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

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

序章 1
用語と定義 2
継続的な回復力 3
ステージ 1: 目標を設定する 4
重要なアプリケーションのマッピング 4
ユーザーストーリーのマッピング 5
測定値の定義 5
追加の測定値の作成 6
ステージ 2: 設計と実装 8
AWS Well-Architected フレームワーク 8
依存関係について 9
ディザスタリカバリ戦略 9
CI/CD 戦略の定義 10
ORRs の実施 11
AWS 障害分離の境界について 11
レスポンスの選択 11
耐障害性モデリング 12
安全に失敗する 13
ステージ 3: 評価とテスト 14
デプロイ前のアクティビティ 14
環境設計 14
インテグレーションテスト 14
自動デプロイパイプライン 15
負荷テスト 16
デプロイ後のアクティビティ 16
レジリエンス評価の実施 16
DR テスト 16
ドリフト検出 17
合成テスト 17
カオスエンジニアリング 17
ステージ 4: 運用 19
オブザーバビリティ 19
イベント管理 19
継続的な回復力 20

ステージ 5: 応答して学習する 22
インシデント分析レポートの作成 22
運用レビューの実施 23
アラームパフォーマンスの確認 24
アラームの精度 24
誤検出 24
偽陰性 24
重複アラート 25
メトリクスレビューの実施 25
トレーニングと有効化の提供 25
インシデントナレッジベースの作成 25
レジリエンスを深く実装する 26
結論とリソース 27
寄稿者 28
ドキュメント履歴 29
用語集 30
..... 30
A 31
B 34
C 36
D 39
E 43
F 45
G 46
H 48
I 49
L 51
M 52
O 56
P 59
Q 62
R 62
S 65
T 69
U 70
V 71

W 71
Z 72
..... lxxiii

耐障害性ライフサイクルフレームワーク: 耐障害性向上への継続的なアプローチ

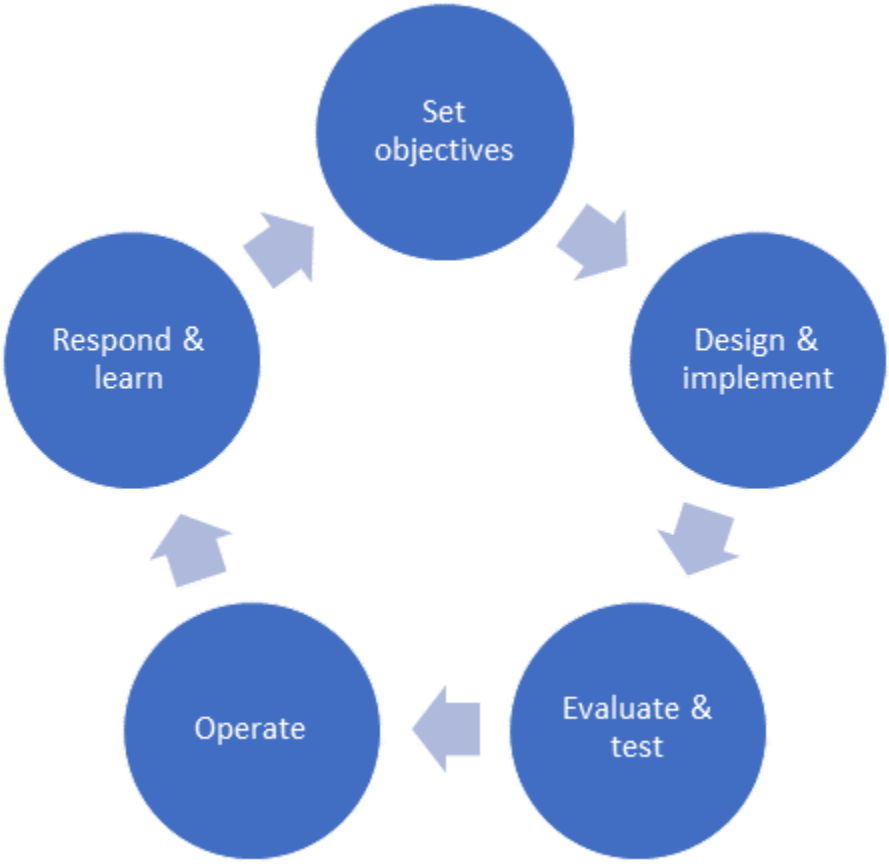
Amazon Web Services (寄稿者)

2023 年 10 月 (ドキュメント履歴)

現代の組織は、特に顧客からの期待が常にオンで常に利用可能な考え方にシフトするにつれて、回復力関連の課題がますます増え続けています。リモートチームと複雑な分散アプリケーションは、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回復力が必要です。

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、中断に耐えたり、中断から回復したり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。（「 Well-Architected AWS 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、望ましいレベルの回復性を実現するには、トレードオフが必要になることがよくあります。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。

は、顧客や社内チームとの長年の連携に基づいて、回復力の学習とベストプラクティスをキャプチャする回復力ライフサイクルフレームワークを開発 AWS 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。各段階で、戦略、サービス、メカニズムを使用してレジリエンス体制を改善できます。



これらのステージについては、このガイドの以下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: 応答して学習する

