Unable to locate subtitle

AWS Well-Architected Framework

AWS Well-Architected Framework: ***Unable to locate subtitle***

Table of Contents

AWS Well-Architected Framework

公開日: 2022 年 10 月 20 日 (改訂履歴)

AWS Well-Architected フレームワークは、AWS でシステムを構築する際に行う決定の長所と短所を理解するのに役立ちます。効率が良く、費用対効果が#く、安全で信頼のおけるクラウド対応システムを設計して運#するために、アーキテクチャに関するベストプラクティスをこのフレームワークに従って学ぶことができます。

はじめに

AWS Well-Architected フレームワークは、AWS でシステムを構築する際に行う決定の長所と短所を理解するのに役立ちます。このフレームワークを利用すると、安全で信頼性が高く、効率的で、費用対効果が高く、持続可能なワークロードを AWS クラウド で設計および運用するための、アーキテクチャに関するベストプラクティスを学ぶことができます。フレームワークにより、アーキテクチャをベストプラクティスに照らし合わせて一貫的に測定し、改善すべき点を特定する手段を提供します。アーキテクチャのレビュープロセスは、アーキテクチャに関する決定についての前向きな話し合いであって、監査過程ではありません。当社では、Well-Architected システムを持つことが、ビジネスで成功を収める可能性を大幅に向上させると考えています。

AWS ソリューションアーキテクトは、さまざまな業種やユースケースに対応したソリューションのアーキテクトとして長年の経験を持っています。これまで何千ものお客様の AWS でのアーキテクチャの設計とレビューをお手伝いしてきました。その経験に基づいて、クラウド対応システムを設計するための核となる戦略とベストプラクティスを確立しました。

AWS Well-Architected フレームワークは、特定のアーキテクチャがクラウドのベストプラクティスと整合しているかどうかを理解するための一連の基本的な質問を文書化したものです。このフレームワークは、現代のクラウドベースのシステムに期待する品質を評価するための一貫したアプローチと、その品質を達成するために必要な対応を提供します。AWS が進化し続け、お客様との共同作業からより多くを学び続ける中で、Well-Architected (よくできたアーキテクチャ) の定義に磨きをかけていきます。

このフレームワークは、最高技術責任者 (CTO)、設計者、デベロッパー、オペレーションチームメンバーなどの技術担当者を対象としています。本書では、クラウドワークロードを設計、運用する際に使用する AWS のベストプラクティスや戦略について説明し、さらなる実装の詳細やアーキテクチャパターンへのリンクも提供しています。詳細については、 AWS Well-Architected ホームページへ.

AWS では、ワークロードをレビューする無料サービスも提供しています。それらの AWS Well-Architected ツール (AWS WA ツール) は、AWS Well-Architected フレームワークを使用してアーキテクチャをレビューおよび測定するための一貫したプロセスを提供するクラウド上のサービスです。AWS WA ツールは、ワークロードの信頼性、安全性、効率性、コスト効率を高めるための推奨事項を提供します。

ベストプラクティスの適用を支援するために、 AWS Well-Architected ラボこのラボでは、コードとドキュメントのリポジトリを使用して、ベストプラクティスの実装を実践的に体験できます。また、右記のプログラムに参加しているAWS パートナーネットワーク (APN) パートナーと提携しています : AWS Well-Architected パートナープログラム.これらの AWS パートナーは AWS に関する深い知識を持っており、ワークロードのレビューと改善をサポートします。

定義

AWS のエキスパートは、クラウドのベストプラクティスを活用したシステムの構築において、#々お客様を支援しています。私たちは、設計が進化するにつれて発生するアーキテクチャとのトレードオフをお客様とともに考えてきました。お客様がライブ環境にシステムをデプロイするたびに、当社はそのシステムの実際のパフォーマンスやトレードオフの結果を学びます。

当社はその学びに基づいて、AWS Well-Architected フレームワークを確立しました。このフレームワークでは、お客様とパートナーがアーキテクチャを評価するための一貫したベストプラクティスや、アーキテクチャが AWS のベストプラクティスにどれだけ準拠しているのかを評価するための質問を提供しています。

AWS Well-Architected 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、サステナビリティという 6 つの柱に基づいています。

表 1AWS Well-Architected フレームワークの柱

名前	説明
運用上の優秀性	開発をサポートし、ワークロードを効率的に実行し、運用に関する洞察を得て、ビジネス価値をもたらすためのサポートプロセスと手順を継続的に改善する能力。
セキュリティ	セキュリティの柱は、クラウドテクノロジーを活用し、セキュリティ体制の向上を可能にする

名前	説明
	やり方でデータ、システム、資産を保護する方法を表します。
信頼性	信頼性の柱には、意図した機能を期待どおりに正しく一貫して実行するワークロードの能力が含まれます。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。本資料では、AWS に信頼性の高いワークロードを実装するための、詳しいベストプラクティスのガイダンスを提供します。
パフォーマンス効率	コンピューティングリソースを効率的に使ってシステム要件を満たし、需要の変化と技術の進化に合わせてこの効率性を維持する能力です。
コスト最適化	最も低い価格でシステムを運用してビジネス価値を実現する能力。
サステナビリティ	プロビジョニングされたリソースのメリットを最大化し、必要な合計リソースを最小化することにより、エネルギー消費を削減し、ワークロードのすべてのコンポーネントにおいて効率を向上させ、持続可能性への影響を継続的に改善する能力。

AWS Well-Architected フレームワークでは、以下の用語を使用します。

- コンポーネントとは、コード、設定、AWS リソースのことで、これらを組み合わせて要件に対応するものです。コンポーネントは多くの場合、技術的所有権の単位であり、他のコンポーネントとは切り離されています。
- ワークロードという用語は、ビジネス価値を実現する一連のコンポーネントを識別するために使用されます。ワークロードの詳細レベルは通常、ビジネスリーダーとテクノロジーリーダーが話し合いで決定します。

- アーキテクチャとは、 コンポーネントがワークロードで 連携する方法であると考えます。コンポーネントが通信や対話を行う方法は、アーキテクチャ図の中心となることがよくあります。
- Milestones/マイルストーン は、アーキテクチャにおける設計、テスト、稼働、本番という製品のライフサイクル全体を通した進化において、重要な変更を記録します。
- 組織内の テクノロジーポートフォリオは、 ビジネスの運用に必要なワークロードの集合体です。
- それらの 工数レベル は、タスクの実行に必要な時間、労力、複雑さの度合いを分類したものです。各組織は、組織の工数レベルを適切に分類するために、チームの規模や専門性、ならびにワークロードの複雑性など、補足的なコンテキストを考慮する必要があります。
  - 高: 作業には数週間または数か月かかる可能性があります。これは複数のストーリー、リリース、およびタスクに分割することができます。
  - 中: 作業には数日または数週間かかる可能性があります。これは複数のリリースおよびタスクに分割することができます。
  - 低: 作業には数時間または数日かかる可能性があります。これは複数のタスクに分割することができます。

ワークロードを設計するときには、ビジネスの状況に応じて各柱の間でトレードオフを行います。これらのビジネス上の決定により、エンジニアリングの優先順位を決めることができます。開発環境では、信頼性を犠牲にして持続可能性への影響を改善し、コストを削減するために最適化するかもしれませんし、ミッションクリティカルなソリューションでは、コストと持続可能性への影響を増加させて信頼性を最適化するかもしれません。e コマースソリューションでは、パフォーマンスが収益と顧客の購買傾向に影響することがあります。セキュリティと運用性は、通常、他の柱とトレードオフされることはありません。

アーキテクチャについて

オンプレミス環境では、多くの場合、テクノロジーアーキテクチャの中心チームがあり、製品や機能を担当する他のチームがベストプラクティスに従うように、まとめ役として機能します。テクノロジーアーキテクチャチームには、通常、テクニカルアーキテクト (インフラストラクチャ)、ソリューションアーキテクト (ソフトウェア)、データアーキテクト、ネットワーキングアーキテクト、セキュリティアーキテクトなどの担当者が含まれます。テクノロジーアーキテクチャチームでエンタープライズアーキテクチャ機能の一部としてよく使用されるのが、 TOGAF や Zachman フレームワーク です。

AWS では、能力を持つチームを一元化するのではなく、各チームに能力を分散させることを好んでいます。決定権限を分散することには、複数のチームを内部標準に準拠させるといった点でリスクが

あります。当社はそのようなリスクを 2 つの方法で軽減します。第 1 の方法は、各チームがその職能を持てるようにするための プラクティス (物事の進め方、プロセス、基準、通念) であり、チームが満たすべき基準の水準を確実に引き上げるために専門家が配置されています。次に、実装する メカニズム を導入して、標準への準拠を徹底させることです。

> 「善意だけでは十分ではない。仕組みづくりが重要だ」- ジェフ ベゾス。

つまり、人間の最善の努力を、ルールやプロセスへの準拠をチェックするメカニズム (多くは自動化) に置き換えるということです。分散というこの方法は Amazon のリーダーシッププリンシパルに基づいており、お客様を起点として 考える 文化をすべての職務で確立します。逆行は当社のイノベーションプロセスの基本です。当社はお客様とその要望を出発点に当社の取り組みを定義して進めます。お客様のことを真剣に考えているチームが、お客様の要件に応じて商品を開発します。

アーキテクチャについては、すべてのチームがアーキテクチャを作成し、ベストプラクティスに従う能力があることを期待しています。新しいチームがこれらの能力を獲得するため、あるいは既存のチームがそのレベルを上げるために、彼らの設計をレビューし、AWS のベストプラクティスを理解するのに役立つプリンシパルエンジニアの仮想コミュニティへのアクセスを可能にしています。プリンシパルエンジニアリングコミュニティの仕事はベストプラクティスを周知させ、わかりやすくすることです。その方法の例としては、ベストプラクティスを実例に適用することについてランチタイムトークで取り上げます。その話を録音して、新しいチームメンバー向けのオンボーディング教材として使用できます。

AWS のベストプラクティスは、インターネット規模で数千のシステムを運用してきた当社の経験から生まれました。ベストプラクティスの定義には主にデータを活用しますが、プリンシパルエンジニアなどの専門分野に精通した人がベストプラクティスを設定することもあります。プリンシパルのエンジニアは、新しいベストプラクティスチームとして機能するようにコミュニティに従います。やがてそれらのベストプラクティスは当社の内部評価プロセスやコンプライアンス遵守メカニズムに取り込まれて正式なものになります。Well-Architected フレームワークは当社の内部評価プロセスを顧客向けに実施しているものであり、フィールドの役割全体で、ソリューションアーキテクチャや内部エンジニアリングチームなどのプリンシパルエンジニアリングの考えが体系化されています。Well-Architected フレームワークは、学んだことを活用できるスケーラブルなメカニズムです。

プリンシパルエンジニアリングコミュニティが取り組んでいるアーキテクチャの分散所有に従うことによって、お客様の要件に基づいて Well-Architected のエンタープライズアーキテクチャが生まれると当社は確信しています。テクノロジーリーダー (CTO や開発マネージャーなど) がお客様のワークロードのすべてに対して Well-Architected の評価を実施することで、お客様のテクノロジーポート

フォリオのリスクがよくわかるようになります。このアプローチを使用して、チーム全体に関わる課題を特定し、その課題に取り組むことができます。プリンシパルエンジニアはメカニズムや、トレーニング、ランチタイムトークを活用して、特定の領域についての考えを複数のチームで共有することができます。

一般的な設計原則

AWS Well-Architected フレームワークは、クラウド上における適切な設計を可能にする一般的な設計の原則を提供します。

- キャパシティーニーズの推測が不要: ワークロードのデプロイ時にキャパシティーを決定する場合、費用のかかるアイドル状態のリソースが発生したり、キャパシティーの制約によるパフォーマンスへの影響に対処することになる可能性があります。クラウドコンピューティングにはこのような問題はありません。必要な分のみキャパシティーを使用し、自動的にスケールアップまたはスケールダウンできます。
- 本稼働スケールでシステムをテストする: クラウドでは、オンデマンドで本稼働規模のテスト環境を作成し、テストが完了したらリソースを削除することができます。テスト環境の#払いは実#時にのみ発#するため、オンプレミスでテストを実施する場合と比べてわずかなコストで、本番環境をシミュレートできます。
- #動化によってアーキテクチャでの実験が容易に: 自動化により、低コストでワークロードを作成、レプリケートすることが可能になり、手作業による負担を回避できます。#動化に対する変更を追跡し、影響を監査して、必要な場合は以前のパラメータに戻すことができます。
- 発展するアーキテクチャが可能に: 従来の環境では、アーキテクチャに関する決定は 1 回限りの静的イベントとして実装されることが多く、システムの存続期間中に主要なバーションがいくつか発生します。ビジネスとその状況が進化し続けるにしたがって、当初の決定によって、変化するビジネス要件にシステムが対応できなくなる可能性があります。クラウド上では、自動化し、オンデマンドでテストできるので、設計変更によって#じる影響のリスクを軽減できます。そのため、イノベーションを標準プラクティスとしてビジネスで活用できるように、システムを時間とともに進化させることができます。
- データに基づいてアーキテクチャを駆動: クラウドでは、アーキテクチャの選択がワークロードの動作に与える影響に関するデータを収集できます。これにより、ワークロードの改善について事実に基づいた意思決定を#うことができます。クラウドのインフラストラクチャはコードですので、そのデータに基づいてアーキテクチャに関する選択と改善を徐々に進めることができます。
- ゲームデーを利用して改善する: ゲームデーを定期的にスケジュールし、本番環境のイベントをシミュレートすることで、アーキテクチャとプロセスのパフォーマンスをテストします。これは、改善できる箇所を把握し、組織がイベントに対応することを経験するのに役#ちます。

フレームワークの柱

ソフトウェアシステムの作成はビルの建設に似ています。基礎がしっかりしていなければ、ビルの健全性と機能を損なう構造上の問題が発生することがあります。技術ソリューションを設計する際、運用性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、および持続可能性の 6 本の柱を疎かにすると、要件に従って意図したとおりに稼働するシステムの構築は困難になるでしょう。これらの柱をアーキテクチャに組み込むことで、安定した効率的なシステムを作成することができます。こうすることで、要求される機能など設計の他の要素に集中できます。

柱

- 運用上の優秀性
- セキュリティ
- 信頼性
- パフォーマンス効率
- コスト最適化
- サステナビリティ

運用上の優秀性

運用上の優秀性の柱には、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ための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。

運用上の優秀性の柱では、設計原則、ベストプラクティス、質問の概要について説明します。実装に関する規範的なガイダンスとして 運用上の優秀性の柱に関するホワイトペーパーを参照してください。.

トピック

- 設計原則
- 定義
- ベストプラクティス
- リソース

設計原則

クラウドでの運用上の優秀性には次の 5 つの設計原則があります。

- 運用をコードとして実行する: クラウドでは、アプリケーションコードに使用するものと同じエンジニアリング原理を、環境全体に適用できます。ワークロード全体 (アプリケーション、インフラストラクチャ) をコードとして定義し、コードを使用して更新できます。運用手順をコードとして実装し、イベントに対応してそのコードをトリガーすることで自動的に実行できます。運用をコードとして実行することで、人為的なミスを抑制し、イベントへの一貫性のある対応を実現できます。
- 小規模かつ可逆的な変更を頻繁に行う: コンポーネントを定期的に更新できるようにワークロードを設計します。変更は、失敗した場合に元に戻すことができるように小規模に行います (可能な場合は、顧客に影響がないようにします)。
- 運用手順を頻繁に改善する: 運用手順を使用する際に、それらを改善する機会を探します。ワークロードを改良するときに、手順もそれに応じて改良します。定期的なゲームデーを計画し、すべての手順が効果的で、チームがその手順を熟知していることを確認および検証します。
- 障害を予想する: 「プレモータム」演習を実施して、潜在的な障害の原因を特定して排除または軽減できるようにします。障害シナリオをテストし、その影響に関する理解を検証します。対応手順をテストし、手順が効果的で、チームが手順の実行を十分に理解していることを確認します。定期的なゲームデーを計画し、ワークロードと、シミュレートされたイベントに対するチームの応答をテストします。
- 運用上の障害すべてから学ぶ: すべての運用イベントと障害から学んだ教訓を通じて、改善を促進します。チーム間と組織全体で教訓を共有します。

定義

クラウドにおける「運用上の優秀性」には 4 つのベストプラクティス領域があります。

- 組織
- 準備
- 運用
- 進化

組織のリーダーシップは、ビジネス目標を定義します。組織は、要件と優先順位を理解し、これらを使用してビジネスの成果を達成するための作業を整理し、指導する必要があります。ワークロードは

サポートに必要な情報を送出する必要があります。ワークロードの統合、デプロイ、提供を可能にするサービスを実装することで、反復プロセスが自動化され、本番環境への有益な変化の流れを増やすことができます。

ワークロードの運用に固有のリスクが存在する可能性があります。本番環境へ移行するためにこれらのリスクを理解し、十分な情報に基づく決定を下す必要があります。チームがワークロードをサポートできる必要があります。希望するビジネス上の成果から得られたビジネスおよび運用上のメトリクスにより、ワークロードの状態や運用上のアクティビティを把握し、インシデントに対応できます。優先順位はビジネスニーズやビジネス環境の変化に応じて変化します。これらをフィードバックループとして使用して、組織とワークロードの運用を継続的に改善します。

ベストプラクティス

トピック

組織

チームは、ビジネスの成功を実現する優先順位を設定するために、ワークロード全体、その役割、共有されるビジネス目標に関する理解を共有する必要があります。優先順位を明確に定義することで、努力を通じて得られるメリットが最大限に活かされます。ビジネス、開発、運用チームなど、主要な利害関係者が関わる社内外の顧客のニーズを評価し、重点領域を決定します。顧客ニーズを評価することにより、ビジネス成果を達成するために必要なサポートについて十分に理解できるようになります。組織のガバナンスによって定義されたガイドラインや義務、および特定の重点領域の必須化や重視が必要となる可能性のある規制コンプライアンス要件や業界標準などの外部要因をしっかりと認識します。内部ガバナンスおよび外部コンプライアンス要件への変更を識別するメカニズムがあることを検証します。要件が特定されていない場合は、必ずこの決定にデューデリジェンスが適用されていることを確認します。ニーズの変化に応じて更新できるように、優先順位を定期的に確認します。

ビジネスに対する脅威 (たとえば、ビジネスリスクと負債や情報セキュリティの脅威) を評価し、この情報をリスクレジストリに保持します。リスクの影響を評価し、競合する利益のトレードオフや代替アプローチを評価します。たとえば、新しい機能の市場投入までの時間を短縮することは、コストの最適化よりも重視されます。または、非リレーショナルデータ用にリレーショナルデータベースを選択すれば、リファクタリングなしで、システムの移行が簡素化されます。メリットとリスクを管理

し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。一部のリスクや選択肢は、一定期間許容される可能性があり、関連するリスクを軽減できる場合もあれば、リスクが残るのを容認できない場合もあります。その場合、リスクに対処するための措置を講じることになります。

