実際の例

AWS は、ラテンアメリカ最大の銀行の 1 つが AI/ML を長年使用して、ハイパーパーソナライズされた安全なデジタルバンキングエクスペリエンスを提供していることを確認しました。銀行は AgentCore を使用してエージェント AI サービスを拡張し、直感的なインタラクション、セキュリティの強化、自動化の強化をお客様に提供しています。CTO によると、AgentCore は顧客コミットメントを大規模に達成するための取り組みをサポートすることが期待されています。AgentCore は、開発者にエージェントを構築および管理するためのツールと柔軟性を提供し、財務規制へのコンプライアンスを確保します。

プロトコル

AI エージェントは、他のエージェントやサービスとやり取りするために標準化された通信プロトコルを必要とします。エージェントアーキテクチャを実装している組織は、相互運用性、ベンダーの独立性、投資の将来性に関して大きな課題に直面しています。

このセクションでは、柔軟性と相互運用性を最大化するオープンスタンダードに焦点を当ててagent-to-agentプロトコルランドスケープをナビゲートするのに役立ちます。(agent-to-toolプロトコルの詳細については、このガイドの後半にある「ツール統合戦略」を参照してください)。

このセクションでは、2024 Anthropic 年に によって最初に開発されたオープンスタンダードである Model Context Protocol (MCP) について説明します。現在、 はプロトコルの開発と実装に貢献することで MCP AWS を積極的にサポートしています。 AWS は、、LangGraph、 CrewAIなどの主要なオープンソースエージェントフレームワークと協力してLlamaIndex、プロトコルでのエージェント間通信の未来を形成しています。詳細については、「Open Protocols for Agent Interoperability Part 1: Inter-Agent Communication on MCP」(AWS ブログ) を参照してください。

このセクションでは、次の操作を行います。

- プロトコルの選択が重要な理由
- Agent-to-agentプロトコル
- エージェントプロトコルの選択
- エージェントプロトコルの実装戦略
- MCP の開始方法
- ???

プロトコルの選択が重要な理由

プロトコルの選択は、AI エージェントアーキテクチャを構築および進化する方法を根本的に形成します。エージェントフレームワーク間の移植性をサポートするプロトコルを選択することで、特定のニーズに合わせてさまざまなエージェントシステムとワークフローを柔軟に組み合わせることができます。

オープンプロトコルを使用すると、エージェントを複数のフレームワークに統合できます。たとえば、 LangChainを使用してラピッドプロトタイプを作成し、 を使用して本番稼働用システムを実装

しStrands Agents、MCP や Agent2Agent (A2A) プロトコルなどの一般的なプロトコルを介して通信します。この柔軟性により、特定の AI プロバイダーへの依存を減らし、既存のシステムとの統合を簡素化し、時間の経過とともにエージェントの機能を強化できます。

また、適切に設計されたプロトコルは、エージェントエコシステム全体で認証と認可のための一貫したセキュリティパターンを確立します。最も重要なのは、プロトコルの移植性により、新しいエージェントフレームワークと機能の導入の自由が維持されることです。オープンプロトコルを選択すると、サードパーティーシステムとの相互運用性を維持しながら、エージェント開発への投資が保護されます。

オープンプロトコルの利点

独自の拡張機能を実装する場合やカスタムエージェントシステムを構築する場合、オープンプロトコルには魅力的な利点があります。

- ドキュメントと透明性 — 通常、包括的なドキュメントと透過的な実装を提供します。
- コミュニティサポート – トラブルシューティングとベストプラクティスのためのより広範な開発者コミュニティへのアクセス
- 相互運用性の保証 – 拡張機能がさまざまな実装で機能する保証を強化
- 将来の互換性 — 変更が中断されたり、廃止されたりするリスクを軽減
- 開発への影響 — プロトコルの進化に貢献する機会

Agent-to-agentプロトコル

次の表は、複数のエージェントがコラボレーション、タスクの委任、および情報の共有を可能にするエージェントプロトコルの概要を示しています。

[プロトコル]	に最適	考慮事項
MCP エージェント間通信	柔軟なエージェントコラボレーションパターンを求める組織	• エージェントagent-to-agent通信の既存の基盤を構築する AWS 、 によって提案されたモデルコンテキストプロトコル (MCP) の拡張 • OAuth ベースのセキュリティによるシームレスなエー