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、いくつかの用語を明確に定義する必要があります。

- コンポーネントは、 関数を実行する要素であり、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、さらにはサーバー、データストア、多要素認証 (MFA) デバイスなどの外部依存関係などがあります。

- アプリケーションは、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です。アプリケーションは、1 つの AWS アカウント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 および リージョンにまたがる複数の AWS アカウント コンポーネントのコレクションである場合もあります。
- システムは、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、プロセスの集合です。これには、関数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。
- 中断とは、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です。
- 障害は、アプリケーションが軽減されない場合に中断がアプリケーションに与える影響です。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。

継続的な回復力

レジリエンス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、アプリケーションチームは、アプリケーションの要件に応じて、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は高くなります。

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、アプリケーションの開発と運用の過程で、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、ソフトウェア開発ライフサイクル (SDLC) と似た耐障害性ライフサイクルを意図的にモデル化しました。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の反復ごとに繰り返すことができます。 ライフサイクルの各段階のプラクティスは、時間の経過とともに段階的に深めることをお勧めします。

ステージ 1: 目標を設定する

必要なレジリエンスのレベルと測定方法を理解することが、設定された目標ステージの基礎です。目標がなく、測定できない場合、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。これをよく例えると、車です。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。乗車中に複数のフラットタイヤを入手する可能性は低く、余分なスペアがあると、船舶スペースや燃料効率などの他の機能から解放される可能性があるため、これは合理的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て、目標が満たされているかどうかを理解します。

重要なアプリケーションのマッピング

レジリエンス目標の定義は、技術的な会話にとどまるべきではありません。代わりに、ビジネス指向の焦点から始めて、アプリケーションが何を提供するべきか、障害の影響を理解します。このビジネス目標の理解は、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。定義したレジリエンス目標はすべてのアプリケーションに適用される場合がありますが、目標の測定方法は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可能性があり、このアプリケーションに障害が発生すると、組織に大きな収益が失われたり、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、組織のビジネス能力に悪影響を及ぼすことなく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。

例として、小売会社の注文管理アプリケーションを考えてみましょう。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、正しく実行されない場合、新しい販売は実行されません。この小売会社には、その建物の 1 つにある従業員向けのコーヒーショップもあります。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、必ずしも会社に金銭的な損害をもたらすとは限りません。この例に基づいて、企業は注文管理アプリケーションに対してより積極的なレジリエンス目標を持つことを選択する可能性が高くなりますが、ウェブアプリケーションのレジリエンスを確保するために大きな投資を行いません。

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、本番環境における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。障害の影響をよりよく理解するために、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順位付けし、潜在的なリスクと影響を評価し、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の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の概要を説明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、小売ビジネスは大きな収益を失う可能性があります。

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を担当していること、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場合があります。前の小売会社の例を使用すると、注文管理機能でチェックアウト、プロモーション、料金設定に別々のアプリケーションが必要になる場合があります。1 つのアプリケーションに障害が発生した場合、その影響はビジネスや会社とやり取りするユーザーによって引き起こされる可能性があります。例えば、会社が新しい注文の追加、プロモーションや割引へのアクセスの提供、製品の価格の更新を行うことができない場合があります。注文管理関数に必要なこれらの異なる機能は、複数のアプリケーションに依存する場合があります。これらの関数には複数の外部依存関係がある場合もあります。これにより、コンポーネントに重点を置いた純粋な回復力を実現するプロセスが複雑になります。このシナリオに対処するより良い方法は、1 つのアプリケーションまたは一連のアプリケーションを操作するときにユーザーが期待するエクスペリエンスの概要を示す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メカニズムを構築できます。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、料金アプリケーションに依存しています。別のユーザーストーリーとして、プロモーションアプリケーションを含むプロモーションの表示があります。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

復旧時点目標 (RPOs)、復旧時間目標 (RTOs)、サービスレベル目標 (SLOs) 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が許容できるデータ損失の量を指します。一方、RTO は停止後にアプリケーションを再度利用できる速度の尺度です。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。また、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 SLOs は、顧客が受け取る予定のサービスレベルを詳細に示し、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オペレーションとリカバリプロセスが中断されることを前提としています。SLOs は、アプリケーションのダウンタイムを短縮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、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 (SLAs) の定義で一般的に使用されます。SLAs 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罰則の概要が記載されています。ただし、SLA はレジリエンス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レジリエンスは向上しません。

SLOs、RPOs、RTOs に基づいて目標の設定を開始できます。レジリエンス目標を定義し、RTO と RTO 目標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、潜在的なレジリエンス関連の弱点を発見できます。 AWS Resilience Hub は AWS 、 Well-Architected Framework 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、定義された RTO と RPO 目標を達成する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、SLOsはレジリエンスの優れた指標ですが、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% を上回ったままになります。 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、平均から標準偏差の範囲内にとどまります。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。例えば、これらの既知の問題が発生した場合、復旧時間は x % 短縮されます。ビジネスニーズに合った目標を作成すると、アプリケーションが許容する障害のタイプを予測するのに役立ちます。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方法を特定するのにも役立ちます。