チームはビジネスの成果を達成するうえでの役割を理解する必要があります。チームは他のチームが成功するためのそれぞれの役割を理解し、自分たちのチームが成功するための他のチームの役割を理解し、目標を共有する必要があります。責任、所有権、意思決定方法、意思決定を行う権限を持つユーザーを理解することは、労力を集中的に投入し、チームの利点を最大化するのに役立ちます。チームのニーズは、サポートする顧客、組織、チームの構成、およびワークロードの特性によって形成されます。1 つの運用モデルによって、組織内のすべてのチームとそのワークロードをサポートできると期待するのは合理的ではありません。

アプリケーション、ワークロード、プラットフォーム、インフラストラクチャの各コンポーネントの所有者が特定されていること、および各プロセスと手順の定義を担当する所有者、およびそのパフォーマンスに責任を持つ所有者が特定されていることを確認します。

各コンポーネント、プロセス、手順のビジネス価値、それらのリソースが配置されている理由やアクティビティが実行されている理由、所有権が存在する理由を理解することで、チームメンバーのアクションが明らかになります。チームメンバーの責任を明確に定義することで、当該メンバーが適切に行動し、責任と所有権を識別するメカニズムを持つことができます。イノベーションを制約しないように、追加、変更、例外をリクエストするメカニズムを備えます。チームがどのように連携して相互にサポートするのか、また、ビジネスの成果について、チーム間の合意を定義します。

チームメンバーにサポートを提供することで、チームメンバーがより効果的に行動し、ビジネスの成果をサポートできるようにします。業務を委嘱された上級リーダーは、目標値を設定し、成功を測定する必要があります。上級リーダーは、ベストプラクティスの採用と組織の進化の協賛者、支持者、および推進者である必要があります。影響を最小限に抑えるために、結果にリスクがある場合は措置を講じるようにチームメンバーの意識を高めるとともに、リスクに対処し、インシデントを回避できるようにするため、リスクがあるとチームメンバーが考える場合は、意思決定者や利害関係者にエスカレーションすることを推奨します。チームメンバーがタイムリーで適切な措置を講じることができるように、既知のリスクと計画されたイベントについて、適時かつ明確で実用的なコミュニケーションを行います。

学習を加速し、チームメンバーが関心と当事者意識を持ち続けるための実験を推奨します。チームは、新しいテクノロジーを採用し、需要と責任の変化をサポートするために、スキルセットを強化する必要があります。学習に専念するために設定された時間を提供することで、これをサポートし、推奨します。チームメンバーが成功し、ビジネスの成果をサポートするためにスケールできるように、ツールとチームメンバーの両方のリソースを持っていることを確認します。組織間の多様性を活用

して、複数のユニークな視点を追求します。この視点を使用して、イノベーションを高め、想定に挑み、確証バイアスに傾くリスクを軽減します。チーム内のインクルージョン、多様性、アクセシビリティを向上させ、有益な視点を得ます。

組織に適用される外部の規制またはコンプライアンス要件がある場合は、AWS クラウドコンプライアンス が提供するリソースを使用して、優先順位への影響を判断できるようにチームを教育する必要があります。Well-Architected フレームワークは学習、測定、改善に重点を置いています。アーキテクチャを評価し、時間の経過とともにスケールする設計を実装するための一貫したアプローチを提供します。AWS は、AWS Well-Architected Tool が開発前にアプローチを、本番稼働前にワークロードの状態を、本番稼働中にワークロードの状態をレビューするのに役立ちます。最新の AWS アーキテクチャのベストプラクティスと比較して、ワークロードの全体的な状態をモニタリングし、潜在的なリスクについてのインサイトを得ることができます。AWS Trusted Advisor は、最適化を推奨する中心的なチェックへのアクセスを提供するツールで、優先順位を決定するのに役立ちます。ビジネスおよびエンタープライズサポートの顧客は、優先順位をさらに高めることができるセキュリティ、信頼性、パフォーマンス、コストの最適化に重点を置いた追加のチェックにアクセスできます。

AWS は、AWS とそのサービスについてチームを教育し、選択がどのようにワークロードに影響を与えるかについての理解を深めるのに役立ちます。チームを教育するには、AWS (AWS ナレッジセンター、AWS ディスカッションフォーラム、AWS Support センター) および AWS ドキュメントが提供するリソースを使用する必要があります。AWS の質問については、AWS Support センターから AWS Support を参照してください。AWS は、Amazon Builders' Library の AWS の運用を通じて学んだベストプラクティスとパターンも共有しています。AWS ブログと公式の AWS ポッドキャストでは、その他さまざまな有益情報を入手できます。AWS トレーニングと認定では、AWS の基礎に関するセルフペースデジタルコースを通じて無料のトレーニングを提供しています。また、インストラクターが実施するトレーニングに登録して、チームの AWS スキルの開発をさらにサポートすることもできます。

AWS Organizations などの、アカウント間で環境を一元管理できるツールやサービスを使用して、運用モデルの管理を支援する必要があります。AWS Control Tower などのサービスでは、この管理機能が拡張されており、アカウントのセットアップに関する設計図 (運用モデルのサポート) を定義し、AWS Organizations を使用して進行中のガバナンスを適用し、新しいアカウントのプロビジョニングを自動化できます。AWS Managed Services、AWS Managed Services パートナー、または AWS パートナーネットワークのマネージドサービスプロバイダをはじめ、各種のマネージドサービスプロバイダは、専門的な実装型のクラウド環境を提供し、セキュリティとコンプライアンスの要件、ビジネスの目標をサポートしています。マネージドサービスを運用モデルに追加すると、時間とリソースを節約でき、新しいスキルや能力を開発するのではなく、戦略的成果に集中して社内チームを維持できます。

以下の質問は、運用の優秀性に関する考慮事項に焦点を当てています。(運用の優秀性に関する質問、回答、ベストプラクティスの一覧については、 付録を参照してください)。

OPS 1: 優先順位はどのように決定すればよいですか?

だれもが、ビジネスを成功させるうえで自分が果たす役割を理解する必要があります。リソースの優先順位を設定するため、共通の目標を設定してください。これにより、取り組みから得られるメリットが最大化されます。

OPS 2: ビジネスの成果をサポートするために、組織をどのように構築すればよいですか?

チームはビジネスの成果を達成するうえでの役割を理解する必要があります。チームは他のチームが成功するためのそれぞれの役割を理解し、自分たちのチームが成功するための他のチームの役割を理解し、目標を共有する必要があります。責任、所有権、意思決定方法、意思決定を行う権限を持つユーザーを理解することは、労力を集中的に投入し、チームの利点を最大化するのに役立ちます。

OPS 3: 組織の文化はビジネスの成果をどのようにサポートするのですか?

チームメンバーにサポートを提供することで、チームメンバーがより効果的に行動し、ビジネスの成果をサポートできるようにします。

ある時点で、優先順位の小さなサブセットに注力したい場合に遭遇するかもしれません。必要な機能の開発とリスクの管理を確実にするために、長期的にバランスのとれたアプローチを使用します。優先順位を定期的に見直し、ニーズの変化に応じて優先順位を更新します。責任と所有権が未定義または不明な場合、必要な活動をタイムリーに処理せず、これらのニーズに対応するために重複し、競合する可能性のある取り組みが発生するリスクがあります。組織文化は、チームメンバーのジョブに対する満足度と定着率に直接影響します。チームメンバーのやる気と能力を引き出して、ビジネスの成功につなげます。イノベーションを起こし、アイデアを成果に変えるには、実験が必要です。望ましくない結果は、成功につながらないパスを特定することに成功した実験であると認識します。

準備

運用上の優秀性を準備するには、ワークロードと期待される動作を理解する必要があります。そうすることでワークロードの状況を把握し、ワークロードをサポートする手順を構築するように設計できます。

ワークロードを設計する際には、可観測性と問題調査への対応においてすべてのコンポーネントにわたって内部状態 (メトリクス、ログ、イベント、トレースなど) を理解するために必要な情報が送出されるようにします。ワークロードの稼働状態を監視し、結果にリスクがあった場合にそれを特定し、効果的な対応を可能にするために必要なテレメトリの開発を繰り返します。ワークロードを計測する際は、フィルターを使用して時間の経過とともに最も有用な情報を選択できるので、状況認識を可能にする幅広い情報 (状態の変化、ユーザーのアクティビティ、権限アクセス、使用量のカウンターなど) を取得します。

本番環境への変化の流れを改善し、リファクタリング、品質に関する迅速なフィードバック、バグ修正を可能にするアプローチを採用します。これらは、本番環境移行時における有益な変更を促進し、デプロイされた問題を制限するとともに、お客様の環境において、デプロイメントアクティビティを通じて生じた問題、または検出された問題をすばやく特定し、修正できるようにします。

品質に関する迅速なフィードバックを提供し、望ましい結果をもたらさない変更から迅速に復旧できるようにするアプローチを採用します。これらを実践することにより、変更のデプロイメントによって生じる問題の影響が軽減されます。変更が失敗した場合の計画を立てて、必要な場合は迅速に対応し、変更をテストして検証できるようにします。環境で計画されたアクティビティに注意して、計画されたアクティビティに影響する変更のリスクを管理できるようにします。頻繁で小さく可逆的な変更に重点を置いて、変更の範囲を制限します。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また、変更をロールバックすることもできます。また、より頻繁に重要な変更の恩恵を受けることができることを意味します。

ワークロード、プロセス、手順、および従業員の運用準備状況を評価し、ワークロードに関連する運用上のリスクを理解します。一貫性のあるプロセス (手作業または自動化によるチェックリストを含む) を使用して、いつワークロードまたは変更を本稼働する準備ができるかを知る必要があります。また、これにより、対処する計画を立てる必要がある領域を見つけることもできます。日常的な活動を文書化したランブックと、問題解決のためにプロセスを導くプレイブックを備えます。メリットとリスクを理解し、十分な情報に基づく決定を下して、変更が本稼働環境に入ることを可能にします。

AWS では、ワークロード全体 (アプリケーション、インフラストラクチャ、ポリシー、ガバナンス、運用) をコードとして表示できます。つまり、アプリケーションコードに使用しているのと同じエンジニアリング規律をスタックのあらゆる要素に適用し、チームや組織間でこれらを共有することで、開発作業のメリットを拡大できます。クラウド上でコードとしてオペレーションを使用するとと

もに、安全に実験を行う機能を使用して、ワークロードや運用手順を開発し、障害に備えた練習を実施します。AWS CloudFormation を使用すると、運用管理のレベルが向上し、テンプレート化された整合性のあるサンドボックスの開発環境、テスト環境、本番環境を構築することができます。

以下の質問は、運用の優秀性に関する考慮事項に焦点を当てています。

OPS 4: ワークロードの状態を理解できるようにするためには、ワークロードをどのように設計すればよいですか?

ワークロードを設計する際には、すべてのコンポーネント (メトリクス、ログ、トレースなど) にわたって内部状態を理解するために必要な情報が送出されるようにします。そうすることによって、適時に有用な返答を提供できるようになります。

OPS 5: どのように欠陥を減らし、修正を容易にして、本番環境への流れを改善するのですか?

リファクタリング、品質についてのすばやいフィードバック、バグ修正を可能にし、本番環境への変更のフローを改善するアプローチを採用します。これらにより、本番環境に採用される有益な変更を加速させ、デプロイされた問題を制限できます。またデプロイアクティビティを通じて挿入された問題をすばやく特定し、修復できます。

OPS 6: デプロイのリスクを軽減するにはどうすればよいですか?

品質に関する迅速なフィードバックを提供し、望ましい結果をもたらさない変更から迅速に復旧できるようにするアプローチを採用します。このような手法を使用すると、変更のデプロイによって生じる問題の影響を軽減できます。

OPS 7: ワークロードをサポートする準備ができていることはどうすれば確認できますか?

ワークロード、プロセス、手順、従業員の運用準備状況を評価し、ワークロードに関連する運用上のリスクを理解するようにします。

運用アクティビティをコードとして実装することに投資することにより、運用担当者の生産性を最大に引き上げ、エラーの発生を最小限に抑え、自動応答を可能にします。「事前予測」のアプロー

チで、失敗を予測し、必要に応じて手順を作成します。リソースタグと AWS Resource Groups を使用し、一貫したタグ付け戦略に従ってメタデータを適用し、リソースを識別できるようにします。組織、原価計算、アクセスコントロールのリソースにタグを付け、自動化された運用アクティビティの実行に的を絞ります。クラウドの伸縮性を活用したデプロイ方法を導入し、開発活動を促進し、システムの事前デプロイを促進して実装を高速化します。ワークロードを評価するために使用するチェックリストに変更を加える場合は、もう準拠していない本番システムで行うことを計画します。

運用

ワークロードの運用の成功は、ビジネスの成果と顧客の成果の達成度によって評価されます。予想される成果を定義し、成功を評価する方法を決定します。また、ワークロードおよび運用が成功したかどうかを判断するための計算で使用するメトリクスを特定します。運用状態には、ワークロードの状態と、そのワークロードのサポートにおいて実行されるオペレーション活動の状態と成功 (デプロイとインシデント対応など) の両方を含みます。改善、調査、介入のためのメトリクスのベースラインを確立し、メトリクスを収集して分析し、オペレーションの成功と経時的な変化について理解していることを検証します。収集したメトリクスを使用して、顧客とビジネスのニーズを満たしているかどうかを確認し、改善の余地がある分野を特定します。

運用上の優秀性を実現するには、運用上のイベントを効率的かつ効果的に管理する必要があります。計画的および予期しない運用イベントの両方に適用されます。十分に把握しているイベントには既定のランブックを使用し、問題の調査および解決にはプレイブックを使用します。ビジネスと顧客への影響に基づいてイベントへの応答に優先順位を付けます。イベントへの応答でアラートが発生する場合、実行する関連プロセスがあり、所有者が具体的に指名されていることを確認します。イベントを解決する担当者を事前に決めておき、緊急性および影響に基づき、必要に応じて他の担当者を関与させるためにエスカレーションするトリガーを含めます。以前に処理したことがないイベント応答によってビジネスに影響が及ぶ場合は、アクションの方針を決定する権限を持つ担当者を特定し、関与させます。

対象 (顧客、ビジネス、開発者、運用など) に合わせたダッシュボードと通知によってワークロードの運用状況が伝えられるため、適切なアクションの実行や予測の管理、通常の運用が再開される時期の把握を行うことができます。

AWS では、ワークロードおよび AWS からネイティブに収集したメトリクスのダッシュボードビューを作成できます。CloudWatch またはサードパーティアプリケーションを利用して、運用アクティビティのビジネス、ワークロード、および運用レベルのビューを集約し、表示できます。AWS は、AWS X-Ray、CloudWatch、CloudTrail、および VPC フローログを含むログ機能を通じてワークロードインサイトを提供し、ワークロード問題の識別を可能にし、根本原因分析と改善をサポートします。

以下の質問は、運用の優秀性に関する考慮事項に焦点を当てています。

OPS 8: ワークロードの正常性はどのように把握するのですか?

ワークロードメトリクスの定義、キャプチャ、分析をすると、適切なアクションを取れるようにワークロードイベントの可視性を高めることができます。

OPS 9: オペレーションの正常性はどのように把握するのですか?

オペレーションメトリクスを定義し、キャプチャし、分析することで、オーペレーションイベントの可視性を高め、適切なアクションがとれるようになります。

OPS 10: ワークロードと運用イベントはどのように管理するのですか?

イベントに対応するための手順を準備、検証してワークロードの中断を最小限にします。

収集するすべてのメトリクスは、ビジネスニーズとそれらがサポートする結果に合わせて調整する必要があります。十分に理解されたイベントに対するスクリプト化されたレスポンスを開発し、イベントの認識に応じてパフォーマンスを自動化します。

進化

運用上の優秀性を維持するには、学び、共有し、継続的に改善する必要があります。継続的かつ段階的な改善を行うために専用の作業サイクルを作成します。顧客に影響を与えるすべてのイベントについて、インシデント後の分析を実行します。反復を制限または防止する要因と予防措置を特定します。必要に応じて、影響を受けたコミュニティと貢献要因を伝達します。ワークロードと運用手順の両方について、改善の機会 (機能のリクエスト、問題の修正、コンプライアンス要件など) を定期的に評価し、優先順位を付けます。

手順にフィードバックループを取り入れ、改善が必要な分野をすばやく特定し、実際に運用して教訓を学びます。

チーム間で学んだ教訓を共有し、その教訓の利点を活用します。学んだ教訓に見られる傾向を分析し、運用のメトリクスに関してチーム間で遡及的分析を行い、改善の機会とその方法を特定します。改善をもたらす変更を実施し、結果を評価して成功の判断を行います。

AWS では、ログデータを Amazon S3 にエクスポートしたり、ログを直接 Amazon S3 に送信して、長期保存したりできます。AWS Glue を使用すると、分析のために Amazon S3 でログデータを検出して準備し、関連するメタデータを AWS Glue Data Catalog に保存できます。Amazon Athena は AWS Glue とのネイティブな統合により、ログデータを分析し、標準 SQL を使用してクエリを実行できます。Amazon QuickSight のようなビジネスインテリジェンスツールを使用して、データの可視化、調査、分析を行うことができます。改善を促進する傾向や関心のあるイベントを発見します。

以下の質問は、運用の優秀性に関する考慮事項に焦点を当てています。

OPS 11: 運用はどのように進化させるのですか?

漸進的な継続的改善に時間とリソースを費やすことで、オペレーションを効果的かつ効率的に進化させることができます。

運用の進化を成功させるためには、頻繁な小規模の改善、実験と開発およびテストの改善のための安全な環境と時間、失敗から学ぶことを推奨する環境が重要です。運用では、サンドボックス、開発、テスト、本番の各環境をサポートします。運用管理レベルが向上し、開発を促進します。また、本番環境にデプロイした変更の成果に関する予測可能性が向上します。

リソース

運用の優秀性のベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメント

- DevOps と AWS

ホワイトペーパー

- 運用上の優秀性の柱

動画

- DevOps at Amazon

セキュリティ

このセキュリティの柱では、データ、システム、資産を保護して、クラウドテクノロジーを活用してセキュリティを強化する能力について説明します。

このセキュリティの柱では、設計原則、ベストプラクティス、質問の概要を説明します。実装に関する規範的なガイダンスとして セキュリティの柱に関するホワイトペーパーを参照してください.