		ジェントコラボレーションを実現
A2A プロトコル	クロスプラットフォームエージェントエコシステム	• によってバックアップ Google • MCP と比較して導入が限られている新しい標準

プロトコルオプション間の決定

agent-to-agent通信を実装する場合は、特定の通信要件を適切なプロトコル機能と一致させます。異なるインタラクションパターンには、異なるプロトコル機能が必要です。次の表は、一般的な通信パターンの概要を示し、各シナリオに最適なプロトコルの選択を推奨しています。

パターン:	説明	理想的なプロトコルの選択
シンプルなリクエストとレスポンス	エージェント間の 1 回限りのインタラクション	ステートレスフローを持つ MCP
ステートフルな対話	コンテキストを使用した継続的な会話	セッション管理による MCP
マルチエージェントコラボレーション	複数のエージェント間の複雑なやり取り	MCP エージェント間または AutoGen
チームベースのワークフロー	ロールが定義された階層エージェントチーム	MCP エージェント間、CrewAI、または AutoGen

コミュニケーションパターン以外にも、いくつかの技術的および組織的な要因がプロトコルの選択に影響を与える可能性があります。次の表は、特定の実装要件に最も近いプロトコルを評価するのに役立つ重要な考慮事項の概要を示しています。

考慮事項	説明	例
セキュリティモデル	認証と認可の要件	MCP の OAuth 2.0

デプロイ環境	エージェントが実行して通信する場所	分散マシンまたは単一マシン
エコシステムの互換性	既存のエージェントフレームワークとの統合	LangChain、または Strands Agents
スケーラビリティのニーズ	エージェントインタラクションの予想される増加	MCP のストリーミング機能

エージェントプロトコルの選択

本番稼働用エージェントシステムを構築するほとんどの組織では、モデルコンテキストプロトコル (MCP) はagent-to-agent通信の最も包括的で十分にサポートされている基盤を提供します。MCP は、 AWS およびオープンソースコミュニティからの積極的な開発貢献からメリットを得ます。

適切なエージェントプロトコルを選択することは、エージェント AI を効果的に実装したい組織にとって重要です。考慮事項は、組織のコンテキストによって異なります。

エージェントプロトコルの選択に関する考慮事項

エージェント AI システムのプロトコルを選択するときは、次のベストプラクティスを考慮する必要があります。

- オープンスタンダードの優先順位付け – 組織は、長期的な相互運用性、拡張性を確保し、ベンダーロックインのリスクを減らすために、MCP などのオープンプロトコルを採用する必要があります。
- スピードと柔軟性のバランス — スタートアップと早期導入者は、迅速な開発のために十分にサポートされている独自のプロトコルから始めることができますが、システムが成熟するにつれて標準を開くための移行パスを定義する必要があります。
- 抽象化レイヤーを実装する – 企業はプロトコル抽象化を実装して、移行を簡素化し、ハイブリッド導入を可能にし、将来を見越した統合戦略を実現する必要があります。
- セキュリティとコンプライアンスを強調する – 規制対象業界の組織は、ガバナンスとコンプライアンスの要件を満たすために、堅牢な認証、暗号化、監査機能を備えたプロトコルを選択する必要があります。
- エコシステムの成熟度を評価する – すべての組織は、持続可能性を確保し、技術的負債を最小限に抑えるために、各プロトコルのヘルス、導入、コミュニティサポートを評価する必要があります。

- 標準の開発に取り組む – 組織は、プロトコルの進化を形成し、ベストプラクティスに影響を与えるために、標準団体またはオープンソースコミュニティに参加する必要があります。
- データ主権の考慮 — 政府機関と規制対象セクターは、プロトコルの選択が、デプロイリージョン全体のデータレジデンシーと主権の要件と一致していることを確認する必要があります。
- マネージドサービスを活用する – 可能な限り、エージェントプロトコルのマネージド実装またはサーバーレス実装を使用して、運用の複雑さを軽減し、デプロイを高速化します。

エージェントプロトコルの実装戦略

組織全体でエージェントプロトコルを効果的に実装するには、以下の戦略的ステップを検討してください。

1. 標準の調整から始める – 可能な場合は、確立されたオープンプロトコルを採用します。
2. 抽象化レイヤーの作成 – システムと特定のプロトコルの間にアダプターを実装します。
3. オープンスタンダードに貢献 — プロトコル開発コミュニティに参加します。
4. プロトコルの進化をモニタリングする – 新しい標準と更新について常に情報を得ます。
5. 相互運用性を定期的にテストする – 実装に互換性があることを確認します。