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、引き続き運用するという目標を考えた場合、アプリケーションは事前にスケーリングする必要があるか、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リングできる必要があ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状態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アクションを実行できます。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、レジリエンス目標を設定します。設計と実装の段階では、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、設計の選択肢を特定しようとします。また、変更管理の戦略を定義し、ソフトウェアコードとインフラストラクチャ設定を開発します。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考慮すべき AWS 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

希望するレジリエンス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を行う必要があります。回復力の高い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、運用の側面を考慮する必要があります。AWS Well-Architected フレームワークは、回復力のある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。 AWS Well-Architected フレームワークの 6 つの柱は、回復力、安全性、効率性、コスト効率、持続可能なシステムを設計および運用するためのベストプラクティスを提供します。このフレームワークは、ベストプラクティスに照らして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。

以下は、 AWS Well-Architected フレームワークがレジリエンス目標を達成するアプリケーションの設計と実装にどのように役立つかの例です。

- 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てシンプルにすることをお勧めします。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の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の詳細な説明を見つけることもできます。
- 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推奨します。これにより、アーキテクチャを一貫して測定し、改善すべき領域を特定できます。
- 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を特定して管理するのに役立ちます。潜在的な障害シナリオに事前に対処することで、障害の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: 回復力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て改善することで、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、レジリエンスにとって重要です。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 APIs 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。これらの接続を理解することで、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、中断を分離して管理できます。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、リソースが効果的に使用されていることを確認するのに役立ちます。依存関係を理解すると、代替戦略を作成し、復旧プロセスを調整するのに役立ちます。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を特定するのにも役立ちます。これにより、依存関係に障害が発生した場合でも、アプリケーションは引き続きビジネス機能を提供できます。依存関係は、ロードバランシングとアプリケーションスケーリング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ため、依存関係を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、スケーリング、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、 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AWS X-Ray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、致命的なイベント中であっても、重要な事業運営が障害を最小限に抑えながら維持され、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。多くの場合、複数の場所にまたがる定期的なデータバックアップとデータレプリケーションが組み込まれているため、貴重なビジネス情報を保護し、災害発生時の完全な損失を防ぐのに役立ちます。さらに、多くの業界は、企業が機密データを保護し、災害発生時にサービスを確実に利用できるように DR 戦略を策定する必要があるポリシーによって規制されています。DR 戦略は、最小限のサービス障害を保証することで、顧客の信頼と満足度も強化します。適切に実装され、頻繁に実行される DR 戦略により、災害後の復旧時間が短縮され、アプリケーションが迅速にオンラインに戻されます。さらに、災害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コストからも、かなり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 戦略は、これらの財務上の損失を防ぐのに役立ちます。

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回復力サービスです。

詳細については、 AWS ウェブサイトの「 でのワークロードのディザスタリカバリ 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

アプリケーションの障害の一般的な原因の 1 つは、以前に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。変更の頻度は、影響の機会を増やします。ただし、変更頻度が低いほど変更セットが大きくなり、複雑さが高いために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 (生産性の向上につながる) 維持すると同時に、自動化を通じて各変更を高レベルの検査の対象にす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。

- 完全な自動化: CI/CD の基本的な概念は、ビルドとデプロイプロセスを可能な限り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。この方法により、すべてのコードにテストが関連付けられるため、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、変更を検証します。
- 頻繁なコミットと統合: デベロッパーにコードを頻繁にコミットし、頻繁に統合を実行するよう促します。小規模で頻繁な変更では、テストとデバッグが容易になり、重大な問題のリスクが軽減されます。自動化により、各コミットとデプロイのコストが削減され、頻繁な統合が可能になります。
- イミュータブルな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。インフラストラクチャをできるだけ変更するのではなく置き換え、パイプラインを通じてテストおよびデプロイされる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: 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があります。デプロイの安全性を確保するには、以前の既知の正常な状態をすばやく戻ることが重要です。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、さらには Infrastructure as Code をバージョン管理されたリポジトリに保持します。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 Blue/Green デプロイ: まずアプリケーションの新しいバージョンを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 (Blue/Green) を維持すると、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 は、ツールだけでなく文化に関するものです。自動化、テスト、障害からの学習を重視する文化の構築は、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。ロールバックは、影響を最小限に抑えて非常に迅速に実行された場合、失敗ではなく学習経験と見なす必要があります。

ORRs の実施

運用準備状況レビュー (ORR) は、運用上および手続き上のギャップを特定するのに役立ちます。Amazon では、何十年もの大規模なサービスを運用してきた経験を、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 ORRs を作成しました。ORR は、以前に学んだ教訓をキャプチャし、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していることを確認する必要があります。ORRs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に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できます。詳細については、 AWS Well-Architected Framework ウェブサイトのORRs)」を参照してください。

AWS 障害分離の境界について

AWS 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ます。これらの境界を使用して、提供する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、 ウェブサイトのAWS 「障害分離境界 」を参照してください。 AWS

レスポンスの選択

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、手動操作として自動化できる対応を維持することもできます。アラームに対するレスポンスのタイプは、レスポン

