Amazon Neptune 用の AWS Well-Architected フレームワークの適用

AWS 規範ガイダンス

AWS 規範ガイダンス: Amazon Neptune 用の AWS Well-Architected フレームワークの適用

Table of Contents

Amazon Neptune 用の AWS Well-Architected フレームワークの適用

Amazon Web Services (寄稿者)

2023 年 9 月 (ドキュメント履歴)

Amazon Neptune を使用して、Amazon Web Services (AWS) でグラフベースのソリューションを構築できます。このガイドでは、Neptune デプロイを計画する際に AWS Well-Architected Framework の原則を適用するための規範的なガイダンスを提供します。Amazon Neptune Analytics 用の AWS Well-Architected フレームワークの適用には、Neptune グラフ分析エンジンの同じトピックが含まれます。

AWS Well-Architected フレームワークは、さまざまなアプリケーションやワークロード向けに、安全で高性能、耐障害性、効率的なインフラストラクチャを構築するのに役立ちます。また、アーキテクチャを評価し、スケーラブルな設計を実装するための一貫したアプローチも提供します。

AWS Well-Architected フレームワークは、次の 6 つの柱を中心に構築されています。

- オペレーショナルエクセレンス
- セキュリティ
- 信頼性
- パフォーマンス効率
- コストの最適化
- 持続可能性

このガイドでは、 Well-Architected Framework の設計の柱とベストプラクティス、および Neptune をデプロイする際に留意すべき考慮事項について説明します AWS。

対象者

このガイドは、グラフを使用するソリューションを設計および実装するデータエンジニア、ソリューションアーキテクト、データアナリストを対象としています AWS。

目的

このガイドは、ユーザーと組織が次のことを行うのに役立ちます。

- ユースケースとクエリパターンに基づいて、サポートされているデプロイオプションとクエリ言語から選択します。
- 耐障害性とセキュリティの向上に役立つ AWS Well-Architected 設計パターンに従ってください。
- 最適なパフォーマンスとコスト削減のためにクエリを設計します。
- Neptune クラスターを本番環境で管理する際に運用効率を高める方法について説明します。

運用上の優秀性の柱

AWS Well-Architected フレームワークの運用上の優秀性の柱は、システムの実行とモニタリング、プロセスと手順の継続的な改善に焦点を当てています。これには、開発をサポートし、ワークロードを効果的に実行し、運用に関するインサイトを得て、ビジネス価値を提供するためのサポートプロセスと手順を継続的に改善する機能が含まれます。自己修復ワークロードは、人間の介入なしにほとんどの問題を検出して修復することで、運用の複雑さを軽減できます。このセクションで説明されているベストプラクティスに従うことで、この目標を達成できます。Amazon Neptune メトリクス、APIs、メカニズムを使用して、ワークロードが予想される動作から逸脱したときに適切に対応します。

この運用上の優秀性の柱に関する説明では、以下の主要分野に焦点を当てています。

- Infrastructure as code (IaC)
- 変更管理
- レジリエンシー戦略
- インシデント管理
- コンプライアンスに関する監査レポート
- ログ記録とモニタリング

IaC アプローチを使用してデプロイを自動化する

IaC を使用して Neptune でのデプロイを自動化するためのベストプラクティスは次のとおりです。

- 可能な限り、Infrastructure as Code (IaC) を適用して Neptune クラスターをデプロイします。一貫した環境設定を行うには、 AWS CloudFormation テンプレート、AWS Cloud Development Kit (AWS CDK)、または HashiCorp Terraform を使用して、クラスターに必要なすべてのリソースを作成します。
- 可能な限り、インスタンスのサイズ変更、リードレプリカの追加または削除、グローバルテーブルでの手動フェイルオーバーなど、Neptune の運用手順を自動化します。
- 接続文字列をクライアントから外部に保存します。抽出、変換、ロード (ETL) プロセスを使用して、ブルー/グリーンデプロイ戦略、ディザスタリカバリ (DR)、新しいクラスターへのほぼゼロのダウンタイム移行を容易にします。接続文字列はAWS Secrets Manager、、Amazon DynamoDB、または動的に変更できる任意の場所に保存できます。

- タグを使用して Neptune リソースにメタデータを追加し、タグに基づいて使用状況を追跡します。詳細については、Amazon Neptune リソースのタグ付け」を参照してください。

頻繁で小さく、可逆的な変更を行う

以下の推奨事項は、複雑性を最小限に抑え、ワークロードの中断の可能性を減らすために、小規模で可逆的な変更に焦点を当てています。

- IaC テンプレートとスクリプトを GitHub や GitLab などのソースコントロールサービスに保存します。

> **Important**
>
> ソース管理に AWS 認証情報を保存しないでください。

- IaC デプロイでは、 AWS CodeDeployや などの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) サービスを使用する必要がありますAWS CodeBuild。これらのサービスは、本番環境の Amazon Neptune クラスターに影響を与える前に、一時的な Neptune クラスターを含む非本番環境でコードをコンパイル、テスト、デプロイします。
- インフラストラクチャとアプリケーションのクエリを本番環境にデプロイする前に、より低い環境でテストします。これにより、中断の可能性が最小限に抑えられ、ワークロードやスケールに合わせて適切に機能するようになります。

失敗を予測する

自己修復インフラストラクチャは、障害を予測し、介入なしで問題を解決しようとすることで、運用上の優秀性を実証します。以下の推奨事項は、Neptune でその成熟度を達成するのに役立ちます。

- Amazon CloudWatch メトリクスを使用して DB インスタンスの CPU とメモリの使用状況をモニタリングし、使用パターンを理解するモニタリングプランを作成します。アプリケーションログにある主要なメトリクスと Neptune クライアントレスポンスの CloudWatch ダッシュボードとアラームを作成します。CPU 使用率が高いまたは低い指標の詳細については、Neptune ドキュメントのCloudWatch を使用して Neptune で DB インスタンスのパフォーマンスをモニタリングする」を参照してください。

FreeableMemory が低いときにクエリでout-of-memory例外が頻繁に発生する場合は、X2 ファミリーのインスタンスの使用を検討してください。

- Neptune クラスターの状態をモニタリングする通知を設定します。例えば、BufferCacheHitRatioは常に高く (99.9% 超) 、 は常に低く (理想的には 0 ですが、要件とレイテンシー耐性によって異なります) MainRequestQueuePendingRequestsする必要があります。
- Neptune 内で高可用性を実現するには、リードレプリカの使用を検討してください。フェイルオーバーイベント中にインスタンスが読み取りクエリを処理できるように、ライターインスタンスとは異なるアベイラビリティーゾーンに少なくとも 2 つのリードレプリカが必要です。
- 使用率メトリクスに基づいてリードレプリカを自動的にスケーリングします。詳細については、Amazon Neptune DB クラスターのレプリカ数の自動スケーリング」を参照してください。
- DB インスタンスのフェイルオーバーをテストすることで、そのプロセスでユースケースにかかる時間を把握します。
- アプリケーションが完全に AWS リージョン 停止した場合、DR プランの一部としてグローバルデータベースを使用することを検討してください。