MCP の開始方法

AWS は、プロトコルの開発と実装に貢献することで、モデルコンテキストプロトコル (MCP) を積極的にサポートしています。 AWS は、LangGraph、、 CrewAIなどの主要なオープンソースエージェントフレームワークと連携してLlamaIndex、プロトコルでのエージェント間通信の未来を形成しています。

エージェントアーキテクチャに MCP を実装するには、次のアクションを実行します。

1. Strands Agents SDK などのフレームワークでの MCP 実装について説明します。
2. Model Context Protocol の技術ドキュメントを確認してください。
3. エージェント相互運用性のオープンプロトコル パート 1: MCP でのエージェント間通信 (AWS ブログ) を読み、エージェントの相互運用性について学びます。
4. MCP コミュニティに参加して、プロトコルの進化に影響を与えます。

MCP は、エージェントが外部データやサービスとやり取りできるようにする通信レイヤーを提供し、エージェントが他のエージェントとやり取りできるようにするためにも使用できます。プロト

コルの Streamable HTTP トランスポート実装により、デベロッパーはホイールを再考案することなく、包括的な一連のインタラクションパターンを使用できます。これらのパターンは、ステートレスリクエスト/レスポンスフローと永続的 IDs を使用したステートフルセッション管理の両方をサポートします。

MCP などのオープンプロトコルを採用することで、AI テクノロジーの進化に合わせて柔軟性、相互運用性、適応性を維持するエージェントシステムを構築するように組織を配置できます。agent-to-toolプロトコルの実装の詳細については、このガイドの後半にある「ツール統合戦略」を参照してください。

A2A の開始方法

Agent2Agent (A2A) プロトコルは、共有セマンティックレイヤーを介してエージェント間の分散コラボレーションを可能にします。A2A では、すべての作業を中央オーケストレーター経由でルーティングする代わりに、エージェントは軽量の JSON ベースのプロトコルを使用して、相互検出、機能のアドバタイズ、タスクのネゴシエート、コンテキストの共有を行うことができます。各エージェントは機能マニフェストを発行します。

次の例は、エージェントのサポートされているアクション、必要な入力、運用メタデータをアドバタイズして検出とタスクネゴシエーションを可能にする、簡略化された A2A 機能マニフェストを示しています。

```
{
  "can": ["summarize.text", "extract.keywords"],
  "needs": ["document.input"],
  "meta": { "version": "1.0.3", "latencyMs": 120 }
}
```

このモデルにより、動的な機能マッチング、タスク中の委任、組織間のコラボレーションが可能になります。エージェントはタスクを中心に自己編成し、一時的な作業グループを形成し、新しい機能がシステムに出入りするにつれて適応できます。

A2A は、シンプルなステートレスリクエストから複数ステップのネゴシエーションセッションまで、次のようなインタラクションをサポートします。

- peer-to-peerのダイレクトメッセージングによる低レイテンシーのコラボレーション
- エージェントが最適なピアを選択するセマンティックタスクネゴシエーション
- 能力ベースの検出により、緊急の作業分担を実現

- ステートフルなマルチステップインタラクションのためのセッションアンカーリング

A2A のようなオープンでエージェントネイティブなプロトコルを採用することで、組織はモジュラーで相互運用可能で、クロスボーダーコラボレーションが可能な AI システムを作成します。A2A は、エージェントのエコシステムを柔軟に保ち、リジッドオーケストレーションレイヤーや事前の結合を必要とせずに、新しいエージェント、チーム、または外部システムの導入に合わせて進化させることができます。

