



耐障害性分析フレームワーク

# AWS 規範ガイドンス



# AWS 規範ガイド: 耐障害性分析フレームワーク

Copyright © 2024 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
フレームワークの概要 .....	3
ワークロードの理解 .....	6
フレームワークの適用 .....	8
潜在的な障害の軽減 .....	11
トレードオフとリスクを理解する .....	11
障害モードオブザバビリティ .....	13
一般的な緩和戦略 .....	14
継続的な改善 .....	19
結論とリソース .....	21
ドキュメント履歴 .....	22
用語集 .....	23
# .....	23
A .....	24
B .....	27
C .....	29
D .....	32
E .....	36
F .....	38
G .....	39
H .....	40
I .....	41
L .....	43
M .....	44
O .....	48
P .....	51
Q .....	53
R .....	54
S .....	56
T .....	60
U .....	61
V .....	62
W .....	62
Z .....	63

---

..... lxv

# レジリエンス分析フレームワーク

アマゾンウェブサービスのジョン・フォルメント、ブルーノ・エマー、スティーブン・フーパー、ジェイソン・バート、マイケル・ハーケン (AWS)

2023 年 9 月([ドキュメント履歴](#))

一貫性のある反復可能な基準とプロセスは、継続的改善の重要な部分です。これは分散システムのレジリエンスにも当てはまります。このガイドの目的は、障害モードとそれがワークロードに及ぼす影響を分析する一貫した方法を提供する回復力分析フレームワークを導入することです。このフレームワークを設計から運用に至るまで、ワークロードのライフサイクル全体にわたって使用することで、さまざまな潜在的な障害モードに対するワークロードの耐障害性を一貫性のある再現可能な方法で継続的に向上させることができます。これにより、レジリエンスの目標を達成し、ワークロードの望ましいレジリエンス特性を維持することができます。

このフレームワークは、AWS ソリューションアーキテクチャのフィールドチームがさまざまな業界の顧客と協力してきた経験を基に開発されました。プロダクトマネージャー、ソフトウェア開発者、システムエンジニア、運用チーム、アーキテクトなど、さまざまな役職に就く可能性のあるビルダーを対象としています。これらは、分析対象のシステム、サービス、または製品について最もよく知っている人々です。このフレームワークを継続的に使用することで、段階的な進歩を遂げ、長期的なレジリエンス目標を達成することができます。

フレームワークの焦点は、潜在的な障害モードと、その影響を軽減するために使用できる予防および是正措置を特定することです。依存関係でのエラー率の上昇など、直接制御できないコンポーネントで障害が発生した場合でも、それらの障害がワークロードにどのように影響するか、また障害に対応するようにそのワークロードをどのように設計するかを検討する必要があります。最終的には、次のことに焦点を当てる必要があります。対応できる障害自分でコントロールできる緩和策を用いることによって

このガイドでは、フレームワークの概要を説明し、次にワークロードを特定して文書化する方法、フレームワークをそのワークロードに適用する方法、および潜在的な障害を発見した場合の緩和戦略を評価する方法について説明します。

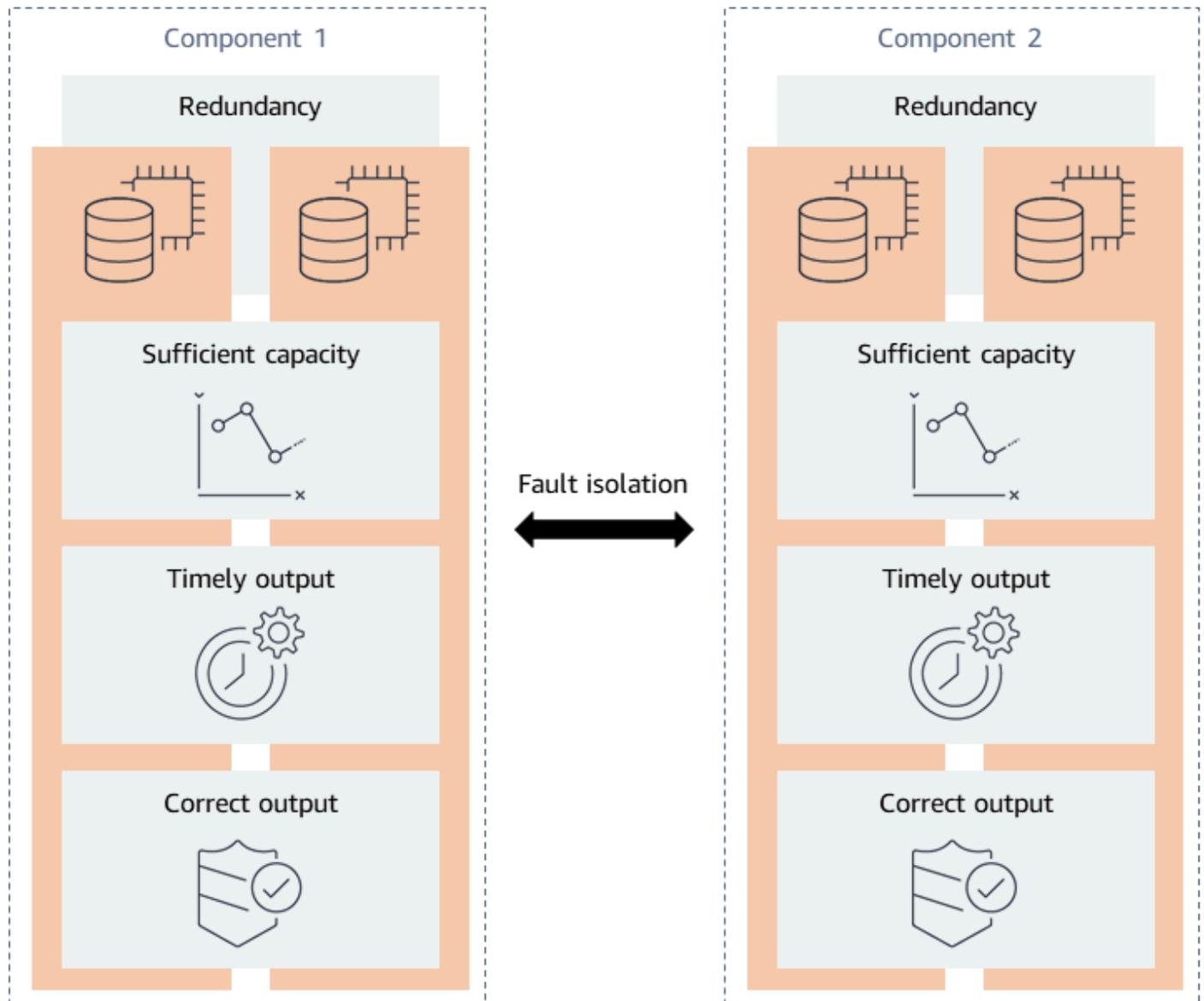
## 目次

- [フレームワークの概要](#)
- [ワークロードの理解](#)
- [フレームワークの適用](#)