すべての運用上の障害から学ぶ

自己修復インフラストラクチャは、まれな問題が発生したり、対応が意図したほど効果的でなくなったりしたときに反復して発生する長期的な労力です。以下のプラクティスを採用することで、その目標に焦点を当てることができます。

- すべての障害から学習することで改善を推進します。
- チームや組織全体で学んだことを共有します。組織内の複数のチームが Neptune を使用している場合は、共通のチャットルームまたはユーザーグループを作成して、学習とベストプラクティスを共有します。

ログ記録機能を使用して、不正または異常なアクティビティをモニタリングする

異常なパフォーマンスとアクティビティのパターンを観察するには、Amazon CloudWatch Logs にログを保存します。以下のベストプラクティスを考慮します。

- スロークエリログ記録を有効にします。ログを定期的に確認し、特定のクエリが遅い理由を診断します。Gremlin、SPARQL、または openCypher の Neptune explain および profile エンドポイントを使用して、これらのクエリが遅い理由に関するインサイトを取得します。
- Neptune 監査ログを有効にし、ログに不正アクセスや異常がないか定期的に確認します。
- スロークエリログ記録または監査ログ記録を使用している場合は、CloudWatch Logs への発行を有効にします。これにより、インスタンスのディスク容量が不足するのを防ぐことができます。Neptune インスタンスのログストレージ容量は限られており、ログスペースを超えると古いログファイルが上書きされます。CloudWatch Logs は、ログの長期保持をサポートしています。CloudWatch Logs で強化されたモニタリング機能を使用すると、ログをクエリして問題を診断する機能が向上します。
- 監査ログの分析ツールを向上させるために、監査ログデータを CloudWatch Logs のロググループに発行するように Neptune DB クラスターを設定できます。CloudWatch Logs を使用すると、ログデータのリアルタイム分析、CloudWatch を使用したアラームの作成およびメトリクスの表示、CloudWatch Logs を使用した、耐久性に優れたストレージへのログレコードの保存を行うことができます。詳細については、Amazon CloudWatch Logs への Neptune ログの発行」を参照してください。
- Neptune は、 を使用したコントロールプレーンアクションのログ記録をサポートしています AWS CloudTrail。詳細については、「 を使用した Amazon Neptune API コールのログ記録 AWS CloudTrail」を参照してください。

セキュリティの柱

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ – AWS は、 AWS のサービス で実行されるインフラストラクチャを保護する責任があります AWS クラウド。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムの一環として、セキュリティの有効性 AWS を定期的にテストおよび検証します。Amazon Neptune に適用されるコンプライアンスプログラムについては、コンプライアンスプログラムによるAWS 対象範囲内のサービスを参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する によって決まり AWS のサービス ます。また、お客様は、 お客様のデータの機密性、企業の要件、および適用可能な法律や規制といった他の要因 についても責任を担います。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、 AWS 責任共有モデルと GDPR ブログ記事を参照してください。

AWS Well-Architected フレームワークのセキュリティの柱は、Neptune を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために Neptune を設定する方法を示します。また、Neptune リソースのモニタリングと保護 AWS のサービス に役立つ他の の使用方法についても説明します。

セキュリティの柱には、以下の主要な重点分野が含まれています。

- データセキュリティ
- ネットワークセキュリティ
- 認証と認可

データセキュリティの実装

データ漏洩や侵害は顧客を危険にさらし、会社に大きな悪影響を与える可能性があります。以下のベストプラクティスは、不注意や悪意のある漏洩から顧客データを保護するのに役立ちます。

- クラスター名、タグ、パラメータグループ、 AWS Identity and Access Management (IAM) ロール、およびその他のメタデータには、請求ログや診断ログに表示される可能性があるため、機密情報や機密情報を含めないでください。
- Neptune にデータとして保存されている外部サーバーへの URIs またはリンクには、リクエストを検証するための認証情報を含めないでください。
- Neptune の暗号化されたインスタンスは、基になるストレージへの不正アクセスからデータを保護することで、データ保護のレイヤーを追加します。Neptune の暗号化を使用すると、クラウドにデプロイされたアプリケーションのデータ保護を強化できます。Neptune 暗号化を使用して、保管中のデータのコンプライアンス要件を満たすこともできます。

  新しい Neptune DB インスタンスの暗号化を有効にするには、Neptune コンソールの暗号化を有効にするセクションで Yes (デフォルトで選択) を選択するか、AWS::Neptune::DBCluster::StorageEncrypted プロパティを設定します AWS CloudFormation。暗号化が有効になっている場合、Neptune はデフォルトで Amazon Relational Database Service (Amazon RDS) AWS マネージドキーを使用するか、カスタマーマネージドキーを作成できます。Neptune DB インスタンスの作成については、「新しい Neptune DB クラスターの作成」を参照してください。詳細については、「保管中の Neptune リソースの暗号化」を参照してください。自動スナップショットと手動スナップショットは、Neptune クラスターに選択したのと同じ暗号化を使用します。
- SPARQL 言語と openCypher 言語を使用する場合は、SQL インジェクションやその他の形式の攻撃を防ぐために、適切な入力検証とパラメータ化の手法を実践してください。ユーザーが指定した入力で文字列連結を使用するクエリを構築しないでください。パラメータ化されたクエリまたは準備済みステートメントを使用して、入力パラメータをグラフデータベースに安全に渡します。詳細については、openCypher パラメータ化されたクエリの例」および「SPARQL Injection Defence」を参照してください。
- Gremlin 言語の場合、潜在的なインジェクションの問題を回避するために、文字列ベースの Gremlin スクリプトを直接渡す代わりに Gremlin 言語バリアントを使用します。

ネットワークを保護する

Amazon Neptune DB クラスターは、 上の仮想プライベートクラウド (VPC) でのみ作成できます AWS。Neptune DB クラスターのエンドポイントには、通常、その VPC で実行されている Amazon Elastic Compute Cloud (Amazon EC2) インスタンスから、その VPC 内でのみアクセスできます。Neptune DB クラスターがある VPC へのアクセスを制限することで、Neptune データを保護できます。詳細については、Amazon Neptune グラフへの接続」を参照してください。

転送中のデータを保護するために、Neptune は安全なプロトコルと暗号を使用して、HTTPS を介した SSL 接続を任意のインスタンスまたはクラスターエンドポイントに強制します。Neptune は、Neptune DB インスタンスの SSL 証明書を提供します。Neptune SSL 証明書は、クラスターエンドポイント、リーダーエンドポイント、インスタンスエンドポイントのホスト名のみをサポートします。ロードバランサーまたはプロキシサーバー (HAProxy など) を使用している場合は、SSL 終了を使用し、プロキシサーバーに独自の SSL 証明書が必要です。提供された SSL 証明書はプロキシサーバーのホスト名と一致しないため、SSL パススルーは機能しません。SSL を使用して Neptune エンドポイントに接続する方法の詳細については、「HTTP REST エンドポイントを使用して Neptune DB インスタンスに接続する」を参照してください。

認証と認可の実装

Neptune DB クラスターと DB インスタンスで Neptune 管理 アクションを実行できるユーザーを制御するには、IAM 認証情報を使用します。IAM 認証情報 AWS を使用して に接続する場合、IAM ロールには、Neptune 管理オペレーションの実行に必要なアクセス許可を付与する IAM ポリシーが必要です。最小特権の原則に従い、タスクを完了するために必要なアクセス許可のみを付与してください。詳細については、「さまざまな種類の IAM ポリシーを使用して Neptune へのアクセスを制御する」および「一時的な認証情報を使用した IAM 認証」を参照してください。

Neptune クラスターに接続してデータをクエリできるユーザーを制御するには、IAM を使用して Neptune DB インスタンスまたは DB クラスターを認証できます。Neptune DB クラスターで IAM 認証を有効にする場合、DB クラスターにアクセスするすべてのユーザーを最初に認証する必要があります。詳細については、「Neptune で IAM データベース認証を有効にする」を参照してください。

IAM データベース認証が有効になっている場合、各リクエストは AWS 署名バージョン 4 を使用して署名する必要があります。IAM 認証が有効になっているすべての Neptune エンドポイントに署名付きリクエストを送信する方法については、AWS 「署名バージョン 4 を使用した接続と署名」を参照してください。awscurl などの多くのライブラリやツールは、 AWS 署名バージョン 4 を既にサポートしています。

他の とやり取りするために AWS のサービス、Amazon Neptune は IAM サービスにリンクされたロールを使用します。サービスにリンクされたロールは、Neptune に直接リンクされた一意のタイプの IAM ロールです サービスにリンクされたロールは Neptune によって事前定義されており、サービスが AWS のサービス ユーザーに代わって他の を呼び出すために必要なすべてのアクセス許可が含まれています。詳細については、「Neptune のサービスにリンクされたロールの使用」を参照してください。

信頼性の柱

AWS Well-Architected フレームワークの信頼性の柱には、ワークロードが意図した機能を正しく一貫して実行する能力が含まれます。これには、ワークロードのライフサイクル全体を通じてワークロードを運用およびテストする能力が含まれます。

信頼性の高いワークロードの実現は、ソフトウェアとインフラストラクチャの両方について事前に設計を決定することから始まります。アーキテクチャの選択は、Well-Architected の柱のすべてにおいて、ワークロードの動作に影響を与えます。高い信頼性を保つには、特定のパターンに従う必要があります。

信頼性の柱は、以下の主要分野に焦点を当てています。

- サービスクォータとデプロイパターンを含むワークロードアーキテクチャ
- 変更管理
- 障害管理

Neptune サービスクォータを理解する

Neptune クラスターボリュームは、クォータが 64 TiB である中国と GovCloud AWS リージョン を除くサポートされているすべての で最大サイズ 128 テビバイト (TiB) まで拡張できます。

128 TiB のクォータは、約 2,000～4,000 億個のオブジェクトをグラフに保存するのに十分です。ラベル付きプロパティグラフ (LPG) では、オブジェクトはノード、エッジ、またはノードまたはエッジ上のプロパティです。Resource Description Framework (RDF) グラフでは、オブジェクトは四角形です。

Neptune Serverless クラスターでは、Neptune キャパシティーユニット (NCUs) の最小数と最大数の両方を設定します。各 NCU は、2 ギビバイト (GiB) のメモリと、関連する vCPU およびネットワークで構成されます。最小および最大 NCU 値は、クラスター内のすべてのサーバーレスインスタンスに適用されます。設定できる最高の最大 NCU 値は 128.0 NCU であり、最低の最小値は 1.0 NCU です。Amazon CloudWatch メトリクスを監視し、一般的に実行される範囲をキャプチャ`ServerlessDatabaseCapacityNCUUtilization`して、アプリケーションに最適な NCU 範囲を最適化し、その範囲内の望ましくない動作やコストを関連付けます。ワークロードが十分に速くスケールされない場合は、スケーリング中に最初のサージに十分な処理を提供するように最小 NCUs を増やします。

各 AWS アカウント には、作成できるデータベースリソースの数に対する各リージョンのクォータがあります。これらのリソースには、DB インスタンスと DB クラスターが含まれます。リソースの制限に達すると、リソースを作成するための追加の呼び出しは、例外が発生して失敗します。一部のクォータは、リクエストによって引き上げることができるソフトクォータです。Amazon Neptune と Amazon RDS、Amazon Aurora、Amazon DocumentDB (MongoDB 互換) の間で共有されるクォータのリストと、利用可能な場合はクォータの引き上げをリクエストするためのリンクについては、「Amazon RDS のクォータ」を参照してください。

Neptune デプロイパターンを理解する

Neptune DB クラスターには、1 つのプライマリ DB インスタンスと最大 15 の Neptune レプリカがあります。プライマリ DB インスタンスは読み取りおよび書き込みオペレーションをサポートし、クラスターボリュームへのすべてのデータ変更を実行します。Neptune レプリカは、プライマリ DB インスタンスと同じストレージボリュームに接続し、読み取りオペレーションのみをサポートします。Neptune レプリカは、プライマリ DB インスタンスから読み取りワークロードをオフロードします。

高可用性を実現するには、リードレプリカを使用します。リードレプリカがプライマリインスタンスのフェイルオーバーターゲットとして機能するため、1 つ以上のリードレプリカインスタンスを異なるアベイラビリティーゾーンで利用できると、可用性が向上する可能性があります。ライターインスタンスが失敗した場合、Neptune はリードレプリカインスタンスをプライマリインスタンスに昇格させます。この場合、昇格されたインスタンスの再起動中に短時間の中断 (通常は 30 秒未満) が発生し、その間、プライマリインスタンスに対する読み取りおよび書き込みリクエストは例外で失敗します。最高の信頼性を得るには、異なるアベイラビリティーゾーンに 2 つのリードレプリカを検討してください。アベイラビリティーゾーン 1 のプライマリインスタンスがオフラインになった場合、アベイラビリティーゾーン 2 のインスタンスはプライマリに昇格されますが、その間はクエリを処理できません。したがって、移行中に読み取りクエリを処理するには、アベイラビリティーゾーン 3 のインスタンスが必要です。

Neptune Serverless を使用している場合、すべてのアベイラビリティーゾーンのリーダーインスタンスとライターインスタンスは、データベースの負荷に応じて、互いに独立してスケールアップおよびスケールダウンします。リーダーインスタンスの昇格階層を 0 または 1 に設定して、ライターインスタンスの容量に合わせてスケールアップまたはスケールダウンできます。これにより、現在のワークロードをいつでも引き継ぐ準備が整います。

Neptune クラスターの管理とスケーリング

Neptune 自動スケーリングを使用して、DB クラスター内の Neptune レプリカの数を自動的に調整し、CPU 使用率のしきい値に基づいて接続とワークロードの要件を満たすことができます。自動スケーリングを使用すると、Neptune DB クラスターはワークロードの急激な増加を処理できます。ワークロードが減少すると、Auto Scaling は不要なレプリカを削除し、未使用の容量に対して料金が発生しないようにします。新しいインスタンスの起動には最大 15 分かかる場合があるため、Auto Scaling だけでは需要の急激な変化に対する十分なソリューションにはならないことに注意してください。

自動スケーリングは、既に 1 つのプライマリライターインスタンスと少なくとも 1 つのリードレプリカインスタンスがある Neptune DB クラスターでのみ使用できます (Amazon Neptune DB クラスターとインスタンス」を参照)。また、クラスター内のすべての read-replica インスタンスが使用可能な状態である必要があります。リードレプリカが使用可能以外の状態にある場合、Neptune 自動スケーリングは、クラスター内のすべてのリードレプリカが使用可能になるまで何も実行しません。

需要が急速に変化する場合は、サーバーレスインスタンスの使用を検討してください。サーバーレスインスタンスは短期間で垂直方向にスケールでき、自動スケーリングは長期間にわたって水平方向にスケールできます。この設定では、サーバーレスインスタンスが垂直方向にスケールし、自動スケーリングは新しいリードレプリカをインスタンス化して、単一のサーバーレスインスタンスの最大容量を超えるワークロードを処理するため、最適なスケーラビリティを提供します。Amazon Neptune Serverless の容量スケーリングの詳細については、「Neptune Serverless DB クラスターの容量スケーリング」を参照してください。

スケーリングのニーズが予測可能なタイミングで変化する場合は、最小インスタンス、最大インスタンス、しきい値の変更をスケジュールして、それらのシフトのニーズをより適切に処理できます。必要に応じてインスタンスがオンラインになるように、少なくとも 15 分前にスケールアウトイベントをスケジュールしてください。

パラメータグループのパラメータを使用して、Amazon Neptune のデータベース設定を管理します。パラメータグループは、1 つ以上の DB インスタンスに適用されるエンジン設定値のコンテナとして機能します。パラメータグループのクラスターパラメータを変更するときは、静的パラメータと動的パラメータの違いと、それらを適用する方法とタイミングを理解します。ステータスエンドポイントを使用して、現在適用されている設定を確認します。

バックアップとフェイルオーバーイベントを管理する

Neptune はクラスターボリュームを自動的にバックアップし、バックアップ保持期間にわたってバックアップされたデータを保持します。Neptune のバックアップは連続的かつ増分的であるため、バックアップ保持期間内の任意の時点にすばやく復元できます。DB クラスターを作成または変更するときに、1～35 日間のバックアップ保持期間を指定できます。

バックアップ保持期間を超えてバックアップを保持するには、クラスターボリューム内のデータのスナップショットを作成することもできます。スナップショットの保存には、Neptune の標準ストレージ料金がかかります。

DB クラスターの Amazon Neptune スナップショットを作成すると、Neptune はクラスターのストレージボリュームスナップショットを作成し、個々のインスタンスだけでなくすべてのデータをバックアップします。新しい DB クラスターは、この DB クラスタースナップショットから復元することで後に作成できます。DB クラスターを復元するときは、復元元の DB クラスタースナップショットの名前を指定し、復元によって作成される新しい DB クラスターの名前を指定します。

フェイルオーバーイベントに対するシステムの応答をテストします。Neptune API を使用してフェイルオーバーイベントを強制します。フェイルオーバーによる再起動は、テストまたはフェイルオーバー後の元のアベイラビリティーゾーンへの復元オペレーションのために DB インスタンスの障害をシミュレートする場合に便利です。詳細については、「マルチ AZ 配置の設定と管理」を参照してください。DB ライターインスタンスを再起動すると、スタンバイレプリカにフェイルオーバーします。Neptune レプリカを再起動しても、フェイルオーバーは開始されません。

クライアントの信頼性を設計します。フェイルオーバーイベント中に動作をテストします。エクスポネンシャルバックオフロジックを使用して、クライアントに再試行ロジックを実装します。このロジックを実装するコード例は、AWS Lambda Amazon Neptune の関数例にあります。

複数のデータベースエンジンに適用される一般的なバックアップ要件のセットAWS Backupがある場合は、 の使用を検討してください。

パフォーマンス効率の柱

AWS Well-Architected フレームワークのパフォーマンス効率の柱は、データの取り込みまたはクエリ中にパフォーマンスを最適化する方法に焦点を当てています。パフォーマンスの最適化は、以下の段階的かつ継続的なプロセスです。

- ビジネス要件の確認
- ワークロードのパフォーマンスの測定
- パフォーマンスの低いコンポーネントを特定する
- ビジネスニーズに合わせてコンポーネントを調整する

パフォーマンス効率の柱は、使用する適切なグラフデータモデルとクエリ言語を特定するのに役立つユースケース固有のガイドラインを提供します。また、Amazon Neptune にデータを取り込んで Amazon Neptune からデータを使用するときに従うべきベストプラクティスも含まれています。

パフォーマンス効率の柱は、次の主要分野に焦点を当てています。

- グラフモデリング
- クエリの最適化
- クラスターの適切なサイジング
- 書き込みの最適化

グラフモデリングを理解する

ラベル付きプロパティグラフ (LPG) モデルとリソース記述フレームワーク (RDF) モデルの違いを理解します。ほとんどの場合、これは優先されます。ただし、1 つのモデルが他のモデルよりも適しているユースケースがいくつかあります。グラフ内の 2 つのノードを接続するパスに関する知識が必要な場合は、LPG を選択します。Neptune クラスターまたは他のグラフトリプルストア間でデータをフェデレーションする場合は、RDF を選択します。

Software as a Service (SaaS) アプリケーションまたはマルチテナンシーを必要とするアプリケーションを構築する場合は、クラスターごとに 1 つのテナントを持つのではなく、データモデルにテナントの論理的な分離を組み込むことを検討してください。このタイプの設計を実現するには、ラベルへの顧客識別子の付加や、テナント識別子を表すプロパティキーと値のペアの追加など、SPARQL の名前付きグラフとラベル付け戦略を使用できます。論理的な分離を維持するために、クライアントレイヤーがこれらの値を挿入していることを確認します。

クエリのパフォーマンスは、クエリの処理で評価する必要があるグラフオブジェクト (ノード、エッジ、プロパティ) の数によって異なります。そのため、グラフモデルはアプリケーションのパフォーマンスに大きな影響を与える可能性があります。可能な限り詳細なラベルを使用し、パスの判定またはフィルタリングに必要なプロパティのみを保存します。より高いパフォーマンスを実現するには、要約ノードの作成や一般的なパスを接続するより直接的なエッジの作成など、グラフの一部を事前に計算することを検討してください。

