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

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 用のマルチテナントアーキテクチャの構築 AWS

Table of Contents

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

Aaron Sempf and Tod golding、アマゾン ウェブ サービス

2025 年 7 月 (ドキュメント履歴)

エージェント AI は、組織がシステムを構築、提供、運用する方法を再検討する必要がある破壊的なパラダイムシフトを表します。エージェントモデルでは、新しいパス、可能性、値を作成する 1 つ以上のエージェントにシステムを分解する新しい方法を検討しています。

エージェントディスカッションの多くは、エージェントの構築と実装に使用されるツール、フレームワーク、パターンを中心に行われます。エージェントを作成するための優れたツールだけでなく、エージェントアーキテクチャの基礎となる新しい統合プロトコル、認証戦略、検出メカニズムを採用する必要があります。

エージェントツールの数は増加しますが、チームはエージェントが従来のアーキテクチャの課題にどのように対処するかも検討する必要があります。スケーリング、ノイズの多い近隣、レジリエンス、コスト、運用効率は、エージェントの設計、構築、デプロイ時に評価する必要がある基本的なトピックです。自律型エージェントとスマートエージェントがどのように機能していても、ビジネスニーズに合ったスケール、効率、俊敏性の経済を確実に達成する必要があります。

このガイドの目的は、エージェントフットプリントのさまざまな側面を調べることです。これには、さまざまなエージェントのデプロイと消費のパターンを確認し、アーキテクチャの目標に対処するエージェントを作成するためのさまざまな戦略に焦点を当てることが含まれます。また、通常はマルチテナント設定で必要な内部コンストラクトを導入することで、マルチテナント環境でエージェントがどのように消費されるかを調べることも意味します。

対象者

このガイドは、AI 駆動型マルチテナントシステムを構築するアーキテクト、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。

- マルチテナントエージェントのデプロイ、サイロモデルとプールモデルの両方の探索、テナントコンテキストがエージェントの実装にどのように影響するかを理解する
- シングルプロバイダー環境とマルチプロバイダー環境におけるオンボーディング、テナント分離、リソース管理などのエージェント管理について調べる
- データの所有権、モニタリング、テストなど、マルチテナントエージェントの側面を評価する

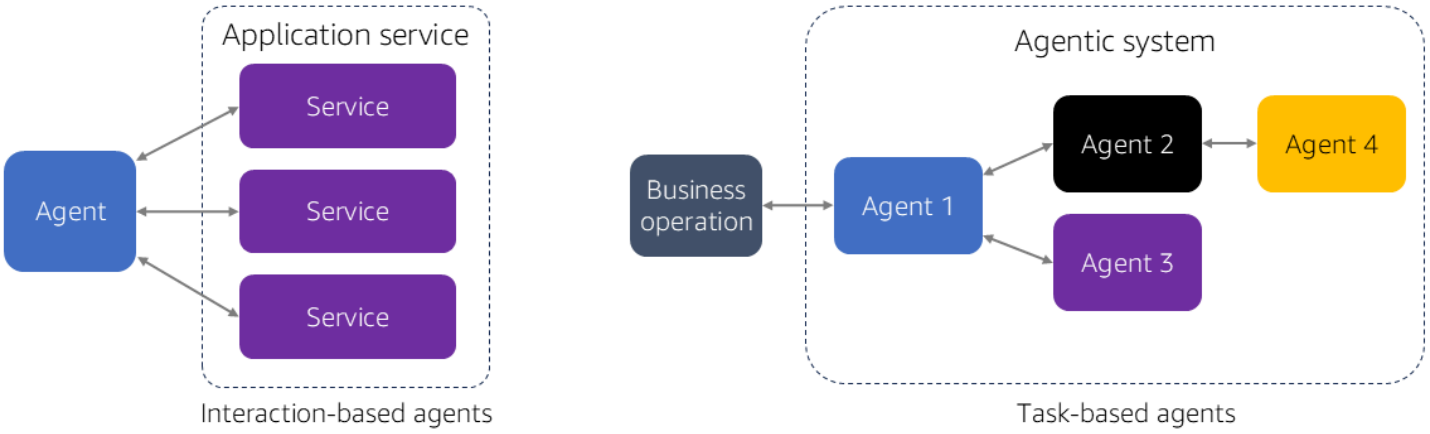
このコンテンツシリーズについて

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。

エージェントの基礎

アーキテクチャの詳細について説明する前に、「エージェント」は多くのユースケースに適用できるオーバーロード用語であるため、エージェントが果たすさまざまな役割について概説する必要があります。分類に役立つ幅広い用語から始めましょう。

最外部レベルでは、まずエージェントのロールと性質を分類する必要があります。これは、エージェントがさまざまな問題に適用できる幅広いシナリオがあるため、困難です。ただし、この説明では、エージェントをアプリケーションまたはシステムに導入することの意味に焦点を当てます。このモデルでは、エージェントがシステムのエクスペリエンスを最も充実させる方法と場所を強調しています。選択したオプションは、エージェントのビルド、統合、さまざまなドメインやユースケースへの適用方法に影響します。次の図は、ビルダーが使用する 2 つのエージェントパターンを示しています。

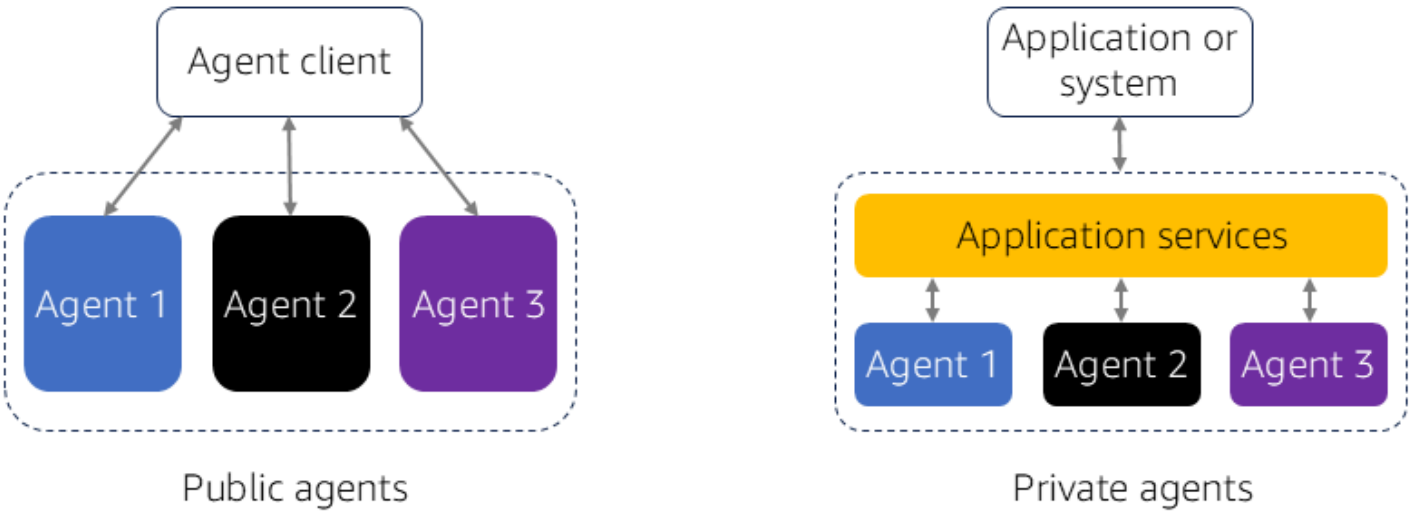


図の左側には、インタラクションベースのエージェントがあります。このモードでは、エージェントは既存のシステムへのビューを作成し、基盤となるサービスとのインタラクションを調整して、目標または成果を達成します。重要なのは、システムの特徴と機能を推進するための代替アプローチとして、エージェントがシステムに追加されることです。たとえば、独立系ソフトウェアベンダー (ISV) に、オペレーションの実行に使用される UX を備えた会計システムがあるとします。インタラクションベースのエージェントは、これらの既存の機能とのやり取りを簡素化します。大まかに定義された目標を達成する方法を学ぶことではなく、既知の経路をオーケストレーションする方法を提供することです。

対照的に、図の右側にあるタスクベースのシステムは、異なるアプローチを表しています。そのシステムのエージェントは、知識と能力を使用して、タスクを完了し、ビジネス成果を促進する方法を学習します。両方のモデルがビジネス成果を達成すると主張できますが、タスクベースのモデルはエージェント自身に依存して成果を達成する方法を決定します。このようなエージェントは決定論的では

なく、学習して進化する能力に依存しています。対照的に、インタラクションベースのエージェントは、主に一連の既知の機能をオーケストレーションするように設計されています。これらの違いは、ビジネスをサポートするためにエージェントを構築、範囲指定、統合する方法に影響します。

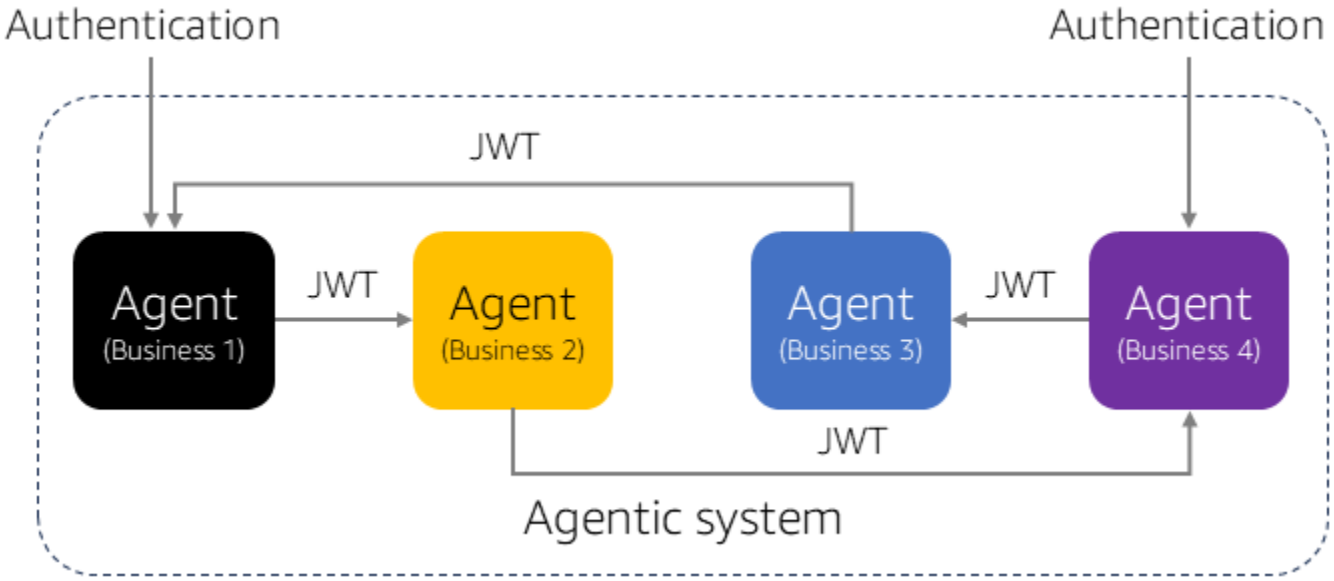
また、エージェントのデプロイ方法とデプロイ場所を特徴付ける用語も必要です。エージェントがシステムのフットプリント内に存在する場所は、エージェントの構築、スコープ設定、保護方法に影響を与える可能性があります。次の図は、エージェントに適用できる 2 つの異なるモデルの概要を示しています。



図の左側には、3 つの異なるエージェントを持つデプロイシステムがあります。エージェントは、他のエージェントまたはアプリケーションである可能性のある外部クライアントに公開されます。このモデルでは、エージェントはパブリックエージェントと呼ばれます。

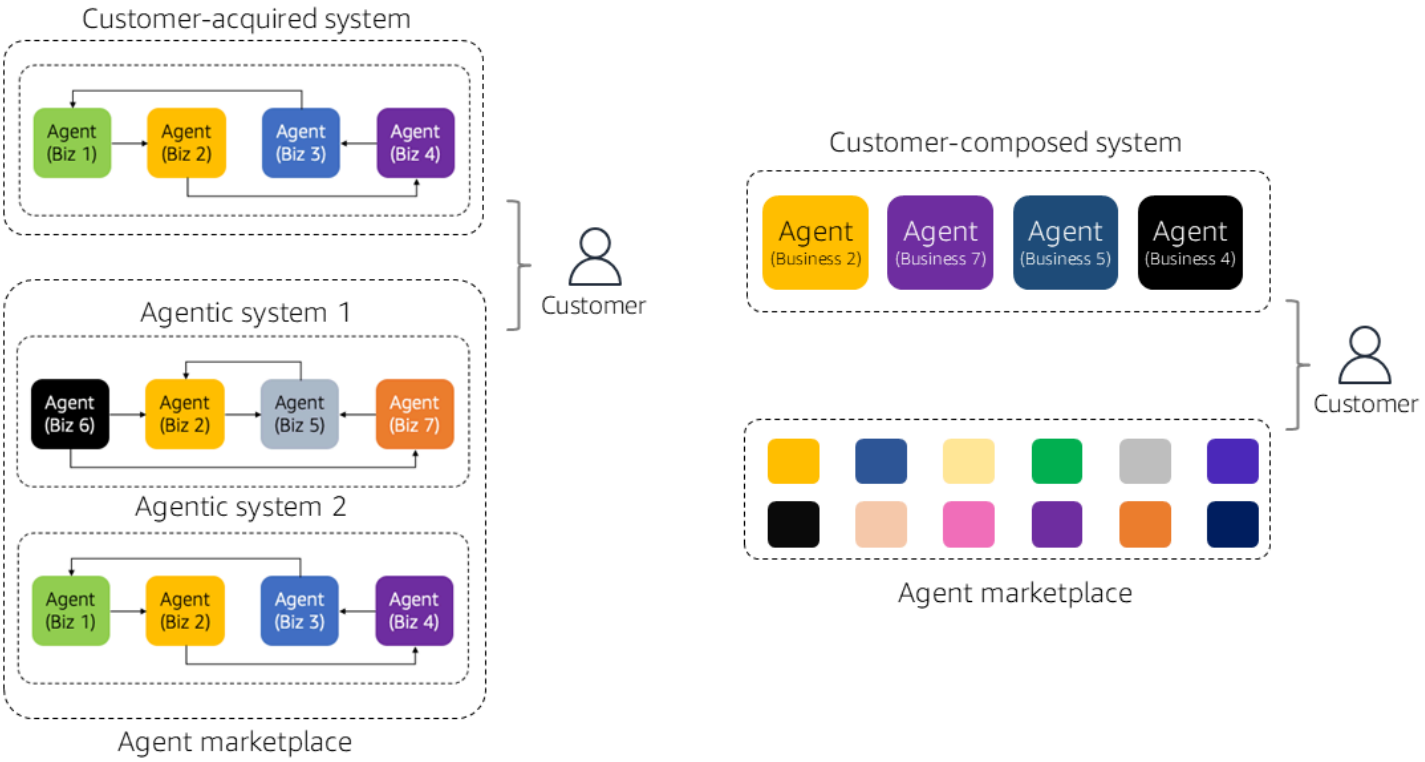
対照的に、右側の図は、ソリューションの実装内のエージェントを示しています。この場合、ユーザーまたはシステムによって消費される一連のアプリケーションサービスがあります。これらのユーザーは、エージェントがエクスペリエンスの一部であることを認識せずにアプリケーションとやり取りします。その後、エージェントは基盤となるシステムのサービスによって呼び出され、オーケストレーションされます。この方法でデプロイされたエージェントは、プライベートエージェントと呼ばれます。