トピック

設計原則

クラウドでのセキュリティには 7 つの設計の原則があります。

- 強力なアイデンティティ基盤を実装する最小特権の原則を実装し、各 AWS リソースとの各インタラクションに適切な権限を付与して役割分担を実施します。ID 管理を一元化し、長期間にわたって一つの認証情報を使用し続けないようにします。
- トレーサビリティの実現: 環境に対して、リアルタイムでモニタリング、アラート、監査のアクションと変更を行います。ログとメトリクスの収集をシステムに統合して、自動的に調査しアクションを実行します。
- 全レイヤーでセキュリティを適用する: 複数のセキュリティコントロールを使用して多層防御アプローチを適用します。ネットワークのエッジ、VPC、ロードバランシング、すべてのインスタンスとコンピューティングサービス、オペレーティングシステム、アプリケーション、コードなど、すべてのレイヤーに適用します。
- セキュリティのベストプラクティスを自動化する: 自動化されたソフトウェアベースのセキュリティメカニズムにより、安全でより高速かつ費用対効果の高いスケーリングが可能になります。バージョン管理されているテンプレートにおいてコードとして定義および管理されるコントロールを実装するなど、セキュアなアーキテクチャを作成します。
- 伝送中および保管中のデータを保護する: データを機密性レベルに分類し、暗号化、トークン分割、アクセスコントロールなどのメカニズムを適宜使用します。

- データに人の手を入れない: データへの直接アクセスや、手作業による処理の必要性を低減または排除するためのメカニズムやツールを使用します。これにより、機密性の高いデータを扱う際の誤処理、改変、ヒューマンエラーのリスクを軽減します。
- セキュリティイベントに備える: 組織の要件に合わせたインシデント管理および調査のポリシーとプロセスを導入し、インシデントに備えます。インシデント対応シミュレーションを実行し、ツールとオートメーションにより、検出、調査、復旧のスピードを上げます。

定義

クラウド内でのセキュリティには、6 つのベストプラクティス領域があります。

- セキュリティ
- アイデンティティとアクセスの管理
- 検知
- インフラストラクチャの保護
- データの保護
- インシデント対応

ワークロードを設計する前に、セキュリティに影響を与えるプラクティスを実施する必要があります。誰が何を実行できるのかという、権限の管理が必要になります。また、セキュリティインシデントを特定し、システムやサービスを保護し、データ保護によってデータの機密性と完全性を維持できる必要があります。セキュリティインシデントに対応するための、明確に定義された経験豊富なプロセスを利用できます。これらのツールやテクニックは、金銭的な損失の予防や規制遵守という目的を達成するためにも重要です。

AWS の責任共有モデルにより、クラウドを採用する組織は、セキュリティとコンプライアンスの目標を達成することができます。AWS がこのクラウドサービスの基盤となるインフラストラクチャを物理的に保護しているため、AWS のお客様はサービスを使用して自分たちの目標を達成することに集中できます。また、AWS クラウドは、セキュリティデータへのアクセスを向上させ、セキュリティイベントへの対応を自動化することもできます。

ベストプラクティス

トピック

- セキュリティ

- ID とアクセス管理
- 検知
- インフラストラクチャ保護
- データ保護
- インシデント対応

セキュリティ

ワークロードを安全に運用するには、セキュリティのすべての領域に包括的なベストプラクティスを適用する必要があります。組織レベルおよびワークロードレベルにおいて、「運用上の優秀性」で定義した要件とプロセスを抽出し、それらをすべての領域に適用します。

AWS や業界のレコメンデーションおよび脅威インテリジェンスを最新に保つことで、脅威モデルと管理目標を進化させることができます。セキュリティプロセス、テスト、検証を自動化することで、セキュリティオペレーションを拡張できます。

以下の質問は、セキュリティに関する考慮事項に焦点を当てています。(セキュリティに関する質問、回答、およびベストプラクティスの一覧については、 付録を参照してください)。

SEC 1: ワークロードを安全に運用するには、どうすればよいですか?
ワークロードを安全に運用するには、セキュリティのすべての領域に包括的なベストプラクティスを適用する必要があります。組織レベルおよびワークロードレベルにおいて、「運用上の優秀性」で定義した要件とプロセスを抽出し、それらをすべての領域に適用します。AWS や業界筋、脅威インテリジェンスからの推奨事項を常に把握することで、脅威モデルや管理目標を向上させることができます。セキュリティプロセス、テスト、検証を自動化することで、セキュリティオペレーションを拡張できます。

AWS における推奨アプローチは、機能およびコンプライアンスまたはデータの機密性要件に基づいて、アカウントごとに異なるワークロードを分離することです。

ID とアクセス管理

アイデンティティとアクセスの管理は情報セキュリティプログラムの主要パートです。これにより、お客様が意図した方法で認可、認証されたユーザーおよびコンポーネントのみがリソースにアクセスできるようになります。たとえば、プリンシパル (つまり、お客様のアカウントに対してアクション

をとるアカウント、ユーザー、ロール、サービス) を定義し、これらのプリンシパルに合わせたポリシーを構築し、強力な認証情報管理を実装できます。これらの権限管理機能は認証と承認の中枢となっています。

AWS では、権限管理は主に AWS Identity and Access Management (IAM) サービスによってサポートされており、AWS のサービスとリソースへのユーザーやプログラムによるアクセスの制御を可能にしています。詳細なポリシーを適用し、ユーザー、グループ、ロール、またはリソースに権限を割り当てることができます。また、複雑性レベル、再利用禁止、多要素認証 (MFA) の強制など、強力なパスワード設定をする機能があります。また既存のディレクトリサービスでフェデレーションを使用することもできます。AWS へのアクセス権を持つシステムを必要とするワークロードでは、IAM は、ロール、インスタンスプロファイル、ID フェデレーション、一時的認証情報によって安全なアクセスを実現します。

以下の質問は、セキュリティに関する考慮事項に焦点を当てています。

SEC 2: ユーザー ID とマシン ID はどのように管理するのですか?

AWS ワークロードを安全に運用するには、2 種類の ID を管理する必要があります。管理およびアクセス権を付与する必要があるアイデンティティのタイプを理解することで、適切な ID が適切な条件下で適切なリソースにアクセスできるようになります。

ユーザー ID: 管理者、開発者、オペレーター、エンドユーザーは、AWS 環境とアプリケーションにアクセスするために ID を必要とします。これらは、組織のメンバー、または共同作業を行う外部ユーザーであり、ウェブブラウザ、クライアントアプリケーション、またはインタラクティブなコマンドラインツールを介して AWS リソースを操作する人たちです。

マシン ID: サービスアプリケーション、運用ツール、ワークロードには、データの読み取りなどのために、AWS のサービスにリクエストを送信するための ID が必要です。このような ID には、Amazon EC2 インスタンスや AWS Lambda 関数など、AWS 環境で実行されているマシンが含まれます。また、アクセスを必要とする外部関係者のマシン ID を管理することもできます。さらに、AWS 環境にアクセスする必要があるマシンが AWS 外にある場合もあります。

SEC 3: ユーザーとマシンのアクセス許可はどのように管理するのですか?

アクセス許可を管理して、AWS とワークロードへのアクセスを必要とするユーザー ID やマシン ID へのアクセスを制御します。権限を分けることで、どのような条件で誰が何にアクセスできるかを制御します。

すべてのユーザーやシステムが認証情報を共有してはいけません。ユーザーアクセス権は、パスワード要件や MFA の強制などのベストプラクティスを実践した上で、最小権限で与えられるべきです。AWS サービスへのAPIコールを含むプログラムによるアクセスは、AWS Security Token Service が発行するような一時的かつ限定的な認証情報を使用して実行する必要があります。

AWS では、Identity and Access Management に役立つリソースを提供しています。ベストプラクティスを学ぶには、 認証情報と認証の管理io1 人為的なアクセスの制御、および プログラムによるアクセスの制御のハンズオンラボを参照してください。

検知

発見的統制により、セキュリティの潜在的な脅威やインシデントを特定できます。これはガバナンスフレームワークの最重要機能であり、品質管理プロセス、法的義務またはコンプライアンス義務、脅威の特定とその対応のサポートのために、この機能を使用できます。さまざまな種類の発見的統制があります。例えば、アセットとそれらの詳細な属性のインベントリを実行することで、より効果的に意思決定やライフサイクル管理を行い、運用の基準を確立できます。また、内部監査という、情報システムに関連するコントロールの検査を行って、ポリシーと要件に準拠し、定義した条件に基づいて正確に自動化されたアラート通知を設定できます。これらのコントロールは、組織が異常なアクティビティの範囲を特定し把握するのに役立つ重要な対応機能です。

AWS では、監査、自動分析、アラームを可能にするログ、イベント、監視を処理することで、検出制御を実装することができます。CloudTrail ログ、AWS API コール、CloudWatch はアラームによるメトリクスのモニタリングを、AWS Config は設定履歴を提供します。Amazon GuardDutyは、悪意ある動作や不正な挙動を継続的に監視し、AWS アカウントとワークロードの保護を支援するマネージド脅威検知サービスです。サービスレベルのログも利用できます。たとえば、Amazon Simple Storage Service (Amazon S3) を使用してアクセスリクエストをログに記録することができます。

以下の質問は、セキュリティに関する考慮事項に焦点を当てています。

SEC 4: セキュリティイベントは、どのように検出して調査するのですか?

ログやメトリクスからイベントを可視化して把握し、分析します。セキュリティイベント、および潜在的な脅威に対する措置を講じて、ワークロードの保護に役立てます。

ログ管理は、セキュリティやフォレンジックから、規制や法的要件に至るまで、Well-Architected ワークロードにとって重要です。潜在的セキュリティインシデントを特定するためには、ログを分析して対応することが不可欠です。AWS では、データ保持のライフサイクルを定義したり、データの

保存先、アーカイブ先、最終的な削除先を定義することで、ログ管理を容易にする機能を提供しています。予測可能で信頼性の高いデータ処理が、さらに簡単かつ費用対効果の高いものになります。

インフラストラクチャ保護

インフラストラクチャの保護には、ベストプラクティスと組織の義務または規制上の義務に準拠するために必要な、多層防御などの制御手段が含まれます。これらの手段を用いることは、クラウドやオンプレミスの環境で滞りなく運用していくために特に重要です。

AWS では、AWS ネイティブのテクノロジーを使用して、または AWS Marketplace から利用できるパートナー製品およびサービスを使用して、ステートフルおよびステートレスのパケットインスペクションを実装できます。Amazon Virtual Private Cloud (Amazon VPC) を使用して、プライベートでセキュアかつスケーラブルな環境を構築でき、この環境内でゲートウェイ、ルーティングテーブル、パブリックおよびプライベートのサブネットなどのトポロジーを定義できます。

以下の質問は、セキュリティに関する考慮事項に焦点を当てています。

SEC 5: ネットワークリソースはどのように保護するのですか?

何らかの形式のネットワーク接続があるワークロードは、インターネットでもプライベートネットワークでも、外部および内部ネットワークベースの脅威から保護するために、複数の防御レイヤーが必要です。

SEC 6: コンピューティングリソースはどのように保護するのですか?

ワークロード内のコンピューティングリソースを内外の脅威から守るには、複数の防御レイヤーを設ける必要があります。コンピューティングリソースには、EC2 インスタンス、コンテナ、AWS Lambda 関数、データベースサービス、IoT デバイスなどがあります。

すべての環境で複数レイヤーを防御するのが賢明です。インフラストラクチャ保護では、そのコンセプトとメソッドの多くがクラウドとオンプレミスの両方に対して有効です。境界保護の強制、イングレスおよびエグレスのモニタリングポイント、包括的なログ記録、モニタリング、アラートはすべて、効果的な情報セキュリティ計画には必須です。

AWS のお客様は、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Elastic Container Service (Amazon ECS) コンテナ、または AWS Elastic Beanstalk インスタンスの設定をカスタマイ

ズまたは強化し、その設定を変更不能な Amazon マシンイメージ (AMI) に永続化することができます。そして、この AMI を使用して起動するすべての新しい仮想サーバー (インスタンス) は、Auto Scaling でトリガーするか手動で起動して、その強化した設定を引き継ぐことができます。

データ保護

システムを設計する前に、セキュリティに影響を与える基本的なプラクティスを実施する必要があります。例えば、データ分類は組織のデータを機密性レベルに基づいてカテゴリーに分類し、暗号化は認証されていないアクセスに対してデータが開示されてしまうことを防ぎます。これらのツールやテクニックは、金銭的な損失の予防や規制遵守という目的を達成するためにも重要です。

AWS では、以下の取り組みによりデータの保護に努めています。

- AWS ユーザーとして、お客様はご自身のデータを完全に管理することができます。
- AWS では、データの暗号化や定期的な鍵の口ーテーションなど、鍵の管理を AWS で簡単に自動化したり、お客様自身でメンテナンスすることができます。
- ファイルのアクセスや変更など、重要なコンテンツを含む詳細なログを記録できます。
- AWS には優れた回復力を持つストレージシステムを設計しています。たとえば、Amazon S3 S3 Standard、S3 Standard–IA、S3 One Zone-IA、Amazon Glacier はすべて、1 年間にオブジェクトの 99.999999999% の堅牢性を実現するよう設計されています。この堅牢性レベルは、オブジェクトの予想される年平均損失の 0.000000001% に相当します。
- 大規模データライフサイクル管理プロセスの一部であるバージョニングにより、間違って上書きしたり削除したりしてデータが損なわれることを防ぎます。
- AWS ではリージョン間のデータの移動は発生しません。1 つのリージョンにあるコンテンツは、リージョン間の移動を可能にする機能を明示的に有効にしたり、その機能を提供するサービスを使用したりしない限りは、そのリージョンにとどまります。

以下の質問は、セキュリティに関する考慮事項に焦点を当てています。

SEC 7: データをどのように分類すればよいですか?

分類方法を確立すると、重要度と機密性に基づいてデータをカテゴリ別に分類して、各カテゴリに適した保護と保持方法でデータを管理できるようになります。

SEC 8: 保管中のデータはどのように保護するのですか?

複数のコントロールを実装して保管中のデータを保護し、不正アクセスや不正処理のリスクを低減します。

SEC 9: 転送中のデータはどのように保護するのですか?

複数のコントロールを実装して、転送中のデータを保護し、不正アクセスや損失のリスクを軽減します。

AWS には、保存中および伝送中のデータを暗号化する手段が複数あります。データの暗号化を容易にする機能を各サービスに搭載しています。たとえば、Amazon S3 にはサーバー側の暗号化 (SSE) を実装しているため、簡単にデータを暗号化して保存することができます。HTTPS の暗号化と復号化のプロセス全体 (一般に SSL termination として知られているプロセス) を、Elastic Load Balancing (ELB) によって処理されるように調整することもできます。

インシデント対応

非常に優れた予防的、発見的統制が実装されていてもなお、組織はセキュリティインシデントの潜在的な影響に対応し、影響を緩和する手段を講じる必要があります。ワークロードのアーキテクチャは、インシデントの際にチームが効果的に対応できるかどうか、システムを隔離するかどうか、運用を既知の正常な状態に復元できるかどうかに大きく影響します。セキュリティインシデントが起きる前にツールとアクセスを実践し、本番を想定したインシデント対応を定期的に実施することで、タイムリーな調査と復旧を可能にするアーキテクチャを構築できます。

AWS では、以下の取り組みにより効果的なインシデント対応を実現しています。

- ファイルのアクセスや変更など、重要なコンテンツを含む詳細なログを記録できます。
- イベントを自動的に処理でき、AWS API の使用によって、対応を自動化するツールを起動することができます。
- AWS CloudFormation を使用して、ツールと「クリーンルーム」を事前にプロビジョニングできます。これにより、安全で隔離された環境でフォレンジックを実行できます。

以下の質問は、セキュリティに関する考慮事項に焦点を当てています。

SEC 10: インシデントの予測、対応、復旧はどのように行うのですか?

組織に支障をきたすことを最小限に抑えるために、セキュリティインシデントのタイムリーで効果的な調査、対応、復旧に備えることが重要です。

お客様のセキュリティチームにすばやくアクセス権を付与し、インスタンスの隔離を自動化するとともに、フォレンジックのデータと状態のキャプチャを自動化します。

リソース

当社のセキュリティのベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメント

- AWS クラウドセキュリティ
- AWS コンプライアンス
- AWS セキュリティブログ

ホワイトペーパー

- セキュリティの柱
- AWS セキュリティの概要
- AWS リスクとコンプライアンス

動画

- AWS セキュリティの現状
- 責任共有モデルの概要

信頼性

信頼性の柱には、意図した機能を期待どおりに正しく一貫して実行するワークロードの能力が含まれます。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。このホワイトペーパーでは、AWS で信頼性の高いワークロードを実装するための詳しいベストプラクティスガイダンスを提供します。

この信頼性の柱では、設計原則、ベストプラクティス、質問の概要を説明します。実装に関する規範的なガイダンスとして 信頼性の柱に関するホワイトペーパーを参照してください.

トピック

設計原則

クラウドでの信頼性の設計原則が 5 つあります。

- 障害から自動的に復旧する: ワークロードで主要業績評価指標 (KPI、key performance indicator) をモニタリングすることで、しきい値を超えた場合にオートメーションをトリガーできます。この場合の KPI は、サービスの運用の技術的側面ではなく、ビジネス価値に関する指標であるべきです。これによって障害発生時の自動通知と追跡が可能になり、障害に対処する、または障害を修正するための復旧プロセスを自動化できます。より複雑な自動化を使用すると、障害が発生する前に修正を予期できます。
- 復旧手順をテストする: オンプレミス環境では、ワークロードが特定のシナリオで動作することを実証するためのテストを行うことがよくあります。復旧戦略を検証するためにテストを実施することはあまりありません。クラウドでは、どのようにシステム障害が発生するかをテストでき、復旧の手順も検証できます。自動化により、さまざまな障害のシミュレーションや過去の障害シナリオの再現を行うことができます。このアプローチでは、実際の障害シナリオが発生する前にテストおよび修正できる障害経路が公開されるため、リスクが軽減されます。
- 水平方向にスケールしてワークロード全体の可用性を高める: 1 つの大規模なリソースを複数の小規模なリソースに置き換えることで、単一の障害がワークロード全体に与える影響を軽減します。リクエストを複数の小規模なリソースに分散することで、一般的な障害点を共有しないようにします。
- キャパシティーを推測することをやめる: オンプレミスのワークロードにおける障害の一般的な原因はリソースの飽和状態で、ワークロードに対する需要がそのワークロードのキャパシティーを超えたときに発生します (サービス妨害攻撃の目標となることがよくあります)。クラウドでは、需要とワークロード使用率をモニタリングし、リソースの追加と削除を自動化することで、プロビジョニングが過剰にも過小にもならない、需要を満たせる最適なレベルを維持できます。制限はまだありますが、いくつかのクォータは制御でき、そのほかのクォーターも管理できます (Service Quotasと制約の管理を参照してください)。

- オートメーションで変更を管理する: インフラストラクチャに対する変更は、オートメーションを使用して実行する必要があります。管理する必要がある変更には、自動化に対する変更が含まれており、それを追跡して確認することができます。

定義

クラウド内での信頼性には、4 つのベストプラクティス領域があります。

- 基盤
- ワークロードアーキテクチャ
- 変更管理
- 障害管理

信頼性を実現するには、基盤、つまりサービスクォータとネットワークトポロジがワークロードに対応する環境作りから始める必要があります。分散システムのワークロードアーキテクチャは、障害を防止および軽減するように設計する必要があります。ワークロードは需要または要件の変化に対応する必要があり、障害を検出して自動的に復旧するように設計する必要があります。

ベストプラクティス

トピック

基盤

基盤となる要件は、単一のワークロードまたはプロジェクトの範囲を超える要件です。システムを設計する前に、信頼性に影響を与える基本的な要件を満たしておく必要があります。例えば、データセンターへの十分なネットワーク帯域幅が必要です。

AWS では、このような基本的な要件のほとんどが既に組み込まれており、必要に応じて変更できます。クラウドは、ほぼ制限を持たないように設計されています。つまり、十分なネットワーク性能とコンピューティング性能の要件を満たすのは AWS の責任であり、お客様はリソースのサイズと割り当てを需要に応じて自由に変更できます。

以下の質問は、信頼性に関する考慮事項に焦点を当てています。(信頼性に関する質問、回答、およびベストプラクティスの一覧については、 付録を参照してください)。

REL 1: サービスクォータと制約はどのように管理するのですか?