エージェントアーキテクチャに A2A プロトコルを実装するには、次のアクションを実行します。

1. A2A プロトコル仕様を確認する – Agent2Agent (A2A) プロトコル仕様の最新バージョンを読み、機能マニフェスト、ネゴシエーションフロー、エージェントのハンドシェイクがどのように動作するかを確認してください。
2. A2A-compatibleランタイムの探索 – Strands Agents SDK などのフレームワークやA2A-style機能マニフェストとpeer-to-peerネゴシエーションをサポートするカスタムランタイムレイヤーを評価します。
3. エージェントの機能マニフェストを実装する – 各エージェントの `can`、`needs`、および `meta`フィールドを定義して、検出、マッチメーキング、インテントレベルのコラボレーションを有効にします。
4. A2A ネゴシエーションパターンを試す – リクエストオファー承諾ループ、構造化された機能クエリ、または gossip ベースの検出を使用して、エージェントがタスクを処理するべきユーザーについてどのような理由があるかを理解します。
5. 混合インフラストラクチャ環境で A2A をテストする – A2A ピアネゴシエーションを Amazon EventBridge AWS を介して にネイティブなイベントルーティングと組み合わせて、ハイブリッド調整パターンを評価します。
6. A2A コミュニティに参加する – オープンワーキンググループと連携し、拡張機能、セキュリティの推奨事項、ベンダー間の相互運用性の改善について最新の状態を維持し、プロトコルの開発に貢献します。

ツール

AI エージェントは、外部ツール、APIs、データソースを操作して有用なタスクを実行することで価値を提供します。適切なツール統合戦略は、エージェントの機能、セキュリティ体制、長期的な柔軟性に直接影響します。

このセクションでは、自由と柔軟性を最大化するオープンスタンダードに焦点を当てて、ツール統合の状況をナビゲートするのに役立ちます。このセクションでは、ツール統合用の Model Context Protocol (MCP) に焦点を当て、エージェントワークフローを強化するフレームワーク固有のツールと特殊なメタツールを確認します。

このセクションでは、次の操作を行います。

- ツールカテゴリ
- プロトコルベースのツール
- フレームワークネイティブツール
- メタツール
- ツール統合戦略
- ツール統合のセキュリティのベストプラクティス

ツールカテゴリ

エージェントシステムの構築には、主に 3 つのカテゴリのツールが含まれます。

プロトコルベースのツール

プロトコルベースのツールは、agent-to-tool間の通信に標準化されたプロトコルを使用します。

- MCP ツール – ローカル実行オプションとリモート実行オプションの両方でフレームワーク間で機能する標準ツールを開きます。
- OpenAI 関数呼び出し — OpenAIモデルに固有の独自のツール。
- Anthropic ツール – Anthropic Claude モデルに固有の独自のツールを呼び出すOpenAI関数のバリエーション。

フレームワークネイティブツール

フレームワークネイティブツールは、特定のエージェントフレームワークに直接組み込まれています。

- Strands Agents ツール – Strands Agentsフレームワークに固有の軽量でquick-to-implementツール。
- LangChain ツール – LangChainエコシステムと緊密に統合された Pythonベースのツール。
- LlamaIndex ツール – 内のデータの取得と処理に最適化されたツールLlamaIndex。

メタツール

メタツールは、外部アクションを直接実行することなく、エージェントワークフローを強化します。

- ワークフローツール – エージェント実行フロー、分岐ロジック、状態管理を管理します。
- エージェントグラフツール – 複雑なワークフローで複数のエージェントを調整します。
- メモリツール – エージェントセッション全体で永続的なストレージと情報の取得を提供します。
- リフレクションツール – エージェントが独自のパフォーマンスを分析して改善できるようにします。

プロトコルベースのツール

プロトコルベースのツールを検討する場合、モデルコンテキストプロトコル (MCP) はツール統合の最も包括的で柔軟な基盤を提供します。AWS エージェントの相互運用性に関するオープンソースブログ記事で述べたように、 AWS は MCP を戦略的プロトコルとして採用し、その開発に積極的に貢献しています。

次の表に、MCP ツールのデプロイのオプションを示します。

デプロイモデル	説明	に最適	実装
ローカル stdio ベース	エージェントと同じプロセスで実行されるツール	開発、テスト、シンプルなツール	ネットワークオーバーヘッドなしで迅速に実装

ローカルサーバー送信イベント (SSE) ベース	ツールはローカルで実行されますが、HTTP 経由で通信します	懸念を分離したより複雑なローカルツール	分離は向上するが、レイテンシーは低い
リモート HTTP ストリーミング可能	リモートサーバーで実行されるツール	本番環境と共有ツール	スケーラブルで一元管理

公式 MCP SDKs は MCP ツールの構築に使用できます。

- Python SDK – 完全なプロトコルサポートによる包括的な実装
- TypeScript SDK – ウェブアプリケーションの JavaScript/TypeScript 実装
- Java SDK – エンタープライズアプリケーションの Java 実装