エージェントの価値の多くは、プロバイダーが他のサードパーティーエージェントと統合することを意図してエージェントを公開するパブリックモデルに焦点を当てています。その後、エージェントは相互接続されたサービスのメッシュまたはウェブの一部となり、まとめて多くのユースケースに対処できます。これらのエージェントは多くのドメインで使用できますが、business-to-businessユースケースは自然に適しています。次の図は、特定の問題を解決するコレクションエージェントをアセンブルするとどうなるかを概念的に示しています。



この図は、一連の目標を達成するために協力する 4 人のビジネスエージェントを示しています。エージェントをこのように構成すると、エージェントシステムを表し、そのようなシステムのさまざまな種類があります。これらは、一般的に 1 つのユニットとして消費されるコラボレーションエージェントのパッケージ化されたセットである可能性があります。または、ニーズに最適なエージェントの組み合わせを選択して選択するお客様が、システムを動的に組み立てることもできます。

どちらのアプローチも、エージェント統合のための実行可能なパスを提供します。一部のエージェントは、価値、到達範囲、影響を最大化できる特定のシステムに統合されることを期待して構築されています。このエージェントシステムの概念は、エージェントの取得方法についても疑問を投げかけます。これに対処する方法は多数あります。次の図は、トランザクションエクスペリエンスを通じてこれらのエージェントとシステムを作成する方法の例を示しています。

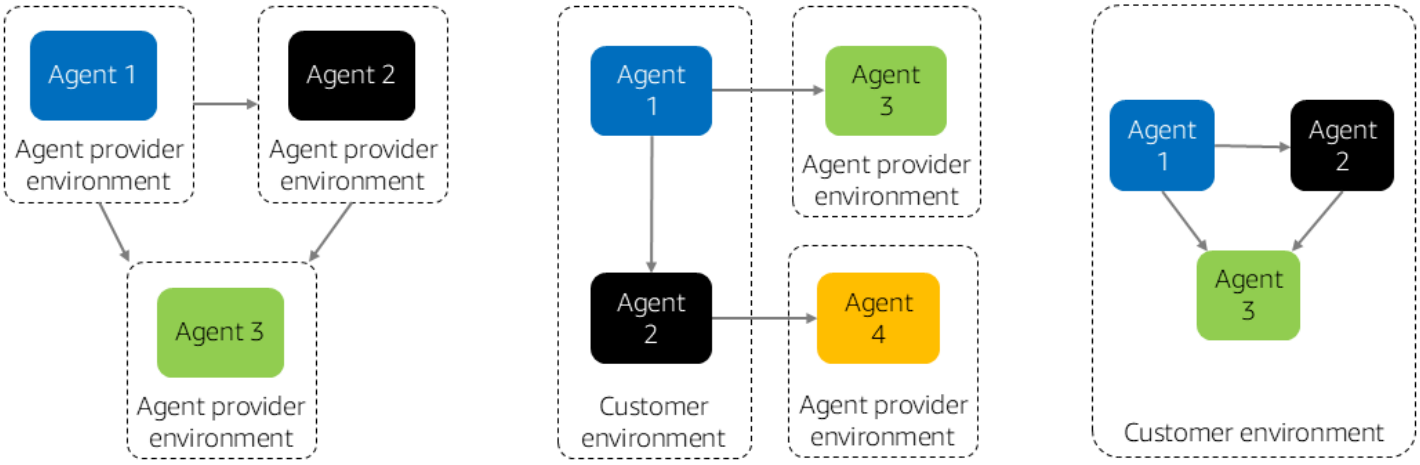


マーケットプレイスエクスペリエンスの 2 つの例を示します。左側では、マーケットプレイスを使用して事前にパッケージ化されたシステムを取得します。このシナリオでは、マーケットプレイスは、複数のエージェントの統合とオーケストレーションを必要とするより広範な目標に対応するシステムを検出してオンボードします。

右側の例は、エージェントが発見され、エージェントシステムに構成されるマーケットプレイスを示しています。このシナリオでは、お客様はニーズに合わせて互換性のある統合エージェントの任意のシステムを構築できます。この方法でエージェントをアセンブルできるかどうかは、個々のエージェントの互換性モデルと統合要件によって異なります。

エージェントホスティングに関する考慮事項

より広範なエージェント概念を理解できたので、これらのエージェントをホストして実行する意味について説明します。計算の実行方法と場所、スケーリング方法、運用方法、管理方法を考慮する必要があります。同時に、エージェントとして見込まれるいくつかのパターンがより広く適用され、採用されています。次の図は、考えられる置換の例を示しています。



ここでは、3 つの異なる戦略を示します。図の左側には、各エージェントプロバイダーの環境内でエージェントがホスト、スケーリング、管理されるモデルが表示されます。これらのエージェントは、サービスとして公開および消費され、サービスとしてのエージェント (AaaS) モデルとしてラベル付けされたモデルで動作します。右側は、プロバイダーのエージェントがすべて専用の顧客環境でホストされるモデルです。

この図の途中には、これら 2 つの戦略を組み合わせた混合デプロイモデルがあり、一部のエージェントをお客様の環境でローカルにホストし、プロバイダーの環境でリモートでホストされているエージェントとやり取りします。

4 番目のオプション (非表示) は、エージェントインフラストラクチャサービスによってスケーリングおよび管理されるローコードサービスまたはノーコードサービスとしてエージェントを構築する場合です。マネージドエージェントのアーキテクチャとホスティングは、主にサービスを所有する組織によって決定されるため、これらについては詳しく説明しません。

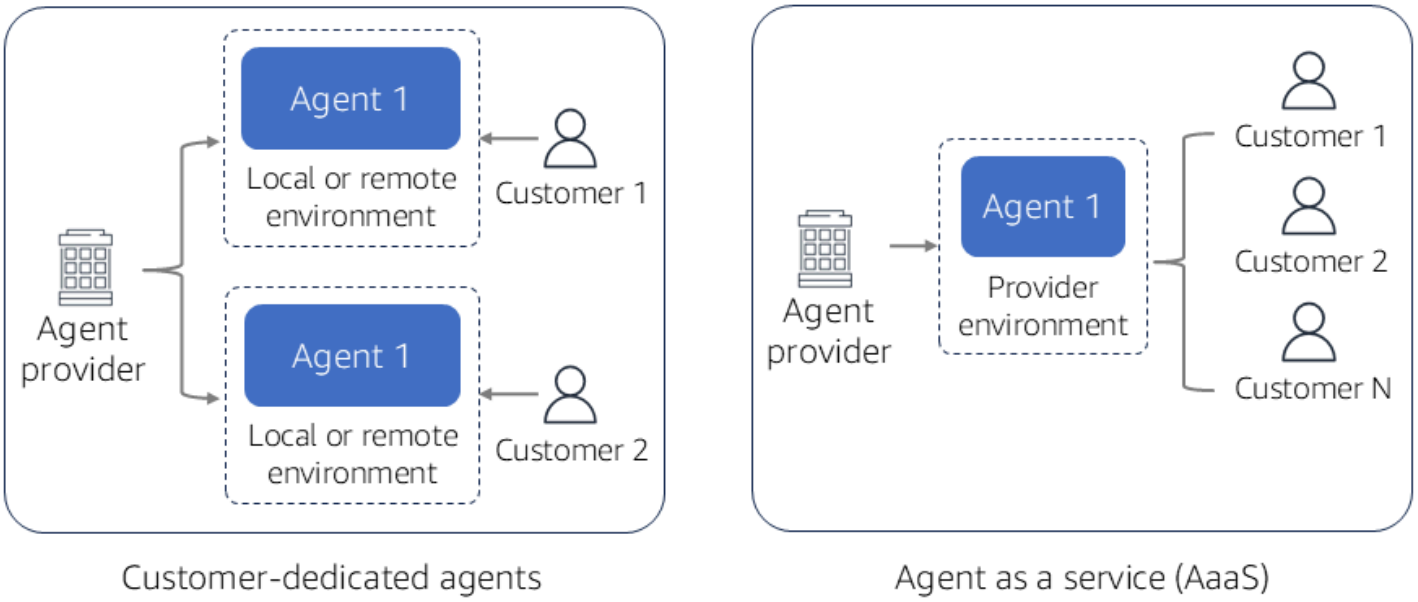
これらのモデルのいずれかの導入に影響を与える可能性のある要因の範囲を想像できます。例えば、コンプライアンス、規制、セキュリティの制約により、顧客がホストするエージェントにプッシュされる可能性があります。スケール、俊敏性、効率性は、組織をより AaaS モデルにプッシュする可能性があります。

ここで重要な概念は、エージェントがさまざまな方法でデプロイおよびホストできることです。エージェントを最適に適用する方法を決定するのは管理者の仕事です。フットプリント、セキュリティ、デプロイなどの要因は、エージェントの構築と運用の方法に大きな影響を与えます。たとえば、プライベートエージェントとパブリックエージェントでは、設計とリリースのライフサイクルが異なる場合があります。

エージェントはマルチテナンシーを満たす

エージェントは、特定のドメインやビジネス上の問題のニーズをサポートするためにアセンブルされた一連の自律コンポーネントと見なされる構成要素と考えるのは簡単です。興味深いのは、これらのエージェントがプロバイダーによってどのようにパッケージ化および消費されるかについて考え始めるときです。多くの点で、エージェントはビジネスのコストと収益の源になります。エージェントプロバイダーは、サービスを利用するさまざまなペルソナ、ペルソナの消費プロファイル、およびエージェントプロバイダーがコンシューマーに合わせた料金モデルと階層化モデルを作成できるようにする収益化戦略を考慮する必要があります。

エージェントプロバイダーは、顧客のニーズを満たすためにエージェントをデプロイするための複数のモデルをサポートできます。次の図は、2 つの主要なエージェントデプロイモデルの概念図を示しています。



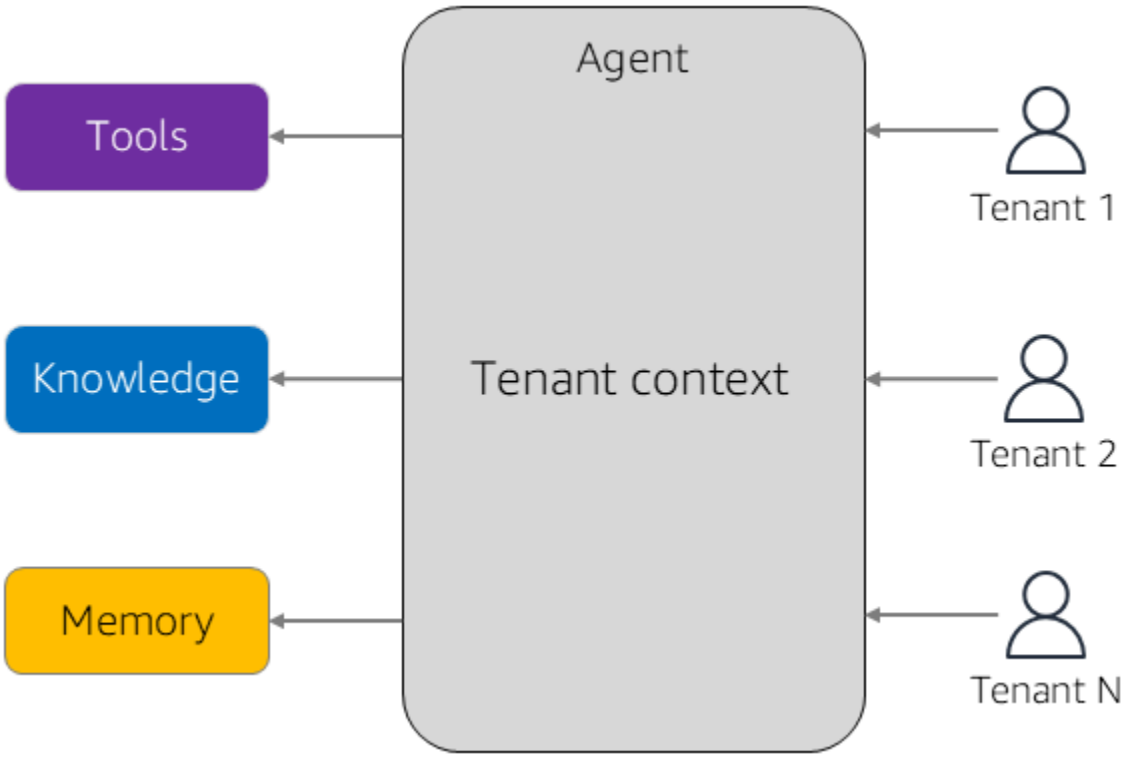
図の左側は、顧客専用エージェントモデルを示しています。エージェントプロバイダーは、オンボーディングされた顧客ごとに個別のエージェントインスタンスをデプロイしてエージェントを構築します。このアプローチでは、エージェントの能力と知識を取得する能力は、特定の顧客の環境の範囲に限定されます。これにより、専用カスタマー環境をサポートする複雑さと利点の一部を継承するカスタマーエクスペリエンスが表されます。