クラウドベースのワークロードアーキテクチャには、サービスクォータ (サービスの制限とも呼ばれます) というものがあります。このようなクォータは、誤って必要以上のリソースをプロビジョニングするのを防ぎ、サービスを不正使用から保護することを目的として API 操作のリクエスト頻度を制限するために存在します。リソースにも制約があります。たとえば、光ファイバーケーブルのビットレートや、物理ディスクの記憶容量などです。

REL 2: ネットワークトポロジはどのように計画するのですか?

多くの場合、ワークロードは複数の環境に存在します。このような環境には、複数のクラウド環境 (パブリックにアクセス可能なクラウド環境とプライベートの両方) と既存のデータセンターインフラストラクチャなどがあります。計画する際は、システム内およびシステム間の接続、パブリック IP アドレスの管理、プライベート IP アドレスの管理、ドメイン名解決といったネットワークに関する諸点も考慮する必要があります。

クラウドベースのワークロードアーキテクチャには、サービスクォータ (サービスの制限とも呼ばれます) というものがあります。このようなクォータは、誤って必要以上のリソースをプロビジョニングするのを防ぎ、サービスを不正使用から保護することを目的として API 操作のリクエスト頻度を制限するために存在します。多くの場合、ワークロードは複数の環境に存在します。これらのクォータは、すべてのワークロード環境でモニタリングおよび管理する必要があります。このような環境には、複数のクラウド環境 (パブリックにアクセス可能なクラウド環境とプライベートの両方) が含まれるほか、既存のデータセンターインフラストラクチャが含まれる場合があります。計画する際は、システム内およびシステム間の接続、パブリック IP アドレスの管理、プライベート IP アドレスの管理、ドメイン名解決といったネットワークに関する項目も考慮に含めなければなりません。

ワークロードアーキテクチャ

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。

AWS では、ワークロード開発者は使用する言語とテクノロジーを選択できます。AWS SDK は、AWS のサービス用に言語固有の API を提供することで、コーディングの複雑さを排除します。これらの SDK と言語の選択により、デベロッパーはここに掲載されている信頼性のベストプラクティスを実装できます。デベロッパーは、Amazon がソフトウェアを構築および運用する方法について Amazon Builders' Library で読み、学ぶことができます。

以下の質問は、信頼性に関する考慮事項に焦点を当てています。

REL 3: ワークロードサービスアーキテクチャをどのように設計すればよいですか?

サービス指向アーキテクチャ (SOA) またはマイクロサービスアーキテクチャを使用して、拡張性と信頼性の高いワークロードを構築します。サービス指向アーキテクチャ (SOA) は、サービスインターフェイスを介してソフトウェアコンポーネントを再利用できるようにする方法です。マイクロサービスアーキテクチャは、その一歩先を行き、コンポーネントをさらに小さくシンプルにしています。

REL 4: 障害を防ぐために、分散システムの操作をどのように設計すればよいですか?

分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。これらのベストプラクティスは、故障を防ぎ、平均故障間隔 (MTBF) を改善します。

REL 5: 障害を緩和または耐えるために、分散システムでの操作をどのように設計すればよいですか?

分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。これらのベストプラクティスに従うことで、ワークロードはストレスや障害に耐え、より迅速に復旧し、そのような障害の影響を軽減できます。その結果、平均復旧時間 (MTTR) が向上します。

変更管理

ワークロードを信頼できる形で運用するには、ワークロードやその環境に対する変化を予測してそれに対応することが不可欠です。変更には、需要の急増といったワークロードに強いられる変更や、機能のデプロイやセキュリティパッチの適用といった内部からの変更があります。

AWS を使用すると、ワークロードの動作をモニタリングし、KPI への応答を自動化できます。たとえば、ワークロードがより多くのユーザーを獲得するにつれ、ワークロードはサーバーを追加できます。ワークロードの変更や変更履歴の監査を行う権限を持つユーザーを制御できます。

以下の質問は、信頼性に関する考慮事項に焦点を当てています。

REL 6: ワークロードリソースをモニタリングするにはどうすればよいですか?

ログとメトリクスは、ワークロードの状態についての洞察を得るための強力なツールです。ワークロードは、しきい値を超えたり重大なイベントが発生したりしたときに、ログとメトリクスがモニタリングされて通知が送信されるように構成できます。モニタリングにより、ワークロードは、低パフォーマンスのしきい値を超えたときや障害が発生したときにそれを認識できるため、それに応じて自動的に復旧できます。

REL 7: 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?

スケーラブルなワークロードには、リソースを自動で追加または削除する伸縮性があるので、リソースは常に、現行の需要に厳密に適合します。

REL 8: 変更はどのように実装するのですか?

変更制御は、新しい機能をデプロイしたり、アプリケーションと運用環境で既知のソフトウェアが実行されており、予測できる方法でパッチを適用または置換できることを確認したりするために必要です。変更が制御されていないと、変更の影響を予測したり、変更によって発生した問題に対処したりすることが困難になります。

需要の変動に対応してリソースの追加や削除を自動で行うワークロードを設計しておけば、信頼性が高まることに加え、ビジネスの成功が負担に変わってしまうことを避けられます。モニタリングを実行しておけば、KPI が予測された基準から逸脱すると、アラートが自動的にチームに送られます。環

境の変更は自動的にログに記録されるため、アクションを監査して、信頼性に影響を与える可能性のあるアクションを特定できます。変更管理をコントロールすることで、必要な信頼性を実現するためのルールに効力を持たせることができます。

障害管理

ある程度複雑なシステムでは、障害が発生することが予想されます。ワークロードで信頼性を確保するには、発生時点で障害を認識し、可用性への影響を回避するための措置を講じることが必要です。ワークロードは、障害に耐え、問題を自動的に修復できる必要があります。

AWS では、自動化を利用してモニタリングデータに反応できます。例えば、特定のメトリクスがしきい値を超えたときに、その問題を解決する自動化されたアクションがトリガーされるよう設定できます。また、障害が発生したリソースを本番環境で診断して修正するのではなく、まずリソースを新しいものに置き換えてから、障害の発生したリソースの分析を別途実施できます。クラウドではシステム全体の一時的なバージョンを低コストで立ち上げることができるため、自動化されたテストで復旧プロセス全体を検証することも可能です。

以下の質問は、信頼性に関する考慮事項に焦点を当てています。

REL 9: データはどのようにバックアップするのですか?

目標復旧時間 (RTO) と目標復旧時点 (RPO) の要件を満たすように、データ、アプリケーション、設定をバックアップします。

REL 10: ワークロードを保護するために障害分離をどのように使用するのですか?

障害部分を分離した境界は、ワークロード内の障害の影響を限られた数のコンポーネントに限定します。境界外のコンポーネントは、障害の影響を受けません。障害部分を切り離した境界を複数使用すると、ワークロードへの影響を制限できます。

REL 11: コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?

高い可用性と低い平均復旧時間 (MTTR) の要件を持つワークロードは、回復力を考慮した設計をする必要があります。

REL 12: どのように信頼性をテストするのですか?

本番環境のストレスに耐えられるようにワークロードを設計した後、ワークロードが意図したとおりに動作し、期待する弾力性を実現することを確認する唯一の方法が、テストを行うことです。

REL 13: 災害対策 (DR) はどのように計画するのですか?

バックアップと冗長ワークロードコンポーネントを配置することは、DR 戦略の出発点です。RTO と RPO は ワークロードの復旧目標です。これは、ビジネスニーズに基づいて設定します。ワークロードのリソースとデータのロケーションと機能を考慮して、目標を達成するための戦略を実装します。ワークロードの災害対策を提供することのビジネス価値を伝えるには、中断の可能性と復旧コストも重要な要素となります。

定期的にデータをバックアップし、バックアップファイルをテストすることで、論理的なエラーや物理的なエラーから確実に復旧できるようにします。障害の管理で重要なのは、ワークロードに対し自動化されたテストを頻繁に実施して障害を発生させ、どのように復旧するかを確認することです。そのためには、このようなテストは定期的なスケジュールでも大きなワークロード変更後にトリガーされます。KPI を積極的に追跡し、目標復旧時間 (RTO)、目標復旧時点 (RPO)、ワークロードの回復力を評価します (特にテストシナリオで障害が発生した場合)。KPI の追跡は、単一障害点を特定して排除するのに役立ちます。目的は、ワークロード復旧プロセスを徹底的にテストして、問題が持続する場合でも、すべてのデータを復旧し、サービスをお客様に提供し続けられることを確認することです。復旧プロセスも、通常の本番プロセスと同じように実施する必要があります。

リソース

信頼性のベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメント

- AWS ドキュメント
- AWS グローバルインフラストラクチャ
- AWS Auto Scaling: スケーリングプランの仕組み
- 「AWS Backup とは。」

ホワイトペーパー

- 信頼性の柱: AWS Well-Architected
- AWS でのマイクロサービスの実装

パフォーマンス効率

パフォーマンス効率の柱には、システムの要件を満たすためにコンピューティングリソースを効率的に使用し、要求の変化とテクノロジーの進化に対してその効率性を維持する能力が含まれます。

このパフォーマンス効率の柱では、設計原則、ベストプラクティス、質問の概要を説明します。実装に関する規範的なガイダンスとして パフォーマンス効率の柱に関するホワイトペーパーを参照してください。.

トピック

- 設計原則
- 定義
- ベストプラクティス
- リソース

設計原則

クラウドには、パフォーマンス効率のための 5 つの設計原則があります。

- 最新テクノロジーを誰もが利用できるようにする: 複雑なタスクをクラウドベンダーに委託することで、チームがより簡単に高度なテクノロジーを実装できるようにします。IT チームに新しいテクノロジーのホスティングと実行について学んでもらうのではなく、テクノロジーをサービスとして消費することを検討してください。たとえば、NoSQL データベース、メディアトランスコーディング、および機械学習などは、いずれも特化された専門知識を必要とするテクノロジーです。クラウドでは、これらのテクノロジーがチームによる消費が可能なサービスとなり、チームはリソースのプロビジョニングと管理ではなく、製品の開発に集中できるようになります。
- わずか数分でグローバル展開する世界中の複数の AWS リージョンにワークロードをデプロイすることで、最小限のコストで、より低いレイテンシーとより優れたエクスペリエンスをお客様に提供できます。
- サーバーレスアーキテクチャを使用する: サーバーレスアーキテクチャにより、従来のコンピューティングアクティビティのために物理的なサーバーを実行および維持する必要がなくなります。た

とえば、サーバーレスストレージサービスは静的ウェブサイトとして機能させることができ (ウェブサーバーが不要になる)、イベントサービスはコードをホストできます。これによって物理サーバーを管理する運用上の負担が取り除かれます。また、マネージドサービスはクラウド規模で運用されることから、トランザクションコストも削減することができます。

- より頻繁に実験する: 仮想および自動化できるリソースを使用すれば、さまざまなタイプのインスタンス、ストレージ、構成を使用して比較テストを迅速に実行できます。
- メカニカルシンパシーを重視する: クラウドサービスの使用方法を理解し、常にワークロードの目標に最適なテクノロジーアプローチを使用します。例えば、データベースやストレージのアプローチを選択するときには、データアクセスパターンを考慮します。

定義

クラウドのパフォーマンス効率を向上させるには、次の 4 つのベストプラクティス領域があります。

- 選択
- レビュー
- モニタリング
- トレードオフ

データ駆動型アプローチを採#して、#パフォーマンスのアーキテクチャを選択します。ハイレベルな設計から、リソースタイプの選択と設定に至るまで、アーキテクチャのあらゆる側面に関するデータを収集してください。

選択した内容を定期的にレビューして、進化し続ける AWS クラウドの機能を十分に活かしていることを確認します。モニタリングすることで、期待されるパフォーマンスからの逸脱を確実に把握できるようにします。圧縮やキャッシュを使用したり、整合性に関する要件を緩和したりするなど、アーキテクチャにおけるトレードオフを行ってパフォーマンスを向上させます。

ベストプラクティス

トピック

- 選択
- レビュー
- モニタリング

- トレードオフ

選択

特定のワークロードに適したソリューションはさまざまで、大抵の場合、ソリューションには複数のアプローチが組み合わされています。優れた設計のワークロードは、複数のソリューションを使用することで様々な機能を活用しパフォーマンスの改善をします。

AWS のリソースは多くのタイプと設定で利用できるため、ワークロードのニーズにしっかりと適合するアプローチを見つけることが容易になります。また、オンプレミスのインフラストラクチャでは簡単に実現できないオプションも利用できます。例えば、Amazon DynamoDB のようなマネージドサービスでは、あらゆるスケールにおいてレイテンシーが 10 ミリ秒未満であるフルマネージド型の NoSQL データベースを提供します。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。(パフォーマンス効率に関する質問、回答、ベストプラクティスの一覧については、 付録を参照してください)。

PERF 1: 最適なパフォーマンスのアーキテクチャはどのように選択すればよいですか?

多くの場合、ワークロード全体での最適なパフォーマンスのためには、複数のアプローチが必要になります。Well-Architected なシステムでは、パフォーマンスを向上させるために複数のソリューションと機能が使用されています。

データ主導のアプローチを用いて、アーキテクチャのパターンと実装を選択し、コスト効率性に優れたソリューションを実現します。AWS ソリューションアーキテクト、AWS リファレンスアーキテクチャ、および AWS パートナーネットワーク (APN) のパートナーは、業界知識に基づいてアーキテクチャの選択を支援しますが、アーキテクチャを最適化するには、ベンチマークまたは負荷テストを通じて得られるデータが必要になります。

アーキテクチャでは通常、アーキテクチャに関するさまざまなアプローチが組み合わされて使用されます (イベント駆動型、ETL、パイプラインなど)。アーキテクチャの実装には、アーキテクチャのパフォーマンスの最適化に特化した AWS のサービスが使用されます。以下のセクションでは、考慮すべき 4 つの主なリソースタイプ、つまりコンピューティング、ストレージ、データベース、およびネットワークについて説明します。

コンピューティング

要件やパフォーマンスのニーズを満たし、コストや労力を大幅に効率化するコンピューティングリソースを選択することで、同じ数のリソースでより多くを達成できます。コンピューティングオプションを評価するときは、ワークロードのパフォーマンスの要件とコスト要件に留意し、これを使用して十分な情報に基づいて意思決定を行います。

AWS では、インスタンス、コンテナ、関数という 3 つの形式でコンピューティングを利用できます。

- インスタンス は仮想化されたサーバーで、ボタンまたは API コールを使用して機能を変更できます。クラウドではリソースを柔軟に選択できることから、異なるサーバータイプで実験することができます。AWS では、これらの仮想サーバーインスタンスには、さまざまなファミリーおよびサイズがあり、ソリッドステートドライブ (SSD) やグラフィック処理ユニット (GPU) など、多種多様な機能を提供します。
- コンテナ は、オペレーティングシステムを仮想化する手段です。コンテナを使用すると、リソースが分離されたプロセス内でアプリケーションとその依存関係を実行できます。AWS Fargate は、コンテナ用のサーバーレスコンピューティングです。コンピューティング環境のインストール、設定、管理を制御する必要がある場合は、Amazon EC2 を使用できます。Amazon Elastic Container Service (ECS) または Amazon Elastic Kubernetes Service (EKS) といった複数のコンテナオーケストレーションプラットフォームから選択することもできます。
- 関数 実行するコードに基づいて、実行環境を抽象化します。例えば、AWS Lambda を使用すれば、インスタンスを実行することなくコードを実行できます。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 2: コンピューティングソリューションはどのように選択すればよいですか?

ワークロードにとって最適なコンピューティングソリューションは、アプリケーションの設計、使用パターン、設定に応じて異なります。各アーキテクチャでは、コンポーネントごとに異なるコンピューティングソリューションが使用される可能性があるため、パフォーマンスを向上させるための機能も異なります。アーキテクチャに不適切なコンピューティングソリューションを選択すると、パフォーマンス効率が低下する可能性があります。

コンピューティングを使用するアーキテクチャを設計する場合、需要が変化してもパフォーマンスを維持できるだけの十分な容量を確保できるように、伸縮性のメカニズムを導入する必要があります。

ストレージ

クラウドストレージはクラウドコンピューティングに不可欠のコンポーネントで、ワークロードが使用する情報を保持します。クラウドストレージは、通常、従来のオンプレミスストレージシステムよりも信頼性、拡張性、安全性に優れています。ワークロードに適したクラウドデータ移行オプションに加えて、オブジェクト、ブロック、ファイルストレージサービスから選択します。

AWS では、オブジェクト、ブロック、ファイルという 3 つの形式でストレージを利用できます。

- オブジェクトストレージ は、ユーザーが生成したコンテンツ、アクティブなアーカイブ、サーバーレスコンピューティング、ビッグデータストレージ、またはバックアップと復旧のために、任意のインターネットロケーションからデータにアクセスできるようにする、スケーラブルで耐久性のあるプラットフォームを提供します。Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データの可用性、セキュリティ、パフォーマンスを備えたオブジェクトストレージサービスです。Amazon S3 は 99.999999999% (9 が 11 個) の耐久性を実現するように設計されており、世界中の企業向けに数百万ものアプリケーションのデータを保存しています。
- ブロックストレージ は、各仮想ホストに、高可用性かつ一貫性のある低レイテンシーのブロックストレージを提供します。これは、ダイレクトアタッチトストレージ (DAS) またはストレージエリアネットワーク (SAN) に類似しています。Amazon Elastic Block Store (Amazon EBS) は、EC2 インスタンスからアクセスできる永続的ストレージを必要とするワークロード向けに設計されており、適切なストレージ容量、パフォーマンス、コストでアプリケーションを調整することができます。
- ファイルストレージ は、複数のシステムにまたがる共有ファイルシステムへのアクセスを提供します。Amazon Elastic File System (EFS) などのファイルストレージソリューションは、大規模なコンテンツリポジトリ、開発環境、メディアストア、ユーザーのホームディレクトリなどのユースケースに最適です。Amazon FSx を使用すると、一般的なファイルシステムを簡単かつコスト効率よく起動して実行できるため、広く使用されているオープンソースや商用ライセンスが付与されたファイルシステムの豊富な機能セットと高速なパフォーマンスを活用できます。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 3: ストレージソリューションはどのように選択すればよいですか?