これらの SDKsは、プロトコル仕様の一貫した実装により、任意の言語で MCP 互換ツールを作成するための構成要素を提供します。

さらに、 AWS は Strands Agents SDK に MCP を実装しています。Strands Agents SDK を使用すると、MCP 互換ツールを簡単に作成して使用できます。包括的なドキュメントは、Strands AgentsGitHub リポジトリで入手できます。より簡単なユースケースやStrands Agentsフレームワーク外で作業する場合、公式 MCP SDKs は複数の言語でプロトコルを直接実装します。

MCP ツールのセキュリティ機能

MCP ツールのセキュリティ機能には以下が含まれます。

- OAuth 2.0/2.1 認証 – 業界標準認証
- アクセス許可のスコープ — ツールのきめ細かなアクセスコントロール
- ツール機能検出 — 使用可能なツールの動的検出
- 構造化エラー処理 — 一貫したエラーパターン

MCP ツールの開始方法

ツール統合用の MCP を実装するには、次のアクションを実行します。

1. 本番環境対応の MCP 実装については、 Strands Agents SDK をご覧ください。
2. MCP 技術ドキュメントを確認して、主要な概念を理解します。

3. このAWS オープンソースブログ記事で説明されている実用的な例を使用します。
4. リモートツールに進む前に、シンプルなローカルツールから始めます。
5. MCP コミュニティに参加して、プロトコルの進化に影響を与えます。

AgentCore Gateway の詳細

Amazon Bedrock AgentCore Gateway は、開発者が MCP ツールやその他のターゲットエンドポイントを大規模に構築、デプロイ、検出、接続するための簡単で安全な方法を提供します。AgentCore Gateway を使用すると、デベロッパーAPIs、 AWS Lambda 関数、既存のサービスを MCP 互換ツールに変換できます。次に、わずか数行のコードで、これらのツールを AgentCore Gateway エンドポイントを介してエージェントが使用できるようにします。AgentCore Gateway は OpenAPI、、Smithy、および Lambda を入力タイプとしてサポートし、フルマネージドサービスで包括的な進入認証と退出認証の両方を提供する唯一のソリューションです。

フレームワークネイティブツール

Model Context Protocol (MCP) は最も柔軟な基盤を提供しますが、フレームワークネイティブツールは特定のユースケースに利点を提供します。

Strands Agents SDK は、シンプルなオペレーションに最小限のオーバーヘッドを必要とする軽量設計を特徴とする Pythonベースのツールを提供します。これにより、迅速な実装が可能になり、開発者はわずか数行のコードでツールを作成できます。さらに、Strands Agentsフレームワーク内でシームレスに機能するように緊密に統合されています。

次の例は、 を使用してシンプルな気象ツールを作成する方法を示していますStrands Agents。開発者は、最小限のコードオーバーヘッドでPython関数をエージェントアクセス可能なツールにすばやく変換し、関数のドキュメントから適切なドキュメントを自動的に生成できます。

```
#Example of a simple Strands native tool

@tool

def weather(location: str) -> str:

"""Get the current weather for a location""" #

Implementation here

return f"The weather in {location} is sunny."
```

迅速なプロトタイプ作成やシンプルなユースケースでは、フレームワークネイティブツールが開発を加速できます。ただし、本番稼働用システムの場合、MCP ツールはフレームワークネイティブツールよりも相互運用性と将来の柔軟性に優れています。

次の表は、他のフレームワーク固有のツールの概要を示しています。

Framework	ツールタイプ	利点	考慮事項
AutoGen	関数定義	強力なマルチエージェントサポート	Microsoft エコシステム
LangChain	Python クラス	構築済みのツールの大規模なエコシステム	フレームワークのロックイン
LlamaIndex	Python 関数	データオペレーション用に最適化	に制限 LlamaIndex

メタツール

メタツールは外部システムと直接やり取りしません。代わりに、エージェントパターンを実装することでエージェントの機能を強化します。このセクションでは、ワークフロー、エージェントグラフ、メモリメタツールについて説明します。

ワークフローメタツール

ワークフローメタツールは、エージェント実行のフローを管理します。

- 状態管理 – 複数のエージェントインタラクションのコンテキストを維持する
- 分岐ロジック – 条件付き実行パスを有効にする
- 再試行メカニズム – 高度な再試行戦略で障害を処理する

ワークフローメタツールを使用したフレームワークの例には、 LangGraphおよび Strands Agents ワークフロー機能が含まれます。

エージェントグラフのメタツール

エージェントグラフのメタツールは、複数のエージェントを連携させて調整します。

- タスクの委任 – 専門エージェントにサブタスクを割り当てる
- 結果の集約 – 複数のエージェントからの出力を結合する
- 競合の解決 – エージェント間の不一致を解決する

