でのサーバーレスアプリケーションのテスト AWS

# AWS 規範ガイダンス

# AWS 規範ガイダンス: でのサーバーレスアプリケーションのテスト AWS

# Table of Contents

# でのサーバーレスアプリケーションのテスト AWS

アマゾン ウェブ サービス (寄???稿者)

2026 年 3 月 (ドキュメント履歴)

本ガイドでは、サーバーレスアプリケーションのテスト方法について解説し、テスト中に発生する問題について説明し、ベストプラクティスについてご紹介します。ご紹介するテストテクニックは、これまでよりも迅速に反復処理を行い、確信をもってコードをリリースできるようにすることを目指しています。

本ガイドは、サーバーレスアプリケーション用の、テスト戦略の作成を目指すデベロッパーを対象としています。始めに本ガイドを読み、テスト戦略について理解できたら、次は「Serverless Test Samples repository」へお進みください。本ガイドでご紹介したパターンやベストプラクティスに沿うテストの例をご覧いただけます。このガイドでは、サーバーレステストの方法論について説明し、サーバーレスアプリケーションをテストする際にお客様が直面する課題について説明し、サーバーレスアプリケーションをテストするためのベストプラクティスを紹介します。これらの手法は、デベロッパーがより迅速に反復処理し、より自信を持ってリリースできるようにすることを目的としています。

## 概要:

自動テストへの投資は、アプリケーションの質と開発スピードを確保するために欠かせません。テストを行えば、デベロッパーもフィードバックをしやすくなります。デベロッパーなら、アプリケーションの反復処理をすばやく行って、コードの質に関するフィードバックを集めたいと考えるはずです。デベロッパーの多くは、アプリケーションを、自分のデスクトップ環境で (オペレーティングシステムに直接、あるいはコンテナベースの環境内に) デプロイし、作成することに慣れています。デスクトップまたはコンテナベースの環境で作業すると、通常は、そのデスクトップで完全にホストされたコードに対してテストを作成することになります。一方、サーバーレスアプリケーションの場合は、デスクトップ環境にデプロイできず、クラウドにしか存在できないアーキテクチャコンポーネントが含まれることがあります。クラウドベースのアーキテクチャには、永続化レイヤー、メッセージングシステム、セキュリティコンストラクト、APIs、およびその他のコンポーネントが含まれる場合があります。こうしたコンポーネントに依存するアプリケーションコードを作成するときは、テストを設計し実行する最適な方法をみきわめることが、難しくなる場合があります。

本ガイドを読めば、摩擦や混乱を減らすテスト戦略に従い、コードの質を高めることができます。

# 前提条件

本ガイドの内容は、ソフトウェアの質を確保する自動ソフトウェアテストの実行方法など、自動テストの基本を理解していることが前提条件となります。このガイドでは、サーバーレスアプリケーションテスト戦略の概要を説明しており、テストの記述に関する実践的な経験は必要ありません。

# 定義

本ガイドでは、以下の用語を使用します。

- ユニットテストは、1 つのアーキテクチャコンポーネントのコードに対して、隔離して実行されるテストです。
- 統合テスト は、通常はクラウド環境で、2 つ以上のアーキテクチャコンポーネントに対して実行されます。
- End-to-endのテストでは、アプリケーションまたはワークフロー全体の動作を検証します。
- エミュレータは、クラウドリソースをプロビジョニングまたは呼び出すことなくクラウドサービスを模倣するように設計されたアプリケーション (多くの場合、サードパーティーから提供されます) です。
- モック (フェイクともいいます) は、依存関係をそのシミュレーションに置き換える、テストアプリケーションの実装のことです。

# 目的

本ガイドのベストプラクティスは、主に次の 2 つの目的を達成することを目指しています。

- サーバーレスアプリケーションの質を高める
  - アーキテクチャの境界でのテスト
  - コード境界でのテスト
- 機能の実装や変更にかかる時間を短縮する

## ソフトウェアの質を高める

アプリケーションの品質は、開発者がさまざまなシナリオをテストして機能を検証する能力に大きく依存します。自動テストを実装しない場合、またはより一般的には、テストが必要なシナリオを適切にカバーしていない場合、アプリケーションの品質を決定または保証することはできません。

サーバーベースのアーキテクチャの場合、チームは、テストすべき範囲を容易に規定することができます。アプリケーションサーバーで実行されるコードをすべてテストすればよいからです。それ以外の、サーバーを呼び出すコンポーネントや、サーバーが呼び出す依存関係などは、サーバーのアプリケーションを担当しているチームから無関係とみなされ、テストの範囲から外されることがよく起こります。

サーバーレスアプリケーションは、多くの場合、 AWS Lambda 関数のように、独自の環境で実行している小さな作業単位で構成されています。チームはこうした、1 つのアプリケーションの中にある、より小さな単位を複数担当することになります。アプリケーションの機能の中には、内部で開発されたコードを一切使用せずに、Amazon Simple Storage Service (Amazon S3) や Amazon Simple Queue Service (Amazon SQS) といったマネージドサービスに完全に委ねることができるものもあります。従来のサーバーベースのソフトウェアテストでは、マネージドサービスは、アプリケーションとは無関係とみなされて除外されることがあります。除外されるとテスト範囲が不十分になり、重要なシナリオが、手動での探索的テストや環境によって結果が変わる少数の統合テストのケースなどに限られる可能性があります。したがって、マネージドサービスの動作とクラウド構成までを含めたテスト戦略を採用すれば、ソフトウェアの質を高めることができるのです。

## アーキテクチャの境界でのテスト

サーバーレスアプリケーションは成長するにつれて、複数のアーキテクチャコンポーネントに自然に分散します。これは AWS 分散機能を使用しますが、end-to-endの動作を理解するのが難しい場合があります。

### 自然境界の特定

サーバーレスのベストプラクティス (1 つの関数 = 1 つのジョブ、デカップリング) に従ってアーキテクチャを設計すると、サブシステムの自然な境界に気付くでしょう。これらの境界は、アプリケーションの論理的な分離ポイントを表します。

### テスト契約としての境界

これらのアーキテクチャの境界は、エッジをテストするための優れた候補です。各境界を契約として扱い、定義された仕様に従って動作することを確認します。これらの境界は、テスト検証を挿入できるアプリケーションの継ぎ目と考えてください。

### 主な利点

以下は、アーキテクチャの境界でテストする主な利点です。

- 重点テストスコープ – アプリケーション全体を理解することなく、サブシステムを個別にテストします。

- 契約の検証 - システムが進化するにつれて、各境界が期待される動作を維持していることを確認します。
- 二用途計測 – これらの同じ境界により、本番環境で優れたオブザーバビリティフックになります。
- テストハーネス – 非同期サーバーレスシステムをテストできます。これは、サブシステムを通過するイベントをキャプチャして検証することで、イベント駆動型アーキテクチャをテストするのに役立ちます。

## コード境界でのテスト

Lambda コードなどのインフラストラクチャコードをコアビジネスロジックから分離して、明確なコードの境界を定義します。この分離により、テスト戦略を簡素化する個別のテストスコープが作成されます。

### 境界パターン

Lambda 関数に 2 つの明確なコード境界を確立します。

- 外部境界 (Lambda ハンドラー) – 固有の懸念を処理するスリムなアダプターレイヤー AWS Lambda
- 内部境界 (ビジネスロジック) – Lambda ランタイムから独立した純粋なビジネスロジックメソッド

### アダプターとしてのハンドラー (外部スコープ)

Lambda 関数ハンドラーは、次の薄いレイヤーである必要があります。

- 受信オブジェクト`event`と `context` オブジェクトからデータを抽出します。
- 抽出されたデータを検証します
- 関連する詳細のみをビジネスロジックメソッドに渡します
- Lambda に期待される形式で結果を返します

### ビジネスロジック (内部スコープ)

コアビジネスロジックは次の条件を満たす必要があります。

- Lambda に固有の詳細とは無関係に運用する
- シンプルで検証済みの入力を受け入れる

- 予測可能な出力を返す
- 初期化に最小限の依存関係が必要

スコープ別のテストの利点

- 内部境界テスト – Lambda の複雑さや環境設定のないビジネスロジックに関する包括的なユニットテスト
- 外部境界テスト – アダプターレイヤーのイベント処理とデータ抽出を検証する集中統合テスト
- 最小限のテストオーバーヘッド – テストの大部分に複雑な環境や広範な依存関係は必要ありません。

この境界ベースのアプローチにより、Lambda テストを最小限に抑えながら、ほとんどのコードを純粋な関数としてテストできます。

## 機能の実装や変更にかかる時間を短縮する

反復的な開発サイクル中にこれらの問題をキャッチすることで、ソフトウェアバグや設定の問題がコストやスケジュールに与える影響を最小限に抑えることができます。開発者がこれらの問題を検出できなかった場合、より多くの人が問題を特定するために追加の労力を費やす必要があります。

サーバーレスアーキテクチャには、API 呼び出しを通じて重要なアプリケーション機能を提供するマネージドサービスが含まれていることがあります。このため、開発サイクルには、ハッピーパス (これらのサービスとのやり取りが期待どおりに動作する) と悲しいパス (呼び出しが失敗する、予期しない応答を返す、または環境間で異なる動作をする) の両方を検証するテストを含める必要があります。これらのテストを実施しないと、ローカル環境とデプロイされた環境の違いに起因する問題が発生する可能性があります。この場合、修正の再現と検証にさらに時間を費やす必要があります。これは、各反復で、希望するセットアップとは異なる環境に対する変更の検証が必要になるようになったためです。

適切なサーバーレステストの戦略を使用すれば、他のサービスの呼び出しを含め正確な結果が得られるため、反復処理の時間を短縮することができます。

# サーバーレスアプリケーションのテスト方法

サーバーレスアプリケーションコードに対してテストを実行するには、主にモックテスト、エミュレーションテスト、クラウドでのテストの 3 つのアプローチがあります。通常、1 つのプロジェクトでは、これら 3 つの方法が混在して使用されているはずです。たとえば、コードをローカルで開発する場合はモックテスト、デプロイ前のサービス動作をより厳密にレプリケートするエミュレーションテスト、夜間の継続的インテグレーションと継続的デリバリー (CI/CD) プロセスの一環としてクラウドテストを使用できます。

## モックテスト

モックテストとは、コード内に、クラウドサービスの動作をシミュレートする代替オブジェクトを作成する戦略のことです。例えば、Amazon S3 サービスのモックを使用するテストを記述し、`PutObject`メソッドが呼び出されるたびに特定のレスポンスオブジェクトを返すようにモックを設定できます。テストを実行すると、このモックは、Amazon S3 やその他のサービスエンドポイントを呼び出すことなく応答を返します。

これらの代替オブジェクトはモックのフレームワークによって生成されることが多く、所定のテストで必要とされる実装の量を減らします。モックフレームワークの中には汎用的なものもあれば、AWS SDKs。

モックは、スタブとともに、フェイクのさらに広いカテゴリーに分類されます。モックは、エミュレータ (本セクションの後半で解説します) とは異なり、一般にデベロッパーによって、テストコードの一部として作成または設定されます。一方エミュレータは、エミュレート対象のシステムと同じ方法で API (REST API など) を公開する、スタンドアロンアプリケーションです。