システムにとって最適なストレージソリューションは、アクセス方法 (ブロック、ファイル、オブジェクト)、アクセスパターン (ランダム、シーケンシャル)、必要なスループットやアクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、可用性と耐久性に関する

PERF 3: ストレージソリューションはどのように選択すればよいですか?

制約によって異なります。優れた設計のシステムでは、複数のストレージソリューションを使用し、さまざまな機能を有効にしてパフォーマンスとリソースの使用効率を高めています。

ストレージソリューションを選択する際は、必要なパフォーマンスを実現できるように、お客様のアクセスパターンに合ったソリューションを選択することが重要です。

データベース

クラウドは、ワークロードで発生するさまざまな問題に対処する、専用のデータベースサービスを提供します。これは、リレーショナル、key-value、ドキュメント、インメモリ、グラフ、時系列、および台帳データベースを含めた多数の専用データベースエンジンから選択できます。特定の問題 (または一連の問題) を解決するために最適なデータベースを選択することによって、制約の多い汎用的なモノリシックデータベースから脱却し、顧客のパフォーマンスニーズを満たすアプリケーションの構築に専念することができます。

AWS では、リレーショナル、key-value、ドキュメント、インメモリ、グラフ、時系列、台帳データベースなど、複数の専用データベースエンジンから選択できます。AWS データベースでは、サーバーのプロビジョニング、パッチの適用、セットアップ、設定、バックアップ、復旧などのデータベース管理タスクについて心配する必要がありません。AWS はクラスターを継続的にモニタリングして、自己修復ストレージと自動スケーリングによってワークロードを実行状態に保つため、より価値の高いアプリケーション開発に専念できます。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 4: データベースソリューションはどのように選択すればよいですか?

システムにとって最適なデータベースソリューションは、可用性、整合性、分断耐性、レイテンシー、耐久性、スケーラビリティ、クエリ機能などの要件に応じて異なります。多くのシステムでは、各種サブシステムに異なるデータベースソリューションを使用しているため、パフォーマンスを向上させるために活用する機能も異なります。システムに対して適切でないデータベースソリューションや機能を選択すると、パフォーマンス効率が低下する場合があります。

ワークロードのデータベースアプローチは、パフォーマンス効率に大きな影響を与えます。多くの場合、データ駆動型のアプローチではなく、組織のデフォルトに従って選択されます。ストレージと同様、ワークロードのアクセスパターンを検討することが重要です。また、データベースではないソ

リューション (グラフ、時系列、インメモリストレージデータベースなど) を使用してより効率的に問題を解決できないか検討することも重要です。

ネットワーク

ネットワークはすべてのワークロードコンポーネント間に存在するため、ワークロードのパフォーマンスと動作に対して、良くも悪くも大きな影響を及ぼす可能性があります。また、ハイパフォーマンスコンピューティング (HPC) などのネットワークパフォーマンスに大きく依存するワークロードもあり、この場合はネットワークを深く理解することがクラスターのパフォーマンスを向上させるうえで重要になります。帯域幅、レイテンシー、ジッター、およびスループットに関するワークロードの要件を判断する必要があります。

AWS ではネットワーキングが仮想化されており、数多くの異なるタイプと設定で利用することができます。このため、ニーズに合うネットワーキング手法が取りやすくなります。AWS は、ネットワークトラフィックを最適化する製品機能を提供します (例えば、拡張ネットワーキング、Amazon EBS 最適化インスタンス、Amazon S3 Transfer Acceleration、動的 Amazon CloudFront など)。また、AWS は、ネットワーク距離またはジッターを軽減するネットワーク機能も提供します (例えば、Amazon Route 53 レイテンシールーティング、Amazon VPC エンドポイント、AWS Direct Connect、AWS Global Accelerator など)。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 5: ネットワークソリューションはどのように選択すればよいですか?

ワークロードに最適なネットワークソリューションは、レイテンシー、スループット要件、ジッター、および帯域幅に応じて異なります。ロケーションのオプションは、ユーザーまたはオンプレミスのリソースなどの物理的な制約に左右されます。これらの制約は、エッジロケーションまたはリソースの配置で相殺することができます。

ネットワークをデプロイする際は、場所を検討する必要があります。距離を縮めるために、リソースが使用される場所の近くに配置することもできます。ネットワーキングメトリクスを使用して、ワークロードの進化に合わせてネットワーキング設定を変更します。リージョンや、プレイスメントグループ、エッジサービスを活用すれば、パフォーマンスを大幅に向上させることができます。クラウドベースのネットワークは素早い再構築または変更が可能なことから、パフォーマンス効率を維持するためにもネットワークアーキテクチャを徐々に進化させることが必要です。

レビュー

クラウドテクノロジーは急速に進化しており、最新のテクノロジーとアプローチを使用してワークロードコンポーネントのパフォーマンスを継続的に向上させる必要があります。ワークロードコンポーネントに対する変更を継続的に評価して検討し、パフォーマンスとコストの目標を確実に満たすようにする必要があります。機械学習や人工知能 (AI) などの新しいテクノロジーにより、カスタマーエクスペリエンスを再考し、ビジネスワークロード全体にわたってイノベーションを起こすことができます。

お客様のニーズによって推進される AWS の継続的なイノベーションをご活用ください。AWS では、新しいリージョン、エッジロケーション、サービス、および機能を定期的にリリースしています。これらのリリースはいずれも、アーキテクチャのパフォーマンス効率を向上させることができます。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 6: ワークロードを進化させるために、新機能をどのように活用すればよいですか？

ワークロードを設計する際に選択できるオプションには限りがありますが、時間と共に、ワークロードのパフォーマンスを向上させることができる新しいテクノロジーとアプローチが利用できるようになります。

アーキテクチャのパフォーマンスが低いのは、通常、パフォーマンスレビュープロセスが存在しないか、破損していることに起因します。アーキテクチャのパフォーマンスレベルが低い場合は、パフォーマンスレビュープロセスを導入することで、デミングの PDCA (計画 – 実施 – 評価 – 改善) サイクルを適用して反復的な改善を促すことができます。

モニタリング

ワークロードを実装した後は、そのパフォーマンスをモニタリングして、問題が顧客に影響を及ぼす前に修正できるようにする必要があります。モニタリングメトリクスを使用して、しきい値を超えたときにアラームが発報されるようにします。

Amazon CloudWatch は、ワークロードの監視、システム全体のパフォーマンス変化への対応、リソース利用の最適化、運用の健全性の統合ビューを得るためのデータと実行可能なインサイトを提供する、監視、観測のためのサービスです。CloudWatch は、AWS およびオンプレミスのサーバーで実行されるワークロードから、ログ、メトリクス、イベントの形式で監視、運用データを収集します。 AWS X-Ray は、デベロッパーが本番稼働用の分散型アプリケーションを分析し、デバッグする

のに役立ちます。AWS X-Ray を使用すると、アプリケーションの動作に関するインサイトを得て、根本原因を検出し、パフォーマンスのボトルネックを特定できます。これらのインサイトを使用することで、速やかに対応し、ワークロードのスムーズな動作を維持できます。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 7: リソースが稼働していることを確認するには、リソースをどのようにモニタリングすればよいですか?

システムのパフォーマンスは徐々に低下することがあります。劣化を特定し、オペレーティングシステムまたはアプリケーション負荷などの内部および外部の要因を修正するために、システムのパフォーマンスをモニタリングします。

誤判定がないことを確実にすることは、効果的なモニタリングソリューションの鍵です。自動化されたトリガーは、人為的なミスを防ぎ、問題の修正にかかる時間を短縮できます。解決までの時間を短縮できます。アラームソリューションをテストするシミュレーションを本番環境で実行するゲームデーを計画し、そのソリューションが問題を正しく認識するか確認してください。

トレードオフ

ソリューションを設計する際は、最適なアプローチを確保するためのトレードオフについて検討します。状況に応じて、整合性、耐久性、および時間とレイテンシーの余裕と引き換えに、より優れたパフォーマンスを実現することができます。

AWS を使用すると、エンドユーザーに近い世界中の場所に数分でリソースをデプロイできます。また、読み取り専用のレプリカをデータベースシステムなどの情報ストアに動的に追加して、プライマリデータベースの負荷を軽減することもできます。

以下の質問は、パフォーマンス効率に関する考慮事項に焦点を当てています。

PERF 8: パフォーマンスを向上させるために、トレードオフをどのように利用すればよいですか?

アーキテクチャの設計にあたって、最適なアプローチとなるトレードオフを特定します。多くの場合、整合性、耐久性、および時間とレイテンシーの余裕と引き換えに、パフォーマンスを向上させることができます。

ワークロードに変更を加えた後、メトリクスを収集して評価し、それらの変更の影響を判断します。システムおよびエンドユーザーに対する影響を測定し、トレードオフがワークロードにどのような影

響を与えるかを理解します。負荷テストなどの体系的なアプローチを使用して、トレードオフによってパフォーマンスが向上するかどうかを調べます。

リソース

パフォーマンス効率に関するベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメント

- Amazon S3 パフォーマンスの最適化
- Amazon EBS ボリュームのパフォーマンス

ホワイトペーパー

- パフォーマンス効率の柱

動画

- AWS re:Invent 2019: Amazon EC2 foundations (CMP211-R2) (Amazon EC2 の基礎知識)
- AWS re:Invent 2019: Leadership session: Storage state of the union (STG201-L) (ストレージの現状)
- AWS re:Invent 2019: Leadership session: AWS purpose-built databases (DAT209-L) (AWS の目的別データベース)
- AWS re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures (NET317-R1) (AWS およびハイブリッド AWS ネットワークアーキテクチャへの接続性)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2) (次世代 Amazon EC2 の実現に向けて: Nitro システムの事例詳細)
- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R) (最初の 1,000 万ユーザーまでのスケールアップ)

コスト最適化

コスト最適化の柱には、最低価格でビジネス価値を実現するシステムを実行できる能力が含まれています。

コスト最適化の柱では、設計原則、ベストプラクティス、質問の概要を説明します。実装に関する規範的なガイダンスとして コスト最適化の柱に関するホワイトペーパーを参照してください。.

トピック

設計原則

クラウドでのコスト最適化には、5 つの設計原則があります。

- クラウド財務管理を実装するクラウドで財務面での成功を達成し、ビジネス価値の実現を加速するには、クラウド財務管理とコスト最適化に投資する必要があります。組織は、テクノロジーと使用状態の管理というこの新しい領域における能力を獲得するために、時間とリソースを投入する必要があります。コスト効率の高い組織にするには、セキュリティや優れた運用力と同様、知識の積み上げ、プログラム、リソース、プロセスを通じて能力を構築する必要があります。
- 消費モデルを導入する: 必要なコンピューティングリソースの使用分のみを支払い、ビジネス要件に応じて使用量を増減できます。複雑なコストの予測は必要ありません。たとえば、通常、1 週間の稼働日に開発環境とテスト環境を使用するのは、1 日あたり 8 時間程度です。未使用時にこのようなリソースを停止することで、コストを 75% 削減できる可能性があります (168 時間ではなく 40 時間になる)。
- 全体的な効率を測定する: ワークロードのビジネス成果とその実現に関連するコストを測定します。この測定値を使用して、生産性を向上し、コストを削減することで得られる利点を把握します。
- 差別化につながらない高負荷の作業に費用をかけるのをやめるラッキング、スタッキング、サーバーへの電源供給など、データセンターの運営に必要な重作業は AWS が行います。また、マネージドサービスを使用することで、オペレーティングシステムやアプリケーションの管理に伴う運用上の負担も解消されます。この結果、IT インフラストラクチャよりも顧客と組織のプロジェクトに集中できるようになります。
- 費用を分析し帰属関係を明らかにする: クラウドを使用すると、システムの使用状況とコストを正確に特定しやすくなり、IT コストを個々のワークロード所有者に透過的に帰属させることができます。これによって投資収益率 (ROI) を把握できるため、ワークロードの所有者はリソースを最適化してコストを削減する機会が得られます。

定義

クラウドでのコスト最適化には、5 つのベストプラクティス領域があります。

- クラウド財務管理を実践する
- 経費支出と使用量の認識
- 費用対効果の高いリソース
- 需要を管理しリソースを供給する
- 継続的最適化

Well-Architected フレームワーク内の他の柱と同様に、この柱にも考慮すべきトレードオフがあります。市場投入スピードとコストのどちらを優先するかはその一例です。市場投入のスピードを向上する、新しい機能を導入する、締め切りに間に合わせるといったケースでは、前払いコストの投資を最適化するよりも、スピードを最適化した方が良い結果が得られます。データでなく時間的制約に基づいて設計上の決断が下されることがあります。また、コストを最適化するデプロイのためにベンチマークを設定することに時間を掛けるより、「万一の備え」でコストを過大に見積もる風潮は常に存在します。その結果、プロビジョニングは過剰に、最適化デプロイは過小になる場合があります。ただし、オンプレミス環境からクラウド環境に「リフト & シフト」式にリソースを移行してから最適化を図る必要がある場合は、妥当な選択といえます。適切な労力を当初からコスト最適化戦略に投入すると、ベストプラクティスが一貫して適用され、不要なオーバープロビジョニングも回避できるため、クラウドの経済的メリットをより早く実感できます。以下のセクションでは、クラウド財務管理の運用とワークロードのコスト最適化について、初期および進行中の両方に使用できるテクニックとベストプラクティスを説明します。

ベストプラクティス

トピック

クラウド財務管理を実践する

クラウドの導入により、承認、調達、インフラストラクチャのデプロイサイクルが短縮されるため、技術チームは、イノベーションをより迅速に起こすことができます。ビジネス価値と財務的な成功を実現するには、クラウドでの財務管理に対する新たなアプローチが必要です。このアプローチとはクラウド財務管理であり、組織全体での知識の蓄積、プログラム、リソース、プロセスを実装することで、組織全体の機能を構築します。

多くの組織は、異なる優先順位を持つ多数の異なる単位で構成されています。合意された一連の財務目標に整合するように組織を調整し、それらを満たすメカニズムを組織に提供することで、より効率的な組織を構築できます。有能な組織は、より迅速にイノベーションを進め、より迅速に構築し、より俊敏になり、内部的または外部的要因に合わせて自らを調整します。

AWS では、Cost Explorer や、オプションで Amazon Athena と Amazon QuickSight をコストと使用状況レポート (CUR) とともに使用することで、組織全体のコストと使用状況を把握できます。AWS Budgets は、コストと使用状況に関する事前通知を提供します。AWS ブログでは、新しいサービスや機能に関する情報を提供し、お客様が新しいサービスのリリースを常に把握できるようにしています。

以下の質問は、コスト最適化に関する考慮事項に焦点を当てています。(コスト最適化の質問、回答、およびベストプラクティスの一覧については、 付録を参照してください)。

COST 1: クラウド財務管理はどのように実装するのですか?

組織はクラウド財務管理の実装により、企業はコストと使用量を最適化し、AWS でスケールアップしながら、ビジネス価値と財務上の成功を実現できます。

コスト最適化機能を構築するときには、メンバーを活用し、CFM とコスト最適化のエキスパートでチームを補完します。既存のチームメンバーは、組織が現在どのように機能しているか、どのように改善を迅速に実施するかを理解します。また、分析やプロジェクト管理など、補足的または専門的なスキルセットを持つ人材を含めることも検討します。

組織にコスト意識を浸透させる際は、既存のプログラムやプロセスを改善または構築します。新しいプロセスやプログラムを構築するよりも、既存のものに追加する方がはるかに迅速です。これにより、結果を得るまでの時間が大幅に短縮されます。

経費支出と使用量の認識

クラウドによる柔軟性と俊敏性の向上は、イノベーションを促進し、開発とデプロイのペースを高めます。クラウドによってオンプレミスインフラストラクチャのプロビジョニングに関連した手動プロセスや時間を省くことができます。これにはハードウェア仕様の決定、価格交渉、注文管理、発送のスケジュール設定、リソースのデプロイなどが含まれます。ただし、この使いやすさと事実上無限のオンデマンドキャパシティーを生かすには、費用に対する新しい考え方が必要になります。

多くのビジネスは、さまざまなチームが運用する複数のシステムによって構成されています。リソースのコストをそれぞれの組織や製品オーナーに帰属させることができると、リソースを効率的に使用し、無駄を削減できます。コストの帰属を明確にすることで、実際に利益率の高い製品を把握し、予算の配分先についてより多くの情報に基づいた決定ができるようになります。

AWS では、AWS Organizations や AWS Control Tower といったアカウント体系を作成し、コストや使用量を分離して配分することができます。リソースのタグ付けを使用して、ビジネスや組織の情報を使用量とコストに適用することもできます。AWS Cost Explorer を使用してコストと使用状況を可視化するか、Amazon Athena と Amazon QuickSight でカスタマイズされたダッシュボードと分析を作成します。コストと使用量の制御は、AWS Budgets による通知と、AWS Identity and Access Management (IAM) や Service Quotas を使用した制御によって行われます。

以下の質問は、コストの最適化に関する考慮事項に焦点を当てています。

COST 2: 使用状況はどのように管理すればよいですか?

発生コストを適正な範囲内に抑えつつ、目的を確実に達成するためのポリシーとメカニズムを設定します。チェックアンドバランスのアプローチを採用することで、無駄なコストを費やすことなくイノベーションが可能です。

COST 3: 使用状況とコストはどのようにモニタリングするのですか?

コストをモニタリングし、適切に配分するためのポリシー手順を定めます。これにより、ワークロードのコスト効率を測定し、向上させることができます。

COST 4: 不要なリソースはどのように削除するのですか？

プロジェクトの開始から終了まで変更管理とリソース管理を実装します。これにより、使用されていないリソースをシャットダウンまたは終了して、無駄を減らします。

コスト配分タグを使用して、AWS の使用量とコストの分類や追跡ができます。AWS リソース (EC2 インスタンスや S3 バケットなど) にタグを付けると、AWS では使用量とタグの情報を使ってコストと使用状況のレポートが生成されます。組織のカテゴリ (コストセンター、ワークロード名、所有者など) を表すタグを付けることで、複数のサービスにわたってコストを整理することができます。

コストと使用状況のレポートとモニタリングで、適切なレベルの詳細および粒度を使用していることを確認します。概括的なインサイトと傾向を知るには、AWS Cost Explorer で日常的な詳細度を使用します。より詳細な分析と検査を行うには、AWS Cost Explorer で時間単位の粒度を使用するか、コストと使用状況レポート (CUR) を使用して時間単位の粒度で Amazon Athena と Amazon QuickSight を使用します。

リソースのタグ付けとエンティティのライフサイクル追跡 (従業員、プロジェクト) を組み合わせることで、組織に価値をもたらしておらず、廃止する必要がある孤立したリソースやプロジェクトを特定できるようになります。予測されている超過支出を通知する請求アラートをセットアップできます。

費用対効果の高いリソース

ワークロードにとって適切なインスタンスとリソースを使用することが、コスト削減の鍵になります。たとえば、レポート処理を小規模なサーバーで実行すると 5 時間かかるものの、費用が 2 倍の大規模なサーバーでは 1 時間で実行できるとします。どちらのサーバーでも同じ結果が得られますが、長期的に見れば小規模なサーバーで発生するコストの方が大きくなります。

優れた設計のワークロードでは最もコスト効率の良いリソースが使用されるため、大幅にプラスの経済的影響が生じます。また、マネージドサービスを使用することで、コストを削減できる場合もあります。例えば、E メールを配信するためにサーバーを保守することなく、メッセージ単位で料金が発生するサービスを使用できます。

AWS には柔軟で費用対効果の高いさまざまな料金オプションがあり、お客様のニーズに最も適した方法で Amazon EC2 やその他のサービスのインスタンスを利用できます。オンデマンド インスタンス 最低基本料金はなく、コンピューティング性能に対して時間単位で利用料金が発生します。Savings Plans およびリザーブドインスタンス では、オンデマンド料金を最大 75% 節約できます。スポットインスタンスでは、使用されていない Amazon EC2 キャパシティーを活用し、オンデマ

ンド料金と比べて最大 90% 節約できます。スポットインスタンス は、ステートレスなウェブサーバー、バッチ処理など、個々のサーバーが動的に追加または削除されるサーバー群の使用をシステムが許容する場合や、HPC やビッグデータを使用する場合に適しています。

サービスを適切に選択することでも、使用量とコストを削減することができます。たとえば、CloudFront を使用するとデータ転送を最小限に抑えられます。コストを完全に排除できる場合もあります。たとえば、RDS で Amazon Aurora を活用すれば、高額のデータベースライセンスコストが発生しません。

以下の質問は、コストの最適化に関する考慮事項に焦点を当てています。

COST 5: サービスを選択する際、どのようにコストを評価するのですか?