AutoGen や などのフレームワークは、エージェントグラフの調整にCrewAI特化しています。

メモリメタツール

メモリメタツールは、永続的なストレージと取り出しを提供します。

- 会話履歴 – セッション間でコンテキストを維持する
- ナレッジベース – ドメイン固有の情報を保存および取得する
- ベクトルストア – セマンティック検索機能を有効にする

MCP のリソースシステムは、さまざまなエージェントフレームワークで動作するメモリメタツールを実装するための標準化された方法を提供します。

ツール統合戦略

ツール統合戦略の選択は、エージェントが達成できることとシステムの進化の容易さに直接影響します。フレームワークネイティブツールとメタツールを戦略的に使用しながら、モデルコンテキストプロトコル (MCP) などのオープンプロトコルを優先します。これにより、AI テクノロジーの進歩に合わせて柔軟で強力なツールエコシステムを構築できます。

ツール統合に対する以下の戦略的アプローチは、組織の当面のニーズを満たしながら柔軟性を最大化します。

1. 基盤として MCP を採用する – MCP は、強力なセキュリティ機能を持つツールにエージェントを接続するための標準化された方法を提供します。以下の主要なツールプロトコルとして MCP から始めます。
   - 複数のエージェント実装で使用される戦略的ツール。
   - 堅牢な認証と認可を必要とするセキュリティ重視のツール。
   - 本番環境でリモート実行が必要なツール。
2. 必要に応じてフレームワークネイティブツールを使用する — 以下のフレームワークネイティブツールを検討します。

- 初期開発中の迅速なプロトタイピング。
- セキュリティ要件を最小限に抑えたシンプルな重要ではないツール。
- 独自の機能を活用するフレームワーク固有の機能。

3. 複雑なワークフローにメタツールを実装する – メタツールを追加してエージェントアーキテクチャを強化します。
   - 基本的なワークフローパターンから簡単に開始します。
   - ユースケースが成熟するにつれて複雑さを追加します。
   - エージェントとメタツール間のインターフェイスを標準化します。
4. 進化の計画 – 将来の柔軟性を念頭に置いて構築します。
   - 実装とは無関係にツールインターフェイスを文書化します。
   - エージェントとツールの間に抽象化レイヤーを作成します。
   - 独自のプロトコルからオープンプロトコルへの移行パスを確立します。

ツール統合のセキュリティのベストプラクティス

ツールの統合は、セキュリティ体制に直接影響します。このセクションでは、組織で考慮すべきベストプラクティスの概要を説明します。

認証と認可

次の堅牢なアクセスコントロールを使用します。

- OAuth 2.0/2.1 を使用する – リモートツールに業界標準の認証を実装します。
- 最小特権を実装する – ツールに必要なアクセス許可のみを付与します。
- 認証情報のローテーション – API キーとアクセストークンを定期的に更新します。

データ保護

データを保護するために、以下の対策を実施してください。

- 入力と出力を検証する – すべてのツールインタラクションにスキーマ検証を実装します。
- 機密データの暗号化 – すべてのリモートツール通信に TLS を使用します。
- データ最小化の実装 – 必要な情報のみをツールに渡します。

モニタリングと監査

次のメカニズムを使用して、可視性と制御を維持します。

- すべてのツール呼び出しを口グに記録する – 包括的な監査証跡を維持します。
- 異常をモニタリングする – 異常なツールの使用パターンを検出します。
- レート制限の実装 — 過剰なツール呼び出しによる不正使用を防止します。

Model Context Protocol (MCP) セキュリティモデルは、これらの懸念に包括的に対処します。詳細については、MCP ドキュメントの「セキュリティに関する考慮事項」を参照してください。

結論

エージェント AI の環境は急速に進化し続けており、組織はインテリジェントで自律的なシステムを構築するための強力な新しい方法を提供します。このガイドでは、実装を成功させるための 3 つの重要なコンポーネント、基盤を提供するフレームワーク、環境を提供するプラットフォーム、通信を可能にするプロトコル、機能を拡張するツールについて説明しました。