- [潜在的な障害の軽減](#)
- [結論とリソース](#)

## フレームワークの概要

レジリエンス分析フレームワークは、ワークロードに必要なレジリエンス特性を特定することで開発されました。望ましいプロパティとは、システムについて真に当てはまることを指します。耐障害性は一般に可用性によって測定されるため、可用性の高い分散システムには、冗長性、十分な容量、タイムリーな出力、正しい出力、障害の切り分けという5つの特性があります。これらのプロパティを次の図に示します。



- 冗長性— 単一障害点 (SPOF) を排除する冗長性により、耐障害性が実現されます。冗長性は、ワークロード内のスペアコンポーネントから、アプリケーションスタック全体の完全なレプリカ

まで多岐にわたります。アプリケーションの冗長性を検討する際には、使用するインフラストラクチャ、データストア、依存関係によって提供される冗長性のレベルを考慮することが重要です。たとえば、Amazon DynamoDB と Amazon シンプルストレージサービス (Amazon S3) は、1 つのリージョン内の複数のアベイラビリティゾーンにデータを複製することで冗長性を実現し、AWS Lambda 複数のアベイラビリティゾーンの複数のワーカーノードで関数を実行します。使用する各サービスについて、そのサービスが提供するものと設計する必要があるものを考慮に入れてください。

- 十分な容量— ワークロードが意図したとおりに機能するには十分なリソースが必要です。リソースには、メモリ、CPU サイクル、スレッド、ストレージ、スループット、サービスクォータなど、さまざまなものがあります。
- タイムリーな出力— 顧客はワークロードを使用するときに、妥当な時間内に意図した機能を実行することを期待します。サービスがレイテンシーに関するサービスレベル契約 (SLA) を定めていない限り、顧客の期待は一般的に経験的証拠、つまり顧客自身の経験に基づいています。これは平均的な顧客体験通常、システム内のレイテンシの中央値 (P50) とみなされます。ワークロードに予想よりも時間がかかる場合、このレイテンシーはカスタマーエクスペリエンスに影響を及ぼす可能性があります。
- 正しい出力— ワークロードが意図した機能を実現するには、ワークロードのソフトウェアの正しい出力が必要です。不正確または不完全な結果は、まったく応答がないより悪い場合があります。
- 障害の切り分け— 障害分離は、障害発生時の影響範囲を意図した障害コンテナに限定します。これにより、ワークロードの特定のコンポーネントが同時に障害を起こし、障害が意図しない他のコンポーネントに連鎖的に影響することを防ぎます。また、ワークロードの顧客への影響範囲を限定するのにも役立ちます。障害の切り分けは、障害がすでに発生していても封じ込める必要があることを受け入れるため、前述の 4 つのプロパティとは多少異なります。インフラストラクチャ、依存関係、ソフトウェア機能から障害分離を行うことができます。

目的のプロパティに違反すると、ワークロードが使用できなくなったり、使用できなくなったりする可能性があります。これらの望ましいレジリエンス特性と、当社が多数の製品を扱ってきた経験に基づいています。AWS お客様から、シングルポイント障害、過剰負荷、過度の遅延、設定ミスやバグ、および運命の共有という 5 つの一般的な障害カテゴリを特定しました。これらを省略して SEEMS と呼んでいます。これらは潜在的な障害モードを分類するための一貫した方法であり、以下の表で説明しています。

障害カテゴリ	違反	定義
シングルポイント障害 (SPOF)	冗長性	1 つのコンポーネントに障害が発生すると、そのコンポー

ネットの冗長性が失われるため、システムが停止します。

過剰な負荷

十分な容量

過剰な需要やトラフィックによってリソースが過剰に消費されると、そのリソースは期待した機能を果たせなくなります。これには、制限やクォータに達することが含まれ、それによってスロットリングやリクエストの拒否が発生することがあります。