モックを使用することのメリットは次のとおりです。

- モックは、API や Software as a Service (SaaS) プロバイダーなど、アプリケーションの制御が及ばないサードパーティのサービスを、それらのサービスに直接アクセスすることなくシミュレートできます。
- モックは障害状態、とりわけシミュレートが困難である状態 (サービスの停止など) のテストにも役立ちます。
- エミュレータと同様に、モックフレームワークは設定後に迅速なローカル開発を提供できます。
- モックは事実上あらゆる種類のオブジェクトの代替動作を提供できるため、モック戦略はエミュレータよりも幅広いサービスを対象とすることができます。

- 新しい機能や動作が利用可能になると、モックテストは一般的なモックフレームワークを使用して (またはそれにフォールバックして) より迅速に対応できます。モックフレームワークは、更新された AWS SDK が利用可能になるとすぐに新機能をシミュレートできます。

モックテストには次のような欠点があります。

- モックは通常、特にレスポンスを適切に模倣するためにさまざまなサービスからの戻り値を特定しようとする際に、かなりの分量のセットアップや設定が必要になります。
- モックは、テストを作成したデベロッパーが作成または設定するため、デベロッパーの責任が増えます。
- サービスの API と戻り値を理解するため、場合によってはクラウドにアクセスする必要があります。
- モックはメンテナンスが難しいこともあります。モックしたクラウド API の署名が変更されたり、戻り値のスキーマが進化したり、テスト対象のアプリケーションのロジックが拡張されて新しい API を呼び出したりするたびに、更新が必要になるためです。こうした変更は、デベロッパーのテスト開発の負担を増やします。
- モックテストはローカルに合格するが、レプリケートするのではなく、実際のサービスの動作をシミュレートし、環境固有の問題が検出されないようにするため、クラウドで失敗する可能性があります。
- エミュレータなどのモックフレームワークは、 AWS Identity and Access Management (IAM) ポリシー違反、サービスクォータ制限、ペイロードサイズの制約を検出することはできません。また、デプロイされた環境でのアプリケーション動作に影響を与える可能性のある Amazon CloudWatch AWS CloudTrail、Amazon GuardDuty などの補助サービスをトリガーすることもできません。
- モックは、認可に失敗した場合やクォータを超えた場合にアプリケーションが何をするかをシミュレートするのに適していますが、モックテストでは、コードが本番環境にデプロイされたときに実際に発生する結果を判断できません。

# エミュレーションテスト

エミュレーションテストは、 のようなエミュレータとして知られるローカルで実行されているアプリケーションによって有効になります AWS のサービス。エミュレータには、クラウド版と同様の戻り値を提供する API があります。また、API 呼び出しによって開始される状態変化をシミュレートすることもできます。例えば、Amazon S3 エミュレータは、 `PutObject`メソッドが呼び出されたときにローカルディスクにオブジェクトを保存する場合があります。が同じキーで呼び`GetObject`出されると、エミュレータはディスクから同じオブジェクトを読み取り、返します。

エミュレーションテストには、以下のような利点があります。

- エミュレーションテストの環境は、ローカル環境専用のコード開発に慣れているデベロッパーにとっては最も使いやすい環境です。例えば、n 層アプリケーションの開発に精通している場合は、本番稼働用と同様にデータベースエンジンとウェブサーバーをローカルマシンで実行して、迅速かつローカルで分離されたテスト機能を提供することができます。
- クラウドインフラストラクチャ (デベロッパークラウドアカウントなど) に変更を加える必要がないため、既存のテストパターンで簡単に実装できます。エミュレーションテストには、ローカルでの開発の反復処理をすばやく実行できるという利点があります。

ただし、エミュレータにはいくつかの欠点があります。

- 特に CI/CD パイプラインの一部として使用する場合、セットアップとレプリケートが難しいことがよくあります。これにより、IT のスタッフや、ソフトウェアのインストールを自分で管理しているデベロッパーの、ワークロードとメンテナンスが増える可能性があります。
- エミュレートされた機能や API はサービスの変更に後れを取るのが一般的で、サポートが追加されるまで新しい機能の導入が妨げられる可能性があります。
- エミュレータには、サポート、更新、バグ修正、機能パリティの強化が必要になる場合がありますが、これらはエミュレータの作成者 (多くの場合サードパーティーの企業) が責任を負います。
- エミュレータに依存するテストはローカルで成功する可能性がありますが、一般的にエミュレートされない IAM ポリシーやクォータ AWS など、 の他の側面とのやり取りが原因でクラウドで失敗する可能性があります。
- 一部の にはエミュレータ AWS のサービス が使用できないため、エミュレーションのみに依存する場合、アプリケーションの一部に対して十分なテストオプションがない可能性があります。

# クラウドでのテスト

クラウドでのテストは、デスクトップ環境ではなくクラウド環境にデプロイされたコードに対して、テストを実行するプロセスです。クラウドでテストする価値は、クラウドネイティブなアプリケーションの開発で増大します。例えば、次のようになります。

- 最新の APIs とサービス機能に対してクラウドでテストを実行することで、 AWS 引き続き新しいサービスや機能を起動する際に、最新の戻り値と動作がテストに反映されるようにできます。
- テストで、IAMポリシー、サービスクォータ、設定、すべてのサービスをカバーできます。

- クラウドのテスト環境は、本番のランタイム環境に最も近いため、クラウドでテストを実行すると、環境の進化とともに最も一貫性のある結果を得られる可能性があります。

クラウドでのテストには、以下のような欠点があります。

- クラウド環境へのデプロイは、通常、デスクトップ環境へのデプロイよりも時間がかかります。AWS Serverless Application Model (AWS SAM) Accelerate、AWS Cloud Development Kit (AWS CDK) watch mode、SST といったツールを使用すると、クラウドでのデプロイのイテレーションに伴う遅延を減らすことができます。
- クラウドのテストでは、ローカルの開発環境を使用するローカルテストとは異なり、サービスコストが追加で発生する場合があります。
- 高速インターネットアクセスがない場合、クラウドでのテストは実行可能性が低くなる可能性があります。
- 規制された業界では、エンタープライズセキュリティポリシーによって開発者のクラウド環境へのアクセスが制限され、ローカル開発ワークフローの一部としてクラウドテストを実行することが困難または不可能になる可能性があります。
- 環境の境界は、デベロッパー環境の共有アカウントではスタックレベルで引かれることが多く、プレフィックスを使用して所有権を特定するような、名前空間タイプの戦略が使用される場合もあります。本番稼働前環境と本番稼働環境では、一般にアカウントレベルで境界線が引かれます。これは、ノイズの多い近隣の問題からワークロードを隔離し、最小特権のセキュリティコントロールをサポートし、機密データを保護するためです。隔離された環境を作成する要件は、特にアカウントとインフラストラクチャを厳密に制御する企業に属している場合、DevOps チームに追加の負担がかかる可能性があります。

# サーバーレスアプリケーションをテストする際の課題

エミュレータとモック呼び出しを使ってローカルのデスクトップでサーバーレスアプリケーションをテストすると、コードが CI/CD パイプラインの環境間を移動したときにテストの不整合が生じることがあります。アプリケーションのビジネスロジックを検証するためにデスクトップで作成するユニットテストには、クラウドサービスの重要な側面が含まれていないか、正確に表現されていない可能性があります。完全なテストは、個別にローカルで実行することはできません。完全なテストを行うには、複数のサービス間でアクセス許可や構成を検証することが必要になります。

以下のセクションでは、クラウドテストの戦略を実装するときに遭遇するさまざまな課題について概説します。以下のセクションでは、クラウドテスト戦略を実装する際にお客様が経験する課題と、効果的なテストカバレッジを達成するためのベストプラクティスに関するガイダンスの概要を説明します。

## 例: Amazon S3 バケットを作成する Lambda 関数

Lambda 関数のロジックが Amazon S3 バケットの作成に依存している場合、完全なテストでは、Amazon S3 が正常に呼び出され、バケットが正常に作成されたことを確認する必要があります。モックテストの設定では、成功レスポンスをモックし、失敗レスポンスを処理するテストケースを追加することもあります。エミュレーションテストシナリオでは、`CreateBucket`API が呼び出される場合がありますが、呼び出しを行う ID はロールを引き受ける Lambda サービスから発信されるのではなく、代わりにプレースホルダー認証が使用されます。これは多くの場合、より寛容なロールまたはユーザー ID です。

以前に解説したモックとエミュレーションのセットアップでは、Amazon S3 の呼び出しに成功 (または失敗) した場合に Lambda 関数がどのような動作をするかが検証されます。ただし、これらのテストでは、関数の設定によっては、Lambda 関数がバケットを正常に作成できるかどうかをキャプチャできません。この設定は AWS CloudFormation、 AWS SAMや HashiCorp Terraform などの製品やサービスのコードとしてのインフラストラクチャ (IaC) によって表される可能性があります。HashiCorp 考えられる問題の 1 つは、関数に割り当てられたロールに `s3:CreateBucket`アクションを許可するポリシーがアタッチされていないことです。そのため、関数はクラウド環境にデプロイされると常に失敗します。

## 例: Amazon SQS からのメッセージを処理する Lambda 関数

Amazon Simple Queue Service (Amazon SQS) キューが Lambda 関数のソースである場合、完全なテストでは、メッセージがキューに入れられたときに Lambda 関数が正常に呼び出されたことを確

認する必要があります。エミュレーションテストとモックテストは通常、Lambda 関数コードを直接実行し、JSON イベントペイロード (または逆シリアル化されたオブジェクト) を関数ハンドラーの入力として渡すことで Amazon SQS 統合をシミュレートするように設定されます。

Amazon SQS 統合をシミュレートするローカルテストでは、特定のペイロードで Amazon SQS によって呼び出されたときに Lambda 関数が何をするかがテストされますが、クラウド環境にデプロイされたときに Amazon SQS が Lambda 関数を正常に呼び出すことは保証されません。

Amazon SQS と Lambda で発生する可能性のある設定の問題として、次のような例があります。

- Amazon SQS の可視性タイムアウトが短すぎるため、意図された 1 回のみの呼び出しが複数回発生する。
- Lambda 関数の実行ロールは、キューからのメッセージの読み取りを許可しません (`sqs:ReceiveMessage`、`sqs:DeleteMessage`、または 経由`sqs:GetQueueAttributes`)。
- Lambda 関数に渡されるサンプルイベントが Amazon SQS メッセージサイズクォータを超えている。したがって、Amazon SQS ではそのサイズのメッセージは送信できず、テストが無効になる。

これらの例が示すように、ビジネスロジックは対象とするがクラウドサービス間の構成は対象としないテストでは、信頼性の低い結果が得られる可能性があります。

# サーバーレスアプリケーションをテストするためのベストプラクティス

以下のセクションでは、サーバーレスアプリケーションをテストする際に効果的なテストカバレッジを達成するための、ベストプラクティスの概要を説明します。

## クラウドでのテストを優先する

優れた設計を持つアプリケーションでは、幅広い要件や条件を満たすさまざまなテスト方法を使用できます。ただし、現行のツールに基づく場合、可能な限りクラウドでのテストに重点をおくことが推奨されます。クラウドでのテストは、デベロッパーの待ち時間が多く、コストがかかり、DevOps の制御に追加の投資が必要になることがありますが、信頼性や精度は最も高く、すべてを網羅したテストカバレッジを達成できます。

テストを実行するユーザーは、隔離された環境にアクセスできる必要があります。理想的には、各デベロッパーには、同じコードで作業している複数のデベロッパーが、同じ名前のリソースに対して API コールをデプロイまたは呼び出そうとしたときに発生する可能性のあるリソース命名に関する問題を回避 AWS アカウント するための専用の が必要です。こうした環境では、不要な支出を避けるため、適切なアラートやコントロールを設定する必要があります。例えば、作成できるリソースのタイプ、階層、またはサイズを制限したり、推定コストが特定のしきい値を超えたときにメールアラートを送信したりすることができます。

1 つの を AWS アカウント 他の開発者と共有する必要がある場合、自動テストプロセスでは、開発者ごとに一意のリソースに名前を付ける必要があります。たとえば、CLI sam デプロイコマンドまたは sam 同期コマンドを引き起こす更新スクリプトまたは TOML AWS SAM 設定ファイルでは、ローカル開発者のユーザー名を含むスタック名を自動的に指定できます。

クラウドでのテストは、ユニットテスト、統合テスト、エンドツーエンドテストなど、テストのあらゆる段階で役立ちます。

## 必要に応じてモックを使用

モックフレームワークは、高速なユニットテストを記述するための有用なツールです。特に、数値の計算や財務計算、あるいはシミュレーションなど、社内の複雑なビジネスロジックをテスト対象とする場合にとても役立ちます。テストケースや入力の変動が多く、それらの入力によって他のクラウドサービスへの呼び出しのパターンや呼び出しの内容が変化しないユニットテストが想定されていま

す。このようなシナリオのモックテストを作成すると、デベロッパーの反復作業に要する時間を短縮することができます。

モックテストを使用しているユニットテストで対象とするコードは、クラウドでのテストでも対象に含める必要があります。これは、モックがまだデベロッパーのラップトップまたはビルドマシンで実行されており、環境がクラウドとは異なる設定になっている可能性があるために必要です。例えば、コードには、特定の入力パラメータで実行されるときに Lambda が割り当てるように設定されたよりも多くのメモリを使用する関数や時間がかかる AWS Lambda 関数が含まれる場合があります。または、コードに、同じ方法 (またはまったく) で設定されていない環境変数が含まれている場合があり、その違いによってコードの動作や障害が発生する可能性があります。

クラウドサービスのモックを使用して、それらのサービス統合の適切な実装を検証しないでください。他の機能をテストする場合は、クラウドサービスをモックしても問題ありませんが、クラウドでクラウドサービスの呼び出しをテストして、正しい設定と機能実装を検証する必要があります。

特に多数のケースを頻繁にテストする場合、モックはユニットテストに価値を追加することができます。この利点は統合テストでは低減します。接続ポイントの数が増えるほど、必要なモックを実装するための手間が増えるためです。エンドツーエンドのテストではモックを使うべきではありません。これらのテストは一般にモックフレームワークでは簡単にシミュレートできない状態や複雑なロジックを扱うためです。

## エミュレーションテストのトレードオフを理解する

エミュレータは、特定のユースケースで実用的な選択肢です。たとえば、インターネットアクセスが制限されている、一貫性がない、または遅い開発チームは、エミュレーションテストがクラウド環境に移行する前にコードで反復する最も信頼性の高い方法であると判断する場合があります。

他のほとんどの状況では、エミュレータを選択的に使用します。エミュレータに大きく依存している場合、エミュレーションベンダーが機能パリティを提供するために更新をリリースするまで、テストに新しい AWS サービス機能を組み込むことが難しくなる可能性があります。エミュレータには、開発システムおよびビルドマシン全体のセットアップと設定のための事前投資と継続的な投資も必要です。さらに、多くのクラウドサービスにはエミュレータがありません。エミュレーションファースト戦略を選択すると、これらのサービスの使用が妨げられるか、実際のサービス動作に対して十分にテストされていないコードや設定が生成される可能性があります。

エミュレーションテストを使用する場合は、可能な限りクラウドテストで補完して、適切なクラウド設定が設定されていることを検証し、エミュレートされた環境でのみシミュレートまたはモックできる サービスとのインタラクションをテストします。

エミュレーションテストは、ユニットテストの迅速なフィードバックを提供することができ、エミュレーションソフトウェアの機能と動作パリティによっては、いくつかの統合テストとend-to-endテストもサポートする場合があります。

# 自然境界によるスコープテスト

サーバーレスアプリケーションがより多くのアーキテクチャコンポーネントにまたがって成長するにつれて、特に単一目的関数やイベント駆動型デカップリングなどのベストプラクティスに従う場合、サブシステムの周囲に自然境界が現れます。これらの境界は、コンポーネント間の契約を検証できる効果的なテストエッジとして機能します。

## アーキテクチャの境界を特定する

アプリケーション設計で自然なシームを探します。

- パブリッシャーをコンシューマーに接続する Amazon EventBridge ルールなどのサービス間
- Lambda 関数の前にある Amazon API Gateway エンドポイントなどの API エッジ
- 複数のサービスの AWS Step Functions オーケストレーションなどのワークフローについて
- ダウンストリーム処理をトリガーする Amazon DynamoDB ストリームなどのストレージレイヤー

## Lambda コードをビジネスロジックから分離する

Lambda コードをコアビジネスロジックから分離することで、テストを簡素化します。Lambda ハンドラーは、 AWS ランタイムとアプリケーションロジックの間のシンアダプターとして機能します。イベントデータを抽出して検証し、Lambda 依存関係のないテスト可能な関数に委任する必要があります。これにより、Lambda オブジェクトをモックしたり、複雑な環境を設定したりすることなく、ビジネスロジックを移植可能にし、推論しやすく、簡単にテストできます。

## 境界を契約として扱う

通過するのではなく、境界でテストします。ダウンストリームシステム全体を必要とせずに、エッジを通過するものを検証します。これらの同じ境界は、本番環境でオブザーバビリティフックとしても機能します。テストするアーキテクチャシームは、Amazon CloudWatch Logs、 AWS X-Ray トレース、EventBridge イベントを使用してモニタリングするために計測できます。

# 非同期ワークフローにテストハーネスを使用する

サーバーレスアプリケーションは、多くの場合非同期パターンに依存します。非同期パターンでは、イベントが処理をトリガーし、メッセージがキューを流れ、ワークフローが複数のサービスにまたがり、すぐに応答することはありません。単に関数を呼び出して戻り値を検査することはできません。結果は、後でデータベース、ログストリーム、または別のサービスに表示されることがあります。

テストハーネスは、アプリケーションと共にデプロイするインフラストラクチャをテストして、この非同期動作を監視および検証します。テストハーネスには通常、次のものが含まれます。

- アプリケーションが生成するのと同じイベントにサブスクライブするイベントリスナー
- テスト結果をキャプチャできるストレージメカニズム (DynamoDB テーブルや Amazon S3 バケットなど)
- 期待される結果が表示されるまで待機するテストコード内のポーリングロジック

テストコードはイベントを開始し、ワークフローが完了するまで待機してから、テストハーネスにクエリを実行して、予想される結果が発生したことを確認します。

ベストプラクティスは以下のとおりです。

- 非同期オペレーションの明確な SLAs を定義する – ワークフローにかかる時間を設定し、それらをテストのポーリングタイムアウトとして使用します。
- テスト分離に一意の識別子を使用する – テスト実行ごとに一意のファイル名、メッセージ IDs、または相関トークンを生成して、テスト間の干渉を防止します。
- アプリケーションと一緒にテストインフラストラクチャをデプロイする – テストハーネスリソースを infrastructure-as-code テンプレートに含めて、アプリケーションの進化に合わせて同期を維持します。
- テスト実行後にテストデータをクリーンアップする – これにより、クラウド環境にテストアーティファクトが蓄積されるのを防ぐことができます。

テストハーネスは、複数のサービスにわたるワークフローを検証する統合テスト、完全なユーザージャーニーを検証するend-to-endテスト、およびサービスが EventBridge、Amazon SNS、Amazon SQS、または Amazon Kinesis を介して通信するイベント駆動型アーキテクチャに最も役立ちます。

# 開発者を分離するためのクラウド環境の整理

クラウドでのテストには、相互に分離された環境が必要です。デベロッパーがチーム開発アカウントなどの単一の を共有する場合は AWS アカウント、デベロッパーまたは機能ブランチごとに個別のアプリケーションスタックを作成することを検討してください。これにより、リソースが分離され、命名衝突が防止され、テスト中のクォータの競合やノイズの多い近隣問題を回避できます。

Parameter Store または同様のツールを使用して、API AWS Systems Manager エンドポイントやキュー名などのスタック固有の設定を管理します。コスト効率を高めるために、Amazon Relational Database Service (Amazon RDS) クラスターなどの高価なリソースを開発者スタック間で共有し、スタックごとに軽量のサーバーレスリソース (Lambda 関数、API Gateway ステージ、DynamoDB テーブルなど) を分離します。

規制された業界では、エンタープライズセキュリティポリシーによって開発者のクラウド環境へのアクセスが制限され、ローカル開発ワークフローの一部としてクラウドテストを実行することが難しくなる可能性があります。このような場合、エミュレーションテストはローカルモックテストと完全なクラウド検証の間のギャップを埋めることができますが、アクセスが許す場合は常にクラウドテストで補完する必要があります。

# フィードバックループを高速化

クラウドでテストを行うときは、いろいろなツールとテクニックを使って開発のフィードバックループを加速させます。たとえば、 AWS SAM Accelerate モードと AWS CDK watch モードを使用すると、コード変更をクラウド環境にプッシュする時間を短縮できます。GitHub のサーバーレステストサンプルリポジトリにあるサンプルでは、これらのテクニックをいくつか取り上げています。

また、ソース管理へのチェックイン後だけでなく、開発中にできるだけ早くローカルマシンからクラウドリソースを作成してテストすることをお勧めします。この方法により、ソリューションを開発する際の調査と実験を迅速に行うことができます。さらに、開発マシンからのデプロイを自動化できれば、クラウド構成内の問題の発見をはやめ、ソース管理の、変更の更新と承認に要する無駄な作業を減らすことができます。

# よくある質問

## 他のサービスを呼び出さずに計算を実行して結果を返す Lambda 関数があります。クラウドでのテストが本当に必要ですか?

はい。 AWS Lambda 関数には、テストの結果を変更する可能性のある設定パラメータがあります。すべての Lambda 関数コードはタイムアウトとメモリ設定に依存しているため、正しく設定されていない場合、関数が失敗する可能性があります。Lambda ポリシーでは、Amazon CloudWatch への標準出力ロギングも可能です。コードが CloudWatch を直接呼び出さない場合でも、ログ記録を有効にするにはアクセス許可が必要であり、そのアクセス許可を正確にモックしたりエミュレートしたりすることはできません。

## クラウドでのテストはユニットテストにどのように役立ちますか? クラウドにあり、他の リソースに接続する場合、統合テストではありませんか?

ユニットテストは、当社では、アーキテクチャコンポーネントで個別に実行されるテストと定義されています。この定義により、必ずしもサービスコールやその他のネットワーク通信の使用が妨げられるわけではありません。

多くのサーバーレスアプリケーションには、クラウドでも個別にテストできるアーキテクチャコンポーネントを備えています。基本的な例は、入力を受け取り、それを解釈して、Amazon Simple Queue Service (Amazon SQS) キューにメッセージを送信する Lambda 関数です。こうした関数のユニットテストは、入力値によってキューに入れられたメッセージに、特定の値が存在するかどうかをテストするといったものになります。整理、行動、アサートパターンを使用して記述されるテストを考えてみましょう。

- 配置 — リソース (メッセージを受信するキュー、およびテスト対象の関数) を割り当てます。
- Act – テスト対象の関数を呼び出します。
- アサート — 関数によって送信されたメッセージを取得し、出力を検証します。

モックテストのアプローチでは、プロセス内のモックオブジェクトでキューをモックし、Lambda 関数コードを含むクラスまたはモジュールのプロセス内インスタンスを作成します。アサートフェーズ中、キューに入れられたメッセージはモックされたオブジェクトから取得されます。

クラウドベースのアプローチでは、テスト用に Amazon SQS キューを作成し、分離された Amazon SQS キューを出力先として使用するように設定された環境変数を使用して Lambda 関数をデプロイします。Lambda 関数の実行後に、テストでメッセージを Amazon SQS キューから取得します。

クラウドベースのテストでは、同じコードを実行し、同じ動作をアサートして、アプリケーションの機能的正確性を検証します。ただし、(IAM) ロール、IAM ポリシー、関数のタイムアウトとメモリ設定など AWS Identity and Access Management 、Lambda 関数の設定を検証できるという追加の利点があります。

# 次のステップとリソース

詳しい説明やその他具体的な例については、こちらのリソースを参照してください。

## 実装例

GitHub の サーバーレステストサンプルリポジトリには、このガイドで説明されているパターンとベストプラクティスに従ったテストの具体例が含まれています。リポジトリには、前のセクションで説明したモック、エミュレーション、クラウドテストプロセスのサンプルコードとガイド付きウォークスルーが含まれています。このリポジトリを使用して、最新のサーバーレステストガイダンスをすぐに取得できます AWS。

## 詳細情報

Serverless Land にアクセスして、 AWS サーバーレステクノロジーに関する最新のブログ、動画、トレーニングにアクセスします。

## リファレンス

- AWS SAM Accelerate によるサーバーレス開発の加速 (AWS ブログ記事)
- CDK Watch の開発速度の向上 (AWS ブログ記事)
- Local とのサービス統合のモック AWS Step Functions (AWS ブログ記事)
- サーバーレスアプリケーションのテストの開始方法 (AWS ブログ記事)
- VS Code IDE での LocalStack 統合によるサーバーレステストの高速化 (AWS ブログ記事)
- を使用して関数をローカルでデバッグ AWS SAMする (AWS ドキュメント)

## ツール

- AWS SAM – サーバーレスアプリケーションのテストとデバッグ
- AWS SAM – 自動テストとの統合
- AWS Lambda – コンソールでの Lambda 関数のテスト
- LocalStack Cloud Emulator – LocalStack を使用してサーバーレスアプリケーションのローカルテストエクスペリエンスを強化する

# 寄稿者

## オーサリング

- Dan Fox、 AWS
- Leslie Raj、 AWS
- Rohan Mehta、 AWS
- Rob Hill、 AWS

## レビューアー

- Brian Krygsman、 AWS

## テクニカルライター

- 「AbouHarb」、 AWS

# ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィードをサブスクライブできます。

| 変更 | 説明 | 日付 |
| --- | --- | --- |
| 更新 | アーキテクチャとコードの境界でのテスト、非同期ワークフローのテストハーネス、開発者環境の分離に関するガイダンスを追加しました。また、エミュレーションテストの推奨事項も更新しました。 | 2026 年 3 月 18 日 |
| 初版発行 | — | 2022 年 12 月 9 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

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

アクティブ/アクティブ移行

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

アクティブ/パッシブ移行

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

集計関数

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

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

匿名化

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

アンチパターン

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

アプリケーション制御

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

アプリケーションポートフォリオ

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

人工知能 (AI)

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

AI オペレーション (AIOps)

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

非対称暗号化

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

原子性、一貫性、分離性、耐久性 (ACID)

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

属性ベースのアクセス制御 (ABAC)

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

信頼できるデータソース

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

アベイラビリティーゾーン (AZ)

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

AWS クラウド導入フレームワーク (AWS CAF)

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

AWS ワークロード認定フレームワーク (AWS WQF)

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

# B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

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

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

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

ブルームフィルター

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

ブルー/グリーンデプロイ

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

ボット

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

ボットネット

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

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発した

り、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

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

ブラウンフィールド戦略

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

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

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

ビジネス継続性計画 (BCP)

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

# C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

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

CCoE

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

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

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

カオスエンジニアリング

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

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

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

クライアント側の暗号化

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

Cloud Center of Excellence (CCoE)

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

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的に、エッジコンピューティング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。AWS 移行戦略との関連性については、「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

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

コールドキャッシュ

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

コールドデータ

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

コンピュータビジョン (CV)

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

設定ドリフト

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

構成管理データベース (CMDB)

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

コンフォーマンスパック

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

継続的インテグレーションと継続的デリバリー (CI/CD)

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

CV

「コンピュータビジョン」を参照してください。

# D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

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

データドリフト

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

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

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

データ最小化

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

データ境界

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

データの前処理

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

データ出所

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

データ件名

データを収集、処理している個人。

データウェアハウス

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

データベース定義言語 (DDL)

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

データベース操作言語 (DML)

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

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

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

深層学習

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

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を採用するときは AWS、リソースの保護に役立つように、 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 、ガバナンス要件のコンプライアンスに影響を与える可能性のあるランディングゾーンの変更を検出したりできます。

DVSM

「開発バリューストリームマッピング」を参照してください。

# E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

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

電子データ交換 (EDI)

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

暗号化

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

暗号化キー

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

エンディアン

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

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

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

エンタープライズリソースプランニング (ERP)

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

エンベロープ暗号化

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

環境

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

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

エピック

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

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

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

# F

ファクトテーブル

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

フェイルファスト

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

障害分離境界

では AWS クラウド、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。

# G

生成 AI

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

ジオブロッキング

「地理的制限」を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リスト

を使って指定します。詳細については、CloudFront ドキュメントの「コンテンツの地理的ディストリビューションの制限」を参照してください。

Gitflow ワークフロー

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

ゴールデンイメージ

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

グリーンフィールド戦略

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

ガードレール

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

# H

HA

「高可用性」を参照してください。

異種混在データベースの移行

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

高可用性 (HA)

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

ヒストリアンのモダナイゼーション

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

ホールドアウトデータ

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

同種データベースの移行

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

ホットデータ

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

ホットフィックス

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

ハイパーケア期間

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

## I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

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

アイドル状態のアプリケーション

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

IIoT

「インダストリアル IoT」を参照してください。

イミュータブルインフラストラクチャ

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

インバウンド (受信) VPC

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

増分移行

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

インダストリー 4.0

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

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

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

インダストリアル IoT (IIoT)

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

インスペクション VPC

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

IoT

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

解釈可能性

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

IoT

「IoT」を参照してください。

IT 情報ライブラリ (ITIL)

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

IT サービス管理 (ITSM)

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

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

# L

ラベルベースアクセス制御 (LBAC)

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

ランディングゾーン

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

大規模言語モデル (LLM)

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

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースアクセス制御」を参照してください。

最小特権

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

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

# M

機械学習 (ML)

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

メインブランチ

「ブランチ」を参照してください。

マルウェア

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

マネージドサービス

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

製造実行システム (MES)

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

MAP

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

メカニズム

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

メンバーアカウント

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

MES

「製造実行システム」を参照してください。

Message Queuing Telemetry Transport (MQTT)

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

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス

機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

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

Migration Acceleration Program (MAP)

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

大規模な移行

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

移行ファクトリー

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

移行メタデータ

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

移行パターン

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

Migration Portfolio Assessment (MPA)

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

移行準備状況評価 (MRA)

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

移行戦略

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

ML

「機械学習」を参照してください。

モダナイゼーション

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

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定されたギャップに対処するためのアクションプランが得られます。詳細については、「AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する」を参照してください。

モノリシックアプリケーション (モノリス)

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

MPA

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

MQTT

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

多クラス分類

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

ミュータブルなインフラストラクチャ

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

# O

OAC

「オリジンアクセス制御」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

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

OI

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」を参照してください。

オンライン移行

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

OPC-UA

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

Open Process Communications - Unified Architecture (OPC-UA)

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

オペレーショナルレベルアグリーメント (OLA)

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

運用準備状況レビュー (ORR)

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

運用テクノロジー (OT)

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

オペレーション統合 (OI)

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

組織の証跡

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

組織変更管理 (OCM)

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

オリジンアクセス制御 (OAC)

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

オリジンアクセスアイデンティティ (OAI)

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

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

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

# P

アクセス許可の境界

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

個人を特定できる情報 (PII)

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

PII

「個人を特定できる情報」を参照してください。

プレイブック

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

PLC

「プログラマブルロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

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

多言語の永続性

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

ポートフォリオ評価

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

述語

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

述語プッシュダウン

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

予防的コントロール

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

プリンシパル

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

プライバシーバイデザイン

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

プライベートホストゾーン

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

プロアクティブコントロール

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

製品ライフサイクル管理 (PLM)

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

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

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

プロンプトチェイニング

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

仮名化

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

発行/サブスクライブ (pub/sub)

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

# Q

クエリプラン

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

クエリプランのリグレッション

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

# R

RACI マトリックス

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

RAG

「検索拡張生成」を参照してください。

ランサムウェア

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

RASCI マトリックス

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

RCAC

「行と列のアクセス制御」を参照してください。

リードレプリカ

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

リアーキテクト

「7 Rs」を参照してください。

目標復旧時点 (RPO)

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

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

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

リグレッション

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

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

リプラットフォーム

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

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

リソースベースのポリシー

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

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

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

レスポンシブコントロール

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

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

検索拡張生成 (RAG)

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

ローテーション

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

行と列のアクセス制御 (RCAC)

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

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

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

# S

SAML 2.0

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

SCADA

「監視制御とデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存する 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

送信先で、それ AWS のサービス を受け取る によるデータの暗号化。

サービスコントロールポリシー (SCP)

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

サービスエンドポイント

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

サービスレベルアグリーメント (SLA)

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

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

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

責任共有モデル

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

SIEM

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

単一障害点 (SPOF)

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

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ」を参照してください。

SPOF

「単一障害点」を参照してください。

スタースキーマ

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

strangler fig パターン

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

サブネット

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

監視制御とデータ取得 (SCADA)

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

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

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

システムプロンプト

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

# T

タグ

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

ターゲット変数

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

タスクリスト

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

テスト環境

「環境」を参照してください。

トレーニング

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

トランジットゲートウェイ

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

トランクベースのワークフロー

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

信頼されたアクセス

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

チューニング

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

ツーピザチーム

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

# U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。詳細については、深層学習システムにおける不確実性の定量化ガイドを参照してください。

未分化なタスク

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

上位環境

「環境」を参照してください。

# V

バキューミング

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

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

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

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

# W

ウォームキャッシュ

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

ウォームデータ

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

ウィンドウ関数

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

ワークロード

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

ワークストリーム

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

WORM

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

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

# Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

ゼロデイ脆弱性

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

ゼロショットプロンプト

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

ゾンビアプリケーション

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

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。