フレームワークが成熟するにつれて、相互運用性の向上、モデルコンテキストプロトコル (MCP) などのプロトコルの標準化、自律型エージェントのより高度なオーケストレーション機能が期待できます。現在、これらのフレームワークに関する専門知識を確立している組織は、ビジネスに大きな価値をもたらす、ますます自律的でインテリジェントなエージェントを構築する態勢が整っています。

プラットフォームは、エージェントシステムが動作する実行、ガバナンス、ライフサイクル環境を提供します。アイデンティティ、セキュリティ境界、オブザーバビリティ、メモリ管理、セッショングラウンディング、ツールやデータとの安全なやり取りなどの懸念に対処します。環境では AWS 、マネージドエージェントランタイムやオーケストレーションサービスなどのプラットフォームにより、組織は自律型エージェントとエージェントシステムを大規模にデプロイ、モニタリング、進化、管理できます。プラットフォームは、基本的なフレームワークを実際の運用要件と橋渡しします。

エージェントプロトコルの選択は、即時の開発ニーズと長期的な柔軟性と相互運用性のバランスを取る戦略的決定を表します。オープンプロトコルを優先し、適切な抽象化レイヤーを作成することで、組織は現在のビジネス要件を満たしながら、進化するテクノロジーに適応可能なエージェントシステムを構築できます。

ほとんどの組織にとって、MCP はオープンスタンダード、成長するエコシステム、agent-to-agent 通信パターンのサポート、ツール統合機能による強力な基盤です。 AWS は MCP と Agent2Agent (A2A) を戦略的プロトコルとして採用し、Strands AgentsSDK などのサービス全体での開発と実装に積極的に貢献しています。MCP または A2A を適切なフレームワークネイティブツールやメタツールとともに使用することで、将来のイノベーションに適応しながらすぐに価値をもたらすエージェントシステムを構築できます。

リソース

次のリソース AWS と、自律型エージェント開発に関連するその他のリソースを使用します。

AWS ブログ

- Amazon Bedrock AgentCore Memory: コンテキスト対応エージェントの構築
- Best practices for building robust generative AI applications with Amazon Bedrock Agents – Part 1
- Best practices for building robust generative AI applications with Amazon Bedrock Agents – Part 2
- LlamaIndexと Amazon Bedrock を使用して強力な RAG パイプラインを構築する
- Amazon Bedrock AgentCore Observability を使用して信頼できる AI エージェントを構築する
- Amazon Bedrock LlamaIndexと RAGAS を使用して RAG レスポンスを評価する
- Amazon Bedrock AgentCore コードインタープリタの紹介
- Amazon Bedrock AgentCore Gateway の紹介: エンタープライズ AI エージェントツール開発の変革
- Amazon Bedrock AgentCore Identity の紹介: エージェント AI を大規模に保護する
- オープンソース AI エージェント SDK Strands Agentsである の紹介
- エージェント相互運用性のオープンプロトコル パート 1: MCP でのエージェント間通信
- Amazon Bedrock AgentCore ランタイムでエージェントとツールを安全に起動およびスケーリングする
- AWS Transform for .NET、.NET アプリケーションを大規模にモダナイズするための最初のエージェント AI サービス
- AWS 毎週の切り上げ: Strands Agents

AWS 規範ガイダンス

- でのエージェント AI の運用 AWS
- でのエージェント AI の基礎 AWS
- でのエージェント AI のパターンとワークフロー AWS
- でのエージェント AI 用のサーバーレスアーキテクチャの構築 AWS
- でのエージェント AI 用のマルチテナントアーキテクチャの構築 AWS

- でのエージェント AI のセキュリティ AWS
- で拡張生成オプションとアーキテクチャを取得する AWS

AWS リソース

- Amazon Bedrock ドキュメント
- Amazon Bedrock AgentCore ドキュメント
- Amazon Bedrock AgentCore スターターツールキット (GitHub リポジトリ)
- Amazon Nova ドキュメント
- AWS MCP サーバー (GitHub リポジトリ)

その他のリソース

- AutoGen ドキュメント (Microsoft)
- 効果的なエージェントの構築 (Anthropic)
- CrewAI GitHub リポジトリ
- LangChainドキュメント
- LangGraph プラットフォーム
- LlamaIndexドキュメント
- Model Context Protocol ドキュメント
- Strands Agentsドキュメント
- Strands Agents ツールの概要
- Strands Agents クイックスタートガイド

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
新規セクション	プラットフォームセクションを追加	2026 年 1 月 16 日
初版発行	—	2025 年 7 月 14 日

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。