過度のレイテンシー

タイムリーな出力

システム処理またはネットワークトラフィックの遅延が、予想時間、サービスレベル目標 (SLO)、またはサービスレベル契約 (SLA) を超えている。

設定ミスやバグ

正しい出力

ソフトウェアのバグやシステムの設定ミスにより、出力が不正確になります。

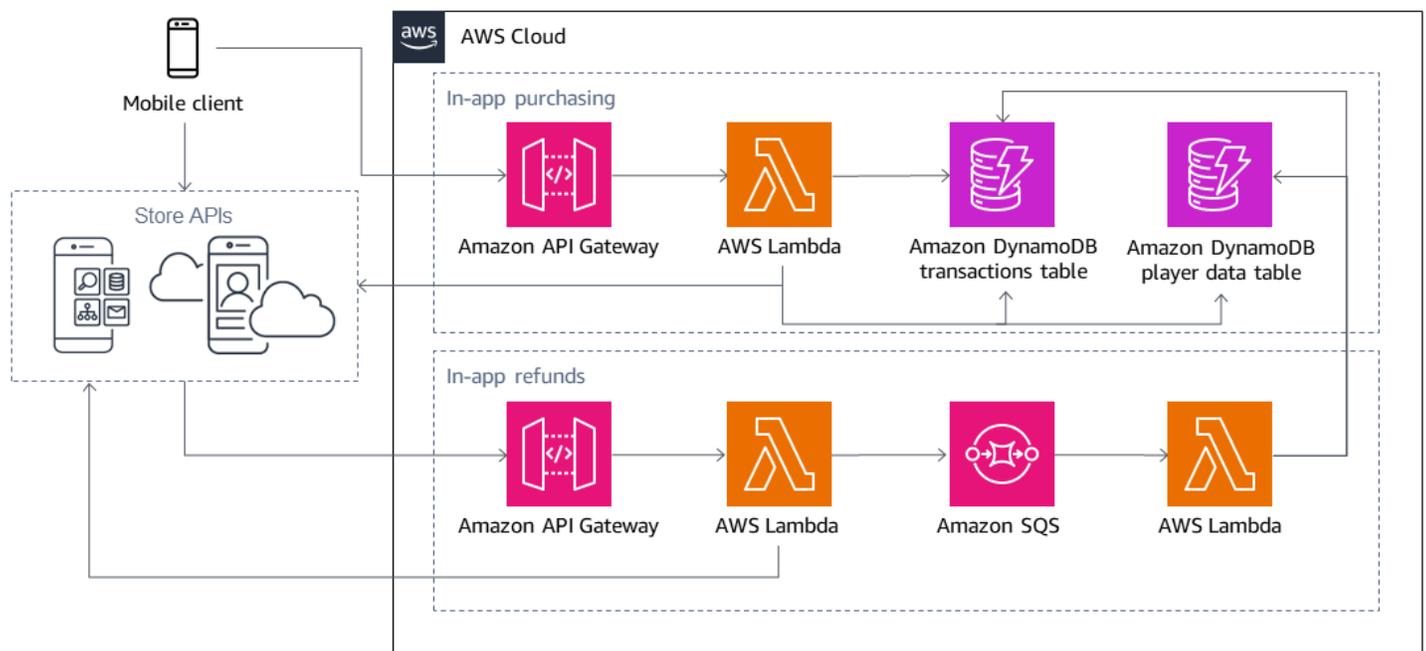
日付を共有

障害分離

前述の障害カテゴリのいずれかが原因で発生した障害は、意図した障害分離の境界を越え、システムの他の部分や他の顧客へと波及します。

## ワークロードの理解

フレームワークを適用するには、まず分析したいワークロードを理解することから始めます。システムアーキテクチャ図は、システムの最も関連性の高い詳細を文書化するための出発点となります。ただし、多くのシステムには多数のコンポーネントと相互作用があるため、ワークロード全体を分析しようとすると複雑になる可能性があります。代わりに、次のことに重点を置くことをお勧めします。ユーザーストーリー、これらはエンドユーザーの視点から書かれたソフトウェア機能の非公式で一般的な説明。その目的は、ソフトウェア機能がどのように顧客に価値をもたらすかを明確にすることです。次に、これらのユーザーストーリーをアーキテクチャ図とデータフロー図でモデル化すると、説明されているビジネス機能を提供する技術コンポーネントを簡単に評価できます。たとえば、アプリ内モバイルゲーム購入ソリューションには、次の図に示すように「アプリ内クレジットの購入」と「アプリ内払い戻しの取得」という2つのユーザーストーリーがある場合があります。(このアーキテクチャ例は、システムをユーザーストーリーに分解する方法を強調しています。耐障害性の高いアプリケーションを表すことを意図したものではありません。)



各ユーザーストーリーは、コードと構成、インフラストラクチャ、データストア、外部依存関係という4つの共通コンポーネントで構成されています。図にはこれらすべてのコンポーネントが含まれ、コンポーネント間の相互作用が反映されている必要があります。たとえば、Amazon API Gateway エンドポイントに過剰な負荷がかかっている場合は、その負荷がシステム内の他のコンポーネント(次のようなコンポーネント)にどのように連鎖するかを検討してください。AWS Lambda関数または Amazon DynamoDB テーブル。これらのインタラクションを追跡すると、障害モードがユーザーストーリーにどのような影響を与えるかを理解するのに役立ちます。このフロー

は、データフロー図で視覚的に把握することも、前の図のようにアーキテクチャ図で単純なフロー矢印を使用することもできます。各コンポーネントについて、送信される情報の種類、受信した情報、通信が同期か非同期か、どの障害境界を越えているかなどの詳細を把握することを検討してください。この例では、DynamoDB テーブルは両方のユーザーストーリーで共有されています。矢印からわかるように、アプリ内返金ストーリーの Lambda コンポーネントがアプリ内購入ストーリーの DynamoDB テーブルにアクセスしていることがわかります。つまり、運命を共有した結果、アプリ内購入のユーザーストーリーによって引き起こされた障害がアプリ内返金ユーザーストーリーに連鎖的に影響する可能性があります。

さらに、各コンポーネントのベースライン構成を理解することも重要です。ベースライン構成では、1 秒あたりの平均トランザクション数と最大トランザクション数、ペイロードの最大サイズ、クライアントのタイムアウト、リソースのデフォルトまたは現在のサービスクォータなどの制約を特定します。新しい設計をモデル化する場合は、設計の機能要件を文書化し、制限を検討することをお勧めします。これにより、故障モードがコンポーネントにどのような形で現れるかを理解しやすくなります。

最後に、提供するビジネス価値に基づいてユーザーストーリーに優先順位を付ける必要があります。この優先順位付けは、ワークロードの最も重要な機能に最初に集中するのに役立ちます。そうすれば、その機能のクリティカルパスに含まれるワークロードコンポーネントに分析を集中させることができ、フレームワークをより早く活用することで得られる価値を実感できます。プロセスを繰り返していくうちに、優先順位の異なる他のユーザーストーリーを調べることができます。

## フレームワークの適用

レジリエンス分析フレームワークを適用する最善の方法は、分析対象のユーザーストーリーの各要素について質問すべき、障害カテゴリ別に整理された標準的な質問セットから始めることです。ワークロードのすべてのコンポーネントに当てはまらない質問がある場合は、最も当てはまる質問を使用してください。

障害モードについては、次の2つの観点から考えることができます。

- 障害は、ユーザーストーリーをサポートするコンポーネントの能力にどのような影響を与えますか？
- 障害はコンポーネントと他のコンポーネントとの相互作用にどのような影響を及ぼしますか？