Amazon EC2、Amazon EBS、Amazon S3 は、いわば AWS のビルディングブロック的なサービスです。Amazon RDS や Amazon DynamoDB などのマネージドサービスは 、より高レベル、つまりアプリケーションレベルの AWS のサービスです。基盤となるサービスやマネージドサービスを適切に選択することで、このワークロードのコストを最適化できます。例えば、マネージドサービスを使用することで、管理や運用によって発生するオーバーヘッドを削減またはゼロにでき、アプリケーション開発やビジネス上の他の活動に注力できるようになります。

COST 6: リソースタイプとサイズを選択する際、どのようにコスト目標を達成するのですか?

対象タスクについて適切なリソースサイズおよびリソース数を選択していることを確認します。最もコスト効率の高いタイプ、サイズ、数を選択することで、無駄を最小限に抑えます。

COST 7: コストを削減するには、料金モデルをどのように使用すればよいですか?

リソースのコストを最小限に抑えるのに最も適した料金モデルを使用します。

COST 8: データ転送料金についてどのように計画すればよいですか ?

データ転送料金を計画し、モニタリングすることで、これらのコストを最小化するためのアーキテクチャ上の決定を下すことができます。小規模でも効果的なアーキテクチャ変更により、長期的な運用コストを大幅に削減できる場合があります。

サービスの選択時にコストを考慮に入れ、Cost Explorer や AWS Trusted Advisor などのツールを使って定期的に AWS の使用状況を確認することで、使用状況を積極的にモニタリングし、必要に応じてデプロイを調整することができます。

需要を管理しリソースを供給する

クラウドに移行すると、お支払いは必要な分のみになります。必要な時にワークロードの需要に合わせたリソースを供給できるため、コストがかかる無駄なオーバープロビジョニングの必要性を排除できます。また、スロットル、バッファ、またはキューを使用して需要を変更することで、需要を円滑に処理し、少ないリソースで供給してコストを削減したり、後でバッチサービスで処理したりできます。

AWS では、ワークロードの需要に合わせてリソースを自動的にプロビジョニングできます。需要または時間ベースのアプローチを使用した Auto Scaling で、必要に応じてリソースを追加および削除することが可能となります。需要の変化が予測できる場合、さらに費用を削減し、リソースをワークロードのニーズに確実に合わせることができます。Amazon API Gateway を使用してスロットリングを実装したり、Amazon SQS を使用してワークロードにキューを実装したりできます。いずれの場合も、ワークロードコンポーネントの需要を変更できます。

以下の質問は、コスト最適化に関する考慮事項に焦点を当てています。

COST 9: どのように需要を管理し、リソースを供給するのですか?

費用とパフォーマンスのバランスが取れたワークロードを作成するには、費用を掛けたすべてのものが活用されるようにし、使用率が著しく低いインスタンスが生じるのを回避します。利用率の指標に偏りがあると、運用コスト (過剰利用によるパフォーマンスの低下)、または AWS の無駄な支出 (過剰プロビジョニングによる) のどちらかで、組織に悪影響を及ぼします。

需要を変更してリソースを提供するように設計する場合、使用パターン、新しいリソースのプロビジョニングにかかる時間、および需要パターンの予測可能性を積極的に検討します。需要を管理する際は、適切なサイズのキューまたはバッファがあり、要求される時間内にワークロードの需要に対応していることを確認します。

継続的最適化

AWS では新しいサービスと機能がリリースされるため、既存のアーキテクチャの決定をレビューし、現在でもコスト効率が最も優れているかどうかを確認することがベストプラクティスです。要件の変化に応じて、不要になったリソース、サービス全体、システムを積極的に廃止してください。

新しい機能またはリソースタイプを実装すると、変更の実装に必要な労力を最小限に抑えながら、ワークロードを段階的に最適化できます。これにより、時間の経過とともに効率が継続的に向上し、最新のテクノロジーを利用して運用コストを削減できます。新しいサービスを使用して、ワークロードに新しいコンポーネントを置き換えたり、追加したりすることもできます。これにより、効率が大幅に向上するため、ワークロードのレビューを定期的に行い、新しいサービスや機能を実装することが不可欠です。

以下の質問は、コスト最適化に関する考慮事項に焦点を当てています。

COST 10: 新しいサービスをどのように評価するのですか？

AWS では新しいサービスと機能がリリースされるため、既存のアーキテクチャの決定をレビューし、現在でもコスト効率が最も優れているかどうかを確認することがベストプラクティスです。

デプロイを定期的に見直すときには、新しいサービスでどのようにコストを節約できるか評価します。たとえば、RDS で Amazon Aurora を使用すると、リレーショナルデータベースのコストを削減できます。Lambda などのサーバーレスを使用すると、コードを実行するためにインスタンスを運用、管理する必要がなくなります。

リソース

コスト最適化に関するベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメント

- AWS ドキュメント

ホワイトペーパー

- コスト最適化の柱

サステナビリティ

持続可能性の柱は、環境に対する影響、特にエネルギーの消費と効率性に焦点を当てています。これらは、アーキテクトにとって、リソースの使用量を削減するための直接的な対応がわかる重要な手段であるためです。実装に関する規範的なガイダンスとして 持続可能性の柱に関するホワイトペーパー.

トピック

設計原則

クラウドでの持続可能性には 6 つの設計原則があります。

- 影響を理解する: クラウドワークロードの影響を計測し、ワークロードの将来の影響をモデル化します。顧客がお客様の製品を使用することによる影響、および最終的に製品を廃止および使用停止する際の影響などを含む、すべての影響源を含めます。作業単位ごとに必要なリソースと排出量を確認し、生産量と、クラウドワークロードの全影響を比較します。このデータを利用して重要業績評価指標 (KPI) を作成し、影響を抑えながら生産性を向上させる方法を評価して、提案された変更による影響を経時的に見積もります。
- 持続可能性の目標を設定する: クラウドワークロードごとに、持続可能性の長期目標を立てます。トランザクションごとのコンピューティングリソースやストレージリソースの削減などです。既存のワークロードに対する持続可能性向上のための投資のリターンをモデル化し、持続可能性目標に必要な投資のリソースを所有者に与えます。成長計画を立て、その成長により、ユーザー単位やトランザクション単位など適した単位に対して計測される影響の大きさが結果的に削減できるようにワークロードを構築します。目標により、ビジネスや組織のより大きな持続可能性目標の支援、回帰の特定、改善できる可能性のある分野の優先順位付けが可能になります。
- 使用率を最大化する: ワークロードのサイズを適正化し効率的な設計を実装して、使用率を高く保ち、基盤となるハードウェアのエネルギー効率を最大化します。ホスト単位のベースライン電力消費量があるため、使用率 30% のホスト 2 つは、使用率 60% のホスト 1 つよりも効率が悪くなります。同時に、アイドル状態のリソース、処理、ストレージをなくすか、最小化して、ワークロードに必要な合計エネルギー量を削減します。
- より効率的なハードウェアやソフトウェアの新製品を予測して採用する: パートナーやサプライヤーが行っているアップストリームの改善をサポートし、お客様のクラウドワークロードへの影響の軽減に役立てます。より効率的なハードウェアやソフトウェアの新製品を継続的にモニターし評価します。新しい効率的なテクノロジーを迅速に採用できるように、設計に柔軟性を持たせます。
- マネージドサービスを使用する: 広範な顧客ベースでサービスを共有することで、リソースの使用率を最大化できます。こうすることで、クラウドワークロードをサポートするために必要なインフラストラクチャ数を削減できます。例えば、ワークロードを AWS クラウド に移行し、サーバーレスコンテナに AWS Fargate などのマネージドサービスを採用することで、電力やネットワーク

など、データセンターに共通するコンポーネントの影響を顧客間で共有できます。マネージドサービスは、AWS が大規模に運用し、効率的なオペレーションについて責任を持ちます。お客様の影響を最小化できるマネージドサービスを使用します。例えば、Simple Storage Service (Amazon S3) ライフサイクル設定を使用して、あまり頻繁にアクセスされていないデータを自動的にコールドストレージに移動したり、Amazon EC2 Auto Scaling を使用して容量を需要に合わせたりできます。

- クラウドワークロードのダウンストリームの影響を軽減する: お客様のサービスを使用するために必要なエネルギーやリソースの量を削減します。お客様のサービスを使用するために顧客がデバイスをアップグレードしなければならない必要性を削減または排除します。Device Farm を使用したテストで予想される影響を理解し、顧客とともにテストしてお客様のサービスを使用することによる実際の影響を理解します。

定義

クラウドでの持続可能性には、6 つのベストプラクティス領域があります。

- リージョンの選択
- ユーザーの行動パターン
- ソフトウェアとアーキテクチャのパターン
- データパターン
- ハードウェアパターン
- 開発とデプロイのプロセス

クラウドでの持続可能性は、プロビジョニングされたリソースを最大限に活用し、必要な合計リソースを最小化することによって、ワークロードのすべてのコンポーネントにわたってエネルギーの削減と効率を主な主眼とした継続的取り組みです。この取り組みは、効率的なプログラミング言語の最初の選択から、最新のアルゴリズムの採用、効率的なデータストレージ技法の使用、適切にサイズ設定された効率的なコンピューティングインフラストラクチャへのデプロイ、高出力のエンドユーザーハードウェアの要件の最小化まで多岐にわたります。

ベストプラクティス

トピック

- ソフトウェアとアーキテクチャのパターン
- データパターン
- ハードウェアパターン
- 開発とデプロイのパターン
- リソース

リージョンの選択

ビジネス要件と持続可能性目標の両方に基づいて、ワークロードを実装するリージョンを選択します。

以下の質問は、持続可能性に関する考慮事項に焦点を当てています。(持続可能性に関する質問とベストプラクティスの一覧については、 付録を参照してください。)

SUS 1: リージョンをどのように選択して、持続可能性目標を目指しますか?

Amazon の再生可能エネルギープロジェクトに近いリージョンであり、グリッドの公開されている炭素集約度が他の場所 (またはリージョン) よりも低いリージョンを選択します。

ユーザーの行動パターン

ユーザーがワークロードやその他のリソースを使用する方法によって、持続可能性の目標を達成するための改善点を特定できます。継続的にユーザーの負荷に合うようにインフラストラクチャをスケールし、ユーザーをサポートするために必要な最小リソースのみがデプロイされているようにします。サービスレベルをお客様のニーズと整合させます。ユーザーがリソースを消費するために必要なネットワークを制限できるようにリソースを配置します。未使用の既存アセットを削除します。作成されたが未使用のアセットを特定し、作成を止めます。チームメンバーには、必要な機能をサポートし持続可能性への影響を最小限にするデバイスを提供します。

以下の質問は、持続可能性に関する、この考慮事項に焦点を当てています。

SUS 2: ユーザーの行動パターンをどのように利用して、持続可能性目標を目指しますか?

ユーザーがワークロードやその他のリソースを使用する方法によって、持続可能性の目標を達成するための改善点を特定できます。継続的にユーザーの負荷に合うようにインフラストラクチャ

SUS 2: ユーザーの行動パターンをどのように利用して、持続可能性目標を目指しますか?

をスケールし、ユーザーをサポートするために必要な最小リソースのみがデプロイされているようにします。サービスレベルをお客様のニーズと整合させます。ユーザーがリソースを消費するために必要なネットワークを制限できるようにリソースを配置します。未使用の既存アセットを削除します。作成されたが未使用のアセットを特定し、作成を止めます。チームメンバーには、必要な機能をサポートし持続可能性への影響を最小限にするデバイスを提供します。

ユーザー負荷に応じたインフラストラクチャのスケーリング: 使用率が低い、または使用されていない期間を特定し、余分な容量を排除し効率性を改善します。

持続可能性目標に応じた SLA の調整: 可用性やデータ保持期間などに関するサービスレベルアグリーメント (SLA) を定義し更新して、引き続きビジネス要件を満たしながら、ワークロードをサポートするために必要なリソース数を最小化します。

使用されていないアセットの作成と保守をなくす: アプリケーションアセット (事前コンパイル済みのレポート、データセット、静的イメージなど) とアセットのアクセスパターンを分析し、冗長性、低使用率、および廃止できそうなターゲットを特定します。生成されたアセットを冗長なコンテンツ (重複または共通のデータセットと出力が含まれる月次レポートなど) と統合し、出力が重複する際に消費されるリソースをなくします。使用されていないアセット (既に販売していない製品の画像など) を廃止し、消費されていたリソースを解放して、ワークロードをサポートするために使用されるリソース数を削減します。

ユーザーの場所に応じてワークロードの地理的配置を最適化する: ネットワークのアクセスパターンを分析し、顧客が地理的にどこから接続しているかを特定します。ネットワークトラフィックが経由しなければならない距離を削減できるリージョンとサービスを選択し、ワークロードをサポートするために必要なネットワークリソースの総量を減らします。

実行されるアクティビティに応じてチームメンバーのリソースを最適化する: チームメンバーに提供されるリソースを最適化することで、ニーズをサポートしながら持続可能性への影響を最小限に抑えます。例えば、レンダリングやコンパイルなどの複雑なオペレーションを、使用率が低く高パワーの単一のユーザーシステムで行うのではなく、使用率の高い共有クラウドデスクトップで行います。

ソフトウェアとアーキテクチャのパターン

負荷平滑化を実行しデプロイされたリソースが一貫して高使用率で維持されるパターンを実装し、リソースの消費を最小化します。時間の経過とともにユーザーの行動が変化したため、コンポーネントが使用されずアイドル状態になることがあります。パターンとアーキテクチャを改定して、使用率の

低いコンポーネントを統合し、全体の使用率を上げます。不要になったコンポーネントは使用停止にします。ワークロードコンポーネントのパフォーマンスを理解し、リソースの消費が最も大きいコンポーネントを最適化します。顧客がお客さまのサービスにアクセスするために使用するデバイスを把握し、デバイスをアップグレードする必要性を最小化するパターンを実装します。

以下の質問は、持続可能性に関する考慮事項に焦点を当てています。

SUS 3: ソフトウェアとアーキテクチャのパターンをどのように利用して、持続可能性目標を目指しますか?

負荷平滑化を実行しデプロイされたリソースが一貫して高使用率で維持されるパターンを実装し、リソースの消費を最小化します。時間の経過とともにユーザーの行動が変化したため、コンポーネントが使用されずアイドル状態になることがあります。パターンとアーキテクチャを改定して、使用率の低いコンポーネントを統合し、全体の使用率を上げます。不要になったコンポーネントは使用停止にします。ワークロードコンポーネントのパフォーマンスを理解し、リソースの消費が最も大きいコンポーネントを最適化します。顧客がお客さまのサービスにアクセスするために使用するデバイスを把握し、デバイスをアップグレードする必要性を最小化するパターンを実装します。

スケジュールされた非同期のジョブに応じてソフトウェアとアーキテクチャを最適化する: 効率的なソフトウェア設計とアーキテクチャを使用し、作業単位ごとに必要な平均リソースを最小化します。コンポーネントの使用率が平均化されるメカニズムを実装し、タスクの合間でアイドルになるリソースを削減して、負荷のスパイクの影響を最小化します。

低使用率または使用されてないワークロードコンポーネントを削除またはリファクタリングする: ワークロードアクティビティをモニタリングして、個別のコンポーネントの時間経過による使用率の変化を特定します。未使用や不要のコンポーネントを削除し、使用率が低いコンポーネントはリファクタリングして、無駄になるリソースを制限します。

時間またはリソースの大部分を消費しているコード領域を最適化する: ワークロードアクティビティをモニタリングして、リソースを最も多く消費しているアプリケーションコンポーネントを特定します。それらのコンポーネント内で実行されているコードを最適化して、パフォーマンスを最大化しながらリソースの使用量を最小化します。

顧客のデバイスや機器への影響を最適化する: お客さまのサービスを利用するために顧客が使用しているデバイスや機器、その予想ライフサイクル、およびそれらのコンポーネントを交換することによる財務および持続可能性に対する影響を理解します。顧客のデバイスの交換や機器のアップグレード

の必要性を最小化するソフトウェアパターンとアーキテクチャを実装します。例えば、新機能の実装には、より古いハードウェアやオペレーティングシステムのバージョンと後方互換性のあるコードを使用します。また、ペイロードのサイズを管理して、対象デバイスのストレージ容量を超えないようにします。

データアクセスとストレージのパターンを最も良くサポートできるソフトウェアパターンとアーキテクチャを使用する: データがワークロード内でどのように使用され、ユーザーによってどのように消費され、どのように転送および保存されているかを理解します。データの処理と保存の要件を最小化するテクノロジーを選択します。

データパターン

負荷平滑化を実行しデプロイされたリソースが一貫して高使用率で維持されるパターンを実装し、リソースの消費を最小化します。時間の経過とともにユーザーの行動が変化したため、コンポーネントが使用されずアイドル状態になることがあります。パターンとアーキテクチャを改定して、使用率の低いコンポーネントを統合し、全体の使用率を上げます。不要になったコンポーネントは使用停止にします。ワークロードコンポーネントのパフォーマンスを理解し、リソースの消費が最も大きいコンポーネントを最適化します。顧客がお客さまのサービスにアクセスするために使用するデバイスを把握し、デバイスをアップグレードする必要性を最小化するパターンを実装します。

以下の質問は、持続可能性に関する考慮事項に焦点を当てています。

SUS 4: データのアクセスパターンおよび使用パターンをどのように利用して、持続可能性目標を目指しますか?