対照的に、図の右側にある図には、プロバイダーの環境にデプロイされた単一のエージェントがあります。エージェントは複数の顧客からのリクエストを処理し、すべての顧客の集合的なエクスペリエンスに基づいて進化し、学習します。追加される新しい顧客はそれぞれ、単にエージェントの別の有効なクライアントを表します。エージェントは、クライアントのニーズをサポートするために共有コ

ンストラクトを使用して、サービスとしてのエージェント (AaaS) モデルのように実行されます。どちらの場合も、エージェントコンシューマーはアプリケーション、システム、またはその他のエージェントである可能性があります。

AaaS モデルを見るには、2 つの方法があります。上記のモデルは、すべてのお客様に同じエクスペリエンスを提供します。つまり、エージェントの内部には、リクエスト元のクライアントのコンテキストを考慮する専門分野のレベルは含まれません。一般的に、このモードでは、エージェントの範囲、目標、価値の性質が、すべてのクライアントに共通に適用されるリソース、知識、結果の共有セットを中心にしていることを前提としています。

AaaS の代替アプローチは、クライアントのコンテキストがエージェントのエクスペリエンスと実装に影響を与えることです。次の図は、このコンテキストでの AaaS エージェントフットプリントの概念図を示しています。



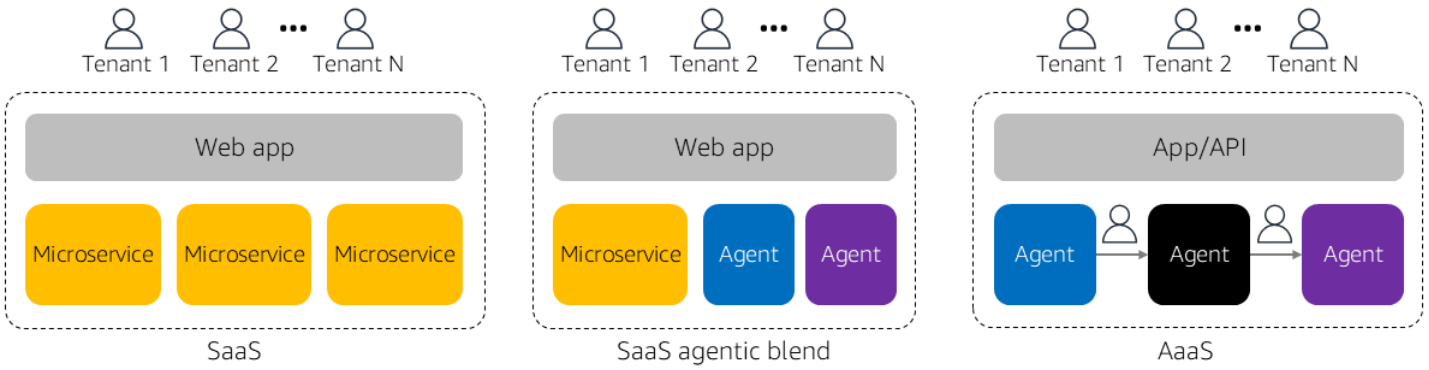
この AaaS ビューでは、受信リクエストのオリジンとコンテキストがエージェントのフットプリントに大きく影響します。エージェントの基礎となる実装の一部であるリソース、アクション、ツールは、受信テナントリクエストごとに異なる場合があります。エージェントの価値は、テナントコンテキストを使用して、テナントの状態、知識、その他の要因の影響を受けるアクションや結果に到達する能力に関連しています。リクエストによってはテナントごとに独自の結果が得られる場合もあれば、テナントごとによりカスタマイズされた結果になる場合もあります。これにより、エージェン

トの学習能力に新しいディメンションが追加されます。これには、よりコンテキスト的になり、ターゲットを絞った成果を高める知識を取得して適用することが含まれます。

プロバイダーにとって、AaaS モデルには多くの利点があります。複数の顧客が 1 つのエージェントを消費しているため、プロバイダーはスケールメリットを実現し、運用効率を高め、コストを制御し、統一された管理エクスペリエンスを創出するより良い機会が得られます。これにより、エージェントビジネスの俊敏性、イノベーション、成長の可能性が高まります。

これらの品質は、Software as a Service (SaaS) モデルの採用を推進するのと同じ原則と重複しています。基本的に、AaaS モデルは、SaaS 環境で見られるのと同じスケール、耐障害性、分離、オンボーディング、運用属性の多くを継承するマルチテナントサービスとして構築されています。多くの点で、AaaS エクスペリエンスは SaaS プロバイダーが使用する戦略とプラクティスから大きく借用しますが、これらの用語を分離するのが合理的です。当社では、主にマルチテナントサポートを必要とするエージェントの構築と運用に伴う影響に重点を置いています。

すべてのユーザーを均等に扱うことができ、永続データ、機密データ、または顧客固有のデータを管理する必要がないシステムでは、テナンシーの概念はエージェントに最小限影響します。データの分離、カスタマイズ、コンテキスト認識を維持しながら複数の顧客にサービスを提供することが予想されるシステムでは、複数のテナントをサポートすることがエージェントの設計、戦略、目標の重要な要素になる可能性があります。次の図は、エージェント環境でマルチテナンシーを使用する方法を示しています。



この図の左側には、従来のマルチテナントアーキテクチャがあります。これには、ウェブアプリケーションと、ビジネスロジックを実装する一連のマイクロサービスが含まれます。複数のテナントがこの環境の共有インフラストラクチャを消費し、進化するテナント人口の変化するワークロードに合わせてスケーリングします。環境は、すべてのテナントに対して 1 つのウィンドウを通じて運用および管理されます。

このメンタルモデルがこの図の右側にあるエージェントにどのようにマッピングされるかを想像してみてください。1 つのエージェントは、1 つ以上のテナントによって消費される AaaS モデルを実行

します。エージェントは、1 つのエージェントの 1 つのインスタンスが複数のテナントからのリクエストを処理する必要があるため、テナントコンテキストがそれらの間を流れる複数のプロバイダーからのものである可能性があります。

この図の真ん中の例は、エージェントが SaaS エクスペリエンス全体の一部であるハイブリッドモデルです。システムの一部の部分はより従来のモデルで実装されており、システムの他の部分はエージェントに依存しています。このパターンは、多くの SaaS サービス、特にエージェントエクスペリエンスに移行している組織で一般的である可能性があります。すべてのシステムが純粋な AaaS として配信されるわけではないため、このモデルが存続することは一般的です。また、マルチテナンシーは引き続きモデルのエージェントに適用されることに注意してください。エージェントはシステム内に埋め込まれていても、複数のテナントからのリクエストを処理する場合があります。

マルチテナンシーが本当に重要であるかどうかを尋ねるのは当然です。エージェントがリクエストを処理するため、テナンシーのサポートはほとんど効果がないと主張できます。ただし、マルチテナントエージェントの影響をより深く掘り下げると、テナントは、個々のテナントをサポートするようにツール、メモリ、データ、その他のエージェントパートへのアクセス、デプロイ、設定にエージェントがどのように影響するかに直接影響する可能性があります。テナンシーは、スケーリング、スロットリング、料金設定、階層化、その他のビジネス側面がエージェントのアーキテクチャにどのように適用されるかにも影響します。

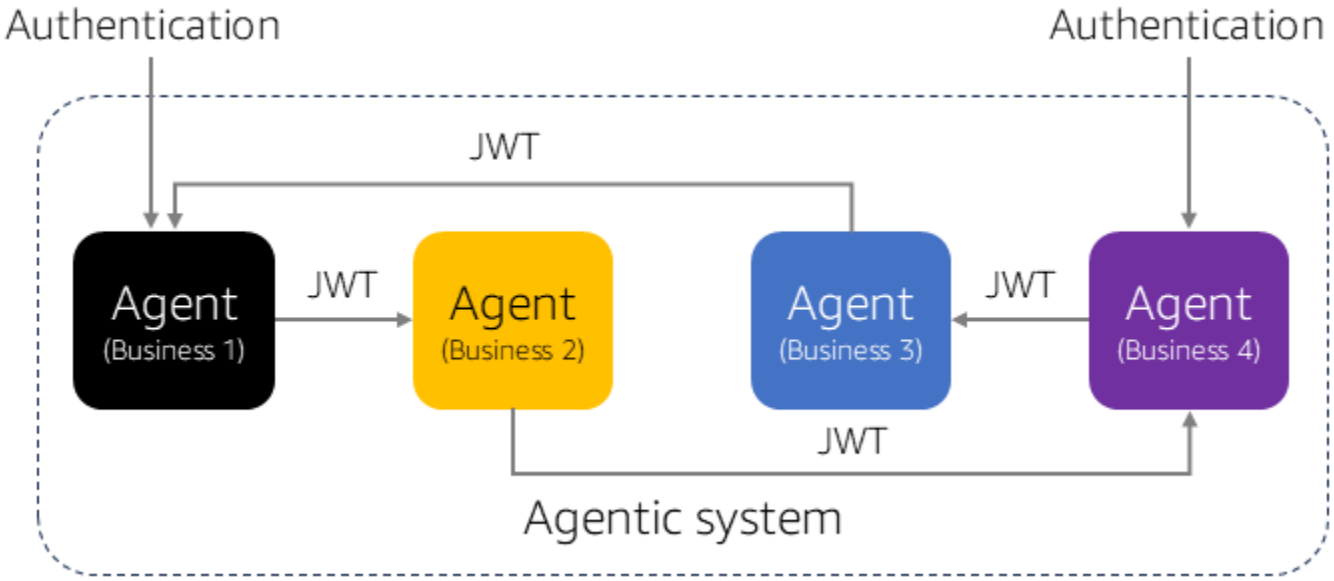
もう 1 つの重要な点は、マルチテナンシーサポートを必要とするエージェントユースケースがあることです。課題は、マルチテナンシーがエージェントエクスペリエンスの全体的な設計とアーキテクチャをどのように形成するかを判断することです。一部のエージェントでは、マルチテナントサポートは差別化機能を表し、エージェントはターゲットを絞った結果をもたらすエージェントにテナント固有のコンテキストを適用できます。

以降のセクションでは、マルチテナント SaaS アーキテクチャを説明するために作成する用語と設計パターンがどのように役立つかを説明します。これらの概念は、有用な側面を借りることで AaaS モデルで採用できます。これにより、必要に応じてエージェント固有の新しい概念が導入されます。

ID、テナントコンテキスト、エージェントシステム

テナントコンテキストを個々のエージェントに追加するのは特に難しくありません。多くの場合、チームはユーザーとシステムをテナントにバインドし、テナント対応トークンをエージェントに渡す一般的なメカニズムに頼ることができます。これは、テナントコンテキストとアイデンティティが複数のエージェントをどのようにサポートしているかを考慮する場合に当てはまります。このモデルでは、テナントはすべてのコラボレーションエージェントにまたがる ID にバインドされている必要があります。

一般に、エージェントドメインには、エージェントシステムの現在および新たなニーズに沿った、よりクロスカットのアイデンティティモデルが必要です。エージェントプロバイダーには、オペレーティングシステムに付属する一意のセキュリティ、コンプライアンス、認可モデルをサポートする ID メカニズムが必要です。これは、システムが顧客または他のエージェントによって構成されている環境では特に困難です。オンボーディングされた各エージェントは、アイデンティティとテナントコンテキストをエージェントのインタラクションに接続する必要があります。次の図は、agent-to-agent (a2a) インタラクションの一部である潜在的なアイデンティティとテナントコンテキストの課題を示しています。



この図は、ここで説明したエージェントシステムの一部としてやり取りするプロバイダー構築のエージェントを示しています。ID とテナントコンテキストが改良されました。このシナリオは、複数のエントリポイントをサポートするエージェントシステムの例です。このシステムの各エージェントは、システムまたはユーザーを特定のテナントに解決するために独自の認証メカニズムが必要であることを前提としています。これらのエージェントがやり取りすると、テナントコンテキストは JSON ウェブトークン (JWT) に渡されます。JWT は、アクセスを許可し、テナントコンテキストをエージェントに挿入するために使用されます。

概念的には、このシナリオの主な違いは、エージェントが個別にデプロイして運用することです。つまり、各エージェントはアイデンティティを解決し、アクセスを許可できる必要があります。重要なのは、そのアイデンティティには、より広範なエージェントシステムのニーズを処理するための分散機能が必要であることです。また、エージェントがテナントコンテキストを共有する方法についても調整する必要があります。

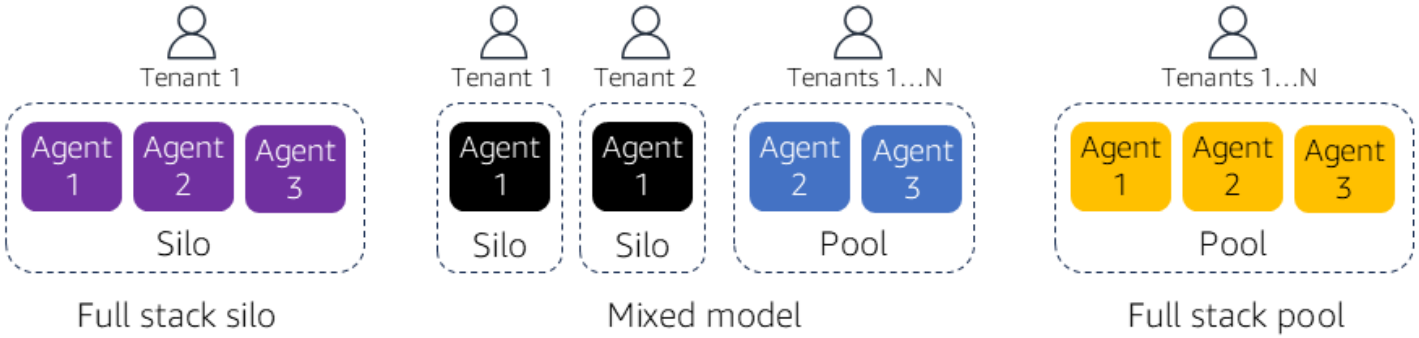
SaaS ビジネス価値を AaaS に適用する

一般的に、as-a-service モデルでシステムを実行する場合、エクスペリエンスの性質と、その技術的および運用上のフットプリントがビジネス成果にどのように影響するかを考慮します。例えば、SaaS を採用する場合、組織は規模、運用効率、コストプロファイル、俊敏性の経済を活用して、成長、マージン、イノベーションを推進します。