スの実装コスト、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潜在的なビジネス損失の関数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、オペレーティングシステムによってプロセスが再起動されたり、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したり、オペレーターにサーバーにリモート接続して再起動するように指示されたりすることがあります。これらのレスポンスは、アプリケーションサーバープロセスを再起動するのと同じ結果になりますが、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**
>
> 複数のレスポンスを選択して、より詳細なレジリエンス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は 3 つのレスポンスすべてを実装し、それぞれに時間遅延を設けることを選択できます。失敗サーバープロセスインジケータが 30 秒後に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想定できます。したがって、自動スケーリング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する場合があります。インジケータが 300 秒後に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、運用スタッフにアラートが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対応には、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ためのビジネスの需要を反映する必要があります。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して、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レスポンスがいくつか存在する可能性があるため、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は、アプリケーションが予想されるさまざまな中断にどのように対応するかを文書化します。 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て、中断しても障害を軽減または防止できます。 は、回復力分析フレームワークを使用して回復力モデルを開発するためのガイダンスを作成 AWS しました。 このフレームワークは、中断や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緩和策を特定できます。回復力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繰り返すたびに回復力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ごとに使用すると、設計段階での中断を予測し、本番デプロイの前後にアプリケーションをテストすることで、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回復力モデルを開発することで、回復力の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、安全に失敗してください。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運用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オペレーションがあり、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。アプリケーションのフェイルセーフ状態を検討し、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設定します。

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の段階では、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は設計済みですが、まだ本番環境にリリースされていません。この段階では、前の段階で実行されたプラクティスをテストし、結果を評価するアクティビティを実装します。アプリケーションがまだアクティブな開発中であるか、プライマリ開発が完了していて、アプリケーションが本番環境にリリースされる前にテスト中である可能性があります。この段階では、アプリケーションが回復力に関する定義された目標を達成することに対する期待を確認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を開発してテストします。ステージ 2: ステージの設計と実装で開発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。テストと評価は、ステージ 4: 運用ステージに進むにつれて続行されます。

評価ステージ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。デプロイ前のアクティビティは、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、ソフトウェアの新しいバージョンのデプロイや、テスト環境への最初のデプロイが含まれます。デプロイ後のアクティビティは、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、これらのフェーズについて詳しく説明します。

デプロイ前のアクティビティ

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何が起こるかを正確に反映していることをどの程度確信しているかに影響します。Amazon DynamoDB などのサービスを使用して、デベロッパー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) 。 DynamoDB ただし、結果の信頼性を最大限に高めるには、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のような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、外部依存関係で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、カスタム開発の他

のコンポーネント、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、すべての統合テストを CI/CD パイプラインの一部として実行し、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、耐障害性目標に違反す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無関係なように見える部分の回復力に大きな影響を与える可能性があります。コミットごとにテストを実行してみてください。 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 を提供します。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

本番稼働前の環境へのデプロイとテストは反復的で複雑なタスクであり、自動化に任せるのが最適です。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減ります。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、本番環境の設定にますます近づく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。最初の環境では、実稼働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、アプリケーションをある程度の負荷下で実行して、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)。この観測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、このプロセスを繰り返して、環境がサポートするとおりにテストまたはロードを追加します。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 戦略を定義する」を参照)。記事「Amazon Builders' Library での安全なハンドオフデプロイの自動化」は、Amazon がコードデプロイを自動化する方法を説明する優れたリソースです。 Amazon Builders' Library 本番デプロイの前にある環境の数は、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

表面では、負荷テストは統合テストに似ています。アプリケーションの個別の関数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ことを確認します。その後、負荷テストは統合テストを超えて、明確に定義された負荷下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、統合テストが成功した後に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、および負荷が予想される負荷を超えたとき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、極端に負荷がかかってもアプリケーションが回復力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。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

レジリエンス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レジリエンスの評価を継続する必要があります。継続的なレジリエンス評価など、デプロイ後のアクティビティの結果では、レジリエンスライフサイクルの前半で実行したレジリエンスアクティビティの一部を再評価して更新する必要がある場合があります。

レジリエンス評価の実施

アプリケーションを本番環境にデプロイした後も、レジリエンスの評価は停止しません。デプロイパイプラインが明確に定義され、自動化された場合でも、変更が本番環境で直接発生することがあります。さらに、デプロイ前のレジリエンス検証でまだ考慮していない要因があるかもしれません。AWS Resilience Hub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性 AWS CodePipeline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、本番環境でレジリエンス体制を継続的に評価するために役立つため、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、システムの一部としてディザスタリカバリ (DR) 戦略を策定しました。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完全に準備され、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、またどのように更新するかを決定する必要があります。最初

に DR テストを開発するときは、事前にテストをスケジュールし、チーム全体が期待する内容、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 テストの実行に習熟したら、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は発生しないため、いつでも計画を実行する準備を整える必要があります。ただし、未発表は計画外の意味ではありません。主要な利害関係者は、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、イベントを計画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予期しない変更が加えられるのは、自動化と明確に定義された手順が講じられている場合でもです。アプリケーションの設定の変更を検出するには、ベースライン設定からの逸脱を参照するドリフトを検出するメカニズムが必要です。 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 APIs をテストするために、スケジュールに基づいて本番環境で実行される設定可能なソフトウェアを作成するプロセスです。これらのテストは、コールマイニングにおける用語の元の使用を参照し、カナリアと呼ばれることもあります。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、アプリケーションが中断した場合に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体系的なプロセスであり、リスクを緩和する方法でアプリケーションを意図的に破壊的なイベントに晒し、その対応を注意深くモニタリングし、必要な改善を実施します。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能力について、前提を検証または検討することです。カオスエンジニアリングは、これらのイベントを偶然に残す代わりに、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ます。通常、トラフィックが少なく、効果的な緩和のためにすぐに利用できるエンジニアリングサポートがあります。