たとえば、データストアと過剰な負荷を考慮すると、データベースに過剰な負荷がかかってクエリがタイムアウトする障害モードについて考えるかもしれません。また、データベースクライアントが再試行でデータベースに負荷をかけたり、データベース接続を閉じられなかったりして、接続プールを使い果たしてしまうことも考えてみてください。もう1つの例として、複数のステップからなる認証プロセスがあります。多要素認証 (MFA) アプリケーションやサードパーティ ID プロバイダー (IdP) の障害が、この認証システムのユーザーストーリーにどのような影響を与えるかを考える必要があります。

以下の質問に答える際には、障害の原因を検討する必要があります。たとえば、過負荷の原因は顧客の急増によるものなのか、それとも保守作業中に多数のノードを稼働停止させた人間のオペレーターによるものなのか。それぞれの質問で複数の障害の原因を特定できる可能性があり、それには異なる緩和策が必要となる可能性があります。質問をする際には、発見した潜在的な障害モード、それらがどのコンポーネントに当てはまるか、各障害の原因を記録しておいてください。

### 単一障害点

- コンポーネントは冗長性を考慮して設計されていますか？
- コンポーネントに障害が発生した場合はどうなりますか？
- 1つのアベイラビリティゾーンが部分的または全体的に失われても、アプリケーションは許容できますか？

### レイテンシーが大きすぎる

- このコンポーネントのレイテンシーが増加したり、相互作用するコンポーネントのレイテンシーが増加した (または TCP リセットなどのネットワーク中断) 場合はどうなりますか？
- リトライストラテジーを使用してタイムアウトを適切に設定していますか？
- 失敗は速いのか、遅いのか。障害の発生が早いために、障害のあるリソースにすべてのトラフィックを意図せず送信してしまう、といった連鎖的な影響はないか。
- このコンポーネントに対して最も高額なリクエストは何か？

### 過剰な負荷

- このコンポーネントに負荷がかかるものは何でしょうか。このコンポーネントが他のコンポーネントを圧倒する原因は何か。
- 決して成功しない仕事にリソースを浪費することを防ぐにはどうすればよいか。
- そのコンポーネント用に構成された回路ブレーカーはありますか？
- 乗り越えられない未処理分を発生させるものはありますか？
- このコンポーネントでバイモーダルな挙動が発生する場所はどこか？
- どの制限またはサービスクォータを超えることができるか (ストレージ容量を含む)？
- 負荷がかかると、コンポーネントはどのようにスケーリングされるのか。

### 設定ミスとバグ

- 設定ミスやバグが本番環境にデプロイされるのを防ぐにはどうすればよいでしょうか？
- 不適切なデプロイを自動的にロールバックしたり、更新や変更がデプロイされたフォールトコンテナからトラフィックを遠ざけたりすることはできますか？
- オペレーターのミスを防ぐためにどのようなガードレールを設置していますか？
- どのような項目 (認証情報や証明書など) に有効期限が切れますか？

### 運命を共有

- 障害隔離の境界は？
- デプロイメントユニットに加えられた変更は、少なくとも意図したとおり小さいですか？ [障害隔離の境界](#)しかし、理想的にはワンボックス環境 (障害分離境界内の 1 つのインスタンス) のような小規模な環境でしょうか？
- このコンポーネントはユーザーストーリーや他のワークロードで共有されていますか？

- このコンポーネントと密接に結びついているコンポーネントは他にありますか？
- このコンポーネントやその依存関係に部分的な障害やグレー障害が発生した場合はどうなるでしょうか。

これらの質問をした後、SEEMS を使用して、ワークロードや各コンポーネントに固有の他の質問を作成することもできます。SEEMSは、故障モードについて体系的に考える方法として、またレジリエンス分析を行う際のインスピレーションの源として使うのが一番です。これは厳密な分類法ではありません。特定の障害モードがどのカテゴリに当てはまるか心配する必要はありません。重要ではありません。何？です大事なのは失敗を思い浮かべて書き留めたこと。間違った答えはありません。創造的で既成概念にとらわれずに考えることは有益です。さらに、障害モードがすでに緩和されていると思えないでください。考えられるすべての潜在的な障害モードを含めてください。

最初の演習では、潜在的な故障モードをすべて予測することはまずありません。フレームワークを複数回繰り返すことで、より完全なモデルを生成できるため、最初の段階ですべてを解決しようとする必要はありません。分析は定期的、週単位、または隔週単位で実行できます。各セッションでは、特定の故障モードまたはコンポーネントに焦点を当てます。これにより、ワークロードの耐障害性の向上を着実かつ段階的に進めることができます。ユーザーストーリーの潜在的な障害モードのリストを収集したら、その対処方法を決定できます。

## 潜在的な障害の軽減

ユーザーストーリーのコンポーネントに潜在的な障害が発生したので、緩和に集中できます。まず、発見した各障害の潜在的な影響と可能性に関連して、潜在的なトレードオフを確認します。次に、必要なオブザーバビリティレベルを決定し、緩和戦略を選択します。トレードオフには、適切なレベルのオブザーバビリティと緩和戦略を測定するための労力を含める必要があります。最後に、定期的なレジリエンス分析レビューを実施するための適切なサイクルを決定します。

