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

Table of Contents

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

Michael Havey、Amazon Web Services (AWS)

2024 年 12 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。Neptune には、大量のグラフデータをすばやく分析してインサイトを取得し、傾向を見つけることができるメモリ最適化グラフ分析エンジンである Neptune Analytics が含まれています。既存の Neptune データベースクラスター内のデータに対して分析を実行したり、外部データセットからデータをロードして分析したりできます。このガイドでは、Neptune Analytics のデプロイを計画するときに AWS Well-Architected フレームワーク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 用の AWS Well-Architected フレームワークを適用すると、Neptune データベースの同じトピックがカバーされます。

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
- コスト最適化
- 持続可能性

このガイドでは、 Well-Architected フレームワークの設計の柱とベストプラクティス、および に Neptune Analytics 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、お客様や組織が以下のことを行うのに役立ちます。

- サポートされているデプロイオプションから選択します。
- 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のクエリを設計します。
- Neptune Analytics グラフ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。開発をサポートし、ワークロードを効率的に実行し、運用に関するインサイトを得て、ビジネス価値をもたらすサポートプロセスと手順を継続的に改善する能力が含まれます。人間の介入なしにほとんどの問題を検出して修復する自己修復ワークロードによって、運用上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い、Amazon Neptune Analytics のメトリクス、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場合に適切に対応することで、この目標を達成できます。

運用上の優秀性の柱に関するこの説明では、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
- コンプライアンスのための監査レポート
- 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。

- IaC を適用して Neptune Analytics グラフと関連リソースをデプロイします。一貫した環境設定を行うには、 が提供する Neptune Analytics のサポートAWS CloudFormationを使用して、グラフとプライベートエンドポイントをプロビジョニングします。
- CloudFormation を使用して、Amazon SageMaker AI で Neptune ノートブックインスタンスをプロビジョニングします。ノートブックを使用して、Neptune Analytics グラフ内のデータをクエリおよび視覚化できます。
- Neptune データベースクラスターやスナップショット、Amazon Simple Storage Service (Amazon S3) でステージングされたデータファイルなどの既存のソースから Neptune Analytics グラフを作成する場合は、一括インポートタスクをモニタリングします。
- グラフのサイズ変更、グラフの削除とスナップショット作成、スナップショットからのグラフの復元、グラフのリセットと再ロードなど、Neptune Analytics の運用手順を自動化します。AWS

Command Line Interface (AWS CLI) または SDK から利用可能な Neptune Analytics API を使用します。

- グラフに必要な稼働時間を評価します。分析は一時的なことが多いため、グラフはアルゴリズムの実行に必要な期間のみ必要です。この場合、 AWS CLI または SDKs を使用して、不要になったグラフをスナップショットして削除します。その後、必要に応じてスナップショットから復元できます。
- 接続文字列をクライアントの外部に保存します。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 Analytics 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。タグはリソースを整理するのに役立ちます。たとえば、特定の環境またはアプリケーションのリソースに共通のタグを適用できます。タグを使用してリソース使用量の請求を分析することもできます。詳細については、AWS 「請求ユーザーガイド」の「コスト配分タグを使用した AWS コストの整理と追跡」を参照してください。さらに、 AWS Identity and Access Management (IAM) ポリシーの条件を使用して、そのリソースで使用されているタグに基づいてリソースへのアクセス AWS を制御できます。これを行うには、グローバル条件キー `aws:ResourceTag/tag-key` を使用します。詳細については、IAM ユーザーガイドのAWS 「リソースへのアクセスの制御」を参照してください。

運用のための設計

Neptune Analytics グラフの操作方法を改善するアプローチを採用します。

- 開発、テスト、本番稼働用に個別の Neptune Analytics グラフを維持します。これらのグラフには、データセット、ユーザー、運用コントロールが異なる場合があります。
- さまざまな用途に合わせて個別の Neptune Analytics グラフを維持します。たとえば、2 つの分析ユーザーのグループで、タイムライン、モデル、パフォーマンスと可用性の SLAs、使用パターンが異なる個別のグラフが必要な場合は、グループごとに個別のグラフを維持します。
- Neptune Analytics メンテナンス更新のためにユーザーと運用スタッフを準備します。

小規模かつ可逆的な変更を頻繁に行う

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実行できる小さな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**
>
> AWS 認証情報をソース管理に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。本番稼働用ではない Neptune Analytics 環境でコードをコンパイル、テスト、デプロイしてから、本番稼働用グラフに昇格させます。

実用的なインサイトのためのオブザーバビリティを実装する

ワークロードの動作、パフォーマンス、信頼性、コスト、ヘルスを包括的に把握できます。以下の推奨事項は、Neptune Analytics でそのレベルの理解に役立ちます。

- Neptune Analytics の Amazon CloudWatch メトリクスをモニタリングします。これらのメトリクスから、グラフのサイズ (ノード数、エッジ数、ベクトル数、合計バイトサイズ) 、CPU 使用率、クエリリクエストとエラー率を判断できます。
- 、NumQueuedRequestsPerSec、NumOpenCypherRequestsPerSec、、 などの主要なメトリクスとGraphSizeBytes、アプリケーションログにある CPUUtilization Neptune GraphStorageUsagePercent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。
- 通知を設定して、グラフサイズ、リクエストレート、CPU 使用率がしきい値を超えたときなど、Neptune Analytics グラフの状態をモニタリングします。例えば、GraphStorageUsagePercentがグラフで 90% まで大幅に拡大する場合は、メモリ最適化 Neptune キャパシティユニット (m-NCU) 容量を増やすかどうかを決定します。現在の m-NCU が 128 の場合、256 に増やすとストレージが約 45% 削減されます。NumQueuedRequestsPerSec が 0 より大きいことがよくある場合は、コンピューティング容量を増やすために m-NCU 容量を増やすことを検討してください。または、クライアント側の同時実行数を減らすこともできます。

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、対応が意図したような効果を出さなかったりするたびに反復的に進化していく長期的な取り組みです。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ぶことで改善を推進します。
- チームと組織で教訓を共有します。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

ログ記録を使用して、異常なパフォーマンスとアクティビティのパターンを確認します。以下のベストプラクティスを考慮します。

- Neptune Analytics 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 Neptune Analytics API コールのログ記録 AWS CloudTrail」を参照してください。この機能を使用すると、Neptune Analytics リソースの作成、更新、削除を追跡できます。堅牢なモニタリングとアラートのために、CloudTrail イベントを Amazon CloudWatch Logs と統合することもできます。Neptune Analytics サービスアクティビティの分析を強化し、 のアクティビティの変更を特定するには AWS アカウント、Amazon Athena を使用して CloudTrail ログをクエリできます。例えば、クエリを使用して傾向を識別したり、アクティビティを属性 (ソース IP アドレスやユーザーなど) でさらに分離したりできます。
- CloudTrail を使用して、クエリ実行などの Neptune Analytics データプレーンアクティビティのログ記録を有効にすることもできます。実行中のクエリ、その頻度、およびソースを表示できます。デフォルトでは、CloudTrail はデータイベントをログ記録しません。追加の変更がイベントデータに適用されます。詳細については、AWS CloudTrail 料金表を参照してください。
- コントロールプレーンまたはデータプレーンのいずれかで、Neptune Analytics へのアプリケーション呼び出しを記録することもできます。たとえば、 を使用してクエリAWS SDK for Python (Boto3)を行う場合、デバッグレベルのログ記録を有効にして、コンソールまたはファイルへのクエリのトレースを取得できます。これは開発時に便利です。また、アプリケーションから例外をキャッチしてログに記録することをお勧めします。

セキュリティの柱

クラウドセキュリティが最優先事項です AWS。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とお客様の間で共有される責任です。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を担います AWS クラウド。 AWS また、 は、安全に使用できるサービスも提供します。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、 AWS セキュリティの有効性を定期的にテストおよび検証します。Neptune に適用されるコンプライアンスプログラムの詳細については、AWS 「コンプライアンスプログラムによる対象範囲内のサービス」を参照してください。
- クラウドのセキュリティ – 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および規制などの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に関するFAQs」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、 責任AWS 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 Analytics 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、セキュリティとコンプライアンスの目的を達成するために Neptune Analytics を設定する方法について説明します。また、Neptune Analytics 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、以下の主要な重点領域が含まれています。

- 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

データ漏洩や違反は、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。

- グラフ名、タグ、IAM ロール、およびその他のメタデータには、機密情報や機密情報を含めないでください。データは請求ログや診断ログに表示される可能性があるためです。

- 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 Analytics グラフは保管時に暗号化されます。選択したデフォルトキーまたは AWS Key Management Service（ AWS KMS) キーを使用して、グラフを暗号化できます。一括インポート中に Amazon S3 にエクスポートされるスナップショットとデータを暗号化することもできます。インポートが完了したら、暗号化を削除できます。
- 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、Neptune ドキュメントのopenCypher パラメータ化されたクエリの例」を参照してください。

ネットワークを保護する

パブリック接続用の Neptune Analytics グラフを有効にして、Virtual Private Cloud (VPC) の外部からアクセスできるようにします。この接続はデフォルトで無効になっています。グラフには IAM 認証が必要です。発信者は ID を取得し、グラフを使用するアクセス許可を持っている必要があります。たとえば、openCypher クエリを実行するには、呼び出し元に特定のグラフに対する読み取り、書き込み、または削除のアクセス許可が必要です。

グラフのプライベートエンドポイントを作成して、VPC 内からグラフにアクセスすることもできます。エンドポイントを作成するときは、VPC、サブネット、セキュリティグループを指定して、グラフを呼び出すアクセスを制限します。

転送中のデータを保護するために、Neptune Analytics は HTTPS 経由でグラフに SSL 接続を適用します。詳細については、Neptune Analytics ドキュメントの「Neptune Analytics でのデータ保護」を参照してください。

認証と認可を実装する

Neptune Analytics グラフを呼び出すには、IAM 認証が必要です。呼び出し元は ID を取得し、グラフでアクションを実行するための十分なアクセス許可を持っている必要があります。API アクションとその必要なアクセス許可の詳細については、Neptune Analytics API ドキュメントを参照してください。条件チェックを適用して、タグによるアクセスを制限できます。

IAM 認証では、AWS 署名バージョン 4 (SigV4) プロトコルを使用します。アプリケーションからの使用を簡素化するために、 AWS SDK を使用することをお勧めします。たとえば、Python では、SigV4 を抽象化する Neptune Graph の Boto3 クライアントを使用します。

グラフにデータをロードすると、バッチロードは発信者の IAM 認証情報を使用します。呼び出し元には、Neptune Analytics が Amazon S3 ファイルからグラフにデータをロードするロールを引き受けられるように、信頼関係を設定して Amazon S3 からデータをダウンロードするアクセス許可が必要です。

一括インポートは、グラフの作成時 (インフラストラクチャチーム) または既存の空のグラフ上 (インポートタスクを開始する権限を持つデータエンジニアリングチーム) に実行できます。どちらの場合も、Neptune Analytics は発信者が入力として提供する IAM ロールを引き受けます。このロールは、入力データがステージングされている Amazon S3 フォルダの内容を読み取って一覧表示するアクセス許可を付与します。

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、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機能が含まれます。

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 のすべての柱にわたってワークロードの動作に影響します。信頼性のために、このセクションで説明するように、従う必要がある特定のパターンがあります。

信頼性の柱は、次の主要分野に焦点を当てています。

- サービスクォータやデプロイパターンなどの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

AWS アカウントには、 ごとにデフォルトのクォータ (以前は制限と呼ばれていました) がありますAWS のサービス。特に明記していない限り、クォータはリージョン固有です。一部のクォータの引き上げをリクエストできますが、すべてではありません。

Neptune Analytics のクォータを確認するには、Service Quotas コンソールを開きます。ナビゲーションペインで、 を選択しAWS のサービス、Amazon Neptune Analytics を選択します。グラフとスナップショットの数、グラフの最大プロビジョニングメモリ、API リクエストレートのクォータに注意してください。

最大プロビジョニングメモリがデータセットに十分でない場合は、意図した分析用途に不可欠なノードとエッジタイプを評価します。データのサブセットをロードして、許容されるプロビジョニングされた容量内で分析できるようにします。多くの分析ワークロード、特にグラフアルゴリズムを実行するワークロードでは、完全なトランザクショングラフではなく、プロパティのセットが制限されたトポロジのみが必要です。(トランザクションワークロードと分析ワークロードの違いについては、「パフォーマンス効率の柱」セクションを参照してください)。

グラフの最大数が意図した用途に十分でない場合：

- 同様の用途を持つグラフを組み合わせることを検討してください。
- 一度に実行する必要があるグラフの数を評価します。エフェメラル分析のユースケースがある場合は、不要になったグラフをスナップショットして削除します。これにより、クォータに対するグラフの数が減少します。
- 異なる でグラフをプロビジョニングすることを検討してください AWS アカウント。

Neptune のデプロイパターンを理解する

Neptune Analytics グラフをデプロイする場合は、次の決定点を理解します。

- シード: Amazon S3、既存の Neptune データベースクラスター、または既存の Neptune データベーススナップショットのデータを使用して、空のグラフを作成するか、作成時にそのグラフにデータをロードするかを決定します。

  推奨事項: ソースが Neptune クラスターまたはスナップショットの場合は、グラフ作成時にデータをロードする必要があります。ソースが Amazon S3 の場合、データをロードする労力が重要で、インフラストラクチャのプロビジョニングアクティビティとして最適である場合は、作成時にデータをロードします。データエンジニアリングまたはアプリケーションアクティビティとしてデータをロードする場合は、空のグラフを作成し、後で Amazon S3 からデータをロードします。
- 容量: データサイズと予想されるアプリケーション使用量を考慮して、グラフに必要なプロビジョニングされた容量を見積もります。

  推奨事項: 作成時に、グラフサイズを制限する最大プロビジョニングメモリを指定します。この設定は必須です。必要に応じて、後で容量を変更できます。
- 可用性と耐障害性: 可用性にレプリカが必要かどうかを決定します。レプリカは、グラフに障害が発生した場合の復旧のためのウォームスタンバイとして機能します。レプリカを含むグラフは、レプリカを含まないグラフよりも速く回復します。また、グラフが必要な期間、エフェメラル分析専用かどうか、必要な場合はいつ削除されるかも考慮してください。

  推奨事項: グラフを作成する前に、グラフを使用できない時間や削除できる時間などの可用性要件を決定します。
- ネットワークとセキュリティ: パブリック接続、プライベート接続、またはその両方が必要かどうか、およびデータを暗号化するかどうかを決定します。

  推奨事項: グラフを作成する前に、パブリック接続が許可されているかどうか、グラフクライアントアプリケーションがデプロイされる場所など、組織の要件を理解します。

- バックアップとリカバリ: スナップショットを作成するかどうか、作成する場合は、いつ、どの条件でスナップショットを作成するかを決定します。組織にディザスタリカバリ (DR) 要件があるかどうかを検討します。

  推奨事項: スナップショットの作成は手動アクティビティです。スナップショットを作成するタイミングを決定し、グラフを作成する前に DR 要件を検討します。

Neptune クラスターの管理とスケーリング

Neptune Analytics グラフは、単一のメモリ最適化インスタンスで構成されます。インスタンスの容量 (m-NCU) は作成時に設定されます。インスタンスは、管理アクションを通じてプロビジョニングされた容量を増やすことで垂直方向にスケーリングできます。また、プロビジョニングされた容量を減らすこともできます。レプリカはパッシブフェイルオーバーターゲットであるため、グラフのスケールは増加しません。この点で、グラフレプリカは、アプリケーションからの読み取りオペレーションを処理できる Neptune クラスター内のアクティブなインスタンスである Neptune データベースリードレプリカとは異なります。

レプリカにはコストがかかります。レプリカの料金はグラフの m-NCU レートです。たとえば、グラフが 128 m-NCU に対してプロビジョニングされ、レプリカが 1 つある場合、コストはレプリカを持たない同等のグラフの 2 倍になります。

分析では、スケールアップの主な理由が 2 つあります。

- 分析クエリとアルゴリズムにより多くのメモリと CPU を提供するために、個々のクエリは高価であるため、実行するグラフアルゴリズムは本質的に複雑で、入力を考慮するとより多くのリソースを必要とするか、同時リクエストレートが高いです。このようなクエリでout-of-memoryエラーが発生した場合は、スケールアップが妥当な解決策です。
- 計画よりも大きいグラフサイズをサポートするには。たとえば、現在のプロビジョニングされた容量が 60 GB のソースデータをサポートするために 128 m-NCU で、さらに 40 GB のソースデータが必要な場合は、256 m-NCU への引き上げが必要です。

NumQueuedRequestsPerSec、、、、 などの Neptune Analytics NumOpenCypherRequestsPerSecの CloudWatch GraphStorageUsagePercentメトリクスをモニタリングしてGraphSizeBytesCPUUtilization、スケーリングが必要かどうかを判断します。グラフの設定は、コンソール AWS CLI、または SDKs を使用して更新できます。(例とベストプラクティスについては、「オペレーショナルエクセレンスの柱」セクションを参照してください)。

バックアップとフェイルオーバーイベントを管理する

レプリカを使用して、障害発生時にグラフを使用できるようにします。グラフはログベースの永続性を使用して、 のアベイラビリティーゾーン間で変更をコミットします AWS リージョン。レプリカはウォームスタンバイとして機能し、このデータにアクセスできます。失敗した場合、グラフはレプリカのオペレーションを再開します。アプリケーションは引き続き同じエンドポイントを使用してグラフに接続します。失敗中の処理中のリクエストは、サービス利用不可の例外を含むエラーを生成します。アプリケーションコードのバックオフパターンで再試行してエラーをキャッチし、短い間隔で再試行することを検討してください。フェイルオーバー中に行われた新しいリクエストはキューに入れられ、レイテンシーが長くなる可能性があります。

レプリカが設定されておらず、グラフが失敗した場合、Neptune Analytics は耐久性のあるストレージから復旧しますが、Neptune がリソースを再初期化する必要があるため、復旧に時間がかかります。

グラフのスナップショットを作成します。(Neptune Analytics は自動スナップショットを作成しません) 。作成後にグラフが定期的に変更される場合は、スナップショットを頻繁に作成して現在の状態をキャプチャします。以前の時点への復元が必要ない場合は、古いスナップショットを削除します。

スナップショットは、他の アカウントと共有できます AWS リージョン。DR 要件がある場合は、スナップショットから別のリージョンのグラフを復元することが、目標復旧時間 (RTO) と目標復旧時点 (RPO) の要件を満たしているかどうか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、以下を段階的かつ継続的に実行するプロセスです。

- ビジネス要件を確認する
- ワークロード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、Neptune Analytics にデータを取り込み、Neptune Analytics からデータを使用する際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
- グラフの正しいサイズ設定
- 書き込みの最適化

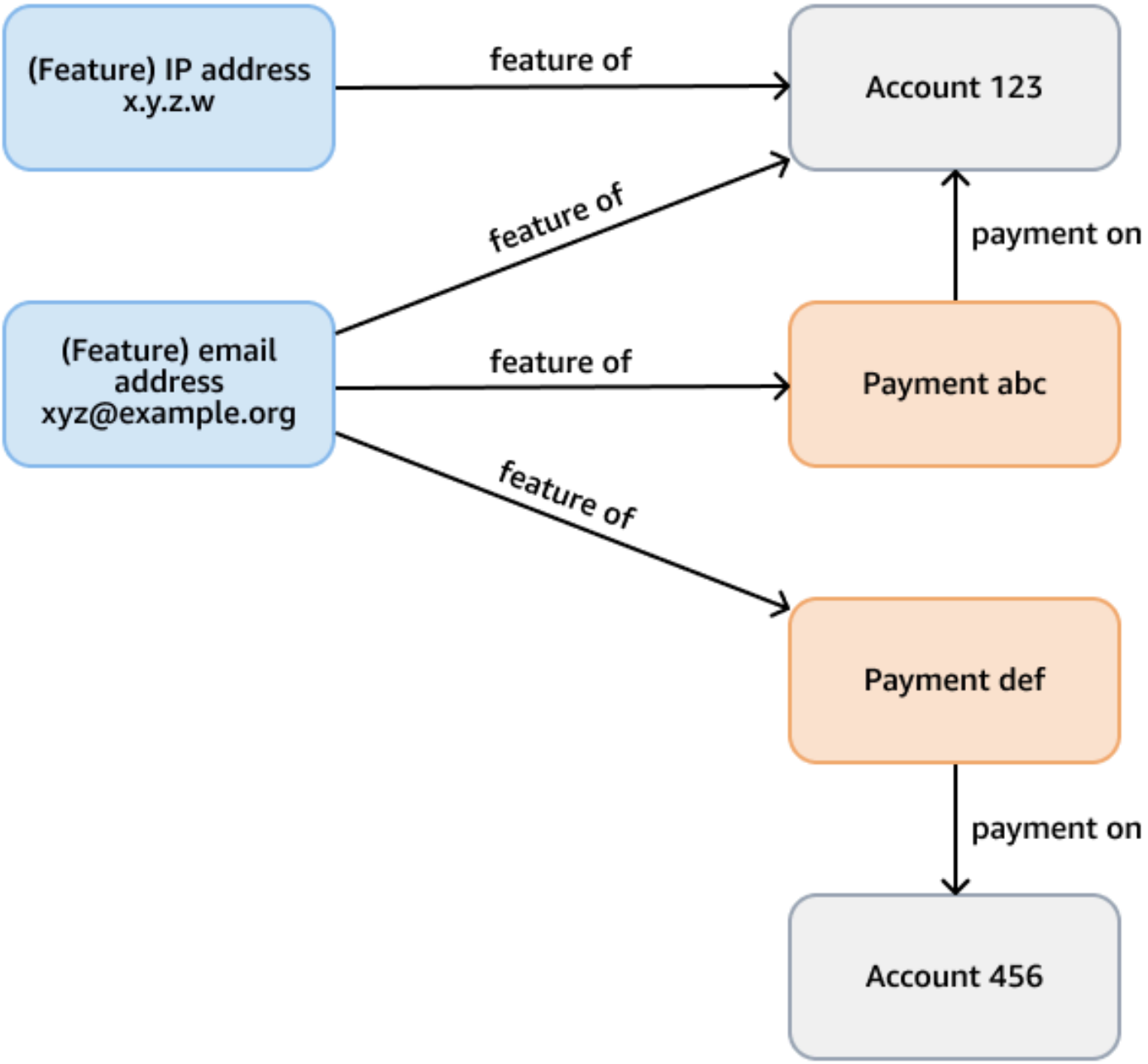
分析用のグラフモデリングを理解する

Amazon Neptune 用の AWS Well-Architected フレームワークの適用」ガイドでは、パフォーマンス効率のためのグラフモデリングについて説明します。パフォーマンスに影響するモデリングの決定には、必要なノードとエッジ、IDs、ラベルとプロパティ、エッジの方向、ラベルが汎用か固有か、一般的にクエリエンジンがグラフをナビゲートして一般的なクエリを処理できる効率の選択が含まれます。

これらの考慮事項は Neptune Analytics にも適用されますが、トランザクション使用パターンと分析使用パターンを区別することが重要です。Neptune データベースなどのトランザクションデータベースのクエリに効率的なグラフモデルは、分析のために再形成する必要がある場合があります。

例えば、Neptune データベースの不正グラフで、クレジットカード支払いの不正パターンをチェックする目的があるとします。このグラフには、アカウントと支払いの両方のアカウント、支払い、機

能 (E メールアドレス、IP アドレス、電話番号など) を表すノードがある場合があります。この接続されたグラフは、特定の支払いから始まる可変長パスを横断するなどのクエリをサポートし、関連する機能やアカウントを見つけるためにいくつかのホップを取ります。次の図は、このようなグラフを示しています。



分析要件は、機能によってリンクされたアカウントのコミュニティを検索するなど、より具体的な場合があります。この目的のために、弱に接続されたコンポーネント (WCC) アルゴリズムを使用できます。前の例のモデルに対して実行するには、いくつかの異なるタイプのノードとエッジをトラバースする必要があるため、非効率的です。次の図のモデルはより効率的です。アカウント自体、または

アカウントからの支払いが機能を共有している場合、accountノードをshares featureエッジにリンクします。たとえば、 Account 123には E メール機能 がありxyz@example.org、支払い () に同じ E メールAccount 456を使用しますPayment def。

WCC の計算の複雑さは です。ここでO(|E|logD)、 |E|はグラフ内のエッジ数、 Dはノードを接続する直径 (最長パスの長さ) です。トランザクションモデルはイネスシャルノードとエッジを省略するため、エッジの数と直径の両方を最適化し、WCC アルゴリズムの複雑さを軽減します。

Neptune Analytics を使用する場合は、必要なアルゴリズムと分析クエリから戻ります。必要に応じて、モデルを再構築してこれらのクエリを最適化します。グラフにデータをロードしたり、グラフ内の既存のデータを変更するクエリを記述する前に、モデルを再構築できます。

クエリを最適化する

Neptune Analytics クエリを最適化するには、次の推奨事項に従ってください。

- パラメータ化されたクエリとクエリプランキャッシュを使用します。これはデフォルトで有効になっています。プランキャッシュを使用すると、クエリが 100 ミリ秒以内に完了すれば、エンジンは後で使用できるようにクエリを準備します。これにより、後続の呼び出しにかかる時間を節約できます。
- スロークエリの場合は、説明プランを実行してボトルネックを特定し、それに応じて改善を行います。
- ベクトル類似度検索を使用する場合は、小さな埋め込みで正確な類似度結果が得られるかどうかを決定します。小さな埋め込みをより効率的に作成、保存、検索できます。
- Neptune Analytics で openCypher を使用するための文書化されたベストプラクティスに従います。たとえば、UNWIND 句でフラット化されたマップを使用し、可能な場合はエッジラベルを指定します。
- グラフアルゴリズムを使用する場合は、アルゴリズムの入力と出力、計算の複雑さ、および広範囲にわたる仕組みを理解します。

- グラフアルゴリズムを呼び出す前に、 MATCH句を使用して入力ノードセットを最小限に抑えます。例えば、幅優先検索 (BFS) を実行するノードを制限するには、Neptune Analytics ドキュメントに記載されている例に従います。
- 可能であれば、ノードラベルとエッジラベルをフィルタリングします。たとえば、BFS には、特定のノードラベル (vertexLabel) または特定のエッジラベル () へのトラバーサルをフィルタリングする入力パラメータがありますedgeLabels。
- 結果を制限するmaxDepthには、 などの境界パラメータを使用します。
- concurrency パラメータを試します。値を 0 にすると、使用可能なすべてのアルゴリズムスレッドを使用して処理を並列化できます。パラメータを 1 に設定して、シングルスレッド実行と比較します。アルゴリズムは 1 つのスレッドでより速く完了できます。特に、並列処理で実行時間が測定可能なほど短縮されず、オーバーヘッドが発生する可能性がある、幅が浅い最初の検索などの小さな入力で完了できます。
- 類似するタイプのアルゴリズムから選択します。例えば、Bellman-Ford とデルタステッピングはどちらも単一ソースの最短パスアルゴリズムです。独自のデータセットでテストする場合は、両方のアルゴリズムを試して結果を比較します。デルタステッピングは、計算の複雑さが低いため、Bellman-Ford よりも高速であることがよくあります。ただし、パフォーマンスはデータセットと入力パラメータ、特に deltaパラメータによって異なります。

書き込みの最適化

Neptune Analytics で書き込みオペレーションを最適化するには、次のプラクティスに従います。

- グラフにデータをロードする最も効率的な方法を探します。Amazon S3 のデータからロードする場合は、データが 50 GB を超える場合は一括インポートを使用します。小さいデータの場合は、バッチロードを使用します。バッチロードの実行時にout-of-memoryエラーが発生した場合は、m-NCU 値を増やすか、ロードを複数のリクエストに分割することを検討してください。これを実現する 1 つの方法は、S3 バケット内の複数のプレフィックスにファイルを分割することです。その場合は、プレフィックスごとにバッチロードを個別に呼び出します。
- 一括インポートまたはバッチローダーを使用して、グラフデータの初期セットを入力します。小さな変更に対してのみ、トランザクション openCypher の作成、更新、削除オペレーションを使用します。
- グラフに埋め込みを取り込むには、同時実行数が 1 (シングルスレッド) の一括インポートまたはバッチローダーを使用します。これらの方法のいずれかを使用して、埋め込みを事前にロードしてみてください。

- ベクトル類似度検索アルゴリズムの正確な類似度検索に必要なベクトル埋め込みのディメンションを評価します。可能であれば、より小さなディメンションを使用します。これにより、埋め込みのロード速度が向上します。
- 必要に応じて、ミューテーションアルゴリズムを使用してアルゴリズムの結果を記憶します。例えば、度可変中心性アルゴリズムは各入力ノードの度合いを見つけ、その値をノードのプロパティとして書き込みます。これらのノードを囲む接続が後で変更されない場合、 プロパティは正しい結果を保持します。アルゴリズムを再度実行する必要はありません。
- グラフリセット管理アクションを使用して、最初からやり直す必要がある場合は、すべてのノード、エッジ、埋め込みをクリアします。openCypher クエリを使用してすべてのノード、エッジ、埋め込みを削除することは、グラフが大きい場合は実行できません。大規模なデータセットに対する 1 回のドロップクエリはタイムアウトする可能性があります。サイズが大きくなると、データセットの削除に時間がかかり、トランザクションサイズも大きくなります。対照的に、グラフのリセットを完了する時間はほぼ一定であり、 アクションには、スナップショットを実行する前にスナップショットを作成するオプションがあります。

適切なサイズのグラフ

全体的なパフォーマンスは、Neptune Analytics グラフのプロビジョニングされた容量によって異なります。容量は、メモリ最適化 Neptune 容量ユニット (m-NCUs。グラフのサイズとクエリをサポートするのに十分なサイズがグラフにあることを確認してください。容量を増やしても、個々のクエリのパフォーマンスは必ずしも向上しないことに注意してください。

可能であれば、Amazon S3 や既存の Neptune クラスターまたはスナップショットなどの既存のソースからデータをインポートしてグラフを作成します。最小容量と最大容量に境界を設定できます。既存のグラフでプロビジョニングされた容量を変更することもできます。

NumQueuedRequestsPerSec、、NumOpenCypherRequestsPerSec、、 などの CloudWatch GraphStorageUsagePercentメトリクスをモニタリングCPUUtilizationしてGraphSizeBytes、グラフのサイズが適切かどうかを評価します。グラフのサイズと負荷をサポートするためにより多くの容量が必要かどうかを判断します。これらのメトリクスの一部を解釈する方法の詳細については、「オペレーショナルエクセレンスの柱」セクションを参照してください。

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、Neptune Analytics 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。

- 時間の経過に伴う支出と資金配分の制御について
- 適切なタイプと数量のリソースの選択
- 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 Analytics を採用する前に、ユースケースがグラフ分析に適しているかどうかを評価してください。

- グラフデータベース: Neptune などのグラフデータベースは、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。このユースケースでは、データがオブジェクトの属性としてより適切に表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 またはバイナリラージオブジェクト (BLOB) データ。

- グラフ分析: Neptune Analytics は、大量のグラフデータをメモリ内ですばやく分析してインサイトを取得し、傾向を見つけることができるグラフ分析データベースエンジンです。Neptune データベースと Neptune Analytics グラフの両方にグラフデータを保存およびクエリできます。Neptune データベースは、スケーラブルなオンライントランザクション処理 (OLTP) のニーズに最適です。Neptune Analytics は、エフェメラル分析ワークロードに最適です。この 2 つを組み合わせて使用するには、トランザクション指向の Neptune データベースから Neptune Analytics グラフにデータをロードして、そのデータの分析を実行します。分析が完了したら、Neptune Analytics グラフを削除できます。詳細については、Neptune Analytics ドキュメントの「Neptune Analytics を使用するタイミング」と「Neptune Database を使用するタイミング」を参照してください。

コストに注意して、Neptune Analytics グラフに入力する最適な方法を決定します。

- S3 バケットにステージングされたグラフデータを一括インポートします。このオプションは、データが以前に Neptune データベースへの一括ロード用にステージングされていた場合、または一括インポートに必要な CSV またはその他のサポートされている形式で分析されるデータがすでにあるか、簡単に生成できる場合にお勧めします。グラフ作成手順の一部として一括インポートを実行できます。最小容量と最大容量に境界を設定できます。 以前に作成した空のグラフでインポートを実行し、実行中にインポートタスクをモニタリングすることもできます。
- 空のグラフを作成し、バッチロードを使用して openCypher クエリで入力できます。このオプションは、ロードするデータが Amazon S3 でステージングされ、50 GB 未満である場合に最適です。
- Neptune データベースクラスターのデータからグラフを入力できます (Neptune データベースバージョン 1.3.0 以降でサポートされています) 。このパターンの目的は、現在グラフデータベースにあるデータに対して分析を実行することです。データベースが最初に一括ロードで入力された場合でも、それ以後大幅に変更された可能性があります。データベースからインポートするために、Neptune Analytics はデータベースのクローンを作成し、クローンから S3 バケットにデータをエクスポートします。この手順にはコストが発生します。特に、クローンを実行するための Neptune データベースコストと、エクスポートされたデータを保存および消費するための Amazon S3 コストです。エクスポートが完了すると、クローンは削除されます。Amazon S3 でエクスポートされたデータを削除できます。
- Neptune データベースクラスターのスナップショットからグラフを入力できます。これは、ソースがデータベーススナップショットである点を除いて、前のオプションと似ています。スナップショットからインポートするには、Neptune Analytics はまずスナップショットを新しいデータベースクラスターに復元し、次にデータを S3 バケットにエクスポートします。この手順ではコストが発生します。特に、復元されたクラスターを実行するための Neptune データベースコストと、エクスポートされたデータの保存と消費のための Amazon S3 コストが発生します。
- openCypher クエリを実行して、グラフでアトミック性、整合性、分離、耐久性 (ACID) 準拠のトランザクションを使用してデータを作成、更新、または削除することもできます。このアプローチは、小さな更新を行う方法としてお勧めしますが、グラフをシードする方法としてはお勧めしません。

分析に必要なデータが Amazon S3 ですでにステージングされている場合は、一括インポートまたはバッチロードをお勧めします。これらは、Neptune データベースクラスターまたはスナップショットからグラフを入力するよりもコスト効率が高くなります。

コストに注意してリソースを選択する

Neptune Analytics の料金は、メモリ最適化 Neptune キャパシティーユニット (m-NCU) と呼ばれる単位を使用します。特定の m-NCU でグラフを実行する場合、時間単位の固定コストが発生します。グラフにはフェイルオーバー用のレプリカが含まれている場合があり、これらのレプリカには時間単位の m-NCU コストも発生します。

容量の見積もり、コストの制限、パフォーマンスに対するコストのモニタリングには、次のベストプラクティスをお勧めします。

- 可能であれば、既存のソースからデータをインポートしてグラフを作成します。Amazon S3 または既存の Neptune クラスターまたはスナップショットでステージングされたデータです。これにより、Neptune Analytics はグラフのシードの重労働を実行し、バインドされた最大容量を指定できるため、労力を節約できます。
- 既存のグラフでプロビジョニングされた容量を変更できます。
- グラフが不要になったら、スナップショットを作成してグラフを削除できます。再度使用する必要がある場合は、スナップショットからグラフを復元できます。
- グラフの作成時にレプリカの数を選択できます。分析の可用性要件に従って値を設定します。この設定を最小化してコストを節約します。最大値 2 では、別々のアベイラビリティーゾーンに 2 つのレプリカインスタンスを使用できます。最小値 0 は、Neptune Analytics がレプリカを実行しないことを意味します。ただし、レプリカが利用可能な場合、復旧が速くなります。グラフの障害と復旧の説明については、「信頼性の柱」セクションを参照してください。
- を使用して、現在および過去の請求期間の Neptune Analytics の費用をモニタリングしますAWS Billing and Cost Management。
- CloudWatch、特に NumQueuedRequestsPerSec、、NumOpenCypherRequestsPerSec、GraphStorageUsagePercent よび GraphSizeBytesの Neptune Analytics メトリクスをモニタリングしてCPUUtlization、プロビジョニングされた容量がグラフに対して適切なサイズであるかどうかを評価します。小さい容量が観測されたリクエストレート、CPU 使用率、グラフサイズに対応できるかどうかを確認します。
- グラフにプライベートエンドポイントが必要な場合は、Elastic IPs、Virtual Private Cloud (VPC) エンドポイント、NAT ゲートウェイ、またはその他の VPC 関連のコストに注意してください。詳細については、「Amazon VPC の料金」とAmazon EC2 の料金」を参照してください。
- 開発者やアナリストがグラフをクエリおよび視覚化するのに役立つクライアントインターフェイスを提供するために、1 つ以上の Neptune ノートブックインスタンスを実行することもできます (「Neptune ワークベンチの料金」を参照)。コストを最小限に抑えるには、ユーザー間でインス

タンスを共有し、ユーザーごとに個別のノートブックフォルダを作成します。インスタンスが使用されていない場合はシャットダウンします。シャットダウンを自動化する方法については、ブログ記事 AWS「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。

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重点を置いています。主なトピックは、持続可能性に関する責任共有モデル、影響の把握、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率の最大化です。

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。

- ユーザーによる影響
- 持続可能性の目標
- 使用率の最大化
- より効率的な新しいソフトウェアサービス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、影響の理解に焦点を当てています。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。

ユーザーによる選択と要件は、環境に影響を及ぼします。炭素強度の低い 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 AWS リージョン が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 選択を検討する

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 Analytics が利用可能なリージョンとリストを相互参照します。

消費の最適化

以下を実践することで、Neptune Analytics の消費量を最小限に抑えます。

- 多くの場合、分析は一時的なものです。グラフは、アルゴリズムを実行して結果を記録する時間にのみ必要です。この場合、不要になったグラフをスナップショットして削除します。必要に応じて、後でスナップショットから復元できます。
- ワークロードがエフェメラルで、分析を実行するタイミングを柔軟に決定できる場合は、電力消費量のday-to-day傾向を考慮してください。電力の需要は、特定の時間帯に高くなります。米国にお住まいの場合は、米国エネルギー情報局 (EIA) のウェブサイトで毎日の電力消費量に関するメトリクスを参照してください。可能であれば、リージョンのオフピーク期間中にワークロードを実行します。
- ワークロードがエフェメラルではなく、限られた期間のみ使用する必要がある場合は、グラフを削除し、必要に応じてスナップショットから復元します。可用性がスケジュールに従っている場合は、スケジュールされた時刻にグラフの準備ができるように、スクリプトを使用して復元プロセスを自動化します。
- データが読み取り専用であるか、前回のスナップショット以降に変更されていない場合は、削除前に再度スナップショットを作成しないでください。
- 使用していない Neptune ノートブックを停止します。
- NumQueuedRequestsPerSec、、NumOpenCypherRequestsPerSec、GraphStorageUsagePercent などの CloudWatch メトリクスをモニタリングCPUUtilizationしてGraphSizeBytes、グラフのサイズが大きすぎるかどうかを評価します。小さいインスタンス容量が観測されたリクエストレート、CPU 使用率、グラフサイズに対応できるかどうかを確認します。

ソフトウェア開発とアーキテクチャ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可能なすべてのリソースを活用します。具体的なベストプラクティスは次のとおりです。

- クエリとグラフアルゴリズムの呼び出しを最適化します。パラメータ化されたクエリを使用し、デフォルトで有効になっているクエリプランキャッシュを使用します。スロークエリの場合は、説明プランを実行して改善を行います。ベクトル類似度検索を使用する場合は、小さい埋め込みの方がより効率的に作成、保存、検索できるため、小さい埋め込みの方が正確な類似度結果が得られるかどうかを判断します。グラフアルゴリズムを呼び出す前に、 MATCH句を使用して入力ノードセットを最小限に抑えます。可能であれば、ノードラベルとエッジラベルをフィルタリングします。
- グラフにデータをロードする最も効率的な方法を探します。Amazon S3 のデータからロードする場合は、データが 50 GB を超える場合は一括インポートを使用します。小さなデータにはバッチロードを使用します。

- 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 ノートブックインスタンスを共有するように依頼します。1 つの Jupyter インスタンスで開発者ごとに個別のノートブックフォルダを作成します。インスタンスが使用されていない場合はシャットダウンします。

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 用の AWS Well-Architected フレームワークの適用
- Neptune Analytics のベストプラクティス

ブログ投稿と動画

- Neptune ブログ投稿 (AWS データベースブログ)
- 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 (AWS データベースブログ)
- Amazon Neptune の詰め物 (短い YouTube 動画)

トレーニング

- Amazon Neptune の開始方法
- Amazon Neptune で構築する
- Data Modeling for Amazon Neptune
- Amazon Neptune Analytics ワークショップ

ドキュメント履歴

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

変更	説明	日付
初版発行	—	2024 年 12 月 20 日

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。