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定常状態と呼ばれる通常の運用条件を理解することから始まります。そこから、中断が発生した場合のアプリケーションの成功した動作を詳述

する仮説を立てます。実験を実行するには、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、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レジリエンスを高めます。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、特に隠れている可能性のある潜在的な問題を発見します。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と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役立ちます。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、継続的な改善の考え方を育みます。最終的には、チームは定期的な練習と繰り返しを通じて運用スキルを構築して強化できます。

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、汎用障害と固有の障害でカオスエンジニアリング実験を実行できます AWS。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。運用段階では、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回復力の成果の多くが決まりますが、このステージでは、システムが回復力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。運用上の優秀性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部分は、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、多様な視点が必要です。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 を使用して測定する必要があります。メトリクスには、障害分離の境界に沿ったアプリケーションの依存関係やディメンションとのやり取り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する追加の詳細を示すログも作成する必要があります。Amazon CloudWatch 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。よりオブザーバビリティが常に必要で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。

以下のリンクは、アプリケーションの計測とアラームの作成に関するベストプラクティスを示しています。

- Amazon での本番サービスのモニタリング (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)
- Amazon でのオブザーバビリティのベストプラクティス (AWS re:Invent 2022 プレゼンテーション)
- 運用を可視化するための分散システムの計測 (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 の記事)

イベント管理

アラーム (または顧客) で問題が発生したことがわかったら、障害を処理するための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、ヒューマンエラー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を理解し、複数のチームのオペレーターを 1 回の電話会議でまとめることで、問題に迅速に対処できます。ただし、組織のサイズと構造によっては、このプロセスに一元化された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てメトリクスを定期的に確認する必要があります。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が本番環境に大きな影響を与える前に特定できます。一貫性のある標準化された方法でメトリクスを確認すると、大きなメリットが得られますが、トップダウンの賛同と時間の投資が必要です。

以下のリンクは、ダッシュボードと運用メトリクスのレビューの構築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 の記事)
- Amazon の失敗へのアプローチ (AWS re:Invent 2019 プレゼンテーション)

継続的な回復力

ステージ 2: 設計と実装、ステージ 3: 評価とテスト中に、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。運用段階では、本番環境でこれらのアクティビティを反復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 (ORRs)、およびレジリエンス分析フレームワークを通じて、アプリケーションのレジリエンス体制を定期的に確認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。これらの継続的なレジリエンスアクティビティは、以前に予期しない中断を発見し、新しい緩和策を策定するのに役立ちます。

また、実稼働前の環境でゲームデーやカオスエンジニアリング実験を正常に実行した後、実稼働環境でのゲームデーやカオスエンジニアリング実験の実行を検討することもできます。ゲームデーは、緩和するための回復力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、ゲームデーは 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かなりの労力が必要になる場合がありますが、どちらの方法も、システムが耐障害性を持つように設計した障害モードに対して回復力があるという確信を構築するのに役立ちます。

アプリケーションを運用し、運用イベントに遭遇し、メトリクスを確認し、アプリケーションをテストすることで、対応して学習する機会が多数あります。

ステージ 5: 応答して学習する

アプリケーションが破壊的なイベントにどのように応答するかは、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中断にどのように対応したかを学ぶことも、信頼性を向上させるために重要です。

応答と学習のステージでは、アプリケーションの破壊的なイベントにより適切に対応する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、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**
>
> 責任を割り当て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。レポートで演算子または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保護するために保護され、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) プロセスで説明されているように、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態をできるだけ詳細に把握しようとします。このレポートは、タイムスタンプ付きの一連のイベントの詳細を示し、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) をキャプチャします。このレポートには、アクションを実行したオペレーターと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間の遅延、インシデント

の解決時間など、さまざまなインジケータ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どのように役立ったかも記録されます。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対応の有効性をチームが理解するのに役立ちます。

過去のイベントのこの詳細な画像は、強力な教育ツールです。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関するチームの直感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マテリアル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、テーブルトップまたはライブゲームデーを鼓舞できます。チームには、レポートにキャプチャ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、タイムラインからの部分的な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開発されるため、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ます。

アプリケーションの信頼性を担当する一元化されたチームは、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。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維持する責任も果たす必要があります。信頼性チームは定期的にレポートを確認して、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、運用レビューは、最近の機能リリース、インシデント、運用メトリクスを確認する機会です。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、およびそれらがどのように処理されたかを確認します。これにより、組織全体のエンジニアは、他の人の経験から学び、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開いて、ビジネスを運営する IT アプリケーションと、それらが遭遇する可能性のある問題の種類について詳しく知ることができるようにしま

す。この知識は、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。

アラームパフォーマンスの確認

操作ステージで説明したように、アラームにより、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応じてアラームが発生した場合、アラームを受信して応答するオペレータは、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や、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学習したら、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。