データ管理プラクティスを実装して、ワークロードのサポートに必要なプロビジョンされたストレージと、それを使用するために必要なリソースを削減します。データを理解し、データのビジネス価値とデータの使用方法を最もよくサポートするストレージテクノロジーと設定を使用します。必要性が小さくなった場合はより効率的で性能を落としたストレージにデータをライフサイクルし、データが不要になった場合は削除します。

データ分類ポリシーを実装する: データを分類して、ビジネス成果にとっての重要性を理解します。この情報を使用して、データをよりエネルギー効率が高いストレージに移動するタイミングや、安全に削除するタイミングを検討します。

データアクセスとストレージのパターンをサポートするテクノロジーを使用する: データへのアクセス方法や保存方法を最も良くサポートするストレージを使用し、ワークロードをサポートしながら、

プロビジョニングされるリソースを最小化します。例えば、ソリッドステートドライブ (SSD) は磁気式ドライブよりもエネルギー消費が大きいため、アクティブなデータのユースケースのみに使用するべきです。アクセスの少ないデータに対して、エネルギー効率の高いアーカイブクラスのストレージを使用します。

ライフサイクルポリシーを使用して、不要なデータを削除する: すべてのデータのライフサイクルを管理し、削除タイムラインを自動的に適用して、ワークロードに必要な合計ストレージを最小化します。

ブロックストレージの過剰プロビジョニングを最小化する: プロビジョニングされる合計ストレージを最小化するには、ワークロードに適したサイズ割り当てのブロックストレージを作成します。伸縮自在なボリュームを使用し、データの増加に合わせて、コンピューティングリソースに添付されたストレージをサイズ変更することなく拡張します。伸縮自在なボリュームを定期的に確認し、現在のデータサイズに合わせてプロビジョンされたボリュームを縮小します。

不要なデータまたは冗長なデータを削除する: データの複製は必要なときにのみ行い、消費される合計ストレージを最小化します。ファイルおよびブロックレベルでデータの重複を排除するバックアップテクノロジーを使用します。SLA の要件を満たすために必要な場合を除き、RAID (Redundant Array of Independent Drives) 設定の仕様を制限します。

共有データへのアクセスには共有ファイルシステムまたはオブジェクトストレージを使用する: 共有ストレージと単一の信頼できるソースを採用し、データの複製を避けてワークロードに必要な合計ストレージを削減します。必要に応じて共有ストレージからデータを取得します。未使用なボリュームをデタッチしてリソースを解放します。ネットワーク間でのデータ移動を最小限に抑える: 共有ストレージを使用し、その地域のデータストアからデータにアクセスして、ワークロードにおけるデータ移動をサポートするために必要なネットワークリソースの総量を最小化します。

再作成が困難なときのみデータをバックアップする: ストレージの消費を最小化するには、ビジネス価値のあるデータまたはコンプライアンス要件を満たすために必要なデータのみをバックアップします。バックアップポリシーを精査し、リカバリーシナリオでは価値のないエフェメラルストレージを除外します。

ハードウェアパターン

ハードウェア管理のプラクティスを変更することで、ワークロードの持続可能性に対する影響を軽減する機会を探します。プロビジョンおよびデプロイする必要があるハードウェア数を最小化し、個別のワークロードにおいて最も効率のいいハードウェアを選択します。

以下の質問は、持続可能性に関する考慮事項に焦点を当てています。

SUS 5: ハードウェアの管理および使用のプラクティスは、持続可能性目標を目指すうえでどのように役立ちますか?

ハードウェア管理のプラクティスを変更することで、ワークロードの持続可能性に対する影響を軽減する機会を探します。プロビジョンおよびデプロイする必要があるハードウェア数を最小化し、個別のワークロードにおいて最も効率のいいハードウェアを選択します。

最小量のハードウェアを使用してニーズを満たす: クラウドの能力を使用して、ワークロードの実装を頻繁に変更できます。ニーズの変化に応じて、デプロイされたコンポーネントを更新します。

影響が最も少ないインスタンスタイプを使用する: 新しいインスタンスタイプのリリースを継続的にモニタリングし、エネルギー効率の改善を活用します。例えば、機械学習のトレーニングや推論、ビデオのトランスコーディングなど、特定のワークロードをサポートするように設計されたインスタンスタイプなどです。

マネージドサービスを使用する: マネージドサービスは、平均使用率を高く保つ責任と、デプロイされたハードウェアの持続可能性に対する最適化の責任を AWS に移します。マネージドサービスを使用して、サービスの持続可能性に対する影響を、そのサービスのすべてのテナント間に分散し、お客様単体の関与を軽減します。

GPU の使用を最適化する: グラフィック処理ユニット (GPU) は高い電力消費のソースになることがあります。GPU ワークロードの種類は多く、レンダリング、トランスコーディング、機械学習トレーニング、モデリングなどさまざまです。GPU インスタンスは必要な時間だけ実行し、必要がないときはオートメーションで廃棄して、消費されるリソースを最小化します。

開発とデプロイのパターン

開発、テスト、デプロイのプラクティスを変更することで、持続可能性に対する影響を減らす機会を探します。

以下の質問は、持続可能性に関する考慮事項に焦点を当てています。

SUS 6: 開発およびデプロイのプロセスは、持続可能性目標を目指すうえでどのように役立ちますか?

開発、テスト、デプロイのプラクティスを変更することで、持続可能性に対する影響を減らす機会を探します。

持続可能性の改善を迅速に導入できる方法を採用する: 潜在的な改善をテストおよび検証してから、本稼働環境にデプロイします。改善に際して将来的に起こりうる利点を計算する際のテストにかかるコストを考慮します。低コストのテスト方法を開発し、小規模な改善を実施します。

ワークロードを最新に保つ: 最新のオペレーティングシステム、ライブラリ、およびアプリケーションを使用すると、ワークロードの効率が高まり、より効率的なテクノロジーを簡単に導入できます。最新のソフトウェアにはまた、ワークロードの持続可能性に対する影響をより正確に測定する機能が含まれている場合があります。これは、ベンダーが独自の持続可能性の目標を満たすための機能でもあります。

ビルド環境の利用率を高める: オートメーションと Infrastructure as Code を使用して、必要に応じて本番稼働前の環境を起動し、使用しないときは停止します。一般的なパターンとしては、開発チームのメンバーの勤務時間と重なるように可用性期間のスケジュールを設定することがあります。休止は、状態を維持し、必要なときにのみインスタンスを迅速にオンラインにする便利な手段です。バーストキャパシティー、スポットインスタンス、伸縮自在なデータベースサービス、コンテナ、その他のテクノロジーを備えたインスタンスタイプ使用して、開発およびテストのキャパシティーを使用に合わせて調整します。

テストにマネージド型のデバイスファームを使用する: マネージド型のデバイスファームは、ハードウェアの製造やリソースの使用が持続可能性に及ぼす影響を複数のテナントに分散させます。マネージド型 Device Farm は、さまざまなデバイスタイプを提供するため、あまり使われない古いハードウェアをサポートすることで、不要なデバイスのアップグレードによるお客様の持続可能性に対する影響を回避できます。

リソース

持続可能性のベストプラクティスの詳細については、以下のリソースを参照してください。

ホワイトペーパー

- 持続可能性の柱

動画

- The Climate Pledge

レビュープロセス

アーキテクチャレビューは、深く掘り下げることを推奨し、問題のない一貫した方法で行う必要があります。話し合いで行う簡易なプロセス (数日ではなく数時間) であり、監査ではありません。アーキテクチャレビューの目的は、対策を必要とする重大な問題や改善可能な領域を特定することです。レビュー結果は、お客様のワークロードの扱いやすさを改善する対策です。

「アーキテクチャ」で説明したとおり、各チームメンバーがアーキテクチャの品質に責任を持つ必要があります。Well-Architected フレームワークに基づいてアーキテクチャを作成するチームメンバーは、形式ばったレビューミーティングを開催するよりも、アーキテクチャを継続してレビューすることが推奨されています。レビューを継続することで、チームメンバーはアーキテクチャの変化に応じて回答を更新したり、機能を提供しながらアーキテクチャを改善したりすることができます。

AWS Well-Architected フレームワークは、AWS がシステムとサービスについて内部でレビューを行う方法に適合しています。これは設計方法を左右する設計原則と、根本原因の分析 (RCA) でよく取り上げられる領域が軽視されないようにするための質問に基づきます。内部システム、AWS のサービス、またはお客様に重大な問題があるときは、RCA を検討し、使用するレビュープロセスを改善できるかどうかを確認します。

レビューは、製品ライフサイクルの主要なマイルストーンで、設計フェーズの早い段階に適用して回避する必要がある 一方向の扉 変更が困難なため、本番稼働日前に行います。(多くの決定はやり直すことができます。つまり「Two Way Door」(双方向の扉) です。こうした決定には簡易なプロセスを使用できます。一方向の扉の場合は、やり直しが困難か不可能であるため、決定前に綿密な調査が必要です。) 本番稼働開始後に新しい機能の追加やテクノロジーの実装の変更を行うにしたがい、ワークロードは進化し続けるようになります。ワークロードのアーキテクチャは時間とともに変わります。アーキテクチャを変えるにつれてアーキテクチャの特性が劣化しないように適切な予防策を取る必要があります。アーキテクチャを大幅に変更するときには、Well-Architected レビューを含めて、予防プロセスに従う必要があります。

1 回限りのスナップショットまたは独立した測定としてレビューを活用するには、すべての適切な担当者を話し合いに参加させる必要があります。レビューが実施されたことにより、チームが実装した内容を初めて本当に理解したということはよくあります。別のチームのワークロードをレビューするときに有効な方法は、そのアーキテクチャについて何度か気軽に話し合うことです。ほとんどの質問に対する回答はそれで得られます。その後に数回の会議で曖昧な領域や気付いたリスクについて解明したり、掘り下げたりすることができます。

会議を進行するために次のアイテムをお勧めします。

- ホワイトボードのある会議室
- 印刷した構成図や設計ノート
- 回答するために帯域外の調査を必要とする質問のアクションリスト (「暗号化を有効にしましたか?」など)

レビューが完了すると、ビジネスの状況に基づいて優先順位を付けることができる問題の一覧が表示されます。それらの課題がチームの日常業務に及ぼす影響を考慮する必要もあります。これらの問題を早期に解決すれば、繰り返し発生する問題を解決するのではなく、ビジネス価値の創出に取り組むための時間ができます。課題に対処する際にレビューを更新することで、アーキテクチャがどのように改善されているのかを確認することができます。

レビューの価値は 1 度やってみると明らかになるのですが、新しいチームが当初抵抗することがあります。レビューの利点をチームに知らせることで、次のような反論に対処できます。

- 「私たちは忙し過ぎる!」(チームが大規模なローンチに向けて準備をしているときによく言われます)
  - 大きな仕事を始める準備をしているならば、それが円滑に進む必要があります。レビューを実施することによって、見逃していたかもしれない問題を把握できます。
  - 製品ライフサイクルの早い段階でレビューを実施して、リスクを洗い出し、機能提供ロードマップに沿ったリスク軽減計画を立てることをお勧めします。
- 「結果が出ても対策をとる時間がない!」 (スーパーボールなどの動かせないイベントを目標としているときによく言われます)
  - そのようなイベントを動かすことはできません。アーキテクチャのリスクを把握せずに、本当に進めるつもりですか。 これらの問題のすべてに対策を講じることができない場合でも、問題が生じたときの対処法を準備しておくことはできます。
- 「ソリューションの実装の秘密を他人に知られたくない!」
  - Well-Architected フレームワークの質問を示した場合、取り引きや技術に関する専有情報を開示する質問が 1 つもないことをチームは理解するでしょう。

組織内の他のチームと数回のレビューを実施すると、テーマに関する問題が見つかることがあります。例えば、特定の柱または主題に関して多くの課題を抱えているチームが複数あるかもしれません。すべてのレビューを総合的に検討し、それらの主題に関する課題に対処するのに役立つメカニズム、トレーニング、またはプリンシパルエンジニアリングの説明を見つける必要があります。

まとめ

AWS Well-Architected フレームワークは、信頼性、安全性、効率性、コスト効率、持続可能性を備えたシステムをクラウドで設計、運用するための 6 つの柱にまたがるアーキテクチャのベストプラクティスを提供します。このフレームワークには、既存のアーキテクチャまたは提案されているアーキテクチャをレビューするための質問が用意されています。それぞれの柱に関する AWS のベストプラクティスも提供します。このフレームワークをアーキテクチャに適用し、安定した効率のよいシステムを構築することにより、機能面の要件に注力できます。

寄稿者

本ドキュメントは、次の人物および組織が寄稿しました。

- Brian Carlson、オペレーションリーダー、優れた設計、アマゾン ウェブ サービス
- Ben Potter、Well-Architected セキュリティリード、アマゾン ウェブ サービス
- Seth Eliot、Well-Architected リライアビリティリード、アマゾン ウェブ サービス
- Eric Pullen, Sr.ソリューションズアーキテクト、アマゾン ウェブ サービス
- Rodney Lester、プリンシパルソリューションズアーキテクト、アマゾン ウェブ サービス
- Jon Steele、シニアテクニカルアカウントマネージャー、アマゾン ウェブ サービス
- Max Ramsay、プリンシパルセキュリティソリューションズアーキテクト、アマゾン ウェブ サービス
- Callum Hughes、ソリューションズアーキテクト、アマゾン ウェブ サービス
- Aden Leirer、Well-Architected コンテンツプログラムマネージャー、アマゾン ウェブ サービス

その他の資料

AWS アーキテクチャセンター

AWS クラウドコンプライアンス

AWS Well-Architected パートナープログラム

AWS Well-Architected Tool

AWS Well-Architected ホームページ

運用上の優秀性の柱に関するホワイトペーパーを参照してください。

セキュリティの柱に関するホワイトペーパーを参照してください

信頼性の柱に関するホワイトペーパーを参照してください

パフォーマンス効率の柱に関するホワイトペーパーを参照してください。

コスト最適化の柱に関するホワイトペーパーを参照してください。

持続可能性の柱に関するホワイトペーパー

Amazon Builders' Library で読み、学ぶことができます

改訂履歴

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

変更	説明	日付
マイナーな更新	付録に労力レベルの定義を追加し、ベストプラクティスを更新しました。	October 20, 2022
ホワイトペーパーの更新	持続可能性の柱を追加し、リンクを更新しました。	December 2, 2021
メジャーアップデート	持続可能性の柱がフレームワークに追加されました。	November 20, 2021
マイナーな更新	非包括的言語を削除しました。	April 22, 2021
マイナーな更新	多数のリンクを修正しました。	March 10, 2021
マイナーな更新	全体を通した編集のマイナー変更。	July 15, 2020
新しいフレームワークの更新	ほぼすべての質問と回答を見直して書き換えています。	July 8, 2020
ホワイトペーパーの更新	AWS Well-Architected Tool、AWS Well-Architected ラボへのリンク、および AWS Well-Architected パートナーの追加と、多言語バージョンのフレームワークを可能にする軽微な修正。	July 1, 2019
ホワイトペーパーの更新	質問の焦点が一度に 1 つのトピックに当たるように、ほ	November 1, 2018

	とんどの質問と回答を見直して書き換えました。これにより、一部の以前の質問が複数の質問に分割されました。定義に共通の用語を追加しました (ワークロード、コンポーネントなど)。本文の質問の表示を説明テキストを含むように変更しました。	
ホワイトペーパーの更新	質問文を平易にし、回答を標準化し、読みやすさを改善しました。	June 1, 2018
ホワイトペーパーの更新	他の柱を包括するように運用性を他の柱の前に移動して書き換えました。その他の柱に AWS の進化を新たに反映させました。	November 1, 2017
ホワイトペーパーの更新	フレームワークを更新しました。運用性の柱を追加し、他の柱を変更および更新して重複箇所を減らしました。多くのお客様と実施した評価から学んだことを盛り込みました。	November 1, 2016
マイナーな更新	付録を最新の Amazon CloudWatch Logs の情報を使用して更新しました。	November 1, 2015
初版発行	AWS Well-Architected フレームワークが公開されました。	October 1, 2015

付録: 質問とベストプラクティス

トピック

運用上の優秀性

トピック

組織

質問

OPS 1 優先順位はどのように決定すればよいですか?

だれもが、ビジネスを成功させるうえで自分が果たす役割を理解する必要があります。リソースの優先順位を設定するため、共通の目標を設定してください。これにより、取り組みから得られるメリットが最大化されます。

ベストプラクティス

- OPS01-BP01 外部顧客のニーズを評価する
- OPS01-BP02 内部顧客のニーズを評価する
- OPS01-BP03 ガバナンス要件を評価する
- OPS01-BP04 コンプライアンス要件を評価する
- OPS01-BP05 脅威の状況を評価する
- OPS01-BP06 トレードオフを評価する
- OPS01-BP07 メリットとリスクを管理する

OPS01-BP01 外部顧客のニーズを評価する

ビジネス、開発、運用チームを含む主要関係者と協力して、外部顧客のニーズに対する重点領域を決定します。これにより、望ましいビジネス成果を達成するために必要なオペレーションサポートについて十分に理解できます。

一般的なアンチパターン:

- あなたは、営業時間外にカスタマーサポートを設けないこととしましたが、サポートリクエストの履歴データを確認していません。あなたには、これが顧客に影響を与えるかどうかはわかりません。
- あなたは、新しい機能を開発していますが、当該機能が望まれているかどうか、望まれている場合はどのような形式なのかを見出すために、顧客に関与してもらっておらず、また、提供の必要性および提供方法を検証するための実験も行っていません。

このベストプラクティスを活用するメリット: ニーズが満たされている顧客は、顧客のままでいる可能性が高くなります。外部の顧客のニーズを評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ビジネスニーズの理解: ビジネスの成功は、ビジネス、開発、運用チームを含む関係者全体で目標と理解を共有することで実現できます。
  - 外部顧客のビジネス目標、ニーズ、優先順位の確認: ビジネス、開発、運用の各チームを含む主要関係者と外部顧客の目標、ニーズ、優先順位について議論します。これにより、ビジネスおよび顧客成果を達成するために必要なオペレーションサポートについて十分に理解できます。

- 共通理解の確立: ワークロードのビジネス機能、ワークロードの運用における各チームの役割、およびこれらの要因が内部および外部顧客の共通のビジネス目標をどのようにサポートするかについて、共通の理解を確立します。

リソース

関連するドキュメント:

- AWS Well-Architected Framework の概念 - フィードバックループ

OPS01-BP02 内部顧客のニーズを評価する

ビジネス、開発、運用チームを含む主要関係者と協力して、内部顧客のニーズに対する重点領域を決定します。これにより、ビジネス成果を達成するために必要なオペレーションサポートについて十分に理解できます。

確立された優先順位を使用して、改善の努力を最も影響があるところに集中させます (チームのスキルの開発、ワークロードのパフォーマンスの改善、コストの削減、ランブックの自動化、モニタリングの強化など)。必要に応じて優先順位を更新します。

一般的なアンチパターン:

- あなたは、ネットワーク管理を容易にするため、製品チームの IP アドレスの割り当てを変更することとしました。あなたは、これが製品チームに与える影響を知りません。
- あなたは、新しい開発ツールを実装しようとしていますが、当該ツールが必要とされているかどうか、または既存のプラクティスと互換性があるかどうかを知るために、社内クライアントを関与させていません。
- あなたは、新しいモニタリングシステムを実装しようとしていますが、検討されるべきモニタリングまたはレポートのニーズがあるかどうかを把握するために社内クライアントに問い合わせていません。

このベストプラクティスを確立するメリット: 社内の顧客のニーズを評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- ビジネスニーズの理解: ビジネスの成功は、ビジネス、開発、運用チームを含む関係者全体で目標を共有し、理解を深めることで実現できます。
  - 内部顧客のビジネス目標、ニーズ、優先順位の確認: ビジネス、開発、運用の各チームを含む主要関係者と連携し、内部顧客の目標、ニーズ、優先順位について議論します。これにより、ビジネスおよび顧客成果を達成するために必要なオペレーションサポートについて十分に理解できます。
  - 共通理解の確立: ワークロードのビジネス機能、ワークロードの運用における各チームの役割、およびこれらの要因が内部および外部顧客の共通のビジネス目標をどのようにサポートするかについて、共通の理解を確立します。

リソース

関連するドキュメント:

- AWS Well-Architected Framework Concepts – Feedback loop (AWS Well-Architected Framework の概念 - フィードバックループ)

OPS01-BP03 ガバナンス要件を評価する

特定のことに焦点を置くように求め、その焦点を強調する組織の定義したガイドラインや義務を把握します。組織のポリシー、標準、要件などの内部要因を評価します。ガバナンスへの変更を識別するメカニズムがあることを検証します。ガバナンス要件が特定されていない場合は、必ずこの決定にデューデリジェンスが適用されていることを確認してください。

一般的なアンチパターン:

- あなたは、監査中であり、社内のガバナンスへのコンプライアンスの証拠を提供するよう求められています。あなたは、自らのコンプライアンス要件が何かを評価したことがないため、遵守しているかどうかがわかりません。
- あなたはセキュリティ侵害の被害に遭い、その結果、金銭的な損失が発生しました。あなたは、金銭的な損失をカバーしたであろう保険が、未実施の、またはガバナンスによって要求されていない、特定のセキュリティに関する統制の実施を条件としていることがわかりました。
- あなたの管理アカウントが侵害され、その結果、会社のウェブサイトが改ざんされ、顧客の信頼が損なわれました。社内のガバナンスでは、管理アカウントのセキュリティを確保するために多要素

認証 (MFA) の使用が要求されています。あなたは、管理アカウントを MFA で保護していなかったため、懲戒処分の対象となりました。

このベストプラクティスを活用するメリット: 組織がワークロードに適用するガバナンス要件を評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ガバナンス要件の理解: プログラムまたは組織のポリシー、プログラムポリシー、問題またはシステム固有のポリシー、標準、手順、ベースライン、ガイドラインなどの社内のガバナンスの要素を評価します。ガバナンスへの変更を識別するメカニズムがあることを検証します。ガバナンス要件が特定されていない場合は、必ずこの決定にデューデリジェンスが適用されていることを確認してください。

リソース

関連するドキュメント:

- AWS クラウド コンプライアンス

OPS01-BP04 コンプライアンス要件を評価する

法令遵守の要件や業界標準などの外的要因を評価して、特定の重点領域の必須化や重視が必要となる可能性のあるガイドラインや義務についてしっかりと認識してください。コンプライアンス要件が特定されていない場合は、必ずこの決定にデューデリジェンスを適用してください。

一般的なアンチパターン:

- あなたは、監査中であり、業界規制へのコンプライアンスの証拠を提供するよう求められています。あなたは、自らのコンプライアンス要件が何かを評価したことがないため、遵守しているかどうかがわかりません。
- あなたの管理アカウントが侵害され、その結果、顧客データがダウンロードされ、顧客の信頼が損なわれました。業界のベストプラクティスでは、管理アカウントのセキュリティを確保するために MFA の使用が要求されています。あなたは、管理アカウントを MFA で保護していなかったため、顧客によって訴訟が提起されました。

このベストプラクティスを確立するメリット: ワークロードに適用されるコンプライアンス要件を評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- コンプライアンス要件の理解: 法令遵守の要件や業界標準などの外的要因を評価して、特定の重点領域の必須化や重視が必要となる可能性のあるガイドラインや義務についてしっかりと認識してください。コンプライアンス要件が特定されていない場合は、この決定にデューデリジェンスが適用されていることを確認してください。
  - 規制コンプライアンス要件の理解: 適合が法的に義務付けられている法令遵守の要件を確認してください。これらの要件に基づいて取り組みに注力してください。例として、プライバシーおよびデータ保護法による義務などがあります。
    - AWS コンプライアンス
    - AWS コンプライアンスプログラム
    - AWS コンプライアンスの最新ニュース
  - 業界標準とベストプラクティスの理解: Payment Card Industry Data Security Standard (PCI DSS) など、ワークロードに適用される業界標準とベストプラクティスの要件を確認してください。これらの要件に基づいて取り組みに注力してください。
    - AWS コンプライアンスプログラム
  - 内部コンプライアンス要件の理解: 組織で確立されているコンプライアンス要件とベストプラクティスを確認してください。これらの要件に基づいて取り組みに注力してください。例としては、情報セキュリティポリシーやデータ分類基準などがあります。

リソース

関連するドキュメント:

- AWS クラウド コンプライアンス
- AWS コンプライアンス
- AWS コンプライアンスの最新ニュース
- AWS コンプライアンスプログラム

OPS01-BP05 脅威の状況を評価する

ビジネスに対する脅威 (競合、ビジネスリスクと負債、運用リスク、情報セキュリティの脅威など) を評価し、リスクのレジストリで現在の情報を維持します。注力する場所を決定する際に、リスクの影響を考慮します。

それらの Well-Architected フレームワーク は、学習、測定、改善を重視しています。アーキテクチャを評価し、時間の経過とともにスケールアップする設計を実装するための一貫したアプローチを提供します。AWS は AWS Well-Architected Tool を提供しており、開発前のアプローチ、本番稼働前のワークロードの状態、本番稼働中のワークロードの状態などを確認するのに役立ちます。最新の AWS アーキテクチャのベストプラクティスと比較して、ワークロードの全体的なステータスをモニタリングし、潜在的なリスクについてインサイトを得ることができます。

AWS をご利用のお客様は、AWS のベストプラクティスと照らし合わせてアーキテクチャを評価するために、 ミッションクリティカルなワークロードの ガイド付き Well-Architected レビューを受けることもできます。エンタープライズサポートをご利用のお客様は、 クラウドでの運用へのアプローチにおけるギャップの特定を支援するように設計された運用レビューの対象となります。

これらのレビューのチーム間での関与は、ワークロードとチームの役割の成功への貢献方法に関する共通理解を確立するのに役立ちます。レビューを通じて特定されるニーズは、優先順位を決定するのに役立ちます。

AWS Trusted Advisor は、最適化を推奨する中心的なチェックのセットへのアクセスを提供するツールであり、優先順位を決定するのに役立ちます。 ビジネスおよびエンタープライズサポートのお客様 は、優先順位をさらに高めることができるセキュリティ、信頼性、パフォーマンス、コストの最適化に重点を置いた追加のチェックにアクセスできます。

一般的なアンチパターン:

- あなたは、製品に古いバージョンのソフトウェアライブラリを使用しています。あなたは、ワークロードに意図しない影響を及ぼす可能性のある問題について、ライブラリのセキュリティ更新が必要なことを認識していません。
- 最近、競合他社は、あなたの製品に関する顧客からの苦情の多くに対処する製品のバージョンをリリースしました。あなたは、これらの既知の問題の対処について優先順位付けを行っていません。
- 規制当局は、法規制コンプライアンス要件を遵守していない企業の責任を追求してきました。あなたは、未対応のコンプライアンス要件への対応に優先順位を付けていません。

このベストプラクティスを確立するメリット: 組織とワークロードに対する脅威を特定して理解することで、どの脅威に対処すべきか、その優先度、およびそれに必要なリソースを判断できます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 脅威の状況の評価: ビジネスに対する脅威 (競合、ビジネスリスクと負債、運用リスク、情報セキュリティの脅威など) を評価し、重点領域を決定する際にその影響を織り込めるようにします。
  - AWS セキュリティ速報
  - AWS Trusted Advisor
  - 脅威モデルの維持: 潜在的な脅威、計画および実施された軽減策、またその優先順位を特定する脅威モデルを確立し、維持します。脅威がインシデントとして出現する確率、それらのインシデントから回復するためのコスト、発生が予想される損害、およびそれらのインシデントを防ぐためのコストを確認します。脅威モデルの内容の変更に伴って、優先順位を変更します。

リソース

関連するドキュメント:

- AWS クラウド コンプライアンス
- AWS セキュリティ速報
- AWS Trusted Advisor

OPS01-BP06 トレードオフを評価する

競合する利益または代替アプローチ間のトレードオフの影響を評価し、重点領域を決定するか、一連のアクションを選択する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、新しい機能の市場投入までの時間を短縮することは、コストの最適化よりも重視されることがあります。または、非リレーショナルデータ用にリレーショナルデータベースを選択すれば、データ型に合わせて最適化されたデータベースに移行してアプリケーションを更新するよりも、システムの移行が簡素化されます。

AWS Support は、AWS とそのサービスについてチームを教育し、選択がどのようにワークロードに影響を与えるかについての理解を深める支援を行います。チームを教育するには、 AWS Support (AWS ナレッジセンター、 AWS ディスカッションフォーラム、および AWS Support センター) および AWS ドキュメント が提供するリソースを使用する必要があります。AWS に関しての質問に対する支援については、AWS Support センターを利用して AWS Support に連絡してください。

また、AWS の運用を通じて学んだベストプラクティスとパターンを Amazon Builders' Library で読み、学ぶことができます。その他のさまざまな有益な情報は、 AWS ブログ および 公式の AWS ポッドキャストで入手できます。

一般的なアンチパターン:

- あなたは、リレーショナルデータベースを使用して、時系列と非リレーショナルデータを管理しています。使用しているデータ型をサポートするように最適化されたデータベースオプションがありますが、あなたはソリューション間のトレードオフを評価していないため、そのメリットを認識していません。
- 投資家からは、Payment Card Industry Data Security Standards (PCI DSS) への準拠を実証することが求められています。あなたは、投資家の要求に応えることと、現在の開発活動を継続することとのトレードオフについて検討しません。代わりに、準拠を実証することなく、開発作業を進めます。あなたの投資家は、プラットフォームのセキュリティと、投資の是非に懸念を抱いて、あなたの会社に対する支援を停止します。

このベストプラクティスを活用するメリット: 選択した影響と結果を理解することで、選択肢に優先順位を付けることができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- トレードオフの評価: 競合する利益間のトレードオフの影響を評価し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、新しい機能を市場に出す速度を上げることが、コストの最適化より重視されることがあります。
- AWS Support は、AWS とそのサービスについてチームを教育し、選択がどのようにワークロードに影響を与えるかについての理解を深める支援を行います。チームを教育するには、AWS Support (AWS ナレッジセンター、AWS ディスカッションフォーラム、AWS Support センター) および AWS ドキュメントが提供するリソースを使用する必要があります。AWS に関しての質問に対する支援については、AWS Support センターを利用して AWS Support に連絡してください。
- また、AWS は Amazon Builders' Library の AWS の運用を通じて学んだベストプラクティスとパターンも共有しています。AWS ブログと公式の AWS ポッドキャストでは、その他のさまざまな有益な情報を入手できます。

リソース

関連するドキュメント:

- AWS ブログ
- AWS クラウド コンプライアンス
- AWS ディスカッションフォーラム
- AWS ドキュメント
- AWS ナレッジセンター
- AWS Support
- AWS Support センター
- Amazon Builders' Library
- 公式の AWS ポッドキャストで入手できます

OPS01-BP07 メリットとリスクを管理する

メリットとリスクを管理し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、重要な新機能を顧客に公開できるように、未解決の問題を記録するワークロードをデプロイしておくと便利な場合があります。関連するリスクを軽減できる場合もあれば、リスクが残るのを容認できない場合もあります。その場合、リスクに対処するための措置を講じることになります。

ある時点で、優先順位の小さなサブセットに注力したい場合に遭遇するかもしれません。必要な機能の開発とリスクの管理を確実にするために、長期的にバランスのとれたアプローチを使用します。必要に応じて優先順位を更新します。

一般的なアンチパターン:

- あなたは、開発者の 1 人が「インターネットで見つけた」「あなたが必要とするすべてのこと」を行うライブラリを含めることにしました。あなたは、不明なソースからこのライブラリを採用するリスクを評価しておらず、脆弱性や悪意のあるコードが含まれているかどうかはわかりません。
- あなたは、既存の問題を修正する代わりに、新しい機能を開発およびデプロイすることに決めました。あなたは、この機能がデプロイされるまで問題をそのままにしておくことのリスクを評価しておらず、顧客への影響がわかりません。
- あなたは、コンプライアンスチームから詳細不明の懸念が寄せられたため、顧客から頻繁にリクエストされる機能をデプロイしないことに決めました。

このベストプラクティスを活用するメリット: 選択肢のメリットを特定し、組織のリスクを認識することで、十分な情報に基づいて意思決定を行うことができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- メリットとリスクの管理: 決定のメリットと関連するリスクのバランスを取ってください。
  - メリットの特定: ビジネスの目標、ニーズ、優先順位に基づいてメリットを特定します。例として、市場投入までの時間、セキュリティ、信頼性、パフォーマンス、コストなどがあります。
  - リスクの特定: ビジネスの目標、ニーズ、優先順位に基づいてリスクを特定します。例として、市場投入までの時間、セキュリティ、信頼性、パフォーマンス、コストなどがあります。
  - リスクに対するメリットの評価と情報に基づく意思決定: ビジネス、開発、運用を含む主要関係者の目標、ニーズ、優先順位に基づいてメリットとリスクの影響を決定します。メリットの価値を、リスクが現実化する可能性とその影響のコストに照らして評価します。たとえば、信頼性よりも市場投入までのスピードを重視すると、競争上の優位性が得られます。ただし、信頼性の問題がある場合、稼働時間が短くなる場合があります。

OPS 2 ビジネスの成果をサポートするために、組織をどのように構築すればよいですか?

チームはビジネスの成果を達成するうえでの役割を理解する必要があります。チームは他のチームが成功するためのそれぞれの役割を理解し、自分たちのチームが成功するための他のチームの役割を理解し、目標を共有する必要があります。責任、所有権、意思決定方法、意思決定を行う権限を持つユーザーを理解することは、労力を集中的に投入し、チームの利点を最大化するのに役立ちます。

ベストプラクティス

- OPS02-BP01 リソースには特定の所有者が存在する
- OPS02-BP02 プロセスと手順には特定の所有者が存在する
- OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する
- OPS02-BP04 チームメンバーが自らの責任範囲を把握する
- OPS02-BP05 責任と所有権を特定するためのメカニズムが存在する
- OPS02-BP06 追加、変更、例外をリクエストするメカニズムが存在する
- OPS02-BP07 チーム間の責任は事前定義済みまたは交渉済みである

OPS02-BP01 リソースには特定の所有者が存在する

各アプリケーション、ワークロード、プラットフォーム、インフラストラクチャコンポーネントの所有権を持つユーザーが誰か、そのコンポーネントによって提供されるビジネス価値、およびその所有

権が存在する理由を理解します。これらの個々のコンポーネントのビジネス価値、およびそれらがビジネスの成果をどのようにサポートするかを理解すると、それらに適用されるプロセスと手順がわかります。

このベストプラクティスを活用するメリット: 所有権を理解すると、改善の承認者、改善を実装する者、またはその両方がわかります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- リソースには特定の所有者が存在する: 環境内のリソースユースケースにおける所有権の意味を定義します。最小限の名前、連絡先情報、組織、チームなどでリソースの所有者を指定し、記録します。タグやリソースグループなどのメタデータを使用して、リソース所有権情報をリソースに保存します。AWS Organizations を使用してアカウントを構築し、ポリシーを実装して、所有権と連絡先情報が確実にキャプチャされるようにします。
  - 所有権の形態と割り当て方法の定義: 所有権は、異なるユースケースにおいて、組織内で複数の定義を持つ場合があります。ワークロード所有者は、ワークロード運用のリスクと責任を所有し、最終的にワークロードに関する決定を行う権限を持つ個人として定義できます。所有権が親組織にロールアップされる場合における財務責任または管理責任の観点から、所有権を定義することができます。開発者は開発環境の所有者であり、運用によって発生するインシデントに責任を負う者とすることができます。製品リーダーは、開発環境の運用に関連する財務コストについて責任を負う者とすることができます。
  - 組織、アカウント、リソースのコレクション、または個々のコンポーネントの所有者の定義: 検出をサポートするために整理され、適切にアクセス可能な状態となっている場所に所有権を定義し、記録します。変更に伴って、定義と所有権の詳細を更新します。
  - リソースのメタデータの所有権のキャプチャ: タグやリソースグループなどのメタデータを使用してリソースの所有権をキャプチャし、所有権と連絡先情報を指定します。AWS Organizations を使用してアカウントを構築し、所有権と連絡先情報がキャプチャされるようにします。

OPS02-BP02 プロセスと手順には特定の所有者が存在する

個々のプロセスと手順の定義を誰が所有しているか、特定のプロセスと手順が使用されている理由、およびその所有権が存在する理由を理解します。特定のプロセスと手順が使用される理由を理解することで、改善の機会を特定できます。

このベストプラクティスを確立するメリット: 所有権を理解すると、改善の承認者、改善を実装する者、またはその両方がわかります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- プロセスや手順には、その定義に責任を持つ所有者が存在する: お客様の環境で使用されているプロセスや手順、およびその定義に責任を持つ個人またはチームを把握します。
  - プロセスと手順の特定: ワークロードのサポートにおいて実施される運用アクティビティを特定します。これらのアクティビティを検出可能な場所に文書化します。
  - プロセスまたは手順の定義の所有者の定義: アクティビティの仕様に責任を有する個人またはチームを一意に特定します。当該個人またはチームは、適切なアクセス許可、アクセス、およびツールを持つ適切なスキルを持つチームメンバーが正常に実行できるようにする責任があります。そのアクティビティの実行に問題がある場合、アクティビティの改善に必要となる詳細なフィードバックを提供する責任はそのチームメンバーにあります。
  - アクティビティアーティファクトのメタデータの所有権のキャプチャ: AWS Systems Manager (ドキュメント)、および AWS Lambda (関数) などのサービスで自動化されている手続きは、メタデータ情報をタグとしてキャプチャするのをサポートしています。タグまたはリソースグループを使用してリソースの所有権をキャプチャし、所有権と連絡先情報を指定します。AWS Organizations を使用してタグ付けポリシーを作成し、所有権と連絡先情報がキャプチャされるようにします。

OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する

定義されたワークロードに対して特定のアