のパフォーマンスエンジニアリングの段階的アプローチ AWS クラウド

AWS 規範ガイダンス

AWS 規範ガイダンス: のパフォーマンスエンジニアリングの段階的アプローチ AWS クラウド

Table of Contents

におけるパフォーマンスエンジニアリングの段階的アプローチ AWS クラウド

Amazon Web Services (寄稿者)

2024 年 4 月 (ドキュメント履歴)

このガイドでは、Amazon Web Services () で実行されているアプリケーションワークロードのパフォーマンスエンジニアリングを計画、構築、有効化するためのベストプラクティスの概要を説明しますAWS。パフォーマンスエンジニアリングには 4 つの柱 があり、アプリケーションのパフォーマンス要件を満たすさまざまなアプローチを提案しています。このガイドでは、各柱について、パフォーマンステストとテスト環境を設定するためのツールとソリューションを一覧表示します。

パフォーマンスエンジニアリングとは

パフォーマンスエンジニアリングには、非機能のパフォーマンス要件 (スループット、レイテンシー、メモリ使用量など) を確実に満たすために、システムの開発ライフサイクル中に適用される手法が含まれます。

パフォーマンステストを開始する前に、パフォーマンス環境を設定する必要があります。一般的なパフォーマンス環境は、次の柱に基づいています。

- テストデータ生成
- オブザーバビリティのテスト
- オートメーションのテスト
- テストレポート

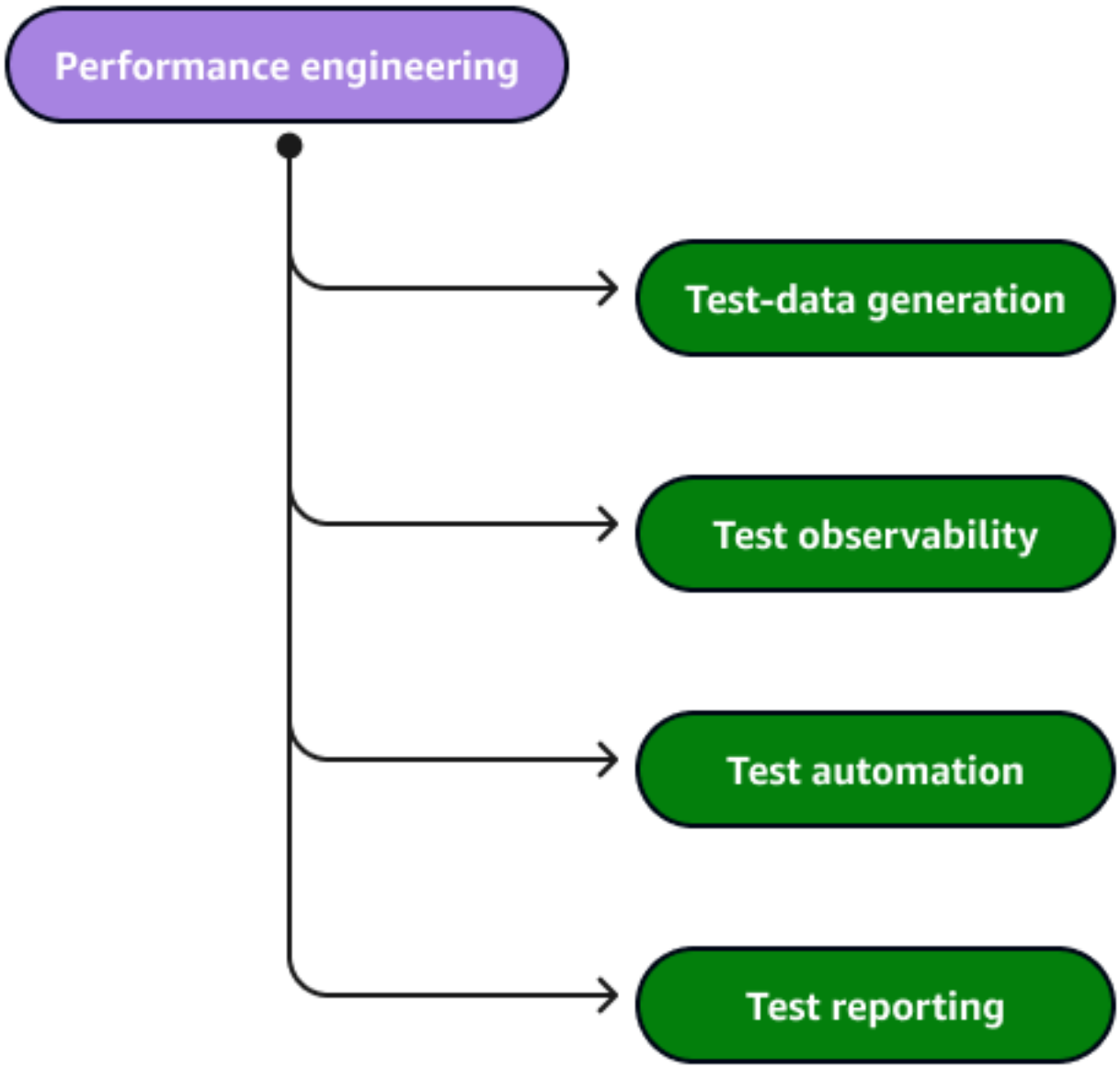
パフォーマンスエンジニアリングを使用する理由

パフォーマンスエンジニアリングは、設計フェーズの開始時点からアプリケーションのパフォーマンスを継続的に最適化するプロセスです。開発サイクルの後の段階でコードのリワークやリファクタリングを回避することで、ビジネスに大きな価値をもたらします。設計段階でパフォーマンスエンジニアリングを開始すると、パフォーマンスを設計に組み込むことができるため、アプリケーションのパフォーマンスが向上します。パフォーマンスエンジニアリングには、システムアーキテクト、デベロッパー DevOps、品質保証の積極的な参加が必要です。

パフォーマンスエンジニアリングの柱

パフォーマンスエンジニアリングの考え方を実現するには、アプリケーションのパフォーマンスエンジニアリングをセットアップしながら、強力な基盤を構築することが重要です。パフォーマンスエンジニアリングには、次の 4 つの主要な柱を設定する必要があります。

- テストデータ生成 – パフォーマンスエンジニアは、テストデータを生成するツールをセットアップします。
- オブザーバビリティのテスト – パフォーマンスエンジニアはオブザーバビリティ環境をセットアップして、パフォーマンス実行をログに記録して追跡できること、および負荷を処理するリソースがモニタリングされていることを確認します。
- テスト自動化 – パフォーマンスエンジニアは、Apache JMeter や ghz などのツールを使用して、ユーザートラフィックとシステム負荷をシミュレートする自動テストを開発します。
- テストレポート – 各テストランの設定とパフォーマンス結果に関するデータが収集されます。このデータにより、設定の変更をパフォーマンスに関連付けることができ、貴重なインサイトが得られます。



これらの柱を組み込むと、設計の初期段階からパフォーマンスの考え方が促進されます。これにより、開発とテストの後のフェーズでアプリケーションまたは環境が変更されるのを防ぐことができます。

テストデータ生成

テストデータの生成には、パフォーマンステストケースを実行するための大量のデータの生成と維持が含まれます。この生成されたデータは、さまざまなデータセットでアプリケーションをテストできるように、テストケースへの入力として機能します。

多くの場合、テストデータの生成は複雑なプロセスです。ただし、データセットの作成が不十分な場合、本番環境でアプリケーションの動作が予測不能になる可能性があります。パフォーマンステストのテストデータ生成は、従来のテストデータ生成アプローチとは異なります。これには実際のシナリオが必要であり、ほとんどのお客様は実際の本番稼働用データに似たデータを使用してワークロードをテストしたいと考えています。生成されたテストデータも、通常、各テスト実行後に元の状態にリセットまたは更新する必要があります。これにより、時間と労力が増大します。

テストデータの生成には、以下の主な考慮事項が含まれます。

- 精度 – データの精度は、テストのあらゆる側面で重要です。データが不正確になると、結果が不正確になります。たとえば、クレジットカード取引が生成された場合、将来の日付にすることはできません。
- 有効 – データはユースケースに対して有効である必要があります。たとえば、クレジットカード取引のテスト中に、1 日あたりユーザーあたり 10,000 件の取引を生成することはお勧めしません。これは、有効なユースケースシナリオから大きく逸脱するためです。
- 自動化 – テストデータ生成の自動化は、時間的労力の利点をもたらす可能性があります。また、効果的なテスト自動化にもつながります。テストデータを手動で生成すると、品質と時間的労力の要件に関して結果が生じる可能性があります。

  ユースケースに基づいて採用できるメカニズムは異なります。

  - API 駆動型 – この場合、開発者はテスターがデータの生成に使用できるテストデータ生成 API を提供します。JMeter などのテストツールを使用して、テスターはビジネス API を使用してデータ生成をスケールできます。たとえば、ユーザーを追加する API がある場合、同じ API を使用して、異なるプロファイルを持つ数百のユーザーを作成できます。同様に、API の削除オペレーションを呼び出してユーザーを削除できます。複雑なワークフローアプリケーションの場合、開発者はさまざまなコンポーネント間でデータセットを生成できる複合 API を提供できます。このアプローチを使用すると、テスターは自動化を記述して、要件に基づいてデータセットを生成および削除できます。

    ただし、システムが複雑な場合、または呼び出しあたりの API 応答時間が長い場合、データの設定と削除に時間がかかることがあります。
  - SQL ステートメント駆動型 – 別の方法は、バックエンド SQL ステートメントを使用して大量のデータを生成することです。開発者は、テストデータ生成用のテンプレートベースの SQL ステートメントを提供できます。テスターは、ステートメントを使用してデータを入力することも、これらのステートメントの上にラッパースクリプトを作成してテストデータの生成を自動化することもできます。このアプローチを使用すると、テストの完了後にデータをリセットする必要がある場合に、テスターはデータを非常に迅速に入力および削除できます。ただし、このアプ

ローチではアプリケーションのデータベースに直接アクセスする必要があります。これは、一般的なセキュリティで保護された環境では不可能な場合があります。さらに、無効なクエリでは、誤ったデータ母集団が発生し、結果に歪みが生じる可能性があります。また、開発者はアプリケーションコードの SQL ステートメントを継続的に更新して、時間の経過とともにアプリケーションに加えられた変更を反映する必要があります。

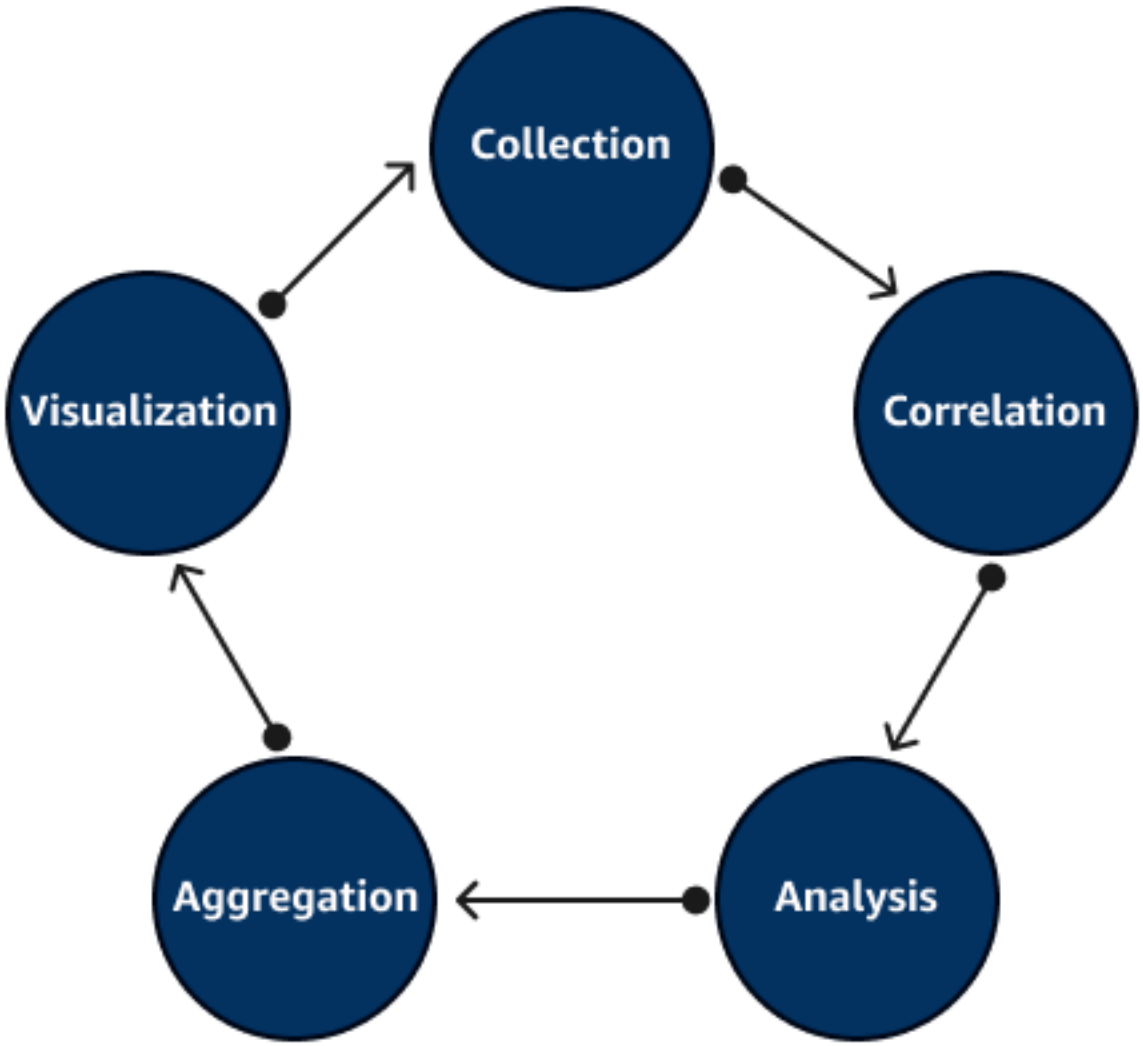
テストデータ生成ツール

AWS には、テストデータ生成に使用できるネイティブカスタムツールが用意されています。

- Amazon Kinesis Data Generator – Amazon Kinesis Data Generator (KDG) は、データを生成して Amazon Kinesis に送信するタスクを簡素化します。このツールは、ブラウザで直接実行される使いやすい UI を提供します。詳細とリファレンス実装については、「新しい Amazon Kinesis Data Generator によるストリーミングデータソリューションのテスト」ブログ記事を参照してください。
- AWS Glue テストデータジェネレーター – AWS Glue テストデータジェネレーターは、 AWS Glue PySpark サーバーレスジョブを使用したテストデータ生成用の設定可能なフレームワークを提供します。必要なテストデータの説明は、YAML 設定ファイルを使用して完全に設定できます。詳細とリファレンスの実装については、AWS Glue 「Test Data Generator GitHub repository」を参照してください。

オブザーバビリティのテスト

テストオブザーバビリティは、パフォーマンステストの実行中にネットワーク、インフラストラクチャ、アプリケーションのテレメトリの収集、関連付け、集約、分析をサポートします。システムの動作、パフォーマンス、状態に関する完全なインサイトが得られます。これらのインサイトは、問題を迅速に検出、調査、修正するのに役立ちます。人工知能と機械学習を追加することで、問題にプロアクティブに対応し、予測し、防止できます。



オブザーバビリティは、 のログ記録、モニタリング、トレースに依存します。これらのアクティビティを正常に実装する責任は、アプリケーションチームとインフラストラクチャチームに及びます。

設計フェーズの開始時に、アプリケーションチームはログ記録、モニタリング、トレースなど、オブザーバビリティスタックの現在の状態を理解する必要があります。その後、オブザーバビリティスタックにツールをよりスムーズに統合できます。

同様に、インフラストラクチャチームはオブザーバビリティインフラストラクチャの管理とスケーリングを担当します。

テストオブザーバビリティについては、次の点を考慮してください。

- アプリケーションログとトレースの可用性
- ログとトレースの相関関係

- ノード、コンテナ、アプリケーションメトリクスの可用性
- オブザーバビリティインフラストラクチャをオンデマンドでセットアップおよび更新する自動化
- テレメトリを視覚化する機能
- オブザーバビリティインフラストラクチャのスケーリング

ログ記録

ログ記録は、システムで発生するイベントに関するデータを保持するプロセスです。ログには、問題、エラー、または現在のオペレーションに関する情報を含めることができます。ログは、次のようなさまざまなタイプに分類できます。

- イベントログ
- サーバーログ
- システムログ
- 認可ログとアクセスログ
- 監査ログ

開発者は、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。ログは、開発者がパフォーマンスの問題の根本原因分析を実行し、システムコンポーネント間で相関するのに役立ちます。

効果的なログ記録ソリューションを構築するには、アプリケーションチームとインフラストラクチャチーム間の緊密な調整が必要です。アプリケーションログは、ログの解析、フィルタリング、バッファリング、相関などのユースケースをサポートするスケーラブルなログ記録インフラストラクチャがない限り、役に立ちません。相関 ID の生成、ビジネスクリティカルなメソッドの実行時間のログ記録、ログパターンの定義などの一般的なユースケースを簡素化できます。

アプリケーションチーム

アプリケーション開発者は、生成されたログがログ記録のベストプラクティスに従っていることを確認する必要があります。ベストプラクティスは次のとおりです。

- 一意のリクエストを追跡するための相関 IDs の生成
- ビジネスクリティカルな方法でかかった時間のログ記録
- 適切なログレベルでのログ記録

- 共通ログ記録ライブラリの共有

さまざまなマイクロサービスとやり取りするアプリケーションを設計する場合は、これらのロギング設計原則を使用してバックエンドでのフィルタリングとログ抽出を簡素化します。

一意のリクエストを追跡するための相関 IDs の生成

アプリケーションはリクエストを受け取ると、相関 ID がヘッダーに既に存在するかどうかを確認できます。ID が存在しない場合、アプリケーションは ID を生成する必要があります。たとえば、Application Load Balancer は というヘッダーを追加しますX-Amzn-Trace-Id。アプリケーションは ヘッダーを使用して、ロードバランサーからのリクエストをアプリケーションに関連付けることができます。同様に、アプリケーションは、リクエストフローの異なるコンポーネントによって生成されたログが相関するように、依存するマイクロサービスを呼び出すtraceId場合は を挿入する必要があります。

ビジネスクリティカルな方法にかかる時間のログ記録

アプリケーションはリクエストを受信すると、別のコンポーネントとやり取りします。アプリケーションは、ビジネスクリティカルなメソッドにかかった時間を定義されたパターンで記録する必要があります。これにより、バックエンドのログを簡単に解析できます。また、ログから有用なインサイトを生成するのにも役立ちます。アスペクト指向プログラミング (AOP) などのアプローチを使用してこのようなログを生成し、ビジネスロジックからログ記録の懸念を分離できます。

適切なログレベルでのログ記録

アプリケーションは、有用な量の情報を含むログを書き込む必要があります。ログレベルを使用して、重要度別にイベントを分類します。例えば、調査が必要な重要なイベントには WARNINGおよびERRORレベルを使用します。詳細なトレースと大量のイベントDEBUGには、 INFOと を使用します。本番稼働に必要なレベルのみをキャプチャするようにログハンドラーを設定します。INFO レベルでのログ作成が多すぎると役に立たず、バックエンドインフラストラクチャにプレッシャーがかかります。 DEBUGログ記録は便利ですが、慎重に使用する必要があります。DEBUG ログを使用すると大量のデータが生成されるため、パフォーマンステスト環境では推奨されません。

共通ログ記録ライブラリの共有

アプリケーションチームは、開発者がプロジェクトの依存関係として使用できる、事前定義された共通のログ記録パターンを持つ AWS SDK for Javaなどの共通のログ記録ライブラリを使用する必要があります。

インフラストラクチャチーム

DevOps エンジニアは、バックエンドでログをフィルタリングおよび抽出するときに、次のログ記録設計原則を使用することで労力を削減できます。インフラストラクチャチームは、次のリソースをセットアップしてサポートする必要があります。

エージェントをログに記録する

ログエージェント (ログシッパー) は、ある場所からログを読み取り、別の場所に送信するプログラムです。ログエージェントは、コンピュータに保存されているログファイルを読み取り、一元化のためにログイベントをバックエンドにアップロードするために使用されます。

ログは非構造化データであり、そこから有意義なインサイトを得る前に構造化する必要があります。ログエージェントは、パーサーを使用してログステートメントを読み取り、タイムスタンプ、ログレベル、サービス名などの関連フィールドを抽出し、そのデータを JSON 形式に構造化します。エッジに軽量ログエージェントを配置すると、リソース使用率が低下するため便利です。ログエージェントはバックエンドに直接プッシュすることも、データをバックエンドにプッシュする中間ログフォワーダーを使用することもできます。ログフォワーダーを使用すると、ソースのログエージェントから作業がオフロードされます。

ログパーサー

ログパーサーは、非構造化ログを構造化ログに変換します。ログエージェントパーサーは、メタデータを追加してログを強化します。データのデータ解析は、ソース (アプリケーションエンド) で行うことも、一元的に行うこともできます。新しいフィールドを追加できるように、ログを保存するためのスキーマは拡張可能である必要があります。JSON などの標準ログ形式を使用することをお勧めします。ただし、場合によっては、検索を改善するためにログを JSON 形式に変換する必要があります。適切なパーサー式を記述することで、効率的な変換が可能になります。

ログバックエンド

ログバックエンドサービスは、さまざまなソースからログデータを収集、取り込み、視覚化します。ログエージェントは、バックエンドに直接書き込むか、中間ログフォワーダーを使用できます。パフォーマンステスト中は、後で検索できるようにログを保存してください。アプリケーションごとにログをバックエンドに個別に保存します。たとえば、アプリケーション専用のインデックスを使用し、インデックスパターンを使用して、さまざまな関連アプリケーションに分散されているログを検索します。ログ検索には、少なくとも 7 日間のデータを保存することをお勧めします。ただし、データを長期間保存すると、不要なストレージコストが発生する可能性があります。パフォーマンステスト中に大量のログが生成されるため、ログ記録インフラストラクチャがログ記録バックエンドをスケーリングして適切なサイズにすることが重要です。

ログの視覚化

アプリケーションログから意味のある実用的なインサイトを得るには、専用の視覚化ツールを使用して raw ログデータを処理してグラフィカル表現に変換します。グラフ、グラフ、ダッシュボードなどの視覚化は、未加工のログを見るときにすぐには明らかにならない傾向、パターン、異常を明らかにするのに役立ちます。

視覚化ツールの主な利点には、複数のシステムやアプリケーション間でデータを関連付けて、依存関係やボトルネックを特定する機能が含まれます。インタラクティブダッシュボードでは、さまざまな詳細レベルでデータをドリルダウンして、問題のトラブルシューティングや使用状況の傾向を把握できます。特殊なデータ可視化プラットフォームは、モニタリングと分析を強化できる分析、アラート、データ共有などの機能を提供します。

アプリケーションログにデータ可視化の機能を使用することで、開発チームと運用チームはシステムおよびアプリケーションのパフォーマンスを可視化できます。得られたインサイトは、効率の最適化、ユーザーエクスペリエンスの向上、セキュリティの強化、キャパシティプランニングなど、さまざまな目的に使用できます。その結果、さまざまなステークホルダーに合わせたダッシュボードが作成され、ログデータを実用的な洞察力のある情報にまとめたビューがat-a-glanceわかります。

ログ記録インフラストラクチャの自動化

アプリケーションによって要件が異なるため、ロギングインフラストラクチャのインストールとオペレーションを自動化することが重要です。Infrastructure as Code (IaC) ツールを使用して、ロギングインフラストラクチャのバックエンドをプロビジョニングします。その後、ロギングインフラストラクチャを共有サービスとして、または特定のアプリケーションの独立したカスタムデプロイとしてプロビジョニングできます。

開発者は、継続的デリバリー (CD) パイプラインを使用して以下を自動化することをお勧めします。

- ログ記録インフラストラクチャをオンデマンドでデプロイし、不要な場合は削除します。
- さまざまなターゲットにログエージェントをデプロイします。
- ログパーサーとフォワーダーの設定をデプロイします。
- アプリケーションダッシュボードをデプロイします。

ログ記録ツール

AWS は、ネイティブロギング、アラーム、ダッシュボードサービスを提供します。以下は、ログ記録によく使用される リソース AWS のサービス と リソースです。

- Amazon OpenSearch Service は、組織がさまざまなソースからログデータを収集、取り込み、視覚化するのに役立ちます。詳細については、OpenSearch による集中ログ記録」を参照してください。
- Amazon CloudWatch エージェントと AWS for Fluent Bit は、 で最も人気のあるログエージェントです AWS。Amazon CloudWatch Logs Insights で CloudWatch エージェントを使用する方法については、ブログ記事「Simplifying Apache server logs with Amazon CloudWatch Logs Insights」を参照してください。 Amazon CloudWatch AWS for Fluent Bit リファレンスの実装については、ブログ記事「Fluent Bit を使用した集中型コンテナログ記録」を参照してください。

モニタリング

モニタリングとは、CPU やメモリなどのさまざまなメトリクスを収集し、Amazon Managed Service for Prometheus などの時系列データベースに保存するプロセスです。モニタリングシステムはプッシュベースでもプルベースでもかまいません。プッシュベースのシステムでは、ソースは定期的に時系列データベースにメトリクスをプッシュします。プルベースのシステムでは、スクレイパーはさまざまなソースからメトリクスをスクレイピングし、時系列データベースに保存します。開発者はメトリクスを分析し、メトリクスをフィルタリングし、経時的にプロットしてパフォーマンスを視覚化できます。モニタリングを正常に実装するには、アプリケーションとインフラストラクチャの 2 つの分野に分けることができます。

アプリケーション開発者にとって、以下のメトリクスが重要です。

- レイテンシー – レスポンスを受信するのにかかる時間
- リクエストスループット – 1 秒あたりに処理されたリクエストの合計数
- リクエストエラー率 – エラーの合計数

ビジネストランザクションに関係する各リソース (アプリケーションコンテナ、データベースなど) のリソース使用率、飽和度、エラー数をキャプチャします。たとえば、CPU 使用率をモニタリングする場合、パフォーマンステストの実行中に平均 CPU 使用率、平均負荷、ピーク負荷を追跡できます。ストレステスト中にリソースが飽和状態に達したが、パフォーマンスの実行中に短時間飽和状態に達しない場合があります。

メトリクス

アプリケーションは、スプリングブートアクチュエータなど、さまざまなアクチュエータを使用してアプリケーションをモニタリングできます。これらの本番稼働用ライブラリは通常、実行中のアプリケーションに関する情報をモニタリングするための REST エンドポイントを公開します。ライブ

ラリは、基盤となるインフラストラクチャ、アプリケーションプラットフォーム、およびその他のリソースをモニタリングできます。デフォルトのメトリクスのいずれかが要件を満たしていない場合、開発者はカスタムメトリクスを実装する必要があります。カスタムメトリクスは、デフォルトの実装のデータでは追跡できないビジネス主要業績評価指標 (KPIs) を追跡するのに役立ちます。たとえば、サードパーティー API 統合のレイテンシーや完了したトランザクションの合計数などのビジネスオペレーションを追跡できます。

カーディナリティ

カーディナリティとは、メトリクスの一意の時系列の数を指します。メトリクスには、追加情報を提供するためにラベルが付けられます。たとえば、特定の API のリクエスト数を追跡する REST ベースのアプリケーションは、カーディナリティが 1 であることを示します。ユーザーラベルを追加してユーザーあたりのリクエスト数を識別すると、カーディナリティはユーザー数に比例して増加します。カーディナリティを作成するラベルを追加することで、さまざまなグループ別にメトリクスをスライスおよびダイスできます。カーディナリティはバックエンドのモニタリング時系列データベースのメトリクス系列の数を増やすため、適切なユースケースに適したラベルを使用することが重要です。

解決策

一般的なモニタリング設定では、モニタリングアプリケーションはアプリケーションからメトリクスを定期的にスクレイプするように設定されています。スクレイピングの周期性は、モニタリングデータの粒度を定義します。より短い間隔で収集されたメトリクスは、より多くのデータポイントが利用可能であるため、パフォーマンスをより正確に把握できる傾向があります。ただし、より多くのエントリが保存されると、時系列データベースの負荷が増加します。通常、粒度は 60 秒が標準解像度で、1 秒が高解像度です。

DevOps チーム

アプリケーション開発者は、多くの場合、DevOps エンジニアにインフラストラクチャとアプリケーションのメトリクスを視覚化するためのモニタリング環境を設定するように依頼します。DevOps エンジニアは、スケーラブルで、アプリケーション開発者が使用するデータ視覚化ツールをサポートする環境を設定する必要があります。これには、さまざまなソースからモニタリングデータをスクレイピングし、そのデータを Amazon Managed Service for Prometheus などの一元的な時系列データベースに送信することが含まれます。

バックエンドのモニタリング

モニタリングバックエンドサービスは、メトリクスデータの収集、保存、クエリ、視覚化をサポートします。通常、Amazon Managed Service for Prometheus や InfluxData InfluxDB などの時系列デー

タベースです。サービス検出メカニズムを使用すると、モニタリングコレクターはさまざまなソースからメトリクスを収集して保存できます。パフォーマンステスト中は、後で検索できるようにメトリクスデータを保存することが重要です。メトリクスには少なくとも 15 日間のデータを保存することをお勧めします。ただし、メトリクスを長期間保存しても大きなメリットは得られず、不要なストレージコストが発生します。パフォーマンステストでは大量のメトリクスを生成できるため、メトリクスインフラストラクチャが高速なクエリパフォーマンスを提供しながらスケールすることが重要です。モニタリングバックエンドサービスは、メトリクスデータの表示に使用できるクエリ言語を提供します。

視覚的表現

アプリケーションデータを表示して有意義なインサイトを提供できる視覚化ツールを提供します。DevOps エンジニアとアプリケーション開発者は、モニタリングバックエンドのクエリ言語を学習し、緊密に連携して再利用できるダッシュボードテンプレートを生成する必要があります。ダッシュボードには、レイテンシーとエラーを含め、インフラストラクチャとアプリケーションリソース全体のリソース使用率と飽和度も表示します。

モニタリングインフラストラクチャの自動化

ログ記録と同様に、モニタリングインフラストラクチャのインストールと運用を自動化して、さまざまなアプリケーションのさまざまな要件に対応できるようにすることが重要です。IaC ツールを使用して、モニタリングインフラストラクチャのバックエンドをプロビジョニングします。その後、モニタリングインフラストラクチャを共有サービスとして、または特定のアプリケーションの独立したカスタムデプロイとしてプロビジョニングできます。

CD パイプラインを使用して以下を自動化します。

- モニタリングインフラストラクチャをオンデマンドでデプロイし、不要な場合は削除します。
- モニタリング設定を更新して、メトリクスをフィルタリングまたは集計します。
- アプリケーションダッシュボードをデプロイします。

モニタリングツール

Amazon Managed Service for Prometheus は、コンテナインフラストラクチャ用の Prometheus 互換のモニタリングサービスであり、コンテナのアプリケーションメトリクスを使用してコンテナ環境を大規模に安全にモニタリングできます。詳細については、ブログ記事「Amazon Managed Service for Prometheus の開始方法」を参照してください。

Amazon CloudWatch はフルスタックのモニタリングを提供します AWS。CloudWatch はネイティブソリューションとオープンソースソリューションの両方 AWS をサポートしているため、テクノロジースタック全体で何が起きているかをいつでも把握できます。

ネイティブ AWS ツールには以下が含まれます。

- Amazon CloudWatch ダッシュボード
- CloudWatch Container Insights
- CloudWatch メトリクス
- CloudWatch アラーム

Amazon CloudWatch は、CloudWatch Container Insights によるコンテナモニタリングなど、特定のユースケースに対応する専用機能を提供します。これらの機能は CloudWatch に組み込まれているため、ログ、メトリクスの収集、モニタリングを設定できます。

コンテナ化されたアプリケーションとマイクロサービスの場合、Container Insights を使用してメトリクスとログを収集、集計、要約します。Container Insights は、Amazon Elastic Compute Cloud (Amazon EC2) 上の Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、および Kubernetes プラットフォームで使用できます。Container Insights は、埋め込みメトリクス形式でパフォーマンスログイベントとしてデータを収集します。これらのパフォーマンスログイベントエントリは、大規模な高カーディナリティデータの取り込みとストレージをサポートする構造化された JSON スキーマを使用します。

Amazon EKS で Container Insights を実装する方法については、ブログ記事「Amazon CloudWatch Container Insights for Amazon EKS Fargate using AWS Distro for OpenTelemetry」を参照してください。

トレース

トレースには、プログラムのプロセスに関するログ情報の特殊な使用が含まれます。ログからのインサイトは、エンジニアが個々のトランザクションをデバッグし、ボトルネックを特定するのに役立ちます。トレースは、自動または手動計測を使用して有効にできます。

アプリケーションはさまざまなサービスと統合されるため、アプリケーションとその基盤となるサービスのパフォーマンスを特定することが重要です。トレースはトレースとスパンで機能します。トレースは完全なリクエストプロセスであり、各トレースはスパンで構成されます。スパンはタグ付けされた時間間隔であり、システムの個々のコンポーネントまたはサービス内のアクティビティです。トレースは、アプリケーションにリクエストが行われたときに何が起こるかの全体像を提供します。

アプリケーションチーム

アプリケーション開発者は、アプリケーション内のインバウンドリクエストとアウトバウンドリクエスト、およびその他のイベントのトレースデータを、各リクエストに関するメタデータとともに送信することで、アプリケーションを計測します。トレースを生成するには、アプリケーションを計測してトレースを生成する必要があります。計測は自動でも手動でもかまいません。

自動計測

ソースコードを変更することなく、自動計測を使用してアプリケーションからテレメトリを収集できます。自動計測エージェントは、アプリケーションまたはサービスのアプリケーショントレースを生成できます。通常、設定変更を使用してエージェントまたは別のメカニズムを追加します。

ライブラリ計測では、アプリケーションコードの変更を最小限に抑えて、構築済みの計測を追加します。計測は、 AWS SDK、Apache HTTP クライアント、SQL クライアントなどの特定のライブラリまたはフレームワークを対象としています。

手動実装

このアプローチでは、アプリケーション開発者はトレース情報を収集する各場所でアプリケーションに計測コードを追加します。たとえば、アスペクト指向プログラミング (AOP) を使用してトレースデータを収集します AWS X-Ray 。開発者は SDKs を使用してアプリケーションを計測できます。

サンプリング

トレースデータは多くの場合、大量の で生成されます。トレースデータをエクスポートするかどうかを決定するメカニズムを持つことが重要です。サンプリングは、エクスポートするデータを決定するプロセスです。これは通常、コストを節約するために行われます。サンプリングルールをカスタマイズすることで、記録するデータの量を制御できます。コードを変更して再デプロイすることなく、サンプリング動作を変更することもできます。適切な量のトレースを生成するには、サンプリングレートを制御することが重要です。

アプリケーションデベロッパーは、メタデータをキーと値のペアとして追加することで、トレースに注釈を付けることができます。注釈はトレースを強化し、バックエンドのフィルタリングを絞り込むのに役立ちます。

DevOps チーム

DevOps エンジニアは、多くの場合、アプリケーション開発者がインフラストラクチャとアプリケーションのトレースを視覚化するためのトレース環境を設定するように求められます。トレース環境の

セットアップでは、さまざまなソースからトレースデータを収集し、視覚化のために中央ストアに送信します。

トレースバックエンド

トレースバックエンドは、アプリケーションが処理するリクエストに関するデータを収集 AWS X-Ray する などのサービスです。これは、そのデータを表示、フィルタリング、インサイトを取得して、最適化の問題と機会を特定するために使用できるツールを提供します。アプリケーションへのトレースされたリクエストについては、リクエストとレスポンス、およびアプリケーションがダウンストリーム AWS リソース、マイクロサービス、データベース、ウェブ APIs に対して行うその他の呼び出しに関する詳細情報を確認できます。

トレースの自動化

アプリケーションごとにトレース要件が異なるため、トレースインフラストラクチャの設定と運用を自動化することが重要です。IaC ツールを使用して、トレースインフラストラクチャのバックエンドをプロビジョニングします。

CD パイプラインを使用して以下を自動化します。

- トレースインフラストラクチャをオンデマンドでデプロイし、不要な場合は削除します。
- トレース設定をアプリケーション全体にデプロイします。

トレースツール

AWS は、トレースとそれに関連する視覚化のために次のサービスを提供します。

- AWS X-Ray は、アプリケーションが既に X-Ray と統合されているサービスからのトレースに加えて、 AWS アプリケーションからトレースを受け取ります。X-Ray トレースシング用アプリケーションを計測するために使用できる SDK、エージェント、およびツールがいくつかあります。詳細については、AWS X-Ray のドキュメントを参照してください。

  開発者は、 AWS X-Ray SDKs を使用して X-Ray にトレースを送信することもできます。 AWS X-Ray は、Go、、Node.jsJava、、Python.NET、および 用の SDKs を提供しますRuby。各 X-Ray SDK は、以下を提供します。

  - インターセプター コードに追加して受信 HTTP リクエストをトレースする
  - アプリケーションが他の AWS サービスを呼び出すために使用する AWS SDK クライアントを計測するクライアントハンドラー

- HTTP クライアント 他の内部および外部 HTTP ウェブサービス呼び出しを計測する

  X-Ray SDKsSQL データベースの計測呼び出し、自動 AWS SDK クライアント計測、およびその他の機能もサポートしています。トレースデータを直接 X-Ray に送信する代わりに、SDK は JSON セグメントドキュメントを UDP トラフィックをリッスンしているデーモンプロセスに送信します。X-Ray デーモンはセグメントをキューにバッファし、バッチで X-Ray にアップロードします。X-Ray SDK を使用してアプリケーションを計測する方法の詳細については、X-Ray ドキュメントを参照してください。

- Amazon OpenSearch Service は、OpenSearch クラスターを実行およびスケーリングするための AWS マネージドサービスであり、ログ、メトリクス、トレースを一元的に保存するために使用できます。可観測性プラグインは、一般的なデータソースからメトリクス、ログ、トレースを収集およびモニタリングするための統合エクスペリエンスを提供します。データ収集とモニタリングを 1 か所で行うことで、インフラストラクチャ全体のフルスタックのend-to-endのオブザーバビリティを実現できます。実装情報については、OpenSearch Service ドキュメントを参照してください。
- AWS Distro for OpenTelemetry (ADOT) は、Cloud Native Computing Foundation (CNCF) OpenTelemetry プロジェクトに基づく AWS ディストリビューションです。ADOT には現在、Java と Python の自動計測サポートが含まれています。さらに、ADOT は、ADOT Managed Lambda Layers を介したJava、、Node.js、Pythonランタイムを使用した AWS Lambda 関数とそのダウンストリームリクエストの自動計測をサポートしています。開発者は ADOT コレクターを使用して、 AWS X-Ray や Amazon OpenSearch Service など、さまざまなバックエンドにトレースを送信できます。

  ADOT SDK を使用してアプリケーションを計測する方法のリファレンス例については、 ドキュメントを参照してください。ADOT SDK を使用して Amazon OpenSearch Service にデータを送信する方法のリファレンス例については、OpenSearch Service ドキュメントを参照してください。

  Amazon EKS で実行されているアプリケーションを計測する方法のリファレンス例については、ブログ記事「Amazon EKS AWS アドオンを使用したメトリクスとトレースのコレクション」を参照してください OpenTelemetry。

自動化をテストする

特殊なフレームワークとツールを使用した自動テストにより、人間の介入を減らし、品質を最大化できます。自動パフォーマンステストは、ユニットテストや統合テストなどの自動化テストとは異なります。

DevOps パイプラインをパフォーマンステストのさまざまな段階で使用します。

テスト自動化パイプラインの 5 つのステージは次のとおりです。

1. セットアップ — このステージの「Test-data generation」セクションで説明されているテストデータアプローチを使用します。有効なテスト結果を取得するには、現実的なテストデータを生成することが重要です。さまざまなユースケースをカバーし、ライブ本番稼働データと密接に一致する多様なテストデータを慎重に作成する必要があります。フルスケールのパフォーマンステストを実行する前に、初期トライアルテストを実行してテストスクリプト、環境、モニタリングツールを検証する必要がある場合があります。
2. テストツール – パフォーマンステストを実行するには、JMeter や ghz などの適切な負荷テストツールを選択します。実際のユーザーの負荷をシミュレートするという点で、ビジネスニーズに最適なものを検討してください。
3. テスト実行 – テストツールと環境が確立されたら、予想されるユーザーの負荷と期間の範囲にわたってend-to-endのパフォーマンステストを実行します。テスト中は、テスト対象のシステムの状態を注意深くモニタリングします。これは通常、長時間実行されるステージです。自動テスト無効化のエラー率をモニタリングし、エラーが多すぎる場合はテストを停止します。

   負荷テストツールは、リソースの使用率、応答時間、潜在的なボトルネックに関するインサイトを提供します。
4. テストレポート – アプリケーションとテスト設定とともにテスト結果を収集します。アプリケーション設定、テスト設定、結果の収集を自動化し、パフォーマンステスト関連のデータの記録と一元的な保存に役立ちます。パフォーマンスデータを一元管理することで、優れたインサイトを提供し、ビジネスの成功基準をプログラムで定義できます。
5. クリーンアップ – パフォーマンステストの実行が完了したら、テスト環境とデータをリセットして後続の実行に備えます。まず、実行中にテストデータに加えられた変更をすべて元に戻します。データベースやその他のデータストアを元の状態に復元し、テスト中に生成された新規、更新、または削除されたレコードを元に戻す必要があります。

パイプラインを再利用して、必要なパフォーマンスが結果に反映されるまでテストを複数回繰り返すことができます。パイプラインを使用して、コードの変更がパフォーマンスを低下させないことを検証することもできます。コード検証テストを営業時間外に実行し、トラブルシューティングに使用できるテストデータとオブザーバビリティデータを使用できます。

ベストプラクティスは次のとおりです。

- 開始時刻と終了時刻を記録し、ログ記録用の URLsを自動的に生成します。これにより、該当する時間枠でオブザーバビリティデータをフィルタリングし、システムをモニタリングおよびトレースできます。
- テストの呼び出し中に、 ヘッダーにテスト識別子を挿入します。アプリケーションデベロッパーは、識別子をバックエンドのフィルターとして使用することで、データのログ記録、モニタリング、トレースを強化できます。
- パイプラインは一度に 1 つの実行のみに制限します。同時テストを実行するとノイズが発生し、トラブルシューティング中に混乱が生じる可能性があります。また、専用のパフォーマンス環境でテストを実行することも重要です。

テスト自動化ツール

テストツールは、テストの自動化において重要な役割を果たします。オープンソーステストツールの一般的な選択肢は次のとおりです。

- Apache JMeter は、長年の経験で培われたパワーホースです。長年にわたって、Apache JMeter は信頼性を向上させ、機能を追加してきました。グラフィカルインターフェイスを使用することで、プログラミング言語を知らなくても複雑なテストを作成できます。BlazeMeter などの企業が Apache JMeter をサポートしています。
- K6 は、サポート、負荷ソースのホスト、および負荷テストを整理、実行、分析するための統合ウェブインターフェイスを提供する無料ツールです。
- Vegeta 負荷テストは別のコンセプトを採用しています。同時実行を定義したり、システムに負荷をかけたりする代わりに、特定のレートを定義します。また、このツールはシステムの応答時間とは無関係に負荷を作成します。
- Apache HTTP サーバーベンチマーキングツールである Hey と ab は、単一のエンドポイントで指定された負荷を実行するためにコマンドラインから使用できる基本的なツールです。ツールを実行するサーバーがある場合、これは負荷を発生させる最も速い方法です。ローカルのラップトップでも動作しますが、高い負荷が発生するほど強力ではない場合があります。
- ghz は、gRPC サービスの負荷テストとベンチマーキングのためのコマンドラインユーティリティおよび Go パッケージです。

AWS は、 での分散負荷テスト AWS ソリューションを提供します。このソリューションは、サーバーをプロビジョニングすることなく、一定のペースでトランザクションレコードを生成する何千人もの接続ユーザーを作成およびシミュレートします。詳細については、AWS 「 ソリューションライブラリ」を参照してください。

を使用して AWS CodePipeline 、パフォーマンステストパイプラインを自動化できます。CodePipeline を使用して API テストを自動化する方法の詳細については、AWS DevOps ブログと AWS ドキュメントを参照してください。

テストレポート

テストレポートは、システム、アプリケーション、サービス、またはプロセスのパフォーマンスに関連するデータの収集、分析、および表示を指します。これには、さまざまなメトリクスと指標を測定して、特定のシステムまたはコンポーネントの効率、応答性、信頼性、および全体的な有効性を評価する必要があります。

パフォーマンステストレポートでは、分析のコンテキストと目標に基づいて関連するメトリクスを選択します。一般的なパフォーマンスメトリクスには、応答時間、スループット、エラー率、リソース使用率 (CPU、メモリ、ディスク) 、ネットワークレイテンシーなどがあります。

パフォーマンス関連のデータを収集したら、中央リポジトリに保存する必要があります。これらのテスト結果は、さまざまな環境、アプリケーション、テストツールから得られる可能性があります。異なる環境で複数のワークロードを実行している場合、パフォーマンス関連のデータを収集し、これらのデータポイントを相関させて情報に基づいた結論を引き出すことは困難です。データストレージと視覚化のための中央リポジトリを使用してパフォーマンスメトリクスデータを収集するための標準的な方法を定義することをお勧めします。

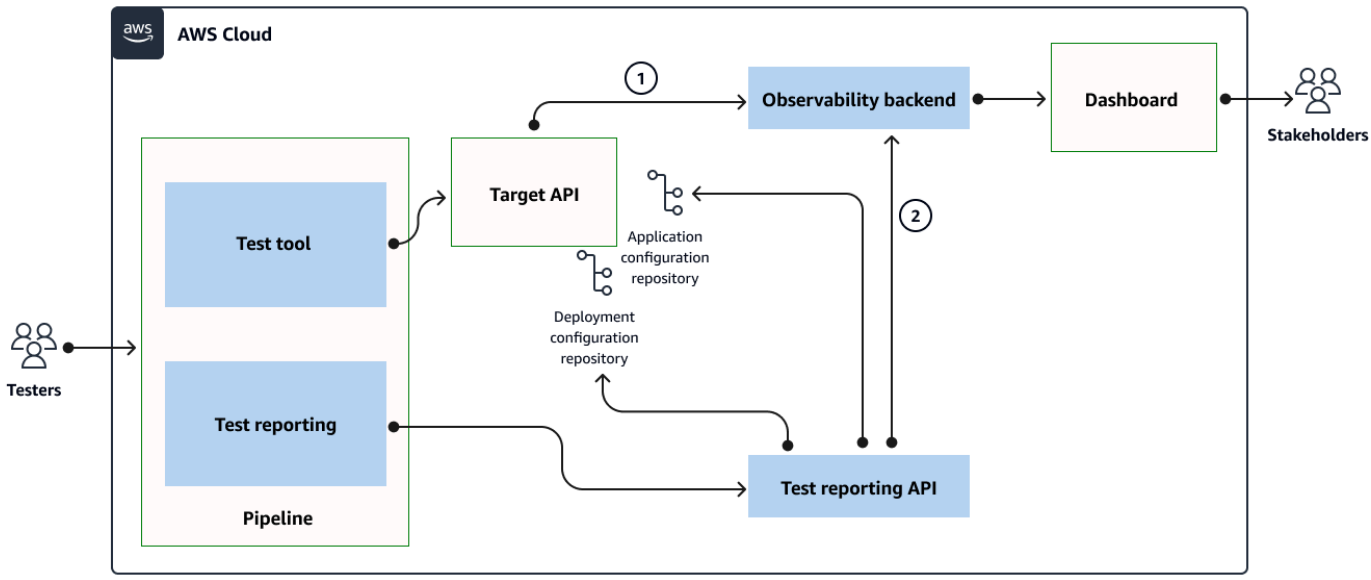
標準化された記録

さまざまな利害関係者がパフォーマンステストを実行し、結果のデータを中央リポジトリに書き込む方法を標準化することをお勧めします。例えば、これは API が結果を受け入れ、永続的ストレージソリューションに保存する形式になります。GitOps や Amazon Managed Service for Prometheus などのソースからデータを取得する必要がある場合、API はデプロイ仕様や Kubernetes 仕様からフィールドを抽出する方法を説明するスキーマファイルに基づいて、指定されたソースからこれらの詳細を直接取得できます。スキーマファイルは、`JSONPath`式または Prometheus クエリ言語 (PromQL) を使用できます。前述のように、収集されるメトリクスは、パフォーマンス分析のコンテキストと目標に関連している必要があります。

API に渡されるデータには、テストが実行されたアプリケーションと環境に関連する詳細とタグを含めることができます。これにより、パフォーマンステストデータに対して分析を実行できます。

実際のパフォーマンスエンジニアリングの柱

次のリファレンスアーキテクチャは、特定の API をテストするためのパフォーマンスエンジニアリングの柱を示しています。



1. ログ記録、モニタリング、トレースデータは、ターゲット API からバックエンドに送信されます。
2. 呼び出されると、テストレポート API は結果と設定情報をバックエンドに送信します。

コアコンポーネントは、テスト対象のターゲット API またはアプリケーションです。ターゲット API は、GitOps でアプリケーション設定リポジトリとデプロイ設定リポジトリと同期して、最新のアプリケーションおよびインフラストラクチャ設定を取得します。この同期により、Git リポジトリで定義されているように、アプリケーションとそのサポートインフラストラクチャの現在の望ましい状態に対して自動テストを実行できます。

テスト自動化パイプラインは、テストデータの生成、テストの実行、ターゲット API のテスト結果の報告を自動化します。

ターゲット API は、オブザーバビリティのベストプラクティスを使用してパフォーマンスインサイト (メトリクス、ログ、トレース) を生成し、オブザーバビリティのバックエンドにメトリクスデータをストリーミングします。

テストレポート API は、すべてのテスト関連のレポートデータ (設定とテスト結果) を収集し、オブザーバビリティバックエンドに保存します。

パフォーマンスインサイトとレポートデータ (設定、テスト結果) を集約すると、ターゲット API のパフォーマンス関連データをクエリするのに役立ちます。例えば、次のような質問をします。

- 最も遅いトランザクションの上位 10 件は何ですか?
- PP99, P90、各テストの平均数を教えてください。
- 2 つのテストランの設定はどのように比較されますか?

テストケースを一定期間の結果、設定、メトリクスと関連付けると、最適な設定とパフォーマンスの結果を特定するのに役立ちます。

これらのテスト結果を使用すると、API に対してより正確でデータ駆動型の意思決定を行い、API を本番環境に持ち込む際に自信を持つことができます。

リソース

AWS サービス

- Amazon CloudWatch
- AWS CodePipeline
- AWS Distro for OpenTelemetry
- Amazon OpenSearch Service
- AWS X-Ray

実装

- amazon-kinesis-data-generator
- AWS Glue テストデータジェネレーター
- での分散負荷テスト AWS

ブログ記事

- Fluent Bit を使用した集中型コンテナログ記録
- 新しい Amazon Kinesis Data Generator を使用してストリーミングデータソリューションをテストする
- AWS Distro for OpenTelemetry を使用した Amazon EKS Fargate 用 Amazon CloudWatch Container Insights の紹介
- を使用した Kubernetes でのアプリケーショントレース AWS X-Ray
- AWS Distro for OpenTelemetry 用の Amazon EKS アドオンを使用したメトリクスとトレースのコレクション
- Amazon Managed Service for Prometheus の開始方法

ワークショップ

- AWS オブザーバビリティの概要

- 負荷テストアプリケーション (ガイド)

サードパーティーアプリケーション

- Apache JMeter
- K6
- Vegeta
- Hey と ab
- ghz

寄稿者

本ドキュメントの寄稿者は次のとおりです。

- Varun Sharma、シニアリードコンサルタント、 AWS
- Akash Kumar、シニアリードコンサルタント、 AWS
- Archana Bhatnagar、プラクティスマネージャー、 AWS
- Pratik Sharma、プロフェッショナルサービス II、 AWS

ドキュメント履歴

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

変更	説明	日付
初版発行	—	2024 年 4 月 24 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。