耐障害性ライフサイクルフレームワーク

AWS 規範ガイダンス

AWS 規範ガイダンス: 耐障害性ライフサイクルフレームワーク

Table of Contents

耐障害性ライフサイクルフレームワーク: 耐障害性向上のための継続的なアプローチ

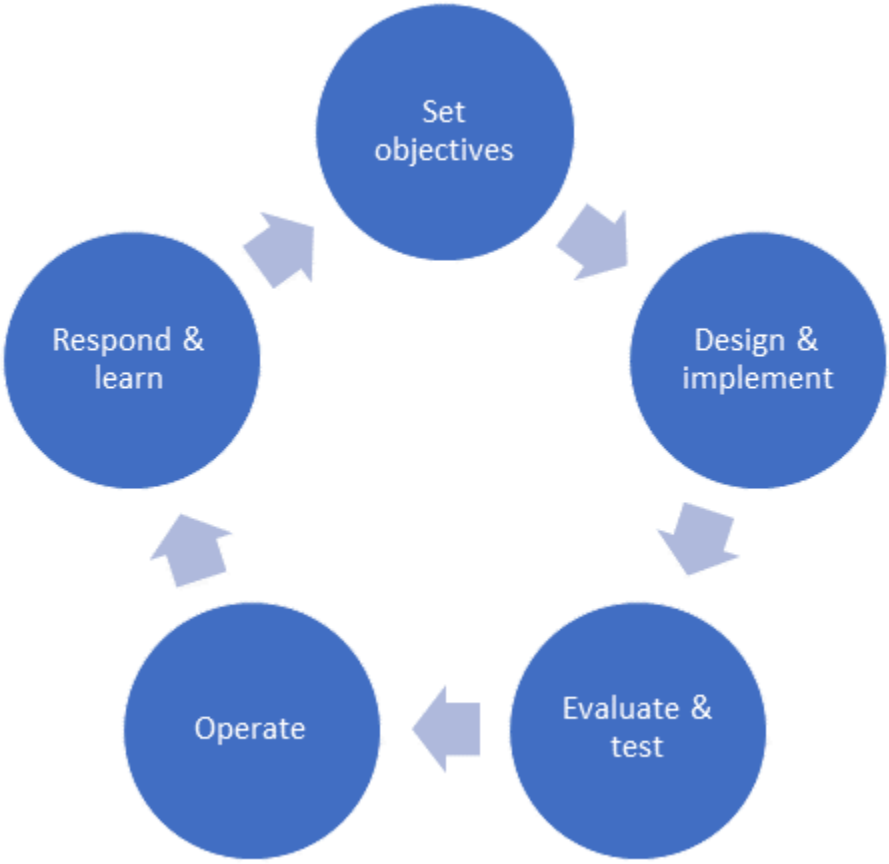
Amazon Web Services (寄稿者)

2023 年 10 月 (ドキュメント履歴)

現代の組織では、特に顧客からの期待が常にオンで常に利用可能という考え方にシフトするにつれて、耐障害性関連の課題が増え続けています。リモートチームと複雑な分散アプリケーションに連動して、頻繁なリリースの必要性が高まっています。そのため、組織とそのアプリケーションはこれまで以上に耐障害性を強化する必要があります。

AWS は、インフラストラクチャ、依存サービス、設定ミス、一時的なネットワーク問題に関連する障害など、中断に抵抗または回復するアプリケーションの能力として回復力を定義します。(AWS「 Well-Architected フレームワークの信頼性の柱」ドキュメントの「回復力」と「信頼性のコンポーネント」を参照してください。) ただし、望ましいレベルの耐障害性を実現するには、多くの場合、トレードオフが必要です。運用の複雑さ、エンジニアリングの複雑さ、コストは、それに応じて評価および調整する必要があります。

AWS は、顧客や社内チームとの長年の協力に基づいて、レジリエンスに関する学習とベストプラクティスをキャプチャするレジリエンスライフサイクルフレームワークを開発しました。このフレームワークは、次の図に示す 5 つの主要なステージの概要を示しています。各ステージで、戦略、サービス、メカニズムを使用して耐障害性体制を改善できます。



以下のステージについて、このガイドの次のセクションで説明します。

- ステージ 1: 目標を設定する
- ステージ 2: 設計と実装
- ステージ 3: 評価とテスト
- ステージ 4: 運用
- ステージ 5: 対応と学習

用語と定義

各ステージの耐障害性の概念は、個々のコンポーネントからシステム全体まで、さまざまなレベルで適用されます。これらの概念を実装するには、以下のいくつかの用語を明確に定義する必要があります。

- コンポーネント: 機能を実行するための要素。ソフトウェアとテクノロジーのリソースで構成されます。コンポーネントの例としては、コード設定、ネットワークなどのインフラストラクチャ、サーバー、データストア、または多要素認証 (MFA) デバイスなどの外部依存関係があります。
- アプリケーション: 顧客向けのウェブストアフロントや機械学習モデルを改善するバックエンドプロセスなど、ビジネス価値を提供するコンポーネントのコレクション。アプリケーションは、単一の AWS アカウント内のコンポーネントのサブセットで構成されている場合もあれば、複数の AWS アカウント およびリージョンにまたがる複数のコンポーネントのコレクションである場合もあります。
- システム: 特定のビジネス機能を管理するために必要なアプリケーション、人材、プロセスのコレクション。システムには、機能の実行に必要なアプリケーション、継続的インテグレーションと継続的デリバリー (CI/CD)、オブザーバビリティ、設定管理、インシデント対応、ディザスタリカバリなどの運用プロセス、およびそのようなタスクを管理するオペレーターが含まれます。
- 中断: アプリケーションがビジネス機能を適切に提供できないようにするイベント。
- 障害: 中断が軽減されない場合にアプリケーションに与える影響。一連の中断が発生した場合、アプリケーションに障害が発生する可能性があります。

継続的な耐障害性

耐障害性ライフサイクルは継続的なプロセスです。同じ組織内であっても、各アプリケーションチームはアプリケーションの要件により、各ステージ内で異なるレベルの完全性で実行できます。ただし、各ステージが完全であればあるほど、アプリケーションの耐障害性も高くなります。

耐障害性ライフサイクルは、組織が運用できる標準プロセスと考える必要があります。 AWS は、アプリケーションを開発および運用しながら、運用プロセス全体に計画、テスト、学習を組み込むことを目的に、耐障害性ライフサイクルをソフトウェア開発ライフサイクル (SDLC) と類似するように意図的にモデル化しています。多くのアジャイル開発プロセス と同様に、耐障害性ライフサイクルは開発プロセスを繰り返すたびに繰り返すことができます。 ライフサイクルの各ステージ内のプラクティスは、時間の経過とともに徐々に深化させていくことをお勧めします。

ステージ 1: 目標を設定する

必要なレベルの耐障害性とその測定方法を理解することが、「目標を設定する」ステージの基礎となります。目標がなく、また目標が測定できないと、何かを改善するのは困難です。

すべてのアプリケーションが同じレベルの耐障害性を必要とするわけではありません。目標を設定するときは、適切な投資とトレードオフを行うために必要なレベルを考慮してください。これは車とよく似ています。タイヤは 4 本ありますが、予備のタイヤは 1 本しか持ちません。乗車中に複数のタイヤがパンクする可能性は低く、余分なスペアがあることで、貨物スペースや燃料効率などの他の機能を損ねる可能性があるため、これは妥当なトレードオフです。

目標を定義したら、後のステージ (ステージ 2: 設計と実装、ステージ 4: 運用) でオブザーバビリティコントロールを実装し、目標が達成されているかどうかを把握します。

重要なアプリケーションのマッピング

耐障害性の目標を定義する際、技術的な会話に限定してはなりません。代わりに、ビジネス指向に焦点をあてた会話から始め、アプリケーションで提供する必要のある内容と障害による影響について理解します。次に、このビジネス目標に対する理解が、アーキテクチャ、エンジニアリング、運用などの分野にカスケードされます。定義する耐障害性目標がすべてのアプリケーションに適用される場合もありますが、目標の測定方法は多くの場合アプリケーションの機能によって異なります。ビジネスにとって重要なアプリケーションを実行している場合に、このアプリケーションに障害が発生すると、組織が重大な収益を損失したり、評判が損なわれたりする可能性があります。または、それほど重要ではなく、組織のビジネス遂行機能に悪影響を及ぼすことなく一部のダウンタイムを許容できる別のアプリケーションがあるかもしれません。

例えば、小売会社の注文管理アプリケーションについて考えてみましょう。注文管理アプリケーションのコンポーネントに障害が発生し、適切に稼働しなくなると、新規販売を遂行できません。この小売会社には、自社の建物の一角で営業を行っている従業員向けのコーヒーショップもあります。このコーヒーショップには、従業員が静的ウェブページでアクセスできるオンラインメニューがあります。このウェブページが利用できなくなった場合、一部の従業員は苦情を言うかもしれませんが、必ずしも会社に財務上の損害をもたらすとは限りません。この例に基づいて、会社では注文管理アプリケーションに対してより積極的な耐障害性目標を持つことを選択する可能性がありますが、ウェブアプリケーションの耐障害性を確保するために多大な投資を行うことはありません。

最も重要なアプリケーション、最も労力をかける場所、トレードオフを行う場所を特定することは、本番環境でのアプリケーションの耐障害性を測定することと同じくらい重要です。障害による影響を

より適切に理解するために、ビジネスインパクト分析 (BIA) を実行できます。BIA は、重要なビジネスアプリケーションを特定して優先付けし、潜在的なリスクと影響を評価してサポートする依存関係を特定するための、構造化された体系的なアプローチを提供します。BIA は、組織の最も重要なアプリケーションに対するダウンタイムのコストを定量化するのに役立ちます。このメトリクスは、特定のアプリケーションに障害が発生し、その機能を完了できない場合のコストを概算するのに役立ちます。前の例では、注文管理アプリケーションに障害が発生した場合、小売ビジネスは大幅な収益損失となる可能性があります。

ユーザーストーリーのマッピング

BIA プロセス中に、アプリケーションが複数のビジネス機能に関与していること、またはビジネス機能に複数のアプリケーションが必要であることに気付くことがあります。前の小売会社の例を使用すると、注文管理機能のチェックアウト、プロモーション、料金設定用に個別のアプリケーションが必要になる可能性があります。1 つのアプリケーションで障害が発生すると、ビジネスや会社とやり取りするユーザーに影響が出る可能性があります。例えば、会社が新規注文の追加、プロモーションや割引へのアクセス提供、製品の価格更新を実行できない場合があります。注文管理機能に必要となるこうしたさまざまな機能は、複数のアプリケーションに依存している可能性があります。また、これらの機能には複数の外部依存関係が存在することもあり、この場合純粋にコンポーネントに焦点を当てた耐障害性を達成するプロセスが極端に複雑になります。このシナリオに対処するより良い方法は、ユーザーストーリーに焦点を当てることです。ユーザーストーリーは 1 つのアプリケーションまたはアプリケーションのセットを操作するときにユーザーが期待する体験について概説しています。

ユーザーストーリーに焦点を当てることで、カスタマーエクスペリエンスのどの部分が最も重要なのかを把握できるため、特定の脅威から保護するためのメカニズムを構築できるようになります。前の例では、1 つのユーザーストーリーがチェックアウトであり、これにはチェックアウトアプリケーションが含まれ、価格アプリケーションとの依存関係があります。別のユーザーストーリーはプロモーションの表示に関するもので、プロモーションアプリケーションが関与しています。最も重要なアプリケーションとそのユーザーストーリーをマッピングしたら、これらのユーザーストーリーの耐障害性を測定するために使用するメトリクスの定義を開始できます。これらのメトリクスは、ポートフォリオ全体または個々のユーザーストーリーに適用できます。

測定値の定義

目標復旧時点 (RPO)、目標復旧時間 (RTO)、サービスレベル目標 (SLO) は、特定のシステムの耐障害性を評価するために使用される標準的な業界測定値です。RPO とは、障害が発生した場合にビジネスで許容できるデータ損失量を指します。一方、RTO は、停止後にアプリケーションを再度利用

できるまでの速度の測定値を表します。これら 2 つのメトリクスは、秒、分、時間の時間単位で測定されます。アプリケーションが正常に動作している時間、つまり、設計どおりに機能し、ユーザーがアクセスできる時間を測定することもできます。これらの SLO では、顧客が享受することを期待するサービスレベルについて詳述し、1 秒未満の応答時間内にエラーなしで処理されたリクエストの割合 (%) などのメトリクスによって測定されます (例えば、リクエストの 99.99% が毎月応答を受け取ります)。 RPO と RTO はディザスタリカバリ戦略に関連しており、バックアップの復元からユーザートラフィックのリダイレクトまで、アプリケーションの運用とリカバリのプロセスが中断されることを前提としています。SLO は、アプリケーションのダウンタイムを軽減する傾向がある高可用性コントロールを実装することで対処されます。

SLO メトリクスは一般的に、サービスプロバイダーとエンドユーザー間の契約であるサービスレベルアグリーメント (SLA) の定義で使用されます。SLA には通常、財務上のコミットメントが伴い、これらの契約が満たされない場合にプロバイダーが支払う必要があるペナルティの概要が記載されています。ただし、SLA は耐障害性体制の測定ではなく、SLA を増やしてもアプリケーションの耐障害性は向上しません。

SLO、RPO、RTO に基づいて目標の設定を開始できます。耐障害性目標を定義し、RPO と RTO のターゲットを明確に理解したら、AWS Resilience Hub を使用してアーキテクチャの評価を実行し、潜在的な耐障害性関連の弱点を発見できます。AWS Resilience Hub は AWS Well-Architected フレームワークのベストプラクティスに照らしてアプリケーションアーキテクチャを評価し、定義済みの RTO と RPO の目標値を満たすために特に改善する必要がある点に関する修復ガイダンスを共有します。

追加の測定値の作成

RPO、RTO、SLO は耐障害性の良い指標となりますが、ビジネスの観点から目標を検討し、アプリケーションの機能に関する目標を定義することもできます。例えば、フロントエンドとバックエンド間のレイテンシーが 40% 増加した場合、1 分あたりの成功注文数は 98% を上回った状態が維持されます。または、1 秒あたりに開始されたストリームは、特定のコンポーネントが失われた場合でも、平均からの標準偏差内にとどまります。また、既知の障害タイプ全体で平均復旧時間 (MTTR) を短縮する目標を作成することもできます (例: これらの既知の問題のいずれかが発生した場合、復旧時間が x% 短縮される)。ビジネスニーズに沿った目標を作成すると、アプリケーションで許容する障害のタイプを予測できるようになります。また、アプリケーションに障害が発生する可能性を減らすアプローチを特定することもできます。

アプリケーションを強化するインスタンスの 5% が失われた場合に運用を継続するという目標を検討している場合、アプリケーションは事前にスケールするか、そのイベント中に発生する追加のトラフィックをサポートするのに十分な速さでスケールすることができると判断することがあります。ま

たは、「ステージ 2: 設計と実装」セクションで説明されているように、さまざまなアーキテクチャパターンを活用する必要があると判断することもできます。

また、特定のビジネス目標に対してオブザーバビリティ対策を実装する必要があります。例えば、平均注文率、平均注文価格、平均サブスクリプション数、またはアプリケーションの動作に基づいてビジネスの正常性に関するインサイトを提供できるその他のメトリクスを追跡できます。アプリケーションにオブザーバビリティ機能を実装することで、アラームを作成し、これらのメトリクスが定義された境界を超えた場合に対応策を取ることができます。オブザーバビリティの詳細については、「ステージ 4: 運用」セクションを参照してください。

ステージ 2: 設計と実装

前のステージでは、耐障害性目標を設定しました。この「設計と実装」ステージでは、前のステージで設定した目標に従って障害モードを予測し、設計における選択肢を特定します。また、変更管理の戦略を定義し、ソフトウェアコードの作成とインフラストラクチャ設定を行います。以下のセクションでは、コスト、複雑さ、運用オーバーヘッドなどのトレードオフを考慮する際に検討する必要のある AWS のベストプラクティスについて説明します。

AWS Well-Architected フレームワーク

望ましい耐障害性目標に基づいてアプリケーションを設計する場合は、複数の要因を評価し、最適なアーキテクチャでトレードオフする必要があります。耐障害性に優れたアプリケーションを構築するには、設計、構築とデプロイ、セキュリティ、運用の各側面を考慮する必要があります。AWS Well-Architected フレームワークでは、AWS で耐障害性に優れたアプリケーションを設計するのに役立つ一連のベストプラクティス、設計原則、アーキテクチャパターンを提供します。AWS Well-Architected フレームワークの 6 つの柱により、耐障害性、安全性、効率性、コスト効率、および持続可能性を備えたシステムを設計、運用するためのベストプラクティスを提供します。フレームワークは、アーキテクチャをベストプラクティスに照らし合わせて一貫的に測定し、改善すべき点を特定する手段を提供します。

AWS Well-Architected フレームワークが耐障害性目標を達成するアプリケーションを設計および実装するのに役立つ方法の例を次に示します。

- 信頼性の柱: 信頼性の柱は、障害や中断が発生した場合でも、正しく一貫して動作できるアプリケーションを構築することの重要性を強調しています。例えば、AWS Well-Architected フレームワークでは、マイクロサービスアーキテクチャを使用してアプリケーションを小さくシンプルにすることをお勧めしています。これにより、アプリケーション内のさまざまなコンポーネントの可用性に対するニーズを区別できます。また、スロットリング、エクスポネンシャルバックオフによる再試行、フェイルファスト (負荷分散)、べき等性、定常作業、サーキットブレーカー、静的安定性を使用して、アプリケーションを構築するためのベストプラクティスに関する詳細な説明を検索することもできます。
- 包括的なレビュー: AWS Well-Architected フレームワークでは、ベストプラクティスと設計原則に照らしてアーキテクチャの包括的なレビューを行うことが推奨されています。このフレームワークでは、アーキテクチャを一貫して測定し、改善すべき分野を特定する方法を提供します。

- リスク管理: AWS Well-Architected フレームワークは、アプリケーションの信頼性に影響を与える可能性のあるリスクの特定と管理に役立ちます。潜在的な障害シナリオに積極的に対処することで、障害の発生可能性や結果として生じる障害を減らすことができます。
- 継続的な改善: 耐障害性は継続的なプロセスであり、AWS Well-Architected フレームワークは継続的な改善を強調しています。AWS Well-Architected フレームワークのガイダンスに基づいてアーキテクチャとプロセスを定期的に見直し、改良することで、進化する課題や要件に直面してもシステムの耐障害性を維持できます。

依存関係について

システムの依存関係を理解することは、耐障害性にとって重要です。依存関係には、アプリケーション内のコンポーネント間の接続、およびサードパーティー API やビジネス所有の共有サービスなど、アプリケーション外のコンポーネントへの接続が含まれます。1 つのコンポーネントの障害が他のコンポーネントに影響を与える可能性があるため、これらの接続を理解すると、中断を分離して管理できるようになります。この知識は、エンジニアが障害の影響を評価し、それに応じて計画を立て、リソースを効果的に使用するのに役立ちます。依存関係を理解することで、代替戦略を作成し、復旧プロセスを調整できるようになります。また、ハード依存関係をソフト依存関係に置き換えることができるケースも特定できるため、依存関係に障害が発生した場合でもアプリケーションでビジネス機能の提供を継続することができます。依存関係は、負荷分散とアプリケーションのスケールの決定にも影響します。アプリケーションに変更を加えるときは、潜在的なリスクと影響を判断するのに役立つ依存関係について理解することが不可欠です。この知識は、安定した回復力のあるアプリケーションを作成し、障害管理、影響評価、障害復旧、負荷分散、スケール、変更管理を支援します。依存関係を手動で追跡することも、AWS X-Ray などのツールやサービスを使用して分散アプリケーションの依存関係を把握することもできます。

ディザスタリカバリ戦略

ディザスタリカバリ (DR) 戦略は、主にビジネス継続性を確保することで、回復力のあるアプリケーションの構築と運用に重要な役割を果たします。これにより、壊滅的な事態の渦中であっても、重要な事業運営が最小限の障害で維持され、ダウンタイムと潜在的な収益損失を最小限に抑えることができます。DR 戦略はデータ保護に不可欠です。この戦略には複数の場所に及ぶ定期的なデータバックアップやデータレプリケーションが組み込まれていることが多いため、貴重なビジネス情報を保護し、災害発生時の全損状態を回避できます。さらに、多くの業界が、企業に機密データを保護し、災害発生時もサービスを確実に利用できるような DR 戦略を策定するよう求めるポリシーによって規制されています。サービスの障害を最小限に抑えることで、DR 戦略による顧客の信頼と満足度も向上します。適切に実装され、頻繁に実践される DR 戦略により、災害後の復旧時間を短縮し、アプリ

ケーションを迅速にオンラインに戻すことができます。さらに、災害時は、ダウンタイムによる収益の損失だけでなく、アプリケーションとデータの復元に要する費用により、相当額のコストが発生する可能性があります。適切に設計された DR 戦略であれば、こうした財務上の損失から保護することができます。

選択する戦略は、アプリケーションの特定のニーズ、RTO と RPO、予算によって異なります。AWS Elastic Disaster Recovery は、オンプレミスアプリケーションとクラウドベースのアプリケーションの両方に DR 戦略を実装するために使用できる専用の耐障害性サービスです。

詳細については、AWS ウェブサイトの「Disaster Recovery of Workloads on AWS」および「AWS マルチリージョンの基礎」を参照してください。

CI/CD 戦略の定義

アプリケーション障害の一般的な原因の 1 つは、以前の既知の動作状態からアプリケーションを変更するコードやその他の変更です。変更管理に慎重に対処しないと、頻繁に障害が発生する可能性があります。変更の頻度が増えるにつれ、影響を及ぼす可能性も高くなります。ただし、変更の頻度を抑えると変更設定が多くなり、より複雑になることで障害が発生する可能性がはるかに高くなります。継続的インテグレーションと継続的デリバリー (CI/CD) のプラクティスは、変更を小規模かつ頻繁に保ち (生産性の向上につながります)、各変更を自動化することで高レベルの検査を受けるように設計されています。基本的な戦略には、次のようなものがあります。

- フルオートメーション: CI/CD の基本的な概念は、できるだけ多くのビルドおよびデプロイプロセスを自動化することです。これには、構築、テスト、デプロイ、さらにはモニタリングが含まれます。自動パイプラインは、人為的ミスの可能性を減らし、一貫性を確保し、プロセスの信頼性と効率を高めるのに役立ちます。
- テスト駆動型開発 (TDD): アプリケーションコードを記述する前にテストを記述します。この手法により、すべてのコードにテストが確実に関連付けられ、コードの信頼性と自動検査の品質が向上します。これらのテストは CI パイプラインで実行され、変更が検証されます。
- 頻繁なコミットと統合: 開発者がコードを頻繁にコミットし、頻繁に統合を実行することを推奨します。小規模で頻繁な変更はテストやデバッグが容易であり、重大な問題のリスクが軽減されます。自動化により各コミットやデプロイでのコストが削減され、頻繁な統合が可能になります。
- イミュータブルインフラストラクチャ: サーバーやその他のインフラストラクチャコンポーネントを静的でイミュータブルなエンティティのように扱います。できるだけ多く変更するのではなく、インフラストラクチャを置き換え、パイプラインを通じてテストされデプロイされたコードを使用して新しいインフラストラクチャを構築します。

- ロールバックメカニズム: 何か問題が発生した場合に変更をロールバックするには、常に簡単で信頼性が高く、頻繁にテストされた方法を使用します。以前の既知の正常な状態にすばやく戻ることは、デプロイの安全性にとって不可欠です。これは、前の状態に戻すシンプルなボタンでも、アラームによって完全に自動化および開始することもできます。
- バージョン管理: すべてのアプリケーションコード、設定、インフラストラクチャをバージョン管理リポジトリのコードとして維持します。この方法により、変更を簡単に追跡し、必要に応じて元に戻すことができます。
- Canary デプロイとブルー/グリーンデプロイ: アプリケーションの新しいバージョンを最初にインフラストラクチャのサブセットにデプロイするか、2 つの環境 (ブルー/グリーン) を維持することで、本番環境での変更の動作を検証し、必要に応じてすばやくロールバックできます。

CI/CD とは、ツールだけでなく文化にも関連します。自動化、テスト、障害からの学習を重視する文化を構築することは、適切なツールやプロセスを実装するのと同じくらい重要です。ロールバックが影響を最小限に抑えながら非常に迅速に行われた場合、失敗ではなく学習経験と考えるべきです。

ORR の実施

運用準備状況レビュー (ORR) は、運用と手続きのギャップを特定するのに役立ちます。Amazon では、何十年にもわたる大規模なサービスを運用してきた教訓を、ベストプラクティスガイダンスによる厳選された質問に抽出するための ORR を作成しました。ORR では、以前に学んだ教訓を取り込み、新しいチームがアプリケーションでこれらの教訓を考慮できるようにします。ORR では、以下の耐障害性モデリングセクションで説明されている耐障害性モデリングアクティビティで実行できる障害モードまたは障害の原因のリストを提供することができます。詳細については、AWS Well-Architected フレームワークウェブサイトの「Operational Readiness Reviews (ORR)」を参照してください。

AWS 障害分離境界について

AWS では、耐障害性目標の達成に役立つ複数の障害分離境界を提供しています。これらの境界を使用して、指定された影響抑制の予測可能な範囲を活用できます。アプリケーションに選択した依存関係について意図的に選択できるように、これらの境界を使用して AWS サービスがどのように設計されるかを理解しておく必要があります。アプリケーションで境界を使用する方法については、AWS ウェブサイトの「AWS 障害分離境界」を参照してください。

応答方法の選択

システムは、アラームにさまざまな方法で応答できます。アラームの中には、運用チームからの応答を必要とするものもあれば、アプリケーション内で自己修復メカニズムをトリガーするものもあります。自動化のコストを制御したり、エンジニアリングの制約を管理したりするために、自動化できる応答を手動操作として維持する場合があります。アラームに対する応答のタイプは、応答の実装コスト、アラームの予想される頻度、アラームの精度、およびアラームにまったく応答しないことによる潜在的なビジネス損失の 1 つの機能として選択される可能性があります。

例えば、サーバープロセスがクラッシュすると、プロセスはオペレーティングシステムによって再起動されるか、新しいサーバーがプロビジョニングされて古いサーバーが終了するか、オペレーターがサーバーにリモート接続して再起動するよう指示される場合があります。これらの応答は同じ結果 (アプリケーションサーバープロセスを再開する) になりますが、実装コストとメンテナンスコストのレベルは異なります。

> **Note**
>
> 複数の応答を選択して、詳細な耐障害性アプローチを取ることができます。例えば、前のシナリオでは、アプリケーションチームが 3 つの応答すべてを実装し、それぞれの間に時間遅延を設けることを選択できます。失敗したサーバープロセスインジケータが 30 秒後もアラーム状態のままである場合、チームはオペレーティングシステムがアプリケーションサーバーの再起動に失敗したと見なすことができます。したがって、Auto Scaling グループを作成して新しい仮想サーバーを作成し、アプリケーションサーバープロセスを復元できます。インジケータが 300 秒経過してもアラーム状態のままである場合、元のサーバーに接続してプロセスの復元を試みるために、アラートが運用スタッフに送信されることがあります。

アプリケーションチームとビジネスが選択する応答には、運用上のオーバーヘッドをエンジニアリング時間への先行投資で相殺するというビジネスの需要を反映する必要があります。応答 (静的安定性などのアーキテクチャパターン、サーキットブレーカーなどのソフトウェアパターン、または運用手順) を選択する際は、各応答オプションの制約と予想されるメンテナンスを慎重に検討する必要があります。アプリケーションチームをガイドするための標準的な応答がいくつか存在する場合があるため、一元化されたアーキテクチャ機能によって管理されるライブラリとパターンを、この考慮事項への入力として使用できます。

耐障害性モデリング

耐障害性モデリングでは、アプリケーションが予想されるさまざまな中断に対応する方法を文書化します。 中断を予測することで、チームはオブザーバビリティ、自動コントロール、復旧プロセスを実装し、中断しても障害を軽減または防止できます。AWS では、耐障害性分析フレームワークを使用して耐障害性モデルを開発するためのガイダンスを作成しました。 このフレームワークは、中断とそれによるアプリケーションへの影響を予測するのに役立ちます。 中断を予測することで、回復力のある信頼性の高いアプリケーションを構築するために必要な軽減策を特定できます。耐障害性分析フレームワークを使用して、アプリケーションのライフサイクルを反復するたびに耐障害性モデルを更新することをお勧めします。 このフレームワークを反復する度に使用すると、設計フェーズ中に中断を予測し、本番環境のデプロイ前後にアプリケーションをテストしてインシデントを減らすことができます。このフレームワークを使用して耐障害性モデルを開発することで、耐障害性目標を達成できます。

安全に失敗する

中断を回避できない場合は、安全に失敗しましょう。重大なビジネス損失が発生しない、デフォルトのフェイルセーフなオペレーションモードでアプリケーションを作成することを検討してください。データベースのフェイルセーフ状態の例としては、デフォルトで読み取り専用操作があり、ユーザーはデータを作成または変更できません。データの機密性によっては、アプリケーションをデフォルトでシャットダウン状態にし、読み取り専用クエリを実行しないようにすることもできます。アプリケーションに必要なフェイルセーフ状態を考慮してください。極端な条件下では、デフォルトでこのオペレーションモードになります。

ステージ 3: 評価とテスト

ライフサイクルの「評価とテスト」ステージとは、アプリケーションまたは既存のアプリケーションへの変更が既に設計済みであるものの、本番環境にはまだリリースされていない段階を指します。このステージでは、前のステージで実践されたプラクティスをテストし、結果を評価するためのアクティビティを実装します。アプリケーションがまだ開発段階にあるか、一次開発が完了し、本番環境にリリースされる前のテスト段階にある可能性があります。このステージでは、アプリケーションが定義された耐障害性目標を達成するという予想を裏付けるまたは否定するテストの開発と実行に焦点を当てます。さらに、システムの運用手順も作成し、テストします。「ステージ 2: 設計と実装」ステージで作成したデプロイ手順が実践され、結果が評価されます。これらのテストおよび評価アクティビティはライフサイクルのこの部分で開始されますが、ここでは終了しません。「ステージ 4: 運用」ステージに移行し、テストと評価が続行されます。

「評価とテスト」ステージは、デプロイ前アクティビティとデプロイ後アクティビティの 2 つのフェーズに分かれています。デプロイ前アクティビティは、アプリケーションを任意の環境にデプロイする前に完了する必要があるタスクで構成されます。これには、新しいソフトウェアバージョンのデプロイや、テスト環境への初期デプロイが含まれます。デプロイ後アクティビティは、ソフトウェアがテスト環境または本番環境にデプロイされた後に行われます。以下のセクションでは、これらの段階について詳しく説明します。

デプロイ前アクティビティ

環境設計

アプリケーションをテストおよび評価する環境は、テストをどの程度徹底できるか、およびそれらの結果が本番環境で起こる内容を正確に反映しているとどの程度信頼できるかに影響します。Amazon DynamoDB などのサービスを使用して、開発者のマシンで一部の統合テストをローカルで実行できる場合があります (DynamoDB ドキュメントの「DynamoDB local のセットアップ」を参照)。ただし、結果の信頼性を最大限に高めるには、ある時点において本番環境をレプリケートする環境でテストする必要があります。この環境にはコストがかかるため、パイプラインの後半に本番環境に似た環境が表示される環境に対して、ステージングまたはパイプライン化されたアプローチを取ることをお勧めします。

インテグレーションテスト

統合テストは、アプリケーションの明確に定義されたコンポーネントが、外部依存関係により動作するときに正しく機能することをテストするプロセスです。これらの外部依存関係には、他のカスタム

開発コンポーネント、アプリケーションに使用する AWS サービス、サードパーティーの依存関係、オンプレミスの依存関係などがあります。 このガイドでは、アプリケーションの耐障害性を示す統合テストに焦点を当てています。ソフトウェアの機能精度を示すユニットテストと統合テストが既に存在することを前提としています。

サーキットブレーカーパターンや負荷分散など、実装した耐障害性パターンを具体的にテストする統合テストを設計することをお勧めします (「ステージ 2: 設計と実装」を参照)。耐障害性指向の統合テストでは、多くの場合、アプリケーションに特定の負荷を適用したり、AWS Fault Injection Service (AWS FIS) などの機能を使用して意図的に環境に中断を導入したりします。理想的には、CI/CD パイプラインの一部としてすべての統合テストを実行し、コードがコミットされるたびにテストを実行する必要があります。これにより、耐障害性目標の違反につながるコードや設定の変更をすばやく検出して対応できます。大規模な分散アプリケーションは複雑であり、わずかな変更でも、アプリケーションの一見無関係に見える部分の耐障害性に大きな影響を与える可能性があります。すべてのコミットでテストを実行してみてください。 には、CI/CD パイプラインやその他の DevOps ツールを運用するための優れたツールセット AWS が用意されています。詳細については、 AWS ウェブサイトの「 での DevOps の概要 AWS」を参照してください。

自動デプロイパイプライン

本番稼働前環境へのデプロイとテストは、反復的かつ複雑なタスクであり、自動化に最適です。このプロセスを自動化することで、人的リソースが解放され、エラーが発生する機会が軽減されます。このプロセスを自動化するメカニズムは、多くの場合パイプラインと呼ばれます。パイプラインを作成するときは、本番環境の設定にさらに近い一連のテスト環境を設定することをお勧めします。この一連の環境を使用して、アプリケーションを繰り返しテストします。最初の環境は本番環境よりも機能が限られていますが、コストが大幅に削減されます。後続の環境では、サービスを追加し、本番環境をより正確に反映するようにスケールする必要があります。

まず、最初の環境でテストします。デプロイが最初のテスト環境ですべてのテストに合格したら、一定期間一定量の負荷でアプリケーションを実行し、時間の経過とともに問題が発生するかどうかを確認します。発生した問題を検出できるように、オブザーバビリティが正しく設定されていることを確認します (このガイドの後半の「アラームの精度」を参照)。この観察期間が正常に完了したら、アプリケーションを次のテスト環境にデプロイし、プロセスを繰り返して、環境でサポートされているテストまたは負荷を追加します。この方法でアプリケーションを十分にテストしたら、以前に設定したデプロイ方法を使用してアプリケーションを本番環境にデプロイできます (このガイドの前半の「CI/CD 戦略の定義」を参照)。Amazon Builders' Library の記事「Automating safe, hands-off deployments」は、Amazon がコードデプロイを自動化する方法について説明された、優れたリソースです。本番環境へのデプロイより前の環境の数は、アプリケーションの複雑さと依存関係のタイプによって異なります。

負荷テスト

負荷テストは、表面的には統合テストに似ています。アプリケーションの個別の機能とその外部依存関係をテストして、期待どおりに動作するか検証します。その後、負荷テストは明確に定義された負荷でアプリケーションがどのように機能するかに焦点を当てるため、統合テストの範囲を超えてテストを実行します。負荷テストでは正しい機能を検証する必要があるため、統合テストが正常に完了してから実行する必要があります。アプリケーションが予想される負荷でどの程度うまく応答するか、および負荷が想定範囲を超えた場合にどのように動作するかを理解することが重要です。これにより、アプリケーションに極端に大きな負荷がかかった場合でも耐障害性を維持するために必要なメカニズムが実装されていることを確認できます。での負荷テストの包括的なガイドについては AWS、AWS ソリューションライブラリの「 での分散負荷テスト AWS」を参照してください。

デプロイ後のアクティビティ

耐障害性は継続的なプロセスであり、アプリケーションのデプロイ後もアプリケーションの耐障害性を継続して評価する必要があります。継続的な耐障害性評価など、デプロイ後アクティビティの結果では、耐障害性ライフサイクルの前半で実行した耐障害性アクティビティの一部を再評価して更新する必要が生じる場合があります。

耐障害性評価の実施

アプリケーションを本番環境にデプロイしたからといって、耐障害性の評価は終わりではありません。デプロイパイプラインが明確に定義され、自動化されていても、本番環境で変更が直接発生することがあります。さらに、デプロイ前の耐障害性検証でまだ考慮していない要素があるかもしれません。AWS Resilience Hub では、デプロイされたアーキテクチャが定義された RPO と RTO のニーズを満たすかどうかを評価できる一元的な場所を提供します。このサービスを使用すると、 AWS ブログ記事「 AWS Resilience Hub と を使用してアプリケーションの耐障害性を継続的に評価する」で説明されているように、アプリケーションの耐障害 AWS CodePipeline性をオンデマンドで評価したり、評価を自動化したり、CI/CD ツールに統合したりできます。これらの評価を自動化することは、本番環境で耐障害性体制を継続的に評価するのに役立つため、ベストプラクティスです。

DR テスト

「ステージ 2: 設計と実装」では、ディザスタリカバリ (DR) 戦略をシステムの一部として開発しました。ステージ 4 では、DR 手順をテストして、チームがインシデントに対して十分に準備でき、手順が期待どおりに機能することを確認する必要があります。フェイルオーバーやフェイルバックを含むすべての DR 手順を定期的にテストし、各演習の結果を確認して、可能な限り最良の結果を得るた

めにシステムの手順を更新するかどうかと、その方法について決定する必要があります。DR テストを最初に作成するときは、事前にテストを適切にスケジュールし、チーム全体が期待される内容、結果の測定方法、結果に基づいて手順を更新するために使用するフィードバックメカニズムを理解していることを確認します。スケジュールされた DR テストの実行に慣れたら、未発表の DR テストの実行を検討してください。実際の災害はスケジュールどおりに発生しないため、いつでも計画を実践する準備を整えておく必要があります。ただし、未発表とは計画外を意味するものではありません。主要な利害関係者は、引き続きイベントを計画し、適切なモニタリングが行われ、顧客や重要なアプリケーションに悪影響が及ばないようにする必要があります。

ドリフト検出

本番稼働用アプリケーションの設定に対する予期しない変更は、自動化や明確に定義された手順が設定されている場合でも発生する可能性があります。アプリケーションの設定に対する変更を検出するには、ドリフトを検出するメカニズムが必要です。これは、ベースライン設定からの偏差を指します。 AWS CloudFormation スタックのドリフトを検出する方法については、 AWS CloudFormation ドキュメントの「スタックとリソースに対するアンマネージド型設定変更の検出」を参照してください。アプリケーションの AWS 環境でドリフトを検出するには、 AWS Control Tower ドキュメントの「 でドリフトを検出して解決 AWS Control Towerする」を参照してください。

合成テスト

合成テストは、エンドユーザーエクスペリエンスをシミュレートする方法でアプリケーションの API をテストするために、本番環境でスケジュールを基に実行される設定可能なソフトウェア作成プロセスです。これらのテストは、元々採炭業で使用されていた用語に関連して、Canary と呼ばれることがあります。合成テストでは、グレー障害の場合と同様に、障害が部分的または断続的であっても、アプリケーションの中断が発生すると早期警告が表示されることがあります。

カオスエンジニアリング

カオスエンジニアリングは、アプリケーションを意図的にリスク軽減の方法で破壊的なイベントにさらし、その対応を注意深くモニタリングし、必要な改善を実装する体系的なプロセスです。その目的は、このような中断を処理するアプリケーションの機能について、前提を検証または疑問視することです。カオスエンジニアリングは、これらのイベントを成り行きに任せるのではなく、エンジニアが制御された環境で実験をオーケストレーションできるようにし、通常はトラフィックが少なく、効果的な緩和のためのエンジニアリングサポートがすぐに利用できる期間に実施します。

カオスエンジニアリングは、検討中のアプリケーションの通常の動作状態 (定常状態と呼ばれます) を理解することから始まります。そこから、中断が発生してもアプリケーションの正常な動作を詳述

する仮説を立て、実験を実行します。実験では、ネットワークレイテンシー、サーバー障害、ハードドライブエラー、外部依存関係の障害などを含みますが、それらに限定しない中断を意図的に注入します。次に、実験の結果を分析し、学習内容に基づいてアプリケーションの耐障害性を強化します。この実験は、パフォーマンスなど、アプリケーションのさまざまな側面を改善するための貴重なツールとして機能し、それ以外の場合では隠されていた可能性のある潜在的な問題を検出します。さらに、カオスエンジニアリングは、オブザーバビリティツールやアラームツールの欠陥を明らかにし、それらを改良するのに役立ちます。また、復旧時間の短縮と運用スキルの向上にも貢献します。カオスエンジニアリングは、ベストプラクティスの導入を加速し、継続的な改善という考え方を育みます。最終的には、チームが定期的な演習と反復により運用スキルを構築し、磨き上げることができるようになります。

AWS では、非本番環境でカオスエンジニアリング作業を開始することを推奨しています。AWS Fault Injection Service (AWS FIS) を使用して、 AWSに固有の障害だけでなく、汎用的な障害でもカオスエンジニアリングの実験を実行できます。このフルマネージドサービスには、停止条件アラームと完全なアクセス許可コントロールが含まれているため、安全性と信頼性に優れたカオスエンジニアリングを簡単に導入できます。

ステージ 4: 運用

「ステージ 3: 評価とテスト」が完了したら、アプリケーションを本番環境にデプロイする準備が整います。「運用」ステージでは、アプリケーションを本番環境にデプロイし、カスタマーエクスペリエンスを管理します。 アプリケーションの設計と実装によって耐障害性の成果の多くが決まりますが、このステージでは、システムが耐障害性を維持および改善するために使用する運用プラクティスに焦点を当てます。オペレーショナルエクセレンスの文化を構築すると、これらのプラクティスに標準と一貫性が生まれます。

オブザーバビリティ

カスタマーエクスペリエンスを理解する上で最も重要なのは、モニタリングとアラームを行うことです。アプリケーションの状態を理解するにはアプリケーションを計測する必要があり、多様な視点が必要となります。つまり、サーバー側とクライアント側の両方から、通常は Canary を使用して測定を行う必要があります。メトリクスには、アプリケーションの依存関係と障害分離境界に沿ったディメンションによる操作に関するデータが含まれている必要があります。また、アプリケーションによって実行されるすべての作業単位に関して追加の詳細を提供するログも作成する必要があります。Amazon CloudWatch の埋め込みメトリクス形式などのソリューションを使用して、メトリクスとログを組み合わせることを検討できます。常にオブザーバビリティを高める必要があることがわかるため、必要なレベルの計測を実装するために必要なコスト、労力、複雑さのトレードオフを検討してください。

以下のリンク先では、アプリケーションを計測し、アラームを作成するためのベストプラクティスについて提供しています。

- Monitoring production services at Amazon (AWS re:Invent 2020 プレゼンテーション)
- Amazon Builders' Library: Operational Excellence at Amazon (AWS re:Invent 2021 プレゼンテーション)
- Observability best practices at Amazon (AWS re:Invent 2022 プレゼンテーション)
- 運用の可視性を高めるために分散システムを装備する (Amazon Builders' Library 記事)
- 運用を可視化するためのダッシュボードの構築 (Amazon Builders' Library 記事)

イベント管理

アラーム (またはさらに悪いケースでは顧客) が何か問題があることを知らせたときは、障害を処理するイベント管理プロセスを用意する必要があります。このプロセスには、オンコールオペレーター

の関与、問題のエスカレーション、人為的ミスの排除に役立つトラブルシューティングへの一貫したアプローチのためのランブックの確立を含める必要があります。ただし、障害は通常、単独では発生しません。単一のアプリケーションが、それに依存する他の複数のアプリケーションに影響を与える可能性があります。影響を受けるすべてのアプリケーションについて理解し、複数のチームのオペレーターを 1 回の電話会議で招集することで、問題に迅速に対処できます。ただし、組織の規模と構造によっては、このプロセスに一元的な運用チームが必要になる場合があります。

イベント管理プロセスの設定に加えて、ダッシュボードを通じたメトリクスの定期的なレビューが必要になります。定期的なレビューは、アプリケーションのパフォーマンスにおけるカスタマーエクスペリエンスと長期的な傾向を理解するのに役立ちます。これにより問題やボトルネックを、それらが本番環境に重大な影響を与える前に特定できるようになります。一貫性のある標準化された方法でメトリクスをレビューすることには大きな利点がありますが、トップダウンの賛同と時間の投資が必要です。

以下のリンク先では、ダッシュボードの構築と運用メトリクスのレビューに関するベストプラクティスを示しています。

- 運用を可視化するためのダッシュボードの構築 (Amazon Builders' Library 記事)
- Amazon's approach to failing successfully (AWS re:Invent 2019 プレゼンテーション)

継続的な耐障害性

「ステージ 2: 設計と実装」、「ステージ 3: 評価とテスト」では、アプリケーションを本番環境にデプロイする前にレビューとテストアクティビティを開始しました。「運用」ステージでは、本番環境でこれらのアクティビティを継続的に実行する必要があります。AWS Well-Architected フレームワークのレビュー、運用準備状況レビュー (ORR)、および耐障害性分析フレームワークを通じて、アプリケーションの耐障害性体制を定期的にレビューする必要があります。これにより、アプリケーションが確立されたベースラインや標準からドリフトしないようにし、新規または更新されたガイダンスを最新の状態に保つことができます。これらの継続的な耐障害性アクティビティは、これまで予期されなかった中断を発見し、新しい軽減策を考えるのに役立ちます。

また、本番前環境でゲームデーとカオスエンジニアリング実験を正常に実行した後、本番環境でもそれらを実行することを検討することもできます。ゲームデーでは、軽減するための耐障害性メカニズムを構築した既知のイベントをシミュレートします。例えば、ゲームデーで AWS リージョンサービスの障害をシミュレートし、マルチリージョンフェイルオーバーを実装する場合があります。これらのアクティビティの実装には相当量の労力が必要になる可能性がありますが、どちらのプラクティスも、システムが耐障害性を設計した障害モードに対して回復力があることを確信するのに役立ちます。

アプリケーションを運用し、運用イベントが発生し、メトリクスをレビューし、アプリケーションをテストすることで、さまざまな対応と学習の機会が得られます。

ステージ 5: 対応と学習

アプリケーションによる中断を伴うイベントへの対応は、その信頼性に影響します。経験と、アプリケーションが過去に行った中断への対応方法から学ぶことも、信頼性を向上させるために重要です。

「対応と学習」ステージでは、アプリケーションの中断を伴うイベントへのより適切な対応のために実装できるプラクティスに焦点を当てます。また、運用チームやエンジニアの経験から最大量の学習内容を抽出するのに役立つプラクティスも含まれています。

インシデント分析レポートの作成

インシデントが発生した場合の最初のアクションは、顧客とビジネスへのさらなる損害をできるだけ早く防止することです。アプリケーションが復旧したら、次のステップは何が起こったのかを理解し、再発を防ぐためのステップを特定することです。このインシデント後分析は、通常、アプリケーションの障害を引き起こした一連のイベントと、アプリケーション、顧客、ビジネスに対する中断の影響を文書化するレポートとしてキャプチャされます。このようなレポートは貴重な学習アーティファクトとなり、ビジネス全体で広く共有する必要があります。

> **Note**
>
> 責任を転嫁することなくインシデント分析を実行することが重要です。すべてのオペレーターが、持っている情報を考慮して、最善かつ適切な一連のアクションを実行したと仮定します。レポートには、オペレーターやエンジニアの名前を使用しないでください。人為的ミスを障害の理由として挙げると、チームメンバーが自分自身を守ろうと用心するようになり、不正確または不完全な情報がキャプチャされる可能性があります。

Amazon Correction of Error (COE) プロセスで文書化されているように、優れたインシデント分析レポートは標準化された形式に従い、アプリケーションの障害の原因となった状況をできるだけ詳細に記載しようとします。このレポートでは、タイムスタンプ付きの一連のイベントを詳述し、タイムライン上のアプリケーションの測定可能な状態を記述する定量的データ (多くの場合、モニタリングダッシュボードからのメトリクスとスクリーンショット) を記載します。このレポートには、アクションを実行したオペレーターやエンジニアの思考プロセスと、それらを結論に導いた情報を記載する必要があります。また、レポートには、発生したアラーム、それらのアラームがアプリケーションの状態を正確に反映しているかどうか、イベントと結果のアラームの間の遅延時間、インシデントの解決時間など、さまざまな指標のパフォーマンスについても詳細に説明する必要があります。タイム

ラインには、開始されたランブックまたはオートメーションと、アプリケーションが有用な状態を取り戻すのにそれらがどのように役立ったかも記載されます。タイムラインのこれらの要素は、問題への対処速度や中断の軽減にどの程度効果的だったかなど、自動化された対応とオペレーターによる対応の有効性をチームが理解するのに役立ちます。

この過去のイベントの詳細な実態は、強力な教育ツールとなります。チームは、これらのレポートをビジネス全体で使用できる中央リポジトリに保存し、他のユーザーがレポートからイベントを確認して学習できるようにする必要があります。これにより、本番環境で何が問題になるかについてのチームの直感力を向上させることができます。

詳細なインシデントレポートのリポジトリも、オペレーターのトレーニング資料のソースになります。チームはインシデントレポートを使用して、テーブルトップデーまたはライブゲームデーを鼓舞します。ここではチームに、レポートに記載されたタイムラインを再生する情報が提供されます。オペレーターは、タイムラインの一部の情報を使用してシナリオを順を追って説明し、実行するアクションを記述できます。その後、ゲームデーのモデレーターは、オペレーターのアクションに基づいてアプリケーションがどのように応答したかに関するガイダンスを提供できます。これにより、オペレーターのトラブルシューティングスキルが育成されるため、オペレーターはより簡単に問題を予測してトラブルシューティングできるようになります。

アプリケーションの信頼性に対する責任を担う中央集中型のチームは、組織全体がアクセスできる一元化されたライブラリにこれらのレポートを維持する必要があります。また、このチームは、インシデント分析レポートの作成方法について、レポートテンプレートとトレーニングチームを管理する責任も担う必要があります。信頼性チームは定期的にレポートを確認し、ソフトウェアライブラリ、アーキテクチャパターン、またはチームプロセスの変更を通じて対処できるビジネス全体の傾向を検出する必要があります。

運用レビューの実施

「ステージ 4: 運用」で説明したように、運用レビューでは、最近の機能リリース、インシデント、運用メトリクスについて確認できます。また、運用レビューは、機能リリースやインシデントから学んだことを組織内の幅広いエンジニアリングコミュニティと共有する機会でもあります。運用レビュー中、チームはロールバックされた機能のデプロイ、発生したインシデント、およびそれらの処理方法を確認します。これにより、組織全体のエンジニアは、他者の経験から学び、質問をする機会が得られます。

社内のエンジニアリングコミュニティに運用レビューを公開し、ビジネスを動かす IT アプリケーションと、発生する可能性のある問題のタイプについて詳しく学習できるようにします。エンジニアは、この知識をビジネス用の他のアプリケーションを設計、実装、デプロイする際に活用できます。

アラームのパフォーマンスの確認

「運用」ステージで説明したように、アラームによってダッシュボードアラート、チケットの作成、E メールの送信、またはオペレーターのページングが発生する可能性があります。 アプリケーションには、オペレーションのさまざまな側面をモニタリングするように多数のアラームが設定されます。時間の経過とともに、これらのアラームの精度と有効性を確認してアラームの精度を高め、誤検出を減らし、重複アラートを統合する必要があります。

アラームの精度

アラームの原因となった特定の中断を解釈または診断するのにかかる時間を短縮するために、アラームはできるだけ具体的である必要があります。アプリケーションの障害に対応してアラームが発生した場合、アラームを受信して対応するオペレーターは、まずアラームが伝達する情報を解釈する必要があります。この情報には、復旧手順などの一連のアクションにマッピングされる単純なエラーコードであったり、アラームが発生した理由を理解するために確認する必要があるアプリケーションログの行が含まれていたりする場合があります。チームがアプリケーションの運用をより効果的に行うには、これらのアラームをできる限り明確かつ簡潔に調整する必要があります。

アプリケーションで発生する可能性のあるすべての中断を予測することはできないため、オペレーターが分析と診断を実行する必要がある一般的なアラームが常に存在します。チームは、対応時間を改善し、平均修復時間 (MTTR) を短縮するために、一般的なアラームの数を減らすように努める必要があります。理想的には、アラームと自動対応または人間による対応との間には 1 対 1 の関係が必要です。

誤検出

オペレーターからのアクションは必要ないが、E メール、ページ、またはチケットとしてアラートを生成するアラームは、時間の経過とともにオペレーターに無視されるようになります。 定期的に、またはインシデント分析の一環としてアラームを確認し、無視されることが多いアラームや、オペレーターからのアクションを必要としないアラーム (誤検出) を特定します。 アラームを削除するか、オペレーターにとって実用的なアラートを発行するようにアラームを改善する必要があります。

偽陰性

インシデント中にアラートするように設定されたアラームは、予期しない方法でアプリケーションに影響を与えるイベントが原因で失敗する可能性があります。 インシデント分析の一環として、発生する必要があったのに発生しなかったアラームを確認する必要があります。イベントで発生する可能性のある条件をより適切に反映するように、これらのアラームを改善する必要があります。または、

同じ中断にマッピングされるが、別の中断症状によって発生する追加のアラームを作成する必要があります。

重複アラート

アプリケーションを損なう中断は、複数の症状を引き起こし、複数のアラームを引き起こす可能性があります。 定期的に、またはインシデント分析の一環として、発行されたアラームとアラートを確認する必要があります。 オペレーターがアラートを重複して受信した場合は、集約アラームを作成して 1 つのアラートメッセージに統合します。

メトリクスレビューの実施

チームは、1 か月あたりの重大度別のインシデント数、インシデントの検出時間、原因の特定時間、修復時間、作成されたチケット数、送信されたアラート、呼び出すページなど、アプリケーションに関する運用メトリクスを収集する必要があります。これらのメトリクスを少なくとも月 1 回確認して、運用スタッフへの負担、それらが処理する信号対雑音比 (情報アラートや実用的なアラートなど)、チームが制御下でアプリケーションを運用する機能を改善しているかどうかを把握します。このレビューを使用して、運用チームの測定可能な側面における傾向を理解します。これらのメトリクスを改善する方法について、チームからアイデアを求めます。

トレーニングと有効化の提供

インシデントや予期しない動作を引き起こしたアプリケーションとその環境の詳細な説明をキャプチャすることは困難です。さらに、アプリケーションの耐障害性をモデル化してこのようなシナリオを予測することは、必ずしも簡単なことではありません。組織は、耐障害性モデリング、インシデント分析、ゲームデー、カオスエンジニアリング実験などのアクティビティに参加するために、運用チームや開発者向けのトレーニングやイネーブルメントの資料に投資する必要があります。これにより、チームが作成するレポートの忠実度と、チームが会得する知識が向上します。また、チームは、スケジュールされたレビューを通じてインサイトを提供する必要がある、より小規模で経験豊富なエンジニアグループに依存することなく、障害を適切に予測できるようになります。

インシデントナレッジベースの作成

インシデントレポートは、インシデント分析からの標準出力です。アプリケーションに障害が発生していなくても、異常なアプリケーション動作を検出したシナリオを文書化するには、同じレポートまたは同様のレポートを使用する必要があります。同じ標準化されたレポート構造を使用して、カオス実験とゲームデーの結果を記載します。レポートは、インシデントやその他の予期しない動作を引き

起こしたアプリケーションとその環境のスナップショットを表します。これらの標準化されたレポートは、ビジネス内のすべてのエンジニアがアクセスできる中央リポジトリに保存する必要があります。

その後、運用チームと開発者は、このナレッジベースを検索して、過去にアプリケーションを中断した要因、中断の原因となった可能性のあるシナリオのタイプ、アプリケーションの障害の防止要因を理解できます。このナレッジベースは、運用チームと開発者のスキルを向上させるためのアクセラレーターになり、チームや開発者が知識と経験を共有できるようになります。さらに、レポートをゲームデーやカオス実験のトレーニング資料やシナリオとして使用し、運用チームの直感と中断のトラブルシューティング能力を向上させることができます。

> **Note**
>
> また、標準化されたレポート形式では、リーダーが読みやすく、探している情報をより迅速に見つけられます。

耐障害性を綿密に実装する

前述のように、高度な組織はアラームに対して複数の対応を実施します。対応が効果的である保証はありません。そのため、対応を重ねることで、アプリケーションのフェイルグレイスフリーに備えることができます。各指標に対し少なくとも 2 つの対応策を実施して、個々の対応が DR シナリオにつながる可能性のある単一障害点にならないように確認することをお勧めします。 これらの複数対応は、前の対応が無効であった場合にのみ連続した対応が実行されるよう、番号順に作成する必要があります。1 つのアラームに対し複数の対応を重ねて実行しないでください。代わりに、対応が失敗だったかどうかを示すアラームを使用し、失敗した場合は次の対応を重ねて開始します。

結論とリソース

このガイドでは、目標の設定、設計と実装、評価とテスト、運用、対応と学習の 5 つのステージでベストプラクティスを実装することでアプリケーションの耐障害性を継続的に向上させることができるライフサイクルについて説明します。

このガイドで説明されているサービスと概念の詳細については、以下のリソースを参照してください。

AWS のサービス:

- AWS Backup
- AWS Elastic Disaster Recovery
- AWS Fault Injection Service (AWS FIS)
- AWS Resilience Hub
- Amazon Application Recovery Controller (ARC)
- AWS X-Ray

ブログ投稿と記事:

- 可用性およびその他: AWS の分散システムの回復力の理解と向上
- AWS 障害分離境界
- AWS マルチリージョンの基礎
- クラウドでのカオスエンジニアリング
- アプリケーションを AWS Resilience Hub と AWS CodePipeline で継続的に評価する
- Disaster Recovery of On-Premises Applications to AWS
- 信頼性の柱 – AWS Well-Architected フレームワーク
- 耐障害性分析フレームワーク

寄稿者

このガイドの寄稿者は次のとおりです。

- AWS プリンシパルソリューションアーキテクト、Bruno Emer
- AWS プリンシパルソリューションアーキテクト、Clark Richey
- AWS リライアビリティサービス部門ジェネラルマネージャー、Elaine Harvey
- AWS プリンシパルソリューションアーキテクト、Jason Barto
- AWS プリンシパルソリューションアーキテクト、John Formento
- AWS プロダクトマーケティング担当シニアマネージャー、Lisi Lewis
- AWS プリンシパルソリューションアーキテクト、Michael Haken
- AWS プリンシパルソリューションアーキテクト、Neeraj Kumar
- AWS プリンシパルソリューションアーキテクト、Wangechi Doble

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	—	2023 年 10 月 6 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、クラウド導入を成功させるための準備に役立つ人材開発、トレーニング、コミュニケーションに関するガイダンスを提供します。詳細については、AWS CAF ウェブサイトと AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

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

B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

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

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

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

ブルームフィルター

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

ブルー/グリーンデプロイ

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

ボット

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

ボットネット

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

ブランチ

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

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

ブレークグラスアクセス

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

ブラウンフィールド戦略

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

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

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

ビジネス継続性計画 (BCP)

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

C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

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

CCoE

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

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

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

カオスエンジニアリング

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

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

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

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。移行戦略との関連性については、 AWS 「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

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

コールドキャッシュ

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

コールドデータ

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

コンピュータビジョン (CV)

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

設定ドリフト

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

構成管理データベース (CMDB)

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

コンフォーマンスパック

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

継続的インテグレーションと継続的デリバリー (CI/CD)

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

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

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

データドリフト

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

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

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

データ最小化

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

データ境界

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

データの前処理

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

データ出所

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

データ件名

データを収集、処理している個人。

データウェアハウス

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

データベース定義言語 (DDL)

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

データベース操作言語 (DML)

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

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

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

深層学習

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

多層防御

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

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS

Organizations ドキュメントの「AWS Organizationsで使用できるサービス」を参照してください。

トラブルシューティング

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

開発環境

「環境」を参照してください。

検出管理

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

開発バリューストリームマッピング (DVSM)

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

デジタルツイン

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

ディメンションテーブル

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

ディザスタ

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

ディザスタリカバリ (DR)

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

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用する方法の詳細については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

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

DVSM

「開発バリューストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

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

電子データ交換 (EDI)

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

暗号化

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

暗号化キー

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

エンディアン

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

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

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

エンタープライズリソースプランニング (ERP)

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

エンベロープ暗号化

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

環境

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

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

エピック

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

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

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

F

ファクトテーブル

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

フェイルファスト

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

障害分離境界

では AWS クラウド、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、、Amazon GuardDuty AWS Security Hub CSPM、、 AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

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。