AaaS として配信されるエージェントは、同様のビジネス成果を目標とする可能性があります。複数のテナントをサポートすることで、エージェントはリソースの消費をテナントアクティビティに合わせることができます。これにより、従来の SaaS 環境に伴うスケールメリットが得られます。AaaS を使用すると、組織はエージェントを頻繁にリリースし、エージェントプロバイダーの俊敏性を高める方法でエージェントを管理、運用、デプロイすることもできます。重要なのは、AaaS モデルがテクノロジーに依存しないことです。成長を促進し、導入を合理化し、運用を簡素化するビジネス戦略を作成および推進します。

エージェントデプロイモデル

基本的な AaaS エクスペリエンスでは、プロバイダーはさまざまなパターンを使用してエージェントをデプロイできます。顧客、パフォーマンス、コンプライアンス、地域、セキュリティのニーズを満たすためにエージェントをデプロイする方法に影響を与えるさまざまな要因があります。さまざまなデプロイ戦略は、エージェントの設計、実装、消費方法に影響します。ここでは、従来のマルチテナント用語を導入して、さまざまなデプロイ戦略にラベルを付けることができます。次の図は、AaaS 環境にエージェントをデプロイするためのさまざまな置換を示しています。

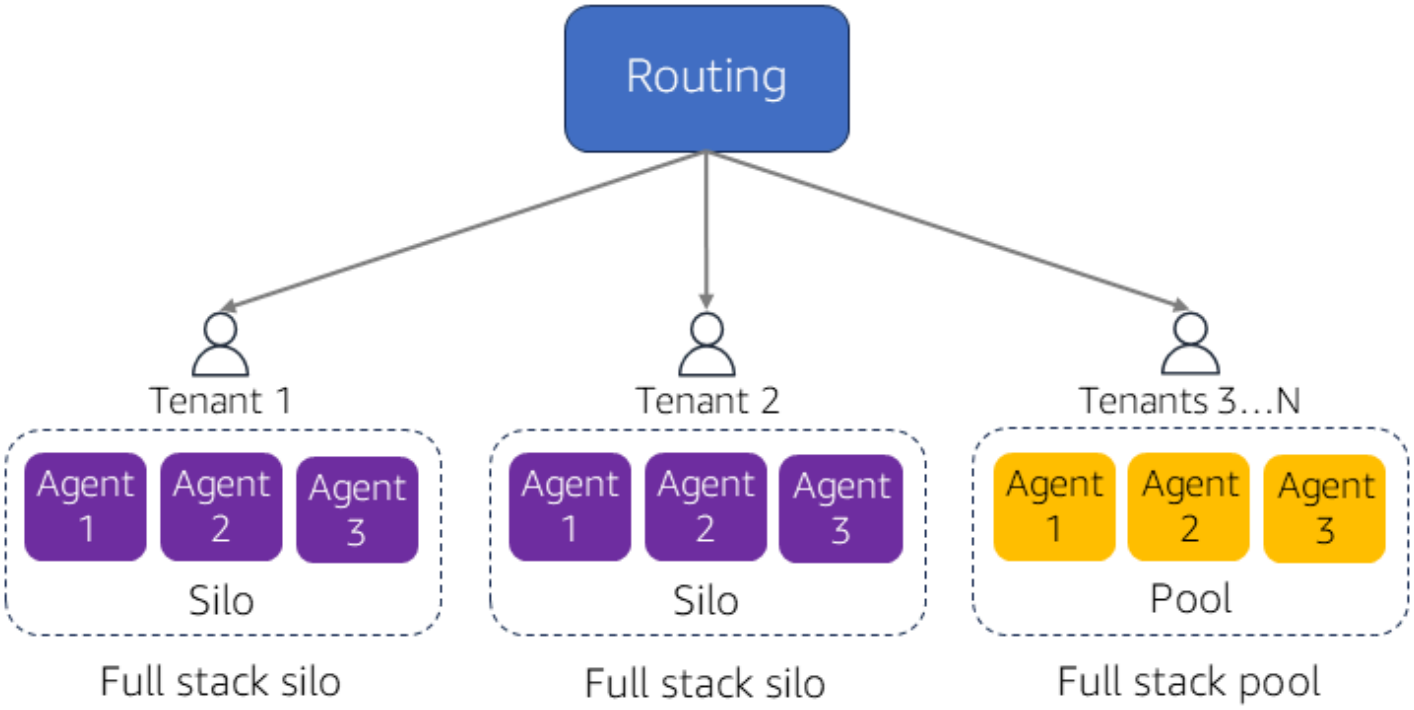


この図は、エージェントのデプロイの 3 つのモードを示しています。左側にはサイロ化されたモデルがあり、各テナントには完全に隔離されたエクスペリエンスと専用のエージェントのセットが提供されます。このシナリオでは、エージェントはコンピューティング、リソース、または実行環境をテナント間で共有しません。

中間の例では、テナントがサイロ化されたエージェントとプールされたエージェントの組み合わせを使用するハイブリッドモデルを示しています。例えば、エージェント 1 はサイロ化されたモードでデプロイされ、各テナントは専用インスタンスを受け取ります。エージェント 2 と 3 はプールされたモデルで動作し、テナント間でリソースを共有します。

右側には、すべてのエージェントがテナント間で共有され、従来のマルチテナントデプロイを提供する、完全にプールされたモデルがあります。このシナリオでは、テナントはエージェント実行に一般的なコンピューティング、メモリ、サービスインフラストラクチャを活用します。

つまり、エージェントは、コンピューティングリソースと依存リソースをテナント間で専用 (サイロ化) または共有 (プール化) して、さまざまなデプロイモデルで運用できます。これらのデプロイ戦略は相互に排他的ではありません。エージェントサービスは、多くの場合、パフォーマンス、分離、コスト、スケーラビリティのバランスを取るために、両方のモデルを組み合わせて、さまざまな顧客のニーズをサポートします。次の図は、同じ運用環境内の複数のデプロイ設定をサポートするエージェントシステムを示しています。

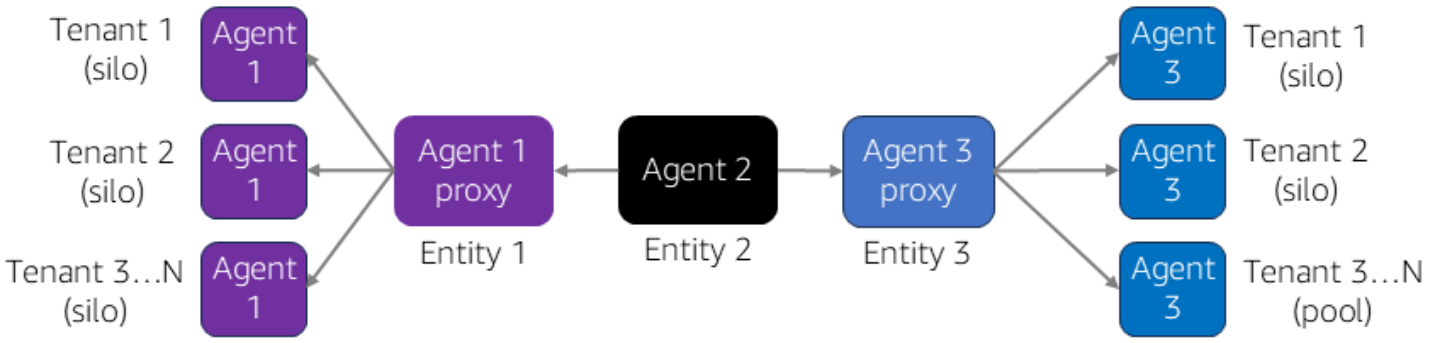


この図では、エージェントプロバイダーには、サービスとしてのエージェント (AaaS) を通じてデプロイされる 3 つのエージェントがあります。2 種類のテナントをサポートしています。左側では、2 つのテナントにフルスタックのサイロモデルを通じて対処するコンプライアンス要件とパフォーマンス要件があります。右側の残りのテナントは、テナントがリソースを共有するプールモデルで実行されます。

俊敏性と運用効率が目標の場合は、テナントごとのデプロイモデルのサポートに関連する影響を制限してみてください。つまり、ルーティングやその他のエクスペリエンスメカニズムを導入することで、エージェントを 1 つの画面で管理、運用、デプロイできます。

ローコード環境またはノーコード環境でエージェントを構築する場合、サイロ化されたエージェントやプールされたエージェントという概念はありません。代わりに、エージェントは別のエージェントによって完全に管理される場合があります。サイロモデルとプールモデルは、組織がエージェントの構造とフットプリントを制御する環境にさらに適用されます。この場合、チームはどのデプロイモデルをサポートするかを検討する必要があります。

表面的には、これらのデプロイモデルは、より広範なシステムでのエージェントの機能に直接影響しません。エージェントは、サイロモデルまたはプールモデルにデプロイされている他のエージェントを直接認識していない可能性があります。代わりに、これらのデプロイ戦略を環境内のルーティングコンストラクトの一部として実装できます。次の図は、サイロ化されたモデルとプールされたモデルをルーティング戦略を使用して実装する方法の例を示しています。



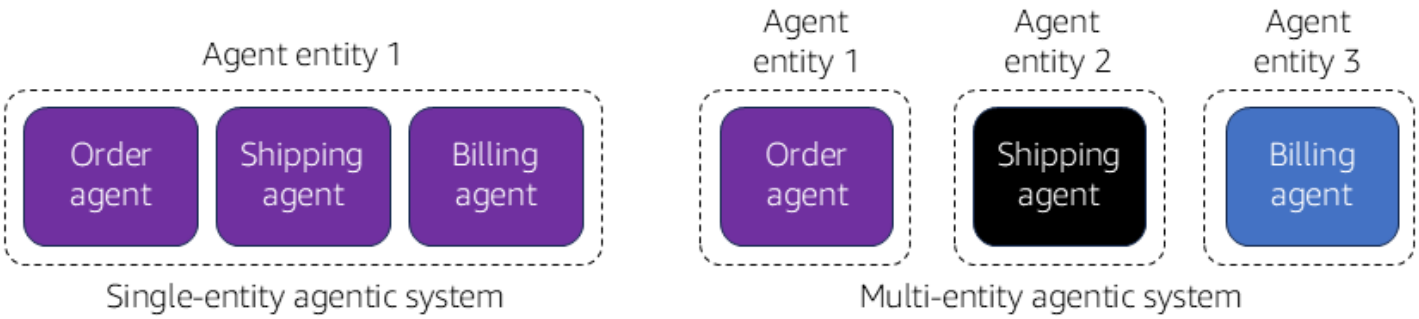
この例では、3 つの異なるプロバイダーの 3 人のエージェントが含まれています。各エージェントプロバイダーには、独自のデプロイ戦略を実装するオプションがあります。たとえば、エージェント 1 はプロキシを使用して、サイロ化されたテナントエージェントのセットにインバウンドリクエストを配信します。エージェント 2 はルーティングを必要とせず、1 つのプールされたエージェントを介したすべてのテナントリクエストをサポートします。エージェント 3 は、一部のテナントがサイロ化され、他のテナントがプールされるハイブリッドモデルデプロイです。

これらのデプロイモデルをサポートするかどうかと方法は、ソリューションの性質によって異なります。どちらのモデルもサポートする必要はありません。ただし、コンプライアンス、ノイズの多い近隣、パフォーマンス、階層化など、この戦略のサポートを検討する必要があるインスタンスがある場合があります。

テナントコンテキストの紹介と適用

マルチテナンシーをサポートするエージェントを構築する場合は、まずテナントコンテキストを設定する方法を検討する必要があります。これは、エージェントの実装内でテナント固有のポリシー、戦略、メカニズムを適用するために使用されます。

最も基本的なレベルでは、従来のマルチテナントアーキテクチャで使用する一般的なツールとメカニズムを通じて、テナントコンテキストをエージェントに導入できます。これは、API キー、OAuth、またはその他のさまざまな検証メカニズムを通じて行うことができます。この多くの例は、認証されたシステムまたはユーザーをテナントコンテキストを保持する JSON ウェブトークン (JWT) キーに解決することに重点を置いています。その後、JWT はシステムを介して伝播されます。これは、エージェントシステムを構成する方法を検討すると、より興味深いものになります。次の図は、2 種類のエージェント環境の例を示しています。



この図では、左側のモデルは、すべてのエージェントが単一のエンティティによって所有、管理、ホストされているエージェントシステムを表します。エクスペリエンス全体を完全に制御できる場合は、一般的な戦略を使用してテナントを各エージェントに渡すことができます。

右側のモデルは、より一般的である可能性があり、複数のエンティティにまたがるエージェントのシステムを表します。エージェントは独立して構築、管理、運用されるため、それぞれに独自の認証および認可スキームがあります。ここでの課題は、これらのエージェント間でテナントコンテキストを解決して共有する共通の方法が必要であることです。これは、各エージェントがシステムまたはユーザーを認証し、適用されたメカニズムに従ってテナントに解決できる必要がある、より分散されたモデルに依存します。

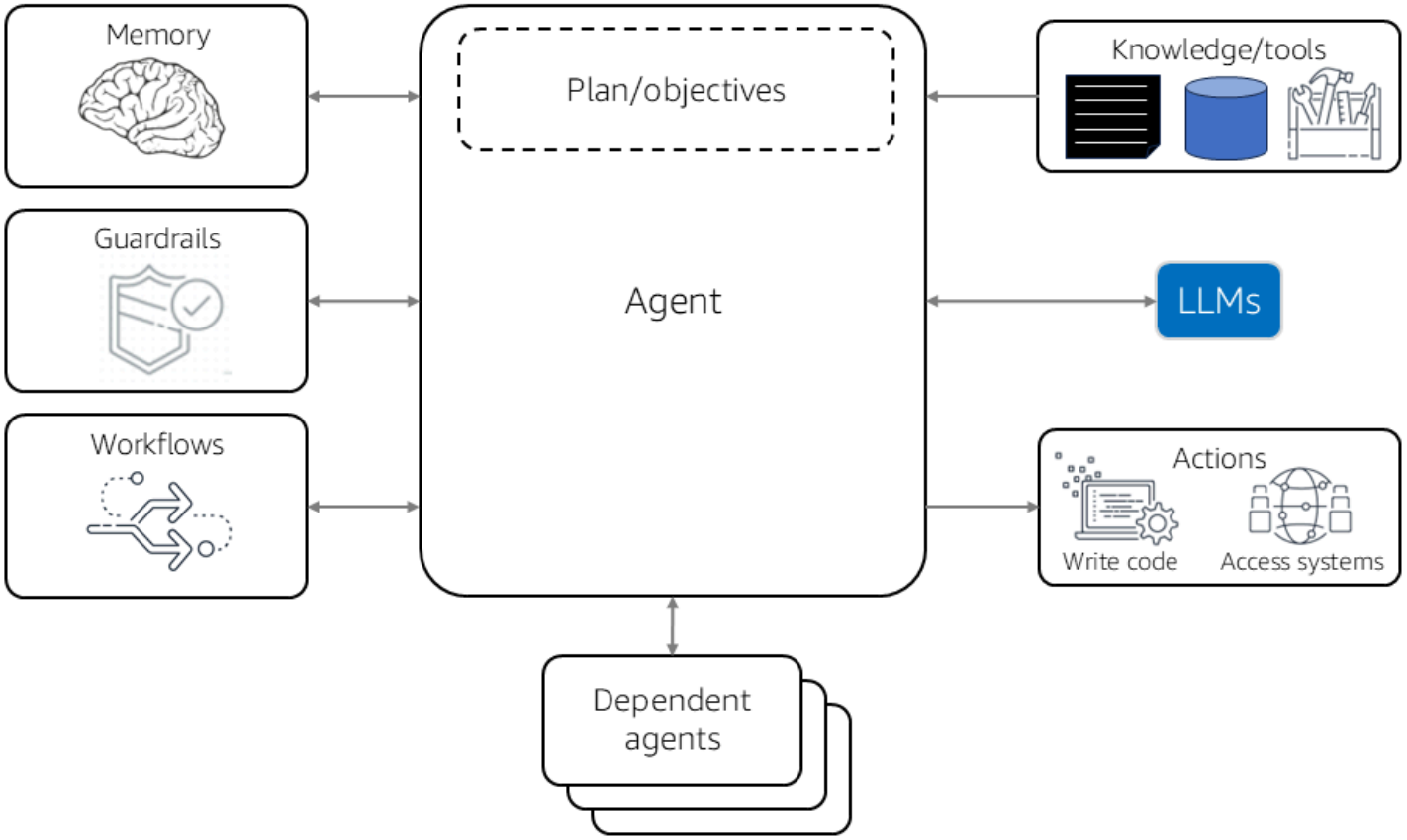
テナント対応エージェントの構築

マルチテナンシーは、個々のエージェントの実装方法に影響します。エージェントがリクエストを処理するときは、テナントコンテキストが、エージェントがデータにアクセスし、決定を行い、ア

クションを呼び出す方法にどのように影響するかを検討してください。マルチテナンシーがエージェントのプロファイルに与える影響と場所をよりよく理解するには、まずコンストラクトを任意のエージェントに含める方法を決定します。

課題は、プロバイダーがエージェントエクスペリエンスの設計について独自の選択を行うため、エージェントの範囲、性質、設計が具体的でないことです。最終的に、エージェントのポイントは、さまざまなツール、データソース、メモリにアクセスしてタスクを解決する方法を決定できる自律学習サービスであることです。

エージェントが使用する戦略とパターンを正確に把握することはそれほど重要ではありません。マルチテナントモデルでは、エージェントのさまざまな部分がどのように設定、アクセス、適用されるかを特定することがより重要です。目標を達成するために一連のリソースとメカニズムに依存する潜在的なエージェント環境を考えてみましょう。次の図は、このようなエージェントの例を示しています。



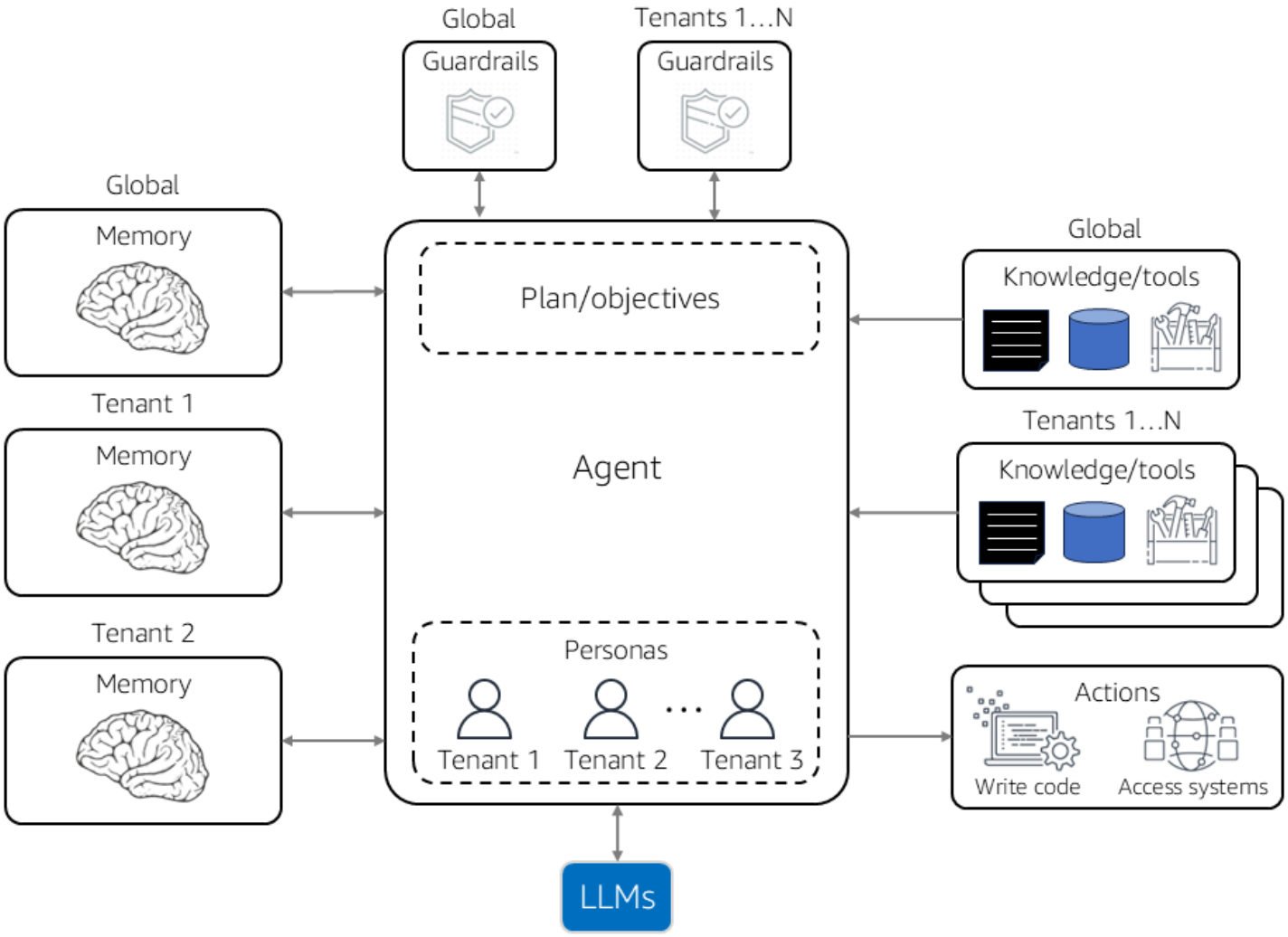
この図は、エージェントの可能性の包括的な範囲を表し、目標を達成するために組み合わせることができるさまざまなツールとメカニズムを示しています。図の左側では、エージェントがコンテキストの一部としてメモリにどのように依存しているか、アクティビティをガイドするポリシーを定義するためのガードレール、および特定のタスクに向けられるワークフローに注意してください。ワークフ

ローをこのコンテキストに含めるべきではないと主張する人もいるかもしれませんが、ワークフローがエージェントエクスペリエンスに不可欠なシナリオがあるかもしれません。

図の右側は、ナレッジやツールなどの入力が、エージェントの機能を強化する追加のインサイトやコンテキストをどのように提供できるかを示しています。次に、エージェントはコードの記述やシステムへのアクセスなどのアクションを出力します。図の下部は、エージェントがより広範なシステムの一部としてオーケストレーションできる 1 つ以上の内部エージェントまたはサードパーティーエージェントにどのように依存しているかを示しています。

マルチテナンシーを導入することの意味について考えてみましょう。テナンシーは、エージェントが行動やアクションを決定する戦略やメカニズムを導入する方法と場所を考慮するよう強制します。これにより、エージェントの知識、学習、ツール、メモリに関する考え方に別の側面が追加されます。

次に、マルチテナンシーをサポートするようにこのモデルを変更する方法を考えてみましょう。次の図は、マルチエージェントモデルの例を示しています。



この図では、エージェントがテナントコンテキストを統合する方法を形成することを目的としたテナントペルソナを紹介します。たとえば、図の左側では、テナント固有のメモリをサポートするようにエージェントメモリが変更されています。これは、エージェントがテナント固有の知識とツールをサポートしている図の右側にも当てはまります。ガードレールにも同じサポートが適用されます。

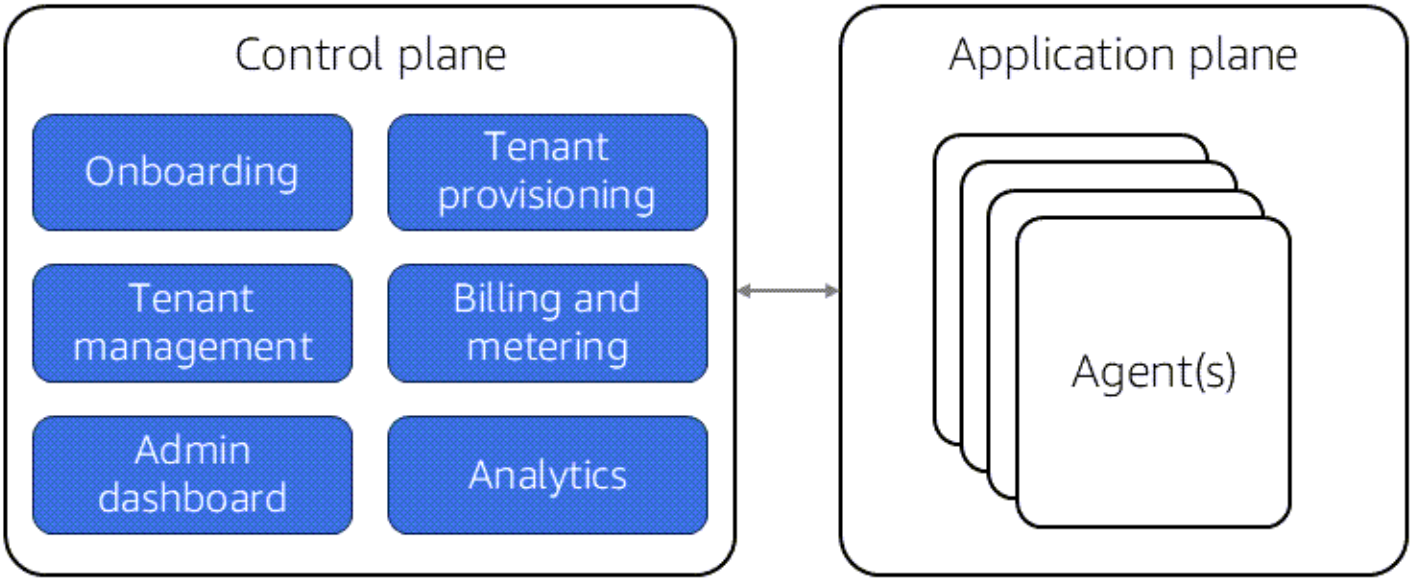
マルチテナントエージェントのすべての側面がテナントごとのリソースを必要とするわけではないため、これは極端な例かもしれません。重要なのは、特定のテナントに合わせてエージェントを調整することで、その効果を高める方法を検討する必要があることです。このアプローチにより、エージェントは影響と価値を高め、より関連性の高いコンテキストをレスポンスに提供し、特殊な機能を開発できます。エージェントは、さまざまなペルソナに一意に適したタスクを学習、適応、実行できるようになります。

主な考え方は、テナントコンテキストがエージェントの構築方法に直接影響することです。また、他のエージェントを含む外部エンティティとのテナントインタラクションを形成することもできます。マルチテナントエージェントを構築すると、ノイズの多い近隣、テナントの分離、階層化、スロットリング、コスト管理などの従来の課題が生じます。エージェントの設計とアーキテクチャは、これらの基本的なマルチテナントの概念に対処する必要があります。これについては、次のセクションで説明します。

エージェント環境でのコントロールプレーンの採用

マルチテナントのベストプラクティスでは、多くの場合、実装をコントロールプレーンとアプリケーションプレーンの 2 つの異なる部分に分割します。コントロールプレーンは、環境のテナントにまたがる運用、管理、オーケストレーションのメカニズムにアクセスするための単一のガラスペインを提供します。アプリケーションプレーンは、ビジネスロジック、機能、機能が存在する場所です。

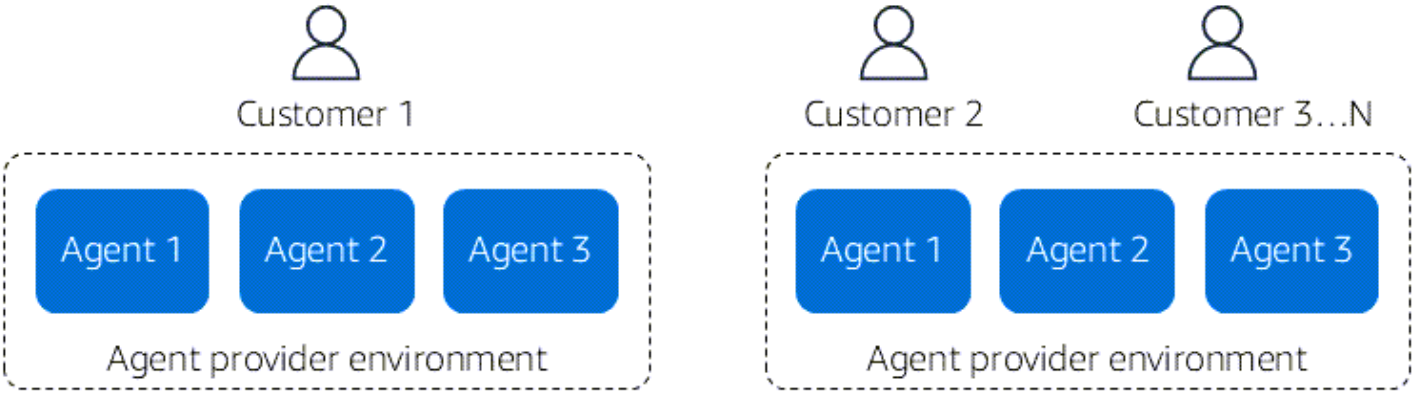
この責任分担は、エージェントモデルにも適用されます。マルチテナントエージェントには、ある程度の一元管理、運用、インサイトが必要であり、コントロールプレーンを通じてこれらのニーズに継続的に対処することが理にかなっています。次の図は、これらのプレーンが Agent as a Service (AaaS) 環境内でどのように分割されるかの概念図を示しています。



この図は、コントロールプレーンとアプリケーションプレーンの従来の分離を示しています。新しい点は、コントロールプレーンが AaaS 環境を構成するエージェントを管理するようになったことです。コントロールプレーンは、エージェントが 1 つのプロバイダーによって構築、管理、デプロイされていることを前提としているため、すべてのエージェントとやり取りします。

このモデルでは、特にエージェントのライフサイクルとサードパーティーの調整において、さらに複雑なレイヤーが導入されますが、懸念の基本的な分離は維持されます。コントロールプレーンは、エージェントの設定をオーケストレーションし、テナントとエージェントのオブザーバビリティを提供し、請求のために消費と計測データを収集し、テナントポリシーを管理することで、引き続き同じコア機能を提供します。

さまざまなプロバイダーのエージェントを組み込んだマルチエージェントシステムを検討すると、このシナリオはより複雑になります。次の図は、このようなモデルの例を示しています。



この図は、マルチエージェントシステムの一部であるさまざまなプロバイダーの 4 人のエージェントを示しています。サードパーティープロバイダーは引き続き各エージェントを運用およびデプロイします。これらのエージェントは、1 つ以上のプロバイダーからの認可されたアクセスを有効にするように設定されています。ただし、エージェントはプロバイダーの管理下にあるため、各エージェントは独自のコントロールプレーンを維持します。

基本的に、これらのマルチテナントエージェントは、他のエージェントと統合するサードパーティーのサービスとして動作します。そのため、エージェントの機能を一元的に操作、設定、管理するには、独自のコントロールプレーンが必要です。

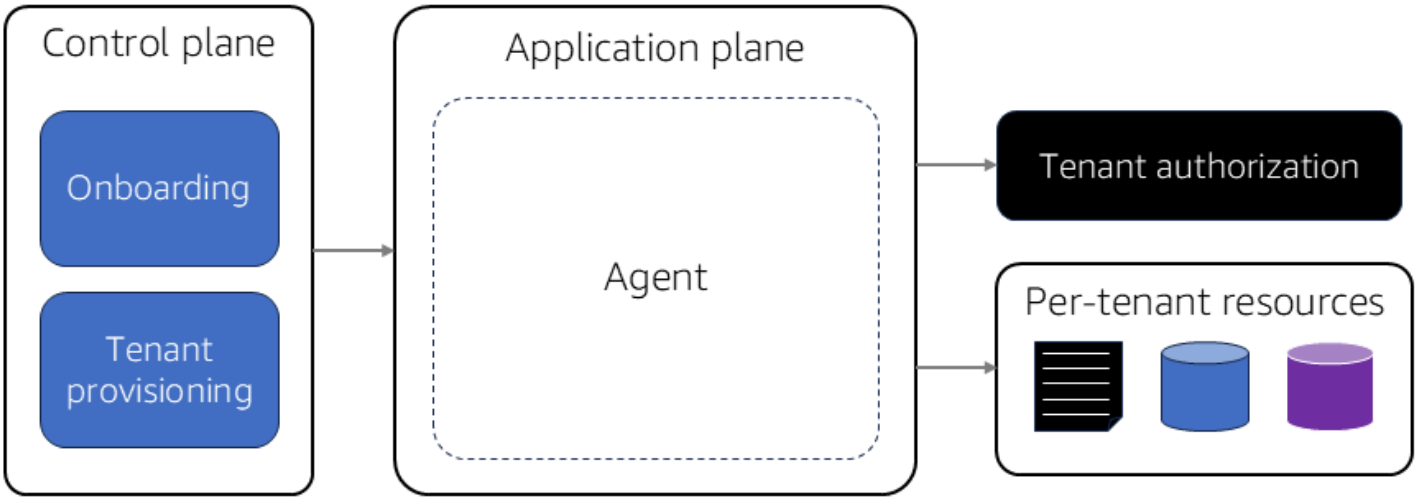
エージェントは、プロバイダーがホストするエクスペリエンスで実行される独立したサービスであることを前提としています。ただし、エージェントコンシューマーがエージェントをホストする方法と場所に対してより多くの制約を課すシナリオでは、これが明確でない場合があります。

エージェントへのテナントのオンボーディング

通常、オンボーディングは AaaS 環境の重要な部分です。テナントの作成、設定、プロビジョニングには、多くの場合、多くの可動部分、統合、ツールが必要です。エージェントオンボーディングエクスペリエンスには、テナントアイデンティティ、階層化、テナントごとのリソースのプロビジョニング、テナントポリシーの設定など、AaaS コントロールプレーンにあるのと同じサービスが必要になる場合があります。

エージェントオンボーディングへのアプローチは、エージェント環境のフットプリントとテナンシーモデルの影響を受けます。サイロ化されたエージェントとプールされたエージェントはそれぞれ独自のニュアンスを持ち、単一のエージェントまたは複数のエージェントのどちらを使用するかの選択も

オンボーディングプロセスに影響します。次の図は、オンボーディングがエージェントの設定にどのように影響するかの概念図を示しています。



エージェントをオンボードするたびに、コントロールプレーンはテナントがエージェントにアクセスするために必要なステップを実行する必要があります。テナントの導入方法はエージェント認可モデルによって異なりますが、エージェントリクエストを個々のテナントに関連付けるテナントアイデンティティを作成することを前提としています。このテナントコンテキストは、ルート、スコープ、およびコントロールアクセスに適用することで、エージェントのエクスペリエンスを決定します。

オンボーディングでは、エージェントが使用するテナントごとのリソースを設定する必要もあります。ここでは、コントロールプレーンのテナントプロビジョニングサービスが、エージェントが相談するテナント固有のデータとリソースにエージェントを接続します。

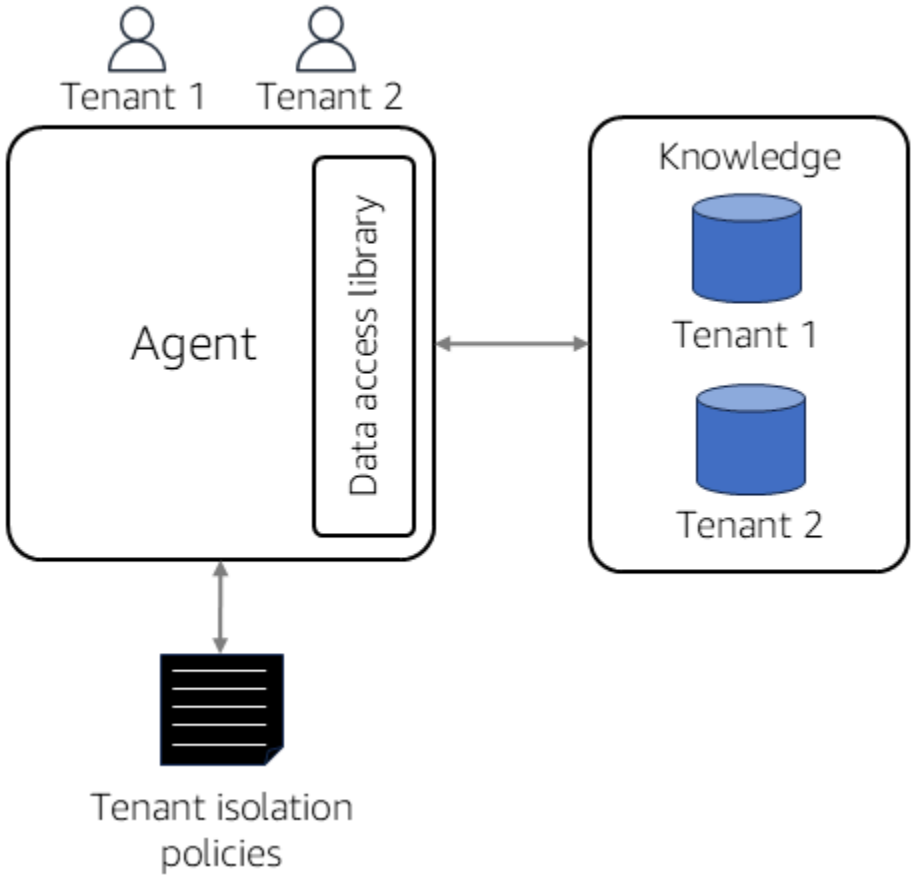
システムがサードパーティーエージェントの統合に依存している場合は、オンボーディングプロセス中にそれらのエージェントのニーズに対応する必要があります。これがどのように機能するかは、エージェント間のアクセスを許可するセキュリティおよび統合メカニズムによって異なります。理想的には、agent-to-agent認証と認可をオーケストレーションして設定するために必要なステップは、自動オンボーディングを通じて対処されます。

テナント分離の強制

テナント分離は、すべてのマルチテナント設定に適用される概念です。つまり、ポリシーと戦略により、1 つのテナントが他のテナントリソースにアクセスできなくなります。マルチテナントエージェントの場合、 とエージェントのテナント分離要件を強制するのに役立つコンストラクトとメカニズムを導入する必要がある場合があります。

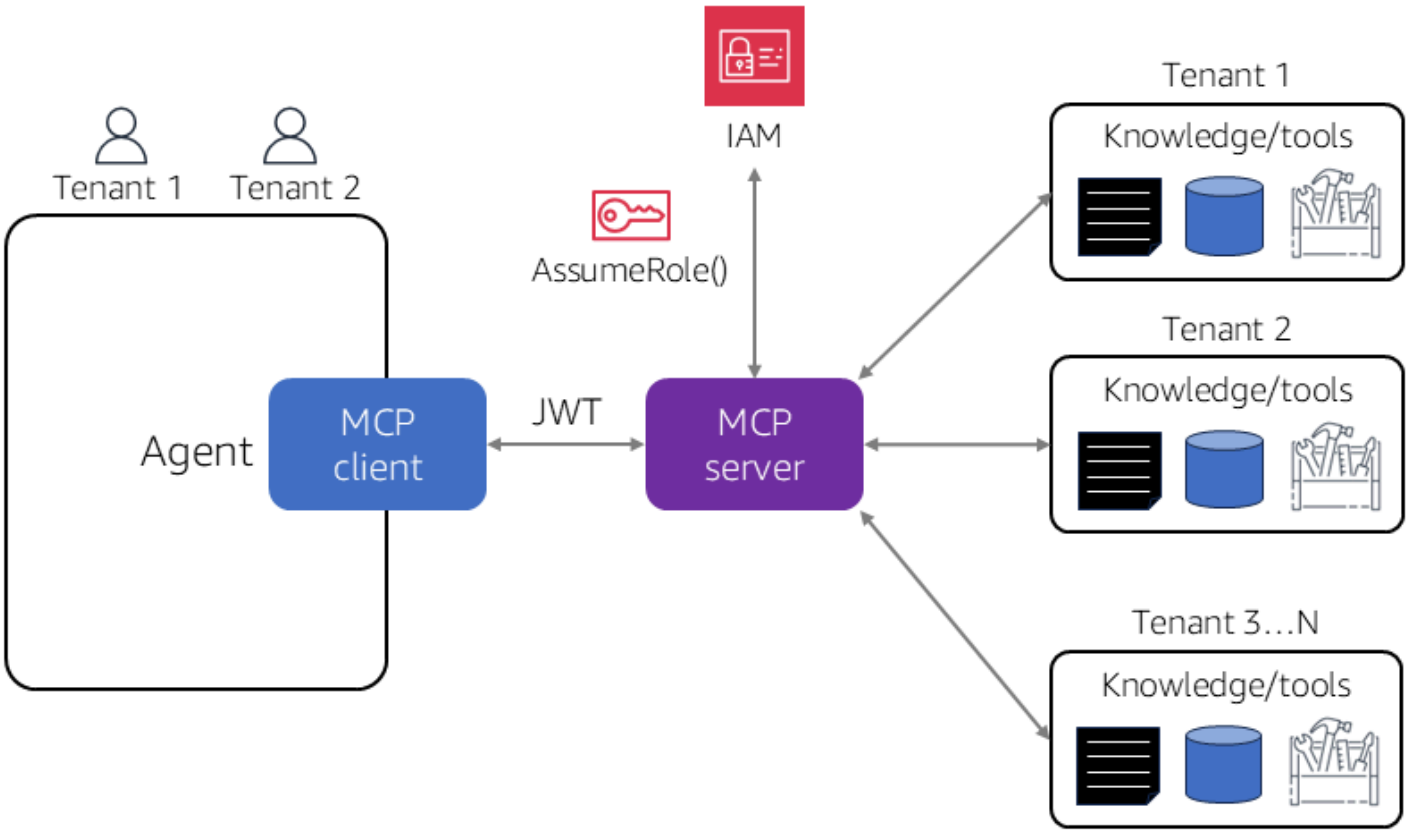
テナント分離の適用は、従来のマルチテナントシステムを使用する他の戦略と同様です。一般的に、AaaS アーキテクチャを構築するときは、リクエストまたはアクションがリソースにアクセスできるシステム内の任意の領域を特定し、リクエストがテナントの境界を超えているかどうかを判断します。たとえば、マイクロサービスにはテナントごとの専用 Amazon DynamoDB テーブルへの依存関係がある場合があります。そのためには、あるテナントのテーブルに別のテナントがアクセスできないようにするポリシーを導入する必要があります。

この場合、エージェントレンズによるテナント分離と、テナントごとのリソースとのやり取りを検討してください。次の図は、エージェントがテナント分離ポリシーを適用してテナントリソースへのアクセスを制御する方法の概念的な例を示しています。



この図の右側では、エージェントには個別のベクトルデータベースに保存されているテナントごとの知識があります。エージェントはリクエストを処理するときに、リクエストを行うテナントのコンテキストを調べます。これに基づいて、エージェントは適切な分離ポリシーを適用し、テナントが指定された境界外でデータまたはリソースにアクセスすることを制限できるようにします。

エージェントがモデルコンテキストプロトコル (MCP) を使用している場合は、テナント分離モデルを実装することもできます。次の図は、MCP を導入し、分離ポリシーを適用する方法の例を示しています。



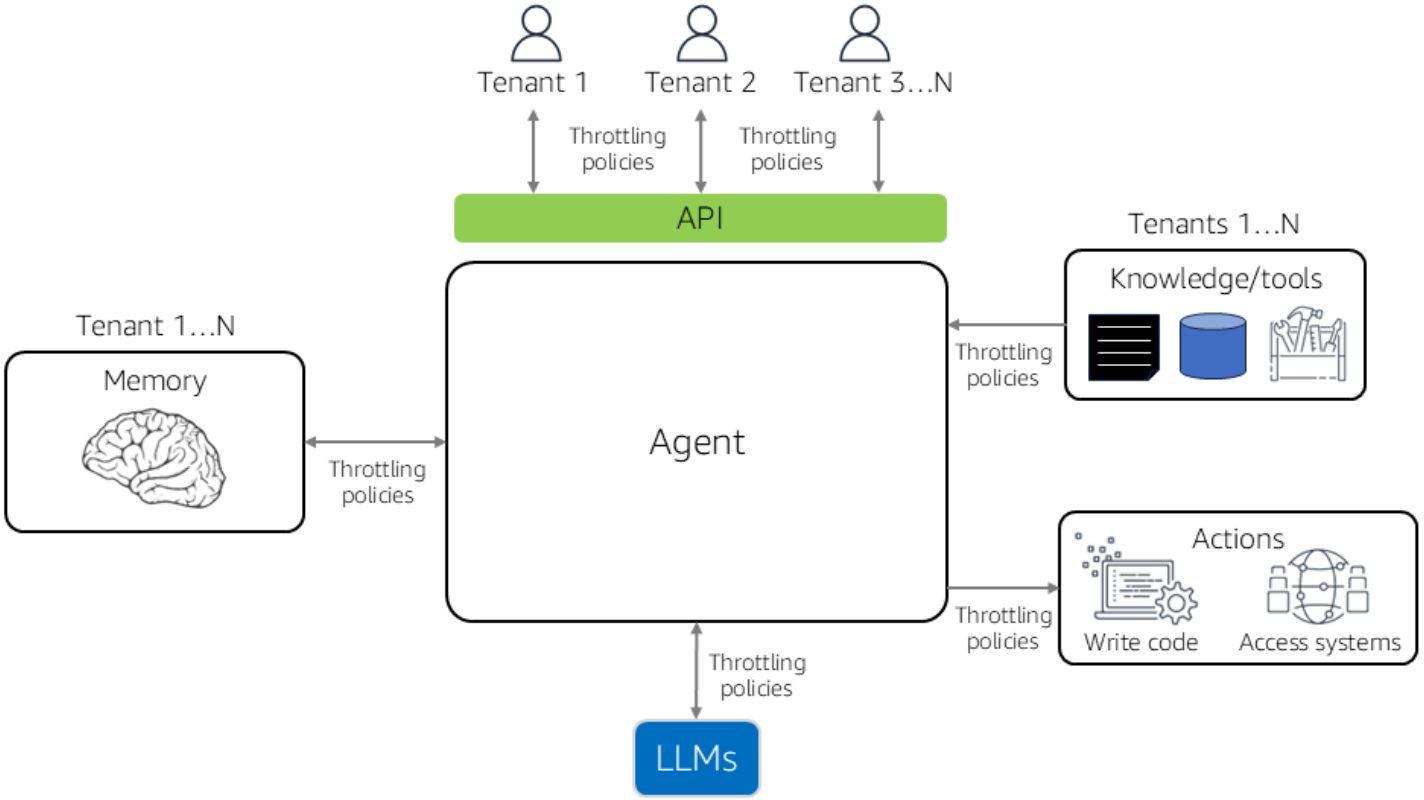
MCP は、エージェントがツール、データ、リソースと統合するために使用する標準化プロトコルです。この例では、MCP クライアントと MCP サーバーは、図の右側に示されているテナント固有の知識とツールとやり取りします。テナントコンテキストはクライアントからサーバーに流れ、サーバーはこのコンテキストを使用して AWS Identity and Access Management (IAM) サービスからテナントスコープの認証情報を取得します。認証情報は、各テナントのリソースへのアクセスを制御し、あるテナントが別のテナントのリソースにアクセスできるようにします。

エージェントはマルチテナンシーを組み込むため、リクエストを処理する際にテナント分離ポリシーを適用するメカニズムを導入する必要があります。場合によっては、IAM はテナントリソースへのアクセスを制限するのに役立ちます。他のインスタンスでは、テナント分離ポリシーを適用するために、他のツールやフレームワークを導入する必要がある場合があります。

ノイズの多いネイバーとエージェント

複数のテナントがエージェントを共有するマルチテナント AaaS 環境では、ノイズの多い近隣条件を防ぐポリシーを導入する場所と方法を検討してください。ポリシーは、すべての消費に適用される汎用スロットリングを導入することも、特定のペルソナに基づいてスロットリングを適用するテナントまたは階層ベースのポリシーを導入することもできます。基本階層のテナントには、プレミアム階層のテナントよりも消費制限が厳しくなる場合があります。

このスロットリングの概念は、複数のアーキテクチャポイントに適用できます。次の図は、ノイズの多い近隣ポリシーを導入できる領域の例を示しています。



マルチエージェント実装の以前のレビューでは、エージェントが使用できるさまざまなリソースを調べ、エージェント内のテナントごとのリソースの可能性を強調しました。各タッチポイントはスロットリングポリシーを導入する可能性のある領域であり、テナントがシステムまたはテナントの階層化ポリシーの消費制限を超えないようにします。

ノイズの多い近隣保護を導入する最適な場所は、テナントがリソースを共有するアーキテクチャのポイントです。コンピューティング、メモリ、APIs、大規模言語モデルなど、これらの共有またはプールされたコンポーネントは、単一のテナントが不釣り合いに消費した場合、パフォーマンス低下の影響を受けやすくなります。

スロットリングを適用する自然な場所の 1 つは、エージェントのエントリポイントにあり、「外側エッジ」と呼ばれることもあります。ここでは、エージェントがリクエストの処理を開始する前に、グローバルレート制限またはtenant-tier-basedレート制限を導入できます。スロットリングは、エージェントが LLM を呼び出すとき、メモリにアクセスするとき、共有ツールを呼び出すときなど、実行パスのより深く適用することもできます。

これらのポリシーは、公平な使用を強制し、負荷がかかってもエージェントの耐障害性を維持し、テナント間で一貫したエクスペリエンスを維持するのに役立ちます。目標によっては、一般的なシステム保護 (耐障害性) やテナントエクスペリエンスの詳細な管理 (階層ベースの使用権限など) に重点を置くことができます。

データ、オペレーション、テスト

エージェントとデータの所有権

エージェント実装のレビューでは、エージェントが特定のテナントのデータに依存するシナリオに焦点を当てています。この場合、データのライフサイクルと、さらに重要な点として、データの保存場所を考慮します。これは、データの性質がエージェントのアクセス方法に影響する業界やユースケースにとって特に重要です。

AaaS プロバイダーは、エージェントのオンボーディング、分離、オペレーションに影響を与える可能性のあるマルチテナント環境でのデータ問題の解決方法を評価する必要があります。適用可能なニュアンスと戦略は、使用するツール、テクノロジー、データによって異なります。これはさまざまな方法でアプローチできます。これは、AaaS サービスを作成する際に注意すべきことです。

マルチテナントエージェントのオペレーション

エージェント環境を構築するときは、エージェントの運用と管理の方法を検討してください。プロバイダーには、エージェントのヘルス、スケール、アクティビティをモニタリングできるメトリクス、データ、インサイト、ログが必要です。これは、個々のテナントがエージェントリソースをどのように消費するかを理解したいマルチテナントエージェント環境でより顕著になります。

これは、エージェントのインタラクションに関するインサイトが必要な場合に、マルチエージェント設定でさらに重要です。エージェント間のアクティビティをプロファイリングして追跡できることは、システムの規模、精度、有効性に影響する問題のトラブルシューティングに不可欠な場合があります。

運用チームは LLM インタラクションをプロファイリングして、エージェントが LLMs に配置する負荷をより適切に把握することもできます。このデータは、エージェントの実装を改良するために不可欠です。また、運用チームに、エージェントとテナンシーがシステムの全体的なコストプロファイルにどのように影響するかを示すこともできます。

マルチテナントエージェントのトレーニングとテスト

エージェントの構築に関連する課題の 1 つは、エージェントが学習して進化することが期待されていることです。また、エージェントを本番環境に移行する前に、エージェントをテストし、改良し、精度を向上させる必要があることも意味します。エージェントがインテントを正しく評価および分類

しているか、適切なツールやアクションを選択して呼び出しているかを検査および評価できる領域は多数あります。変数のリストは多岐にわたりますが、最終的には、エージェントが目標を達成するための結果を見つけられるようにすることです。

テストエージェントに関連するすべての可動部分と原則を調べることは、このドキュメントの範囲外ですが、テスト戦略はマルチテナント AaaS 環境に複雑さを追加することに注意してください。たとえば、エージェントに各テナントにコンテキスト的に適用されるデータ、メモリ、およびその他のコンストラクトがある場合、エージェントの結果はテナントごとのリソースによって形成できます。

エージェントを使用してシナリオをシミュレートする場合は、テナント固有のユースケースに合わせてシミュレーションを拡張する必要がある場合があります。それに応じて、検証基準がテナントごとに異なるインスタンスを許可するには、検証手順を絞り込む必要があります。

考慮事項と議論

SaaS はどこに適していますか?

業界の専門家は、エージェントが Software as a Service (SaaS) ランドスケープにどのように影響するかについて積極的に議論します。エージェントが多くのシステムのソフトウェアを変更していることは事実ですが、エージェントが配信モデルを廃止することを提案するのは難しいです。一部の SaaS プロバイダーは、エージェントを採用することで混乱する可能性が高く、エージェント・アズ・ア・サービス (AaaS) モデルに頼ることで、自社の価値提案を完全に再検討する場合があります。特定のニーズに対応するためにエージェントを選択的に導入することでバランスを取る人もいるかもしれません。

このトピックは、最高の SaaS 原則を採用することが SaaS の次の進化を表す可能性があるため、興味深いものです。これは、SaaS が進行中であることを意味するか、SaaS の基本原則がエージェントベースのモデルでパッケージ化および実現されていることを意味する可能性があります。用語が最終的にどこに到達するかを決定することはおそらくそれほど重要ではありませんが、概念としての SaaS が消える可能性は低いようです。エージェントが SaaS フットプリントを形成する可能性が高くなります。

最終的には、どの戦略を AaaS に適用できるかを決定する必要があります。つまり、プロバイダーがエージェントシステムの効率、価値、影響を最大化できるように、組織がエージェントアーキテクチャとビジネス戦略を採用できるようになります。エージェントはブラックボックスではありません。エージェントはリソースを消費し、オペレーションをスケールし、データに依存し、コストを生成します。プロバイダーが対処する必要があるすべての要因です。エージェントプロバイダーは、マルチテナントの原則がサービス提供をどのように形成し、運用モデルを最適化できるかを評価する必要があります。

説明

エージェントランドスケープは、ドメイン、意図したユースケース、ターゲット業界に応じて設計が変化するにつれて進化し続けています。この進化の一環として、アーキテクトがエージェントを設計および構築する際に考慮する戦略、パターン、トレードオフのビューをさらに改良することが含まれます。

包括的なエージェント戦略は、ビジネス目的と技術目的の両方と一致している必要があります。これには、ターゲット市場とペルソナの定義、料金設定とリソース管理戦略の確立、エージェントがより

大きなシステムにどのように適合するかの判断が含まれます。これらの考慮事項は、スケール、コスト効率、イノベーションが主な目標である AaaS を提供する際に特に重要です。

運用能力も同様に重要です。環境は、エージェントのアクティビティ、ヘルスメトリクス、使用パターンのモニタリングをサポートしている必要があります。これは、オペレーションを独立したエージェント間で調整する必要があるマルチエージェントシステムではより複雑になります。

全体として、エージェントに関するこの説明は、エージェントシステムの一部である可能性のあるさまざまなアーキテクチャ上の考慮事項の表面のみを傷付けます。適切なツール、フレームワーク、LLMs を選択するだけでなく、成功するかどうかは、スケーラビリティ、効率、デプロイ、マルチテナンシーのビジネス要件を満たすアーキテクチャの作成にかかっています。

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 規範ガイダンスによって提供される戦略、ガイド、パターンで一般的に使用される用語です。エントリを提案するには、用語集の最後のフィードバックの提供リンクを使用します。

数字

7 Rs

アプリケーションをクラウドに移行するための 7 つの一般的な移行戦略。これらの戦略は、ガートナーが 2011 年に特定した 5 Rs に基づいて構築され、以下で構成されています。

- リファクタリング/アーキテクチャの再設計 — クラウドネイティブ特徴を最大限に活用して、俊敏性、パフォーマンス、スケーラビリティを向上させ、アプリケーションを移動させ、アーキテクチャを変更します。これには、通常、オペレーティングシステムとデータベースの移植が含まれます。例: オンプレミスの Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行する。
- リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。
- 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: 顧客関係管理 (CRM) システムを Salesforce.com に移行する。
- リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの EC2 インスタンス上の Oracle に移行する。
- 再配置 (ハイパーバイザーレベルのリフトアンドシフト) – 新しいハードウェアを購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例： Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。

- 廃止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

A2A (Agent-to-Agent)

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

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

アクティブ/アクティブ移行

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

アクティブ/パッシブ移行

ソースデータベースとターゲットデータベースを同期させながら、データがターゲットデータベースにレプリケートされている間、接続しているアプリケーションからのトランザクションをソースデータベースのみで処理するデータベース移行方法。移行中、ターゲットデータベースはトランザクションを受け付けません。

[エージェント]

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

エージェントオペレーション

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

集計関数

複数行に処理を行い、グループ全体を対象に単一の戻り値を計算する SQL 関数。集計関数の例としては、SUM や MAX などがあります。

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

匿名化

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

アンチパターン

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

アプリケーション制御

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

アプリケーションポートフォリオ

アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の重要な要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。

人工知能 (AI)

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

AI オペレーション (AIOps)

機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。 AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。

非対称暗号化

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

原子性、一貫性、分離性、耐久性 (ACID)

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

属性ベースのアクセス制御 (ABAC)

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

信頼できるデータソース

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

アベイラビリティーゾーン (AZ)

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

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 のガイドラインとベストプラクティスのフレームワークです。 AWS CAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用という 6 つの重点分野にガイダンスを整理しています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションに関するガイダンスを提供します。詳細については、AWS CAF ウェブサイトと AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

データベース移行ワークロードを評価し、移行戦略を推奨し、作業見積もりを提供するツール。AWS WQF は AWS Schema Conversion Tool (AWS SCT) に含まれています。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。

B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

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

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。

ブルームフィルター

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

ブルー/グリーンデプロイ

それぞれが独立しているが、同一の環境を 2 つ作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (ブルー) で実行し、新しいアプリケーションバージョンを別の環境 (グリーン) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

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

ボットネット

マルウェアに感染しており、ボットハーダーまたはボットオペレーターと呼ばれる単一の当事者によって制御されているボットのネットワーク。ボットネットは、ボットとその影響力を拡大する仕組みとして、非常によく知られています。

ブランチ

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

ブレークグラスアクセス

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

ブラウンフィールド戦略

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

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど)。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、AWSでのコンテナ化されたマイクロサービスの実行ホワイトペーパーの「ビジネス機能を中心に組織化」セクションを参照してください。

ビジネス継続性計画 (BCP)

大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。

C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

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

CCoE

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

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

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

カオスエンジニアリング

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

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。

シチズンデベロッパー

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

クライアント側の暗号化

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

Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、 AWS クラウド エンタープライズ戦略ブログの CCoE 投稿を参照してください。

クラウドコンピューティング

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

クラウド運用モデル

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

導入のクラウドステージ

組織が、 AWS クラウドへの移行時に通常実行する 4 つの段階。

- プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略との関連性については、 AWS「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、GitHub や Bitbucket Cloud があります。コードの各バージョンはブランチと呼ばれます。マイクロサービスの構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

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

コールドデータ

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

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオといった、ビジュアル形式の情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI では、CV 用の画像処理アルゴリズムを利用できます。

設定ドリフト

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

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンの単一のエンティティとしてデプロイ

することも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

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

データドリフト

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

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

非一元的で分散型のデータ所有権を持つとともに、一元的な管理およびガバナンスを行えるアーキテクチャフレームワーク。

データ最小化

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

データ境界

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

データの前処理

raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。

データ出所

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

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには、一般的に、大量の履歴データが含まれており、多くの場合、それらはクエリや分析に使用されます。

データベース定義言語 (DDL)

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

データベース操作言語 (DML)

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

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせます。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

深層学習

人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。

多層防御

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

委任管理者

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

トラブルシューティング

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

開発環境

「環境」を参照してください。

検出管理

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

開発バリューストリームマッピング (DVSM)

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

デジタルツイン

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

ディメンションテーブル

スタースキーマにおいて、ファクトテーブルの定量データに関するデータ属性が含まれる小さいテーブル。ディメンションテーブルの属性は、通常、テキストフィールド、またはテキストのように扱える個別の数値で示されます。これらの属性は、一般的に、クエリの制約、フィルタリング、結果セットのラベル付けに使用されます。

ディザスタ

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

ディザスタリカバリ (DR)

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

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用す

る方法の詳細については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

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

DVSM

「開発バリューストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

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

電子データ交換 (EDI)

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

暗号化

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

暗号化キー

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

エンディアン

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

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

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

エンベロープ暗号化

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

環境

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

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

- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

エピック

アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。たとえば、 AWS CAF セキュリティエピックには、ID とアクセスの管理、検出コントロール、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。 AWS 移行戦略のエピックの詳細については、プログラム実装ガイドを参照してください。

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。

F

ファクトテーブル

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

フェイルファスト

開発ライフサイクルを短縮するために、頻繁かつ段階的にテストを行う哲学であり、アジャイルアプローチでは、この考え方がきわめて重要です。

障害分離境界

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

機能ブランチ

「ブランチ」を参照してください。

特徴量

お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。

特徴量重要度

モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

機能変換

追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。

数ショットプロンプト

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

FGAC

「きめ細かなアクセス制御」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

変更データのキャプチャによる継続的なデータ複製を利用して、段階的なアプローチではなく、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

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

FM ゲートウェイ

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

G

生成 AI

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

ジオブロッキング

「地理的制限」を参照してください。

地理的制限 (ジオブロッキング)

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

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローは古いと見なされている方法であり、トランクベースのワークフローは推奨されている新しい方法です。

ゴールデンイメージ

システムまたはソフトウェアのスナップショットであり、システムまたはソフトウェアの新規インスタンスをデプロイするテンプレートとして使用されます。製造の例で言えば、ゴールデンイメージを使用すると、複数のデバイスにソフトウェアをプロビジョニングして、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

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

ガードレール

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

ガードレール (AI)

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

H

HA

「高可用性」を参照してください。

異種混在データベースの移行

別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCTを提供します。

高可用性 (HA)

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

ヒストリアンのモダナイゼーション

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

ホールドアウトデータ

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

ヒューhuman-in-the-loop (HitL)

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

同種データベースの移行

お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。

ホットデータ

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

ホットフィックス

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

ハイパーケア期間

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

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

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

アイドル状態のアプリケーション

90 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。

IIoT

「インダストリアル IoT」を参照してください。

イミュータブルインフラストラクチャ

既存インフラストラクチャの更新、パッチ適用、変更などを行わずに、本番環境ワークロードに使用する新規インフラストラクチャをデプロイするモデル。本質的に、イミュータブルインフラストラクチャは、ミュータブルインフラストラクチャよりも一貫性、信頼性、予測性に優れています。詳細については、 AWS Well-Architected フレームワークにある「イミュータブルインフラストラクチャを使用してデプロイする」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

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

インダストリー 4.0

2016 年に Klaus Schwab 氏が提唱した用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩による、ビジネスプロセスのモダナイズを意味します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

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

インダストリアル IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「インダストリアル IoT (IIoT) デジタルトランスフォーメーション戦略の構築」を参照してください。

インスペクション VPC

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

IoT

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

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「IoT」を参照してください。

IT 情報ライブラリ (ITIL)

IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。

IT サービス管理 (ITSM)

組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、オペレーション統合ガイドを参照してください。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

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

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロードとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、「安全でスケーラブルなマルチアカウント AWS 環境のセットアップ」を参照してください。

大規模言語モデル (LLM)

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

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースアクセス制御」を参照してください。

最小特権

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

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

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

メインブランチ

「ブランチ」を参照してください。

マルウェア

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

マネージドサービス

AWS のサービス はインフラストラクチャレイヤー、オペレーティングシステム、プラットフォーム AWS を運用し、エンドポイントにアクセスしてデータを保存および取得します。

マネージドサービスの例として、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB が挙げられます。このサービスは、抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するソフトウェアシステムであり、工場では、これによって、原材料から製品を完成させます。

MAP

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

MCP

「モデルコンテキストプロトコル」を参照してください。

モデルコンテキストプロトコル (MCP)

エージェントツーツール通信のステートレスプロトコル。

MCP サーバー

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

メカニズム

ツールを作成してその導入を推進し、導入結果を調べて調整を行うための包括的なプロセス。メカニズムとは、運用中にそれ自体を強化し改善するサイクルを意味します。詳細については、AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

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

MES

「製造実行システム」を参照してください。

Message Queuing Telemetry Transport (MQTT)

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

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれ

る場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS 「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

組織がクラウドに移行するための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー、およびスプリントで作業する DevOps プロフェッショナルが含まれます。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

移行メタデータ

移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例としては、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

移行パターン

移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

オンラインツール。これによって、 AWS クラウドに移行するビジネスケースの検証に必要な情報を得られます。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイドを参照してください。MRA は、AWS 移行戦略の第一段階です。

移行戦略

ワークロードを AWS クラウドに移行するために使用するアプローチ。詳細については、この用語集の 7 Rs エントリと、「組織を動員して大規模な移行を加速する」を参照してください。

ML

「機械学習」を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、「AWS クラウドでのアプリケーションのモダナイズ戦略」を参照してください。

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定された

ギャップに対処するためのアクションプランが得られます。詳細については、「AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する」を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、「モノリスをマイクロサービスに分解する」を参照してください。

MPA

「Migration Portfolio Assessment」を参照してください。

MQTT

「Message Queuing Telemetry Transport」を参照してください。

多クラス分類

複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」 または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。

ミュータブルなインフラストラクチャ

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

O

OAC

「オリジンアクセス制御」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

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

OI

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」を参照してください。

オンライン移行

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

OPC-UA

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

Open Process Communications - Unified Architecture (OPC-UA)

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

オペレーショナルレベルアグリーメント (OLA)

サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。

運用準備状況レビュー (ORR)

質問と関連するベストプラクティスのチェックリスト。インシデントや起こり得る障害を理解、評価、防止したり、その範囲を縮小したりする際に役立ちます。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携させるハードウェアおよびソフトウェアシステム。製造分野では、Industry 4.0 への変革を進める上で、OT と情報技術 (IT) システムの統合に焦点が当てられています。

オペレーション統合 (OI)

クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイドを参照してください。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail をログに記録する によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの「組織の証跡の作成」を参照してください。

組織変更管理 (OCM)

人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイドを参照してください。

オリジンアクセス制御 (OAC)

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

オリジンアクセスアイデンティティ (OAI)

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

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

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

P

アクセス許可の境界

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

個人を特定できる情報 (PII)

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

PII

「個人を特定できる情報」を参照してください。

プレイブック

クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。

PLC

「プログラマブルロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

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

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。

ポートフォリオ評価

移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行の準備状況の評価」を参照してください。

述語

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

述語プッシュダウン

データベースクエリを最適化する手法。これによって、転送前にクエリ内のデータをフィルタリングします。この手法を取ると、リレーショナルデータベースから取得し処理する必要のあるデータの量が減少するため、クエリのパフォーマンスが向上します。

予防的コントロール

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

プリンシパル

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

プライバシーバイデザイン

開発プロセス全体を通してプライバシーが考慮されているシステムエンジニアリングのアプローチ。

プライベートホストゾーン

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

プロアクティブコントロール

非準拠リソースのデプロイ防止を目的としたセキュリティコントロール。このコントロールにより、プロビジョニング前にリソースをスキャンします。コントロールに準拠していないリソースは、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

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

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

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

プロンプトチェイニング

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

仮名化

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

発行/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。これにより、スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES の場合、マイクロサービスは、他のマイクロサービスがサブスクライブ可能なチャネルにイベントメッセージを発行できます。このシステムでは、発行サービスの変更なしに、新規マイクロサービスを追加できます。

Q

クエリプラン

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

クエリプランのリグレッション

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

R

RACI マトリックス

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

RAG

「検索拡張生成」を参照してください。

ランサムウェア

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

RASCI マトリックス

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

RCAC

「行と列のアクセス制御」を参照してください。

リードレプリカ

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

リアーキテクト

「7 Rs」を参照してください。

目標復旧時点 (RPO)

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

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

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

リグレッション

数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

リプラットフォーム

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または中断から回復するアプリケーションの機能。 AWS クラウドでの回復力を計画する際には、一般的に、高可用性とディザスタリカバリが考慮されます。詳細については、「AWS クラウド の耐障害性」を参照してください。

リソースベースのポリシー

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

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

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

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、「 AWSでのセキュリティコントロールの実装」の「レスポンシブコントロール」を参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

検索拡張生成 (RAG)

生成 AI の技術。これにより、LLM では、レスポンスの生成前に、トレーニングデータソースの外部にある信頼できるデータソースが参照されます。例えば、RAG モデルによって、組織のナレッジベースまたはカスタムデータのセマンティック検索を実行できる場合があります。細については、「RAG (検索拡張生成) とは何ですか?」を参照してください。

ローテーション

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

行と列のアクセス制御 (RCAC)

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

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。

S

SAML 2.0

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

SCADA

「監視制御とデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

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

セキュリティバイデザイン

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

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、主に 4 つの種類があります。4 つとは、予防、検出、レスポンシブ、プロアクティブです。

セキュリティ強化

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

Security Information and Event Management (SIEM) システム

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

セキュリティレスポンスの自動化

セキュリティイベントへの自動レスポンスまたは自動修復を目的として、事前定義およびプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例には、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータを、 AWS のサービス が受信する によって暗号化します。

サービスコントロールポリシー (SCP)

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

サービスエンドポイント

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

サービスレベルアグリーメント (SLA)

サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

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

責任共有モデル

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

シャドウ AI

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

SIEM

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

単一障害点 (SPOF)

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

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お

客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ」を参照してください。

SPOF

「単一障害点」を参照してください。

スタースキーマ

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

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

サブネット

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

監視制御とデータ取得 (SCADA)

製造分野において、ハードウェアとソフトウェアを使用して物理アセットと本番運用をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして、起こり得る問題を検出したり、パフォーマンスをモニタリングしたりすることで、システムをテストします。Amazon CloudWatch Synthetics を使用すると、こうしたテストを作成できます。

システムプロンプト

コンテキスト、指示、ガイドラインなどを提示して、LLM に動作を指示する手法。システムプロンプトは、コンテキストを設定して、ユーザーとやり取りするルールを確立するのに有用です。

T

タグ

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

ターゲット変数

監督された機械学習でお客様が予測しようとしている値。これは、結果変数のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。

タスクリスト

ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。

テスト環境

「環境」を参照してください。

トレーニング

お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。

tool

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

トランジットゲートウェイ

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

トランクベースのワークフロー

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

信頼されたアクセス

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

チューニング

機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。

ツーピザチーム

2 枚のピザを分け合えることができるくらい小さな DevOps チーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。

U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。

未分化なタスク

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

上位環境

「環境」を参照してください。

V

バキューミング

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

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

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

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

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

ウォームデータ

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

ウィンドウ関数

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

ワークロード

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

ワークストリーム

特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。

WORM

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

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

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

ゼロデイ脆弱性

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

ゼロショットプロンプト

LLM にタスク実行の手順は提示するが、実行のガイドとして役立つ例 (ショット) は提示しない方法。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

ゾンビアプリケーション

平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

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。