同じラベルを持つエッジ数が異常に多いノード間を移動しないようにします。このようなノードには多くの場合、数千のエッジがあります (ほとんどのノードのエッジカウントは数十です) 。その結果、コンピューティングとデータの複雑さが大幅に高まります。これらのノードは、一部のクエリパターンでは問題にならない場合がありますが、特に中間ステップとしてノード間を移動する場合は、それを回避するためにデータを異なる方法でモデル化することをお勧めします。スロークエリログを使用して、これらのノード間を移動するクエリを識別できます。特にデバッグモードを使用する場合、平均クエリパターンよりもレイテンシーとデータアクセスメトリクスがはるかに高くなります。

Neptune を使用して IDsランダムな GUID 値を割り当てずに、ユースケースでサポートされている場合は、ノードとエッジに決定的なノード IDs を使用します。ID によるノードへのアクセスが最も効率的な方法です。

クエリを最適化する

openCypher 言語と Gremlin 言語は、LPG モデルで互換的に使用できます。パフォーマンスが最大の懸念事項である場合は、2 つの言語を同じ意味で使用することを検討してください。1 つの言語は特定のクエリパターンで他の言語よりもパフォーマンスが良い可能性があるためです。

Neptune は代替クエリエンジン (DFE) に変換中です。openCypher は DFE でのみ実行されますが、オプションでクエリ注釈を使用して Gremlin クエリと SPARQL クエリの両方を DFE で実行するように設定できます。DFE を有効にしてクエリをテストし、DFE を使用しない場合はクエリパターンのパフォーマンスを比較することを検討してください。

Neptune は、グラフ全体を評価する分析クエリではなく、単一ノードまたは一連のノードから開始してそこからファンアウトするトランザクションタイプのクエリに最適化されています。分析クエリワークロードでは、AWS SDK for Pandas を使用するか、neptune-export を AWS Glueまたは Amazon EMR と組み合わせて使用することを検討してください。

モデルとクエリの非効率性とボトルネックを特定するには、各クエリ言語の `profile`および `explain` APIs を使用して、クエリプランとクエリメトリクスの詳細な説明を取得します。詳細については、「Gremlin profile」、openCypher explain」、および「SPARQL explain」を参照してください。

クエリパターンを理解します。グラフ内の個別のエッジの数が大きくなると、デフォルトの Neptune アクセス戦略が非効率になる可能性があります。以下のクエリは、非常に非効率になる可能性があります。

- エッジラベルが指定されていない場合にエッジをまたいで後方に移動するクエリ。
- Gremlin など、内部`.both()`的に同じパターンを使用する句、または任意の言語でノードを削除する句 (ラベルを知らずに受信エッジを削除する必要があります) 。
- プロパティラベルを指定せずにプロパティ値にアクセスするクエリ。これらのクエリは非常に非効率になる可能性があります。これが使用パターンと一致する場合は、OSGP インデックス (オブジェクト、サブジェクト、グラフ、述語) を有効にすることを検討してください。

スロークエリログ記録を使用して、スロークエリを識別します。クエリが遅いのは、クエリプランが最適化されていないか、インデックスルックアップが不必要に多数あることが原因で、I/O コストが増加する可能性があります。Gremlin、SPARQL、または openCypher の Neptune explain および profile エンドポイントは、これらのクエリが遅い理由を理解するのに役立ちます。原因には次のようなものがあります。

- グラフ内の平均ノードと比較してエッジ数が異常に多いノード (例えば、数千と数十) では、計算が複雑になり、レイテンシーが長くなり、リソースの消費が増加する可能性があります。これらのノードが正しくモデル化されているかどうか、またはトラバースする必要があるエッジの数を減らすためにアクセスパターンを改善できるかどうかを決定します。
- 最適化されていないクエリには、特定のステップが最適化されていないという警告が含まれます。これらのクエリを書き換えて最適化されたステップを使用すると、パフォーマンスが向上する可能性があります。
- フィルターが重複すると、不要なインデックスルックアップが発生する可能性があります。同様に、冗長パターンは、クエリを改善することで最適化できるインデックスルックアップの重複を引き起こす可能性があります (プロファイル出力`Index Operations - Duplication ratio`の を参照) 。
- Gremlin などの一部の言語では、厳密に型指定された数値がなく、代わりに型昇格を使用します。例えば、値が 55 の場合、Neptune は整数、長整数、浮動小数点数、および 55 に相当するその他の数値型である値を検索します。これにより、追加のオペレーションが発生します。タイプが事前に一致することがわかっている場合は、クエリヒントを使用してこれを回避できます。
- グラフモデルはパフォーマンスに大きな影響を与える可能性があります。より詳細なラベルを使用するか、マルチホップ線形パスへのショートカットを事前に計算して、評価する必要があるオブジェクトの数を減らすことを検討してください。

クエリの最適化だけではパフォーマンス要件を満たすことができない場合は、Neptune でさまざまなキャッシュ手法を使用してこれらの要件を満たすことを検討してください。

適切なサイズのクラスター

同時実行とスループットの要件に合わせてクラスターのサイズを設定します。クラスター内の各インスタンスで処理できる同時クエリの数は、そのインスタンスの仮想 CPUs (vCPUs) の数の 2 倍に等しくなります。すべてのワーカースレッドが占有されている間に到着する追加のクエリは、サーバー側のキューに入れられます。これらのクエリは、ワーカースレッドが利用可能になったときに first-in-first-out (FIFO) ベースで処理されます。`MainRequestQueuePendingRequests` Amazon CloudWatch メトリクスには、各インスタンスの現在のキューの深さが表示されます。この値が頻繁に 0 を超える場合は、より多くの vCPU を持つインスタンスを選択することを検討してください。vCPUs キューの深さが 8,192 を超えると、Neptune は`ThrottlingException`エラーを返します。

各インスタンスの RAM の約 65% がバッファキャッシュ用に予約されています。バッファキャッシュは、作業データセット (グラフ全体ではなく、クエリ対象のデータのみ) を保持します。ストレージではなくバッファキャッシュから取得されるデータの割合を確認するには、CloudWatch メトリクス をモニタリングします`BufferCacheHitRatio`。このメトリクスが 99.9% を下回ることがよくある場合は、より多くのメモリを持つインスタンスを試して、レイテンシーと I/O コストを削減するかどうかを検討してください。

リードレプリカは、ライターインスタンスと同じサイズである必要はありません。ただし、書き込みワークロードが多いと、レプリケーションに追いつくことができないため、レプリカが小さくなり、再起動する可能性があります。したがって、レプリカをライターインスタンスと同じかそれ以上にすることをお勧めします。

リードレプリカに Auto Scaling を使用する場合は、新しいリードレプリカをオンラインにするまでに最大 15 分かかる場合があることに注意してください。クライアントトラフィックが迅速かつ予測可能な方法で増加する場合は、スケジュールされたスケーリングを使用して、その初期化時間を考慮してリードレプリカの最小数を高く設定することを検討してください。

サーバーレスインスタンスは、いくつかの異なるユースケースとワークロードをサポートしています。以下のシナリオでは、サーバーレスのオーバープロビジョニングされたインスタンスを検討してください。

- ワークロードは 1 日を通して頻繁に変動します。
- 新しいアプリケーションを作成し、ワークロードのサイズが不明な場合。
- 開発とテストを実行している。

サーバーレスインスタンスは、同等のプロビジョニングされたインスタンスよりも RAM の GB 単位でコストがかかることに注意してください。各サーバーレスインスタンスは、2 GB の RAM と、関連する vCPU およびネットワークで構成されます。オプション間でコスト分析を実行して、予想外の請求を回避します。一般的に、1 日に数時間しかワークロードが非常に重く、残りの時間はほぼゼロである場合、またはワークロードが 1 日を通じて大幅に変動する場合にのみ、サーバーレスでコスト削減を達成できます。

書き込みの最適化

書き込みを最適化するには、次の点を考慮してください。

- Neptune Bulk Loader は、最初にデータベースをロードしたり、既存のデータに追加したりするのに最適な方法です。Neptune ローダーはトランザクションではないため、データを削除できないため、これらが要件である場合は使用しないでください。
- トランザクションの更新は、サポートされているクエリ言語を使用して行うことができます。書き込み I/O オペレーションを最適化するには、コミットごとに 50～100 個のオブジェクトのバッチでデータを書き込みます。オブジェクトは、LPG のノードまたはエッジ、または RDF のトリプルストアまたはクワッド上のノード、エッジ、またはプロパティです。
- すべての Neptune 書き込みオペレーションは、接続ごとにシングルスレッドです。Neptune に大量のデータを送信するときは、それぞれがデータを書き込む複数の並列接続を持つことを検討してください。Neptune でプロビジョニングされたインスタンスを選択すると、インスタンスサイズは多数の vCPUs。Neptune はインスタンス上の vCPU ごとに 2 つのデータベーススレッドを作成するため、最適な並列化をテストするときに vCPUs の数の 2 倍から開始します。 サーバーレスインスタンスは、4 NCU ごとに約 1 の速度で vCPUs の数をスケーリングします。 NCUs
- 1 つの接続だけがいつでもデータを書き込んでいる場合でも、すべての書き込みプロセス中に ConcurrentModificationExceptions を計画し、効率的に処理します。`ConcurrentModificationExceptions` 発生時の信頼性を考慮してクライアントを設計します。
- すべてのデータを削除する場合は、同時削除クエリを発行するのではなく、高速リセット API を使用することを検討してください。後者では、前者と比較してはるかに時間がかかり、かなりの I/O コストが発生します。
- ほとんどのデータを削除する場合は、neptune-export を使用して新しいクラスターにデータをロードすることで、保持するデータをエクスポートすることを検討してください。次に、元のクラスターを削除します。

コスト最適化の柱

AWS Well-Architected フレームワークのコスト最適化の柱は、不要なコストを回避することに重点を置いています。以下の推奨事項は、Amazon Neptune のコスト最適化設計原則とアーキテクチャのベストプラクティスを満たすのに役立ちます。

コスト最適化の柱は、次の主要分野に焦点を当てています。

- 経時的な支出の把握と資金配分の制御
- 適切なタイプと数量のリソースを選択する
- 過剰な支出なしでビジネスニーズを満たすスケーリング

必要な使用パターンとサービスを理解する

Neptune は、データモデルに識別可能なグラフ構造があり、クエリが関係を調べて複数のホップをトラバースする必要がある場合、ワークロードに適しています。グラフデータベースは、次のパターンには適していません。

- 主にシングルホップクエリ (データがオブジェクトの属性としてより適切に表現されるかどうかを検討してください)
- プロパティとして保存される JSON または BLOB データ
- 多数のノードにわたる数値プロパティの合計の計算など、データセット全体で集計されるクエリ

特定のアクセスパターンに複数の専用データベースを一緒に使用すると、すべてのニーズに対応できるかどうかを検討してください。以下に例を示します。

- 1 つのノードのプロパティを高度に同時に取得すると同時に、複雑なグラフナビゲーションをあまり必要としない API は、Neptune、DynamoDB、または Amazon DocumentDB の 1 つ以上を使用することが最適です。
- リレーショナルデータベースは Neptune と共存して既存の機能を維持できますが、Neptune はリレーショナルデータベースでうまく動作およびスケーリングしないマルチホップトラバーサルにのみ使用します。

Neptune とやり取りして補完するサービスに関連するコストについて、以下を含めます。

- Neptune に一括ロードされるデータファイルの Amazon Simple Storage Service (Amazon S3) ストレージコスト
- クエリの挿入またはアップサート、クエリの読み取り、Neptune ストリーム処理に使用される Lambda 関数
- または で (データベースに直接接続する代わりに) クライアントアプリケーションとやり取りするために Neptune 上に構築された API レイヤー AWS AppSync
- AWS Glue Neptune との間でデータを転送するために使用する ジョブ
- Amazon Kinesis または Amazon Managed Streaming for Apache Kafka (Amazon MSK) インスタンスは、Neptune へのほぼリアルタイムの取り込みのためにストリーミングデータを受信します。
- AWS Database Migration Service Neptune へのリレーショナルデータの移行用
- Jupyter ノートブックとディープグラフライブラリ機械学習モデルの Amazon SageMaker ランタイムコスト

コストに注意してリソースを選択する

Neptune の料金は、時間単位のインスタンスコスト (またはサーバーレスに消費される Neptune コンピューティングユニット) 、データ I/O、ストレージ使用量に基づいています。インスタンスは平均して全体のコストの 85% を占めるため、適切なサイジングはコストに大きな影響を与える可能性があります。インスタンスのサイズを適切に設定する最善の方法は、さまざまなインスタンスでアプリケーションのパフォーマンスをテストし、次の要因を比較することです。

- `MainRequestQueuePendingRequests` CloudWatch メトリクスは一貫してゼロに近い低い数値に留まりますか？
- `BufferCacheHitRatio` CloudWatch メトリクスは、ほとんどの場合 99.9% 以上に留まりますか？
- インスタンスコストおよび関連するデータ I/O コストのコストとパフォーマンス曲線はどのようなものですか？ ストレージとのバッファキャッシュのスワップが頻繁に必要になる、サイズが小さいインスタンスでは、データ読み取りコストが大幅に増加する可能性があります。`BufferCacheHitRatio`は、これらのシナリオでは頻繁に低下します。

インスタンスのコストは、同じインスタンスファミリー内のサイズに応じて直線的にスケールされます。`db.r6i.2xlarge` インスタンスの時間単位のコストは`db.r6i.xlarge`インスタンスの 2 倍で、リソース割り当ても 2 倍です。`db.r6i.24xlarge` インスタンスは、`db.r6i.xlarge`インスタンスの時間単位のコストの 24 倍です。

サポートする必要がある同時クエリの数を見積もります。読み取り専用クエリを処理するために、0 ~ 15 個のリードレプリカを持つことができます。時間、週、月によって要件が異なる場合は、複数の小さいインスタンスを使用してスケジュールに基づいてスケーリングできます。インスタンスの各 vCPU には、同時クエリを処理するための 2 つのスレッドが用意されています。それぞれ 4 つの vCPU を持つ 3 つのdb.r6i.xlargeリードレプリカは、24 の同時クエリを処理できます。

代わりにトラフィック量が 1 秒あたりのクエリ数 (QPS) で測定される場合は、クエリの平均レイテンシーを試して判断する必要があります。Neptune クラスターがサポートできる 1 秒あたりのクエリ数は、 に等しくなりますvCPU × 2 × (1 second/average query latency)。例えば、4 つの vCPU があり、クエリのレイテンシーが 100 ミリ秒 (0.1 秒) の場合、 ですQPS = 4 × 2 × (1s/0.1s) = 80 queries per second。

プロビジョニングされたインスタンスは、継続的で安定した予測可能なワークロードに対して、サーバーレスよりも安価です。サーバーレスは、1 日あたり数時間のみ非常に高い使用量を必要とするワークロード (などdb.r6i.4xlarge) があり、残りの時間はほとんどトラフィックがない場合 (1 Neptune コンピューティングユニットなど) 、コストを最適化する機会を提供します。数時間スケールアップしてからバックダウンするサーバーレスインスタンスは、プロビジョニングされたdb.r6i.4xlargeインスタンスを終日使用するよりも安価になります。

ワークロードに最適な Neptune インスタンス設定を選択する

Neptune のエントリレベルの実験には、AWS 無料利用枠を使用できます。db.t3.medium および db.t4g.mediumインスタンスの 750 時間の無料利用時間は、Neptune を低規模で十分に理解するのに十分です。クラスターは無料トライアル期間終了後も残ります。ただし、その時点以降の使用量に対しては課金されます。

db.t3.medium および db.t4g.mediumインスタンスは低コストの開発環境に適していますが、RAM と vCPU の比率 (2:1) は R ファミリーインスタンス (8:1) または X ファミリーインスタンス (16:1) よりも小さいことに注意してください。パフォーマンスプロファイルは、特にクエリがグラフの大部分を移動する場合OutOfMemoryExceptionsや移動する場合に、これらのクラスとは異なることがあります。後者の条件が影響を受ける可能性があるかどうかを判断するには、BufferCacheHitRatioCloudWatch メトリクスを確認します。

T ファミリーインスタンスでパフォーマンステストや負荷テストを行わないことを強くお勧めします。これは、本番環境を示していない一貫性のない結果が発生する可能性があるためです。

プロビジョニングされたインスタンスは、ワークロードがかなり安定していて予測可能な場合に、コストとパフォーマンスの最適な組み合わせを提供します。必要なリクエストの同時実行

数とクエリの複雑さに基づいてインスタンスサイズを選択します。同時実行数を増やすには、より多くの vCPUsが必要です。クエリの複雑さが高いほど、より多くの RAM が必要になります。`MainRequestQueuePendingRequests` CloudWatch メトリクスを使用して、前者の影響を判断します (ゼロより大きい場合は、処理可能な数よりも多くの同時リクエストを表します)。`BufferCacheHitRatio` CloudWatch メトリクスを使用して、後者の影響を判断します。比率が 99.9% を頻繁に下回る場合は、評価対象のグラフの作業部分を含むのに十分な RAM がないことを示し、キャッシュスワップの頻度が高くなります。インスタンスの R ファミリーが十分な同時実行性を提供していても RAM が十分でない場合は、インスタンスの X ファミリーを試すことを検討してください。

サーバーレスインスタンスの理想的なユースケースについては、Neptune ドキュメントで説明されています。プロビジョニング済みとサーバーレスのどちらが最適かが不明で、コストが主な懸念事項である場合は、サーバーレスでワークロードをテストして、使用する NCUs の数を決定し、プロビジョニング済み (`N hours × hourly provisioned cost`) のコストをサーバーレス () と比較しますsum of NCUs × hourly cost per NCU。同等のサイズのプロビジョニングインスタンスが不明な場合、1 つの NCU は、約 2 GB の RAM および関連する vCPU とネットワークに相当します。プロビジョニングされたインスタンスが R6i ファミリーのものである場合、その比率は 8 GB の RAM あたり 1 vCPU、または 4 NCUs、関連付けられたネットワークとともに行われます。

プライマリインスタンスとレプリカインスタンスにサーバーレスを使用する場合、昇格階層 0 と 1 のリードレプリカは、フェイルオーバーイベントが発生した場合に適切にスケーリングされるように、ライターインスタンスに合わせて NCUs をスケーリングすることに注意してください。これらのインスタンスの NCU 制限は、ライターまたはリーダーのうち、トラフィックを最も多く受け取るインスタンスに基づいて設定します。

クラスターが 1 日 24 時間、週 7 日不要な環境では、使用していないときに Neptune インスタンスをオフにするスクリプトを記述し、使用前に再度起動することを検討してください。Neptune インスタンスは 7 日ごとに自動的に再起動され、必要なメンテナンス更新が適用されます。インスタンスを長期間オフにする場合は、毎週のスクリプトを使用して再度シャットダウンします。

適切なサイズのデータストレージと転送

より効率的なクエリ (たとえば、グラフ内のノード、エッジ、プロパティの数を減らす必要があるクエリ) では、必要な I/O 転送が少なくなり、バッファキャッシュが少なくて済むため、より小さなインスタンスを使用できる可能性があります。クエリ言語のプロファイルまたは説明エンドポイントを使用してクエリを最適化し、クエリのパフォーマンスに合わせてグラフモデルを最適化することを検討してください。

Neptune は大きな文字列にディクショナリエンコーディングを使用し、そのディクショナリは効率ではなくパフォーマンスのために最適化されています。プロパティの大きな BLOBs、JSON、または頻繁に変更される文字列がある場合は、それらを Neptune の外部に Amazon S3、Amazon DynamoDB、または Amazon DocumentDB に保存することを検討し、Neptune ノード内に参照のみを保存してください。

場合によっては、インスタンスサイズを大きくすると安価になることがあります。が低いために I/O コストが非常に高い場合`BufferCacheHitRatio`、バッファキャッシュが大きいほど、そのコストが大幅に削減される可能性があります。これは、ストレージから頻繁にスワップされ、I/O 転送レートが発生するのではなく、すべてのデータがキャッシュに収まるためです。

Neptune はcopy-on-writeのクローン作成を使用します。クローンを作成してグラフを複数のシャードに分割する場合、クローンクラスター上の不要なデータを削除しない方が効率的です。これは、新しいデータページの作成に伴ってストレージコストが増加するためです。クローン作成イベント前から変更されていないデータは、2 つのクラスター間で共有される 1 つのデータページに存在し、その 1 つのコピーに対してのみ課金されます。

OSGP インデックスを有効にしたりR5d インスタンスを使用したりしないでください。ただし、ワークロードに大きな違いがあることをテストして確認した場合は除きます。どちらもまれに発生するシナリオ向けに設計されており、最小限の利益または利益なしでコストが増加する可能性があります。

持続可能性の柱

AWS Well-Architected フレームワークの持続可能性の柱は、クラウドワークロードの実行による環境への影響を最小限に抑えることに焦点を当てています。主なトピックには、持続可能性、影響の理解、必要なリソースを最小限に抑えてダウンストリームへの影響を軽減するための使用の最大化に関する責任共有モデルが含まれます。

持続可能性の柱には、以下の主要な重点分野が含まれています。

- 影響
- 持続可能性の目標
- 最大使用量
- より効率的な新しいハードウェアおよびソフトウェアの提供を予測して採用する
- マネージドサービスの使用
- ダウンストリームの影響軽減

このガイドでは、影響に焦点を当てます。その他の持続可能性設計原則の詳細については、AWS Well-Architected フレームワークを参照してください。

選択と要件は環境に影響します。炭素強度 AWS リージョン の低い を選択し、稼働時間と耐久性を最大化するだけでなく、要件に実際のワークロードのニーズを反映している場合、ワークロードの持続可能性が向上します。次のセクションでは、ワークロード設計と継続的な運用で採用された場合に環境に悪影響を及ぼすベストプラクティスと考慮事項について説明します。

AWS リージョンの選択

一部の AWS リージョン は Amazon 再生可能エネルギープロジェクトの近くにあるか、グリッドの炭素強度が他のものよりも低い公開されている場所にあります。ワークロードに対して実行可能なリージョンの持続可能性への影響を考慮し、Neptune が利用可能なリージョンとリストを相互参照します。

ユーザー行動パターンに基づく消費

ユーザーのトラフィックと動作に合わせて使用量を適切にサイズ設定することで、サービスが環境に与える影響 AWS を最小限に抑えることができます。ソリューションを設計するときは、次のベストプラクティスを考慮してください。

- CPUUtilization、、 などの Amazon CloudWatch メトリクスをモニタリングTotalRequestsPerSecしてMainRequestQueuePendingRequests、需要が最も高いタイミングと最も低いタイミングを判断し、その間にクラスターリソースのサイズが適切であることを確認します。
- 非本番環境が使用されていない時間帯に停止を自動化します。詳細については、ブログ記事「リソースタグを使用して Amazon Neptune 環境リソースの停止と開始を自動化する」を参照してください。
- トラフィックパターンが頻繁かつ予測不可能な場合は、ピークトラフィック用にプロビジョニングされたインスタンスを使用する代わりに、需要に応じてスケールアップ/ダウンする Neptune Serverless インスタンスを使用することを検討してください。
- ビジネス継続性の目標に加えて、サービスレベルの契約を持続可能性の目標に合わせることを検討してください。マルチリージョンディザスタリカバリ、高可用性、長期バックアップ保持などの簡単な要件、特に非本番環境やミッションクリティカルなワークロードでは、これらの目標を達成するために必要なリソースの量を減らすことができます。

ソフトウェア開発とアーキテクチャのパターンを最適化する

無駄を防ぐには、モデルとクエリを最適化し、コンピューティングリソースを共有して、Neptune インスタンスとクラスターで使用できるすべてのリソースを使用します。具体的なベストプラクティスは次のとおりです。

- 開発者に、それぞれが独自のインスタンスを作成するのではなく、Neptune インスタンスと Jupyter Notebook アプリケーションインスタンスを共有してもらいます。マルチテナンシーパーティショニング戦略を使用して、各デベロッパーに単一の Neptune クラスター内の独自の論理パーティションを提供し、単一の Jupyter インスタンスでデベロッパーごとに個別のノートブックフォルダを作成します。
- データをロードするための並列スレッドやレコードをより大きなトランザクションにまとめてバッチ処理するなど、リソースの使用を最大化し、アイドル時間を最小限に抑えるパターンを実装します。
- クエリとグラフモデルを最適化して、結果の計算に必要なリソースを最小限に抑えます。
- Gremlin クエリの結果については、結果キャッシュ機能を使用して、ページ分割されたクエリまたは頻繁に繰り返されるクエリの再計算に費やされるリソースを最小限に抑えます。
- Neptune 環境を最新の状態に保ちます。Neptune の最新バージョンは、Graviton などの最新の EC2 インスタンスをより効率的にサポートします。また、クエリの最適化が改善され、クエリの計算に必要なリソースの量を減らすバグ修正も行われています。

リソース

リファレンス

- AWS Well-Architected
- AWS Well-Architected フレームワークのドキュメント
- Amazon Neptune Analytics 用の AWS Well-Architected フレームワークの適用
- Amazon Neptune の最新更新
- ベストプラクティス: Neptune を最大限に活用する

ブログ記事

- Apache TinkerPop Gremlin を使用した Amazon Neptune データアクセスの自動テスト TinkerPop
- リソースタグを使用して Amazon Neptune 環境リソースの停止と開始を自動化する
- Amazon Neptune データプレーンアクションのきめ細かなアクセスコントロール
- RDFox を Amazon Neptune と統合して、セマンティック推論を使用して RDF グラフから新しい事実を推測する Amazon Neptune
- Amazon Neptune ML を使用してリアルタイムの不正検出ソリューションを構築する

無料の AWS スキルビルダーコース

- Amazon Neptune の開始方法
- Amazon Neptune でのアプリケーションの構築
- Amazon Neptune のデータモデリング

寄稿者

このガイドの寄稿者は以下のとおりです。

- Brian O'Keefe、プリンシパル Neptune ソリューションアーキテクト、 AWS
- Abhishek Mishra、シニア Neptune ソリューションアーキテクト、 AWS
- Ganesh Sawhney、チームリード - 戦略的パートナー成功ソリューションアーキテクト、 AWS
- マイケル・ヘイビー、シニア Neptune ソリューションアーキテクト、 AWS
- Kevin Phillips、Neptune ソリューションアーキテクト、 AWS
- Melissa Kwok、Neptune ソリューションアーキテクト、 AWS
- Sakti Mishra、プリンシパルソリューションアーキテクト AWS
- Javed Ali、シニアソリューションアーキテクト、 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 9 月 27 日

AWS 規範ガイダンス用語集

以下は、 AWS 規範ガイダンスが提供する戦略、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、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。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。集計関数の例には、SUMおよび が含まれます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、組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るのに役立ちます。 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のガイダンスを提供します。詳細については、AWS CAF ウェブサイト と AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

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

B

不正なボット

個人や組織を混乱させたり、損害を与えたりすることを意図したボット。

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、単一の当事者によって制御されているボットのネットワーク。ボットは、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。移行戦略との関連性については、 AWS「移行準備ガイド」を参照してください。

CMDB

「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。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、通常はクエリや分析に使用されます。

データベース定義言語 (DDL)

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

データベース操作言語 (DML)

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

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせる。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。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件のコンプライアンスに影響を与える可能性のあるランディングゾーンの変更を検出したりできます。

DVSM

「開発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。たとえば、 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、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で、障害の影響を制限し、ワークロードの耐障害性を向上させるのに役立ちます。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。

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)

一般化データと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画像、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できる 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 Framework」の「Deploy using immutable infrastructure best practice」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 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

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

解釈可能性

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

IoT

「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、動作時にそれ自体を強化および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 を除くすべて 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のため、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、1 つの文字列、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。