アプリケーションに発生する可能性のあるすべての中断を予測できないため、オペレーターが分析と診断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、応答時間を改善し、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、アラームと自動または人間が実行するレスポンスの間には one-to-one 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よって無視されます。 定期的に、またはインシデント分析の一環として、アラームを確認して、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

すべきだったが発生しなかったアラームを確認する必要があります。イベントから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、中断の別の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る場合があります。

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。 オペレータが重複するアラートを受信した場合は、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、インシデントを検出する時間、原因を特定する時間、修復時間、作成されたチケット数、送信されたアラート数、発生したページ数など、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signal-to-noise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能力を向上させ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の傾向を理解します。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、レジリエンスモデリング、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、運用チームやデベロッパー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、チームが把握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、障害を予測する準備が強化されます。

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キャプチャします。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、運用チームとデベロッパーは、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、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マテリアルやシナリオとして使用して、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**
>
> 標準化されたレポート形式は、読者に親しみやすさを提供し、探している情報をより迅速に見つけるのに役立ちます。

レジリエンスを深く実装する

前述のように、高度な組織はアラームに対して複数の応答を実装します。レスポンスが有効である保証はありません。そのため、レスポンスをレイヤーすることで、アプリケーションが適切に失敗する準備が整います。各インジケータに少なくとも 2 つのレスポンスを実装して、個々のレスポンスが 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することをお勧めします。 これらのレイヤーは、前のレスポンスが無効な場合にのみ連続したレスポンスが実行されるように、シリアル順に作成する必要があります。1 つのアラームに対して複数のレイヤードレスポンスを実行しないでください。代わりに、レスポンスが失敗し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レイヤードレスポンスを開始します。

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、応答と学習の 5 つの段階にわたって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のに役立つライフサイクルを紹介します。

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 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

ブログの投稿と記事：

- 可用性以上: での分散システムの耐障害性を理解して改善する AWS
- 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
- 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 AWS CodePipeline
- オンプレミスアプリケーションの へのディザスタリカバリ AWS
- 信頼性の柱 — AWS 優れたアーキテクチャのフレームワーク
- 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以下のとおりです。

- Bruno Emer、Principal Solutions Architect、 AWS
- Clark Richey、Principal Solutions Architect、 AWS
- Elaine Harvey、信頼性サービス担当、総マネージャー AWS
- プリンシパルソリューションアーキテクト、Jason Barto AWS
- John Formento、Principal Solutions Architect、 AWS
- Lisi Lewis、シニアプロダクトマーケティングマネージャー、 AWS
- Michael Haken、Principal Solutions Architect、 AWS
- Neeraj Kumar、Principal Solutions Architect、 AWS
- Wangechi Doble、Principal Solutions Architect、 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	—	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します。
- 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: でオンプレミスの Oracle データベースを Oracle 用 Amazon Relational Database Service (Amazon RDS) に移行します AWS クラウド。
- 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: カスタマーリレーションシップ管理 (CRM) システムを Salesforce.com に移行します。
- 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: オンプレミスの Oracle データベースを の EC2 インスタンス上の Oracle に移行します AWS クラウド。
- 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。

- 使用停止 — 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コントロール」を参照してください。

抽象化されたサービス

「 マネージドサービス」を参照してください。

ACID

「アトミック性」、「整合性」、「分離」、「耐久性」を参照してください

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。より柔軟ですが、アクティブ/パッシブ移行よりも多くの作業が必要です。

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の方法。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

行のグループで動作し、グループの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 などがあります`MAX`。

AI

「人工知能」を参照してください。

AIOps

「人工知能オペレーション」を参照してください。

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。

アプリケーションコントロール

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 の需要要素であり、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?」を参照してください。

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

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)

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

AWS ワークロード認定フレームワーク (AWS WQF)

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

B

不正なボット

個人または組織に損害を与えることを目的としたボット。

BCP

「事業継続計画」を参照してください。

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Data in a behavior graph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。エンディアン性も参照してください。

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

2 つの異なる同一の環境を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 (青) 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 (緑) 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、ボットハーダーまたはボットオペレーターとして知られる単一関係者の管理下にあるボットのネットワーク。ボットは、ボットとその影響をスケールするための最もよく知られているメカニズムです。

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なります。詳細については、 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 AWS 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、ホワイトペーパー AWSでのコンテナ化されたマイクロサービスの実行 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。

Canary デプロイ

エンドユーザーへのバージョンのスローリリースと増分リリース。確信が持てば、新しいバージョンをデプロイし、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

「データキャプチャの変更」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します。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テクノロジー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 に移行するときに通常実行する 4 つのフェーズ AWS クラウド：

- 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

「設定管理データベース」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または が含まれますBitbucket Cloud。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などのビジュアル形式から情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI は CV 用の画像処理アルゴリズムを提供します。

設定ドリフト

ワークロードの場合、設定は想定状態から変化します。これにより、ワークロードが非準拠になる可能性があり、通常は段階的かつ意図的ではありません。

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

一元管理とガバナンスを備えた分散型の分散データ所有権を提供す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、通常、大量の履歴データが含まれており、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る。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。

ディープラーニング

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。

デプロイ

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、Implementing security controls on AWSのDetective controls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