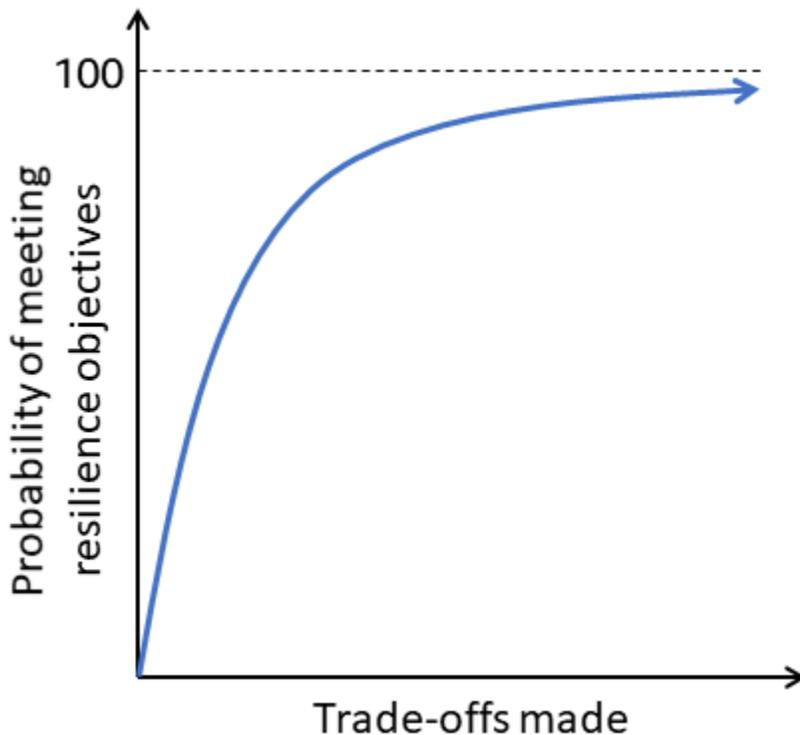
### セクション

- [トレードオフとリスクを理解する](#)
- [障害モードオブザーバビリティ](#)
- [一般的な緩和戦略](#)
- [継続的な改善](#)

## トレードオフとリスクを理解する

回復力のあるアーキテクチャでは、障害に対応するために、十分にテストされ、シンプルで信頼性の高いメカニズムを数多く使用する必要があります。最高レベルの耐障害性を実現するには、ワークロードが可能な限り多くの障害モードを自動的に検出して復旧する必要があります。これを行うには、レジリエンス分析の実行に広範な投資が必要です。つまり、より高いレベルのレジリエンスを実現するには、トレードオフが必要です。ただし、トレードオフを続けると、レジリエンス目標と比較して収益が減少する時点に達します。最も一般的なトレードオフは次のとおりです。

- コスト — 冗長コンポーネント、オブザーバビリティの強化、追加のツール、またはリソース使用率の増加により、コストが増加します。
- システムの複雑さ — 緩和ソリューションなどの障害モードを検出して対応し、マネージドサービスを使用しない場合、システムの複雑さが増します。
- エンジニアリング作業 — 障害モードを検出して対応するためのソリューションを構築するには、追加の開発者時間が必要です。
- 運用オーバーヘッド — より多くの障害モードを処理するシステムをモニタリングして運用すると、特にマネージドサービスを使用して特定の障害モードを軽減できない場合に、運用オーバーヘッドが発生する可能性があります。
- レイテンシーと一貫性 — 可用性を優先する分散システムを構築するには、[PACELC 定理](#) で説明されているように、一貫性とレイテンシーのトレードオフが必要です。



ユーザーストーリーで特定された障害モードの緩和策を検討する際には、必要なトレードオフを検討してください。セキュリティと同様に、レジリエンスは最適化の問題です。特定された障害によってもたらされるリスクを回避、軽減、転送、または受け入れるかどうかを決定する必要があります。回避できる障害モード、受け入れるセット、転送できる障害モードがいくつかあります。特定した障害モードの多くを軽減することもできます。実行するアプローチを決定するには、次の2つの質問をして評価を実行します。障害が発生する可能性はどれくらいですか？ワークロードが発生した場合は、どのような影響がありますか？

ライフは、イベントが発生することに対する懸念です。例えば、ユーザーストーリーに単一の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスで動作するコンポーネントがある場合、パッチ適用手順やオペレーティングシステムエラーなどが原因で、システムのオペレーション中にコンポーネントが中断される可能性があります。または、プライマリインスタンスとセカンダリインスタンス間でデータを同期する Amazon Relational Database Service (Amazon RDS) によって管理されるデータベースは、完全に使用できなくなる可能性が低くなります。

影響は、イベントが引き起こす可能性のある悪影響の推定値です。これは、財務的および評判的の両方の観点から評価される必要があり、それが影響するユーザーストーリーの価値を基準にしています。例えば、処理負荷の高いデータベースは、eコマースシステムの新しい注文を受け入れる能力に大きな影響を与える可能性があります。ただし、ロードバランサーの背後にある20個のインスタンスのフリートから1つのインスタンスが失われても、ほとんど影響がない可能性があります。

これらの質問に対する回答を、リスクを軽減するために実行する必要があるトレードオフのコストと比較できます。この情報をリスクしきい値と耐障害性目標の観点から考慮すると、どの障害モードを積極的に緩和するかを決定するのに役立ちます。

## 障害モードオブザーバビリティ

障害モードを軽減するには、まず、ワークロードに現在影響している、または影響しようとしていることを検出する必要があります。緩和は、アクションを実行する必要があるというシグナルがある場合にのみ有効です。つまり、緩和策の作成には、少なくとも、障害の影響を検出するために必要なオブザーバビリティを構築していることの確認が含まれます。

2つの次元で障害モードの監視可能な症状を考慮する必要があります。

- システムが間もなく影響が見られる可能性がある条件に近づいていることを知らせる主要な指標は何ですか？
- 障害モードが発生した後、できるだけ早く障害モードの影響を示すことができる遅延インジケータは何ですか？

例えば、データベース要素に適用される過剰なロード障害は、主要なインジケータとして接続数を持つ可能性があります。データベースが接続制限をすぐに超える可能性があることを示す主要な指標として、接続数の着実な増加を確認できます。これにより、最近使用されていない接続を終了するなどのアクションを実行して、接続数を減らすことができます。遅延インジケータは、データベース接続の制限を超えたときと、データベース接続エラーが昇格したことを示します。アプリケーションとインフラストラクチャのメトリクスを収集するだけでなく、[主要な業績評価指標 \(KPI\)](#) を収集して、障害がカスタマーエクスペリエンスに与える影響を検出することを検討してください。

可能であれば、オブザーバビリティ戦略に両方のタイプのインジケータを含めることをお勧めします。場合によっては、先頭のインジケータを作成できないことがあります。緩和する障害ごとに常に遅延インジケータを設定するように計画する必要があります。適切な緩和策を選択するには、先頭または遅延インジケータが障害を検出したかどうかも考慮する必要があります。例えば、ウェブサイトへのトラフィックが突然急増するとします。遅延インジケータのみが表示される可能性があります。この場合、自動スケーリングだけでは新しいリソースのデプロイに時間がかかるため、最適な緩和策ではない可能性があります。一方、スロットリングによって過負荷がほぼ即時に防止され、アプリケーションの負荷をスケーリングまたは軽減する時間を与えることができます。逆に、トラフィックが徐々に増加すると、先頭のインジケータが表示されます。この場合、システムを自動的にスケーリングして応答する時間があるため、スロットリングは適切ではありません。

## 一般的な緩和戦略

まず、予防的緩和策を使用して、障害モードがユーザーストーリーに影響を与えないようにすることを検討してください。次に、是正措置を検討する必要があります。正しい緩和策は、システムの自己修復や変化する条件への適応に役立ちます。耐障害性プロパティに沿った各障害カテゴリの一般的な緩和策のリストを次に示します。

失敗カテゴリ	希望する耐障害性プロパティ	緩和策
単一障害点 (SPOFs)	冗長性と耐障害性	<ul style="list-style-type: none"> <li>例えば、Elastic Load Balancing (ELB) の背後にある複数の EC2 インスタンスを使用するなどして、<a href="#">冗長性</a>を実装します。</li> <li><a href="#">AWS グローバルサービスコントロールプレーン</a>の依存関係を削除し、グローバルサービスデータプレーンにのみ依存関係を作成します。</li> <li>リソースが利用できない場合には<a href="#">グレースなパフォーマンス低下</a>を使用してください。これにより、システムは単一障害点に対して静的に安定します。</li> </ul>
過剰な負荷	十分な容量	<ul style="list-style-type: none"> <li>主な緩和戦略は、<a href="#">レート制限</a>、<a href="#">負荷分散</a>と作業の優先順位付け、<a href="#">定常作業</a>、<a href="#">ジッタまたは再試行なしによるエクスポネンシャルバックオフと再試行</a>、<a href="#">小さいサービスの管理</a>、<a href="#">キューの深さの管理</a>、<a href="#">自動スケーリング</a> <a href="https://aws.amazon.com/autoscaling/">https://aws.amazon.com/autoscaling/</a>、<a href="#">コールドキャッ</a></li> </ul>

## シユの回避、サーキットブレーカー

- また、キャパシティプランを検討し、システム内の AWS リソースと制限に関連する将来のキャパシティとスケーリング制限がヒットする可能性があることを考慮する必要があります。

## 過剰なレイテンシー

## タイムリー出力

- 適切に設定された[タイムアウト](#)またはアダプティブタイムアウトを実装します (現在のレイテンシー条件と予測されるレイテンシー条件に基づいてタイムアウト値を変更することで、遅いリクエストを中止する代わりに、遅い依存関係が進行する可能性があります)。
- オンプレミス環境からクラウドサービスに接続し、[特定のルートでレイテンシーが発生するときにマルチパス TCP などのテクノロジーを使用し、疎結合システムとの非同期インタラクション、キャッシュを使用して、作業を破棄しないように、ジッターでエクスポネンシャルバックオフと再試行を実装します。](#) [https://en.wikipedia.org/wiki/Multipath\\_TCP](https://en.wikipedia.org/wiki/Multipath_TCP) [https://docs.aws.amazon.com/wellarchitected/latest/framework/rel\\_posely\\_coupled\\_system.html](https://docs.aws.amazon.com/wellarchitected/latest/framework/rel_posely_coupled_system.html) <https://aws.amazon.com/builders-library/caching-challenges-and-strategies> <https://aws.amazon.com/builders-library/using-load-shedding-to-avoid-overload/>

## 設定ミスとバグ

## 正しい出力

- ソフトウェアで反復可能な機能エラーを検出する主な方法は、[静的分析](#)、[ユニットテスト](#)、[統合テスト](#)、[回帰テスト](#)、[負荷テスト](#)、[耐障害性テスト](#)などのメカニズムによる厳格なテストです。 [https://en.wikipedia.org/wiki/Unit\\_testing](https://en.wikipedia.org/wiki/Unit_testing)
- [Infrastructure as Code \(IaC\)](#) や [継続的インテグレーション](#) および [継続的デリバリー \(CI/CD\) 自動化](#) などの戦略を実装して、設定ミスの脅威を軽減します。
- [1つのボックス](#)、[Canary デプロイ](#)、[障害分離境界に合わせた小数のデプロイ](#)、[ブルー/グリーンデプロイ](#) などのデプロイ手法を使用して、設定ミスやバグを減らします。

## 共有フェード

## 障害分離

- システムに[耐障害性](#)を実装し、複数のコンピューティングクラスターやコンテナクラスター、複数のAWS アカウント、複数のAWS Identity and Access Management (IAM) プリンシパル、複数のアベイラビリティゾーン、場合によっては複数の など、論理的および物理的な障害分離境界を使用しますAWS リージョン。
- [セルベースのアーキテクチャ](#)や[シャッフルシャーディング](#)などの手法により、障害の分離も向上する可能性があります。
- カスケード障害を防ぐために、[疎結合](#)や[グレースダウン](#)などのパターンを検討してください。ユーザーストーリーに優先順位を付けるときは、その優先順位を使用して、主要なビジネス機能に不可欠なユーザーストーリーと、適切に低下する可能性のあるユーザーストーリーを区別することもできます。例えば、e コマースサイトでは、ウェブサイトのプロモーションウィジェットに障害が発生しても、新しい注文を処

理する能力には影響しません。

これらの緩和策の中には、実装に最小限の労力を必要とするものもありますが、特定のユーザーストーリーのコンポーネントだけでなく、ワークロード全体の再設計が必要になるものもあります(たとえば、予測可能な障害分離のためのセルベースのアーキテクチャを採用したり、フェルト失敗を最小限に抑えたりするなど)。前述のように、障害モードの可能性と影響を、その軽減のために行ったトレードオフと比較することが重要です。

各障害モードカテゴリに適用される緩和方法に加えて、ユーザーストーリーまたはシステム全体の復旧に必要な緩和策を検討する必要があります。例えば、障害が発生するとワークフローが停止し、目的の宛先へのデータの書き込みが妨げられる可能性があります。この場合、ワークフローをリドライブしたり、データを手動で修正したりするために、運用ツールが必要になる場合があります。また、障害発生時のデータ損失を防ぐために、ワークロードにチェックポイントメカニズムを構築する必要がある場合もあります。または、ワークフローを一時停止し、さらなる損害を防ぐために新しい作業の受け入れを停止するために、このコードを構築する必要がある場合があります。このような場合は、必要な運用ツールとガードレールを検討する必要があります。

最後に、緩和戦略を策定する際に、人間が間違えることを常に想定する必要があります。最新の DevOps プラクティスでは運用を自動化しようとしていますが、人間はさまざまな理由でワークロードと対話する必要があります。メンテナンス中のノードの削除が多すぎたり、過負荷になったり、機能フラグを誤って設定したりするなど、SEEMS カテゴリのいずれかで誤った人為的なアクションにより障害が発生する可能性があります。これらのシナリオは、予防的ガードレールの障害です。根本原因分析は、「人間が間違えた」という結論で終わるべきではありません。代わりに、そもそもミスがあった理由に対処する必要があります。したがって、緩和戦略では、人間のオペレーターがワークロードコンポーネントとやり取りする方法と、安全ガードレールを通じて人間のオペレーターのミスによる影響を防止または最小限に抑える方法を考慮する必要があります。

## 継続的な改善

耐障害性は[継続的なプロセス](#)です。システムのライフサイクルを通じて、それが動作する環境が変更されます。システムの耐障害性を維持するには、フレームワークを定期的に運用およびアーキテクチャレビューに統合する必要があります。初めて識別しなかった新しい障害モードや、導入できる新しい緩和策または以前に仮定していない緩和策が見つかる場合があります。レジリエンス分析は反復的なプロセスであり、1 回限りの演習ではありません。

[カオスエンジニアリング](#)や[ゲームデー](#)などのプロセスで緩和戦略を経験的にテストし、期待どおりに機能することを検証する必要があります。厳格なテストメカニズムがない場合は、必要なときに緩和策が期待どおりに機能することが保証されません。耐障害性分析中に、障害モードが既に特定の緩和策によって処理されていると判断する場合がありますが、それらの仮定もテストすることが重要です。耐障害性分析フレームワークを使用して作成された既存の緩和策と新しい緩和策の両方をテストする必要があります。

また、チームの遡及的な経験を通じて分析をどの程度うまく実行したかを評価する必要があります。分析中に自分が何に取り組んでいるかは、全員がわかりますか？ 耐障害性分析で見つかった障害モードの数は、チームの期待と一致していますか？ 検出されたすべての障害モードの緩和策を特定できますか？ チームはこのプロセスを役に立ちますか？ ワークロードの耐障害性の向上につながると考えられますか？

ワークロードの可用性に影響する実際の障害イベントが発生した場合は、特定の障害モード、障害の一部であったコンポーネント、使用された緩和パターンを記録します。インシデント後分析ツールでこのメタデータを検索可能にすると、今後どの障害モードとコンポーネントに集中すべきかを判断できます。このプロセスを通じて、AWSアカウントチームやソリューションアーキテクトとやり取りすることができます。

## 結論とリソース

このガイドでは、レジリエンス分析を継続的かつ一貫性のある方法で実施するためのフレームワークを紹介しています。このフレームワークは、単一障害点、過剰な負荷、過度の遅延、設定ミスやバグ、そして共通の運命がワークロードの構成要素にどのように影響するかを特定するのに役立ちます。これらの障害モードを特定することで、回復指向のアーキテクチャを構築する一環として適切な緩和戦略を決定できます。

レジリエンス分析に関するその他の資料については、以下のリンクを参照してください。

- [レジリエンス・ライフサイクル・フレームワーク](#) (AWS規範的ガイド)
- [レジリエンス向けソリューション](#) (ソリューションライブラリ) AWS
- [継続的なレジリエンスに向けて](#) (エイドリアン・ホーンズビー、クラウドアーキテクト、2021年3月24日)

## ドキュメント履歴

このガイドは、このドキュメントの大きな変更点をまとめたものです。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
<a href="#">初版発行</a>	—	2023 年 9 月 5 日

# AWS 規範的ガイドの用語集

以下は、AWS 規範的ガイドが提供する戦略、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: オンプレミスの Oracle データベースをの 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。サーバーを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、およびそれらを行き移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

[「原子性、一貫性、分離性、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。アクティブ/[パッシブ移行](#)よりも柔軟ですが、より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

行のグループを操作し、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 MAXなどがあります。

### 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。

## アベイラビリティゾーン

他のアベイラビリティゾーンの障害から 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)

組織がクラウドに正常に移行 AWS するための効率的で効果的な計画を立てるのに役立つ、から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。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編成します。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、組織が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

個人または組織に混乱や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。[エンディアンネス](#)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一つ) 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は 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は他の環境 (緑) で実行します。この戦略は、影響を最小限に抑えながら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インタラクション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

トの中には、個人や組織に混乱を与えたり、損害を与えたりすることを意図しているものがあります。

## ポットネット

[マルウェア](#)に感染し、[ポット](#)のヘルダーまたはポットオペレーターと呼ばれる、単一関係者の管理下にあるポットのネットワーク。ポットは、ポ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、「[ブランチについて](#) (GitHub ドキュメント)」を参照してください。

## ブレイクグラスアクセス

例外的な状況や承認されたプロセスを通じて、ユーザーが通常アクセス許可を持たない AWS アカウント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、Well-Architected [ガイド](#)の「[ブレイ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低速かつ増分的なリリース。確信できたら、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 ワークロードに負荷をかけ、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 サービス を受信する前に、ローカルでデータを暗号化します。

## Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、AWS クラウド エンタープライズ戦略ブログの[CCoE の投稿](#)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 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。移行戦略とどのように関連しているかについては、AWS [「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 AWS CodeCommit。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ます。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、はオンプレミスカメラネットワークに CV を追加するデバイス AWS Panorama を提供し、Amazon SageMaker は CV 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、設定は想定した状態から変化します。これにより、ワークロードが非標準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、または組織全体に単一のエンティティとしてデプロイできます。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

一元化された管理とガバナンスにより、分散型の分散型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。信頼できる ID のみが、期待されるネットワークから信頼できるリソースにアクセスしていることを確認できます。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、クエリや分析によく使用されます。

## 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。

## defense-in-depth

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。この戦略をに採用するときは 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。例えば、defense-in-depth 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で利用できるサービス](#)を参照してください。

### デプロイメント

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ユファクチャリング・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の追跡。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を与える可能性のある [ランディングゾーンの変更を検出したり](#) できます。

## DVSM

[「開発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。

### 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。

### 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 (アカウンティング、[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利用できる、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](#) の中央テーブル。事業運営に関する定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 種類の列が含まれます。

### フェイルファスト

頻繁で段階的なテストを使用して開発ライフサイクルを短縮する哲学。これはアジャイルアプローチの重要な部分です。

### 障害分離境界

では AWS クラウド、障害の影響を制限し AWS リージョン、ワークロードの耐障害性を向上させるアベイラビリティゾーン、コントロールプレーン、データプレーンなどの境界です。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。

### 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のではなく、[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。

## G

### 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)

Amazon では CloudFront、特定の国のユーザーがコンテンツディストリビューションにアクセスできないようにするオプションです。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。

### 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 AWS Security Hub、GuardDuty、Amazon Inspector AWS Trusted Advisor、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。

#### 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。緊急性のため、通常、修正は一般的な DevOps リリースワークフローの外で行われます。

## 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、本番ワークロード用の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[「変更不可能な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

接続、リアルタイムデータ、自動化、分析、AI/ML の進歩を通じて、のビジネスプロセスのモダナイゼーションを指すために 2016 年に [Klaus Schwab](#) によって導入された用語。

## 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 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、「[AWS を使用した機械学習モデルの解釈](#)」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

[「7R」](#) 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。[エンディアンネス](#) も参照してください。

## 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ているソフトウェア。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 サービスが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、ユーザーが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。これにより、加工品を現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、調整のために結果を検査する完全なプロセス。メカニズムとは、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

内の組織の一部である管理アカウント AWS アカウントを除くすべての 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の 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)

組織がクラウドへの移行のための強固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、オペレーション、ビジネスアナリストと所有者、移行エンジニア、デベロッパー、スプリントに取り組む 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には、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: Application Migration Service を使用して Amazon EC2 AWS 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されるアプローチ 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。

## 変更可能な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 Framework では、一貫性、信頼性、予測可能性を向上させるために、[イミュータブルな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「[運用準備レビュー \(ORR\)](#)」を参照してください。

## 運用テクノロジー (OT)

産業運用、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 と情報テクノロジー (IT) システムの統合が、[Industry 4.0](#) トランスフォーメーションの主要な焦点です。

## 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

の組織 AWS アカウント 内のすべての のすべてのイベントをログ AWS CloudTrail 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、ドキュメントの[「組織の証跡の作成」](#)を参照してください。CloudTrail

### 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から、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)

では CloudFront、Amazon Simple Storage Service (Amazon S3) コンテンツを保護するためのアクセスを制限するための拡張オプションです。OAC は、すべての のすべての S3 バケット AWS リージョン、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)

では CloudFront、Amazon S3 コンテンツを保護するためのアクセスを制限するオプションです。OAI を使用すると、は Amazon S3 が認証できるプリンシパル CloudFront を作成します。認証されたプリンシパルは、特定の CloudFront ディストリビューションを介してのみ S3 バケット内のコンテンツにアクセスできます。[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

[「運用技術」](#)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る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

アクセス許可の定義 ([アイデンティティベースのポリシーを参照](#))、アクセス条件の指定 ([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の定義 AWS Organizations ([サービスコントロールポリシーを参照](#)) が可能なオブジェクト。

## 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。false 通常は 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 ロール AWS アカウント、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

エンジニアリングプロセス全体を通してプライバシーを考慮に入れた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

非準拠のリソースのデプロイを防止するように設計された[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「[での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、成熟、縮小、削除まで、ライフサイクル全体にわたる製品のデータとプロセスの管理。

## 本番環境

[「環境」](#)を参照してください。

## プログラミング可能ロジックコントローラー (NAL)

製造では、マシンをモニタリングし、承認プロセスを自動化する、信頼性が高く、適応性の高いコンピュータです。

## 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してスケーラビリティと応答性を向上させるパターン。例えば、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ン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す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、情報 \(RACI\)](#) 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

読み取り専用で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。

### 再構築

[「7R」](#) 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)

サービスが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

[「7R」](#) 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、[AWS リージョン「を使用できるアカウントを指定する」](#)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HOST

[「7 R」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 R」を参照してください。](#)

## 再購入

[「7 R」を参照してください。](#)

## 回復性

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機能。で障害耐性を計画する場合、[高可用性](#)と[ディザスタリカバリ](#)が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。

## レスポンスコントロール

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

[「7R」を参照してください。](#)

### 廃止

[「7R」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 (IdPs) が使用するオープンスタンダード。この機能により、フェデレーテ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 [AWS](#)

Management Console したり、組織内のすべてのユーザーを IAM で作成しなくても 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収集」](#)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

では AWS Secrets Manager、暗号化された形式で保存する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。

## 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正するように設計された、事前定義され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の実装に役立つ[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

シヨンの例としては、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、認証情報のローテーションなどがあります。

## サーバー側の暗号化

送信先にあるデータの、それを受け取る AWS サービス 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

のエンドポイントの URL AWS 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 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単一の重要なコンポーネントの障害。

## 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。

## split-and-seed モデル

モダナイゼーションプロジェクトのスケールアップ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。

## star スキーマ

トランザクションデータまたは測定データを保存するために1つの大きなファクトテーブルを使用し、データ属性を保存するために1つ以上の小さなディメンションテーブルを使用するデータベースの組織構造。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、宿主にと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ゾーンに存在する必要があります。

## 監視統制とデータ収集 (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生産オペレーション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

ユーザーインタラクション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。

## 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が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するサービスへの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と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 [AWS Organizations の「を他の AWS のサービスで使用する 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 つのピザを食べることができる小さな 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

[「書き込み 1 回」](#)を参照し、[多くの](#)を読み取ります。

## 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 [イミュータブルな](#) と見なされます。

## Z

### ゼロデイ 익스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。

## 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。