スタースキーマでは、ファクトテーブル内の量的データに関するデータ属性を含む小さなテーブル。ディメンションテーブル属性は通常、テキストフィールドまたはテキストのように動作する

離散数値です。これらの属性は、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一般的に使用されます。

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)

災害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に使用する戦略とプロセス。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

「開発値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、応答時間を短縮できます。

電子データ交換 (EDI)

組織間の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プロセス。

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

星スキーマの中央テーブル。事業運営に関する量的データを保存します。通常、ファクトテーブルには、メジャーを含む列とディメンションテーブルへの外部キーを含む列の 2 つのタイプの列が含まれます。

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で段階的なテストを使用する哲学。これはアジャイルアプローチの重要な部分です。

障害分離の境界

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

同様のタスクの実行を求める前に、タスクと必要な出力を示す少数の例を LLM に提供します。この手法は、プロンプトに埋め込まれた例 (ショット) からモデルが学習するコンテキスト内学習のアプリケーションです。少数ショットプロンプトは、特定のフォーマット、推論、またはドメインの知識を必要とする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コントロール」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

段階的なアプローチを使用する代わりに、変更データキャプチャによる継続的なデータレプリケーションを使用して、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)

一般化およびラベル付けされていないデータの大規模なデータセットでトレーニングされている大規模な深層学習ニューラルネットワーク。FMs は、言語の理解、テキストと画像の生成、自然言語の会話など、さまざまな一般的なタスクを実行できます。詳細については、「基盤モデルとは」を参照してください。

G

生成 AI

大量のデータでトレーニングされ、シンプルなテキストプロンプトを使用してイメージ、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る AI モデルのサブセット。詳細については、「生成 AI とは」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 ワークフローはレガシーと見なされ、トランクベースのワークフローはモダンで推奨されるアプローチです。

ゴールデンイメージ

そのシステムまたはソフトウェアの新しいインスタンスをデプロイするためのテンプレートとして使用されるシステムまたはソフトウェアのスナップショット。例えば、製造では、ゴールデンイメージを使用して複数のデバイスにソフトウェアをプロビジョニングし、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、、 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。

ハイアベイラビリティ (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に使用されるデータセットから保留される、ラベル付きの履歴データの一部。モデル予測をホールドアウトデータと比較することで、ホールドアウトデータを使用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

「産業用モノのインターネット」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

既存のインフラストラクチャを更新、パッチ適用、または変更する代わりに、本番環境のワークロード用に新しいインフラストラクチャをデプロイするモデル。イミュータブルインフラストラクチャは、本質的に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、予測性が高くなります。詳細については、 AWS 「 Well-Architected フレームワーク」の「イミュータブルインフラストラクチャを使用したデプロイ」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 によって導入された用語で、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ゼーションを指します。

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。

産業分野における 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、「Building an industrial Internet of Things (IIoT) digital transformation strategy」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 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

「モノのインターネット」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)

大量のデータに対して事前トレーニングされた深層学習 AI モデル。LLM は、質問への回答、ドキュメントの要約、テキストの他の言語への翻訳、文の完了など、複数のタスクを実行できます。詳細については、LLMs」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

「ラベルベースのアクセスコントロール」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。エンディアン性も参照してください。

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または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、不正アクセスにつながる可能性があります。マルウェアの例としては、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。Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB は、マネージドサービスの例です。これら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、制御するためのソフトウェアシステムで、原材料を工場の完成製品に変換します。

MAP

「移行促進プログラム」を参照してください。

メカニズム

ツールを作成し、ツールの導入を推進し、調整を行うために結果を検査する完全なプロセス。メカニズムは、動作中にそれ自体を強化して改善するサイクルです。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。

メッセージキューイングテレメトリトランスポート (MQTT)

リソースに制約のある IoT デバイス用の、パブリッシュ/サブスクライブパターンに基づく軽量 machine-to-machine (M2M) 通信プロトコル。

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)

に移行するためのビジネスケースを検証するための情報を提供するオンラインツール AWS クラウド。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 に移行するために使用するアプローチ AWS クラウド。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、「」の「アプリケーションをモダナイズするための戦略 AWS クラウド」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、『』の「アプリケーションのモダナイゼーション準備状況の評価 AWS クラウド」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

「移行ポートフォリオ評価」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。

ミュータブルインフラストラクチャ

本番ワークロードの既存のインフラストラクチャを更新および変更するモデル。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

「オリジンアクセスコントロール」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

「 オペレーションの統合」を参照してください。

OLA

「運用レベルの契約」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。

オープンプロセス通信 - 統合アーキテクチャ (OPC-UA)

産業用オートメーション用のmachine-to-machine (M2M) 通信プロトコル。OPC-UA は、データの暗号化、認証、認可スキームを備えた相互運用性標準を提供します。

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)

インシデントや潜在的な障害の理解、評価、防止、または範囲の縮小に役立つ質問とそれに関連するベストプラクティスのチェックリスト。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するハードウェアおよびソフトウェアシステム。製造では、OT と情報技術 (IT) システムの統合が、Industry 4.0 変換の主な焦点です。

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。

オリジンアクセスコントロール (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コントロールが可能です。

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

「プログラム可能な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

アクセス許可を定義 (アイデンティティベースのポリシーを参照) 、アクセス条件を指定 (リソースベースのポリシーを参照) 、または の組織内のすべてのアカウントに対する最大アクセス許可を定義 AWS Organizations （ サービスコントロールポリシーを参照) できるオブジェクト。

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。詳細については、マイクロサービスでのデータ永続性の有効化 を参照してください。

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、「移行準備状況ガイド」を参照してください。

述語

`true` または を返すクエリ条件。一般的に`false`は `WHERE`句にあります。

述語プッシュダウン

転送前にクエリ内のデータをフィルタリングするデータベースクエリ最適化手法。これにより、リレーショナルデータベースから取得して処理する必要があるデータの量が減少し、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、Implementing security controls on AWSのPreventative controls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内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じてプライバシーを考慮するシステムエンジニアリングアプローチ。

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を防ぐ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、プロビジョニング前にリソースをスキャンします。リソースがコントロールに準拠していない場合、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)

設計、開発、発売から成長と成熟まで、製品のデータとプロセスのライフサイクル全体にわたる管理。

本番環境

「環境」を参照してください。

プログラム可能なロジックコントローラー (PLC)

製造では、マシンをモニタリングし、製造プロセスを自動化する、信頼性の高い適応可能なコンピュータです。

プロンプトの連鎖

1 つの LLM プロンプトの出力を次のプロンプトの入力として使用して、より良いレスポンスを生成します。この手法は、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善または拡張したりするために使用されます。これにより、モデルのレスポンスの精度と関連性が向上し、より詳細でパーソナライズされた結果が得られます。

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。

パブリッシュ/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。スケーラビリティと応答性を向上させます。たとえば、マイクロサービスベースの MES では、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できるチャネルにイベントメッセージを発行できます。システムは、公開サービスを変更せずに新しいマイクロサービスを追加できます。

Q

クエリプラン

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る手順などの一連のステップ。

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

責任、説明責任、相談、通知 (RACI) を参照してください。

RAG

「取得拡張生成」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

責任、説明責任、相談、情報 (RACI) を参照してください。

RCAC

「行と列のアクセスコントロール」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。

再設計

「7 Rs」を参照してください。

目標復旧時点 (RPO)

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

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

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

回帰

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。

プラットフォーム変更

「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

中断に抵抗または回復するアプリケーションの機能。高可用性とディザスタリカバリは、 で回復性を計画する際の一般的な考慮事項です AWS クラウド。詳細については、AWS クラウド 「レジリエンス」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。サポートを含めると、そのマトリックスは RASCI マトリックスと呼ばれ、サポートを除外すると 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、Implementing security controls on AWSのResponsive controls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。

取得拡張生成 (RAG)

LLM がレスポンスを生成する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を参照する生成 AI テクノロジー。たとえば、RAG モデルは、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する場合があります。詳細については、「RAG とは」を参照してください。

ローテーション

攻撃者が認証情報にアクセスすることをより困難にするために、シークレットを定期的に更新するプロセス。

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 Management Console 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

「監視コントロールとデータ取得」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。シークレット値は、バイナリ、単一の文字列、または複数の文字列にすることができます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの内容」を参照してください。

設計によるセキュリティ

開発プロセス全体でセキュリティを考慮するシステムエンジニアリングアプローチ。

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、予防的、検出的、応答的、プロアクティブの 4 つの主なタイプがあります。

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に自動的に応答または修復するように設計された、事前定義された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。自動応答アクションの例としては、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータの、それ AWS のサービス を受け取る による暗号化。

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、スループットなど、サービスのパフォーマンス側面の測定。

サービスレベルの目標 (SLO)

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

責任共有モデル

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

SIEM

セキュリティ情報とイベント管理システムを参照してください。

単一障害点 (SPOF)

システムを中断する可能性のあるアプリケーションの 1 つの重要なコンポーネントの障害。

SLA

「サービスレベルの契約」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、『』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 AWS クラウド」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

1 つの大きなファクトテーブルを使用してトランザクションデータまたは測定データを保存し、1 つ以上の小さなディメンションテーブルを使用してデータ属性を保存するデータベース組織構造。この構造は、データウェアハウスまたはビジネスインテリジェンスの目的で使用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。

監視制御とデータ収集 (SCADA)

製造では、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稼働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潜在的な問題を検出したり、パフォーマンスをモニタリングしたりする方法でシステムをテストします。Amazon CloudWatch Synthetics を使用して、これらのテストを作成できます。

システムプロンプト

LLM にコンテキスト、指示、またはガイドラインを提供して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、ユーザーとのやり取りのルールを確立するのに役立ちます。

T

tags

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「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

チューニング

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

ツーピザチーム

2 枚のピザで養う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する行のグループに対して計算を実行する SQL 関数。ウィンドウ関数は、移動平均の計算や、現在の行の相対位置に基づく行の値へのアクセスなどの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 「ワークロード認定フレームワーク」を参照してください。

Write Once, Read Many (WORM)

データを 1 回書き込み、データの削除や変更を防ぐストレージモデル。承認されたユーザーは、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

ゼロデイ脆弱性を利用する攻撃、通常はマルウェア。

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 にタスクを実行する手順を提供しますが、タスクのガイドに役立つ例 (ショット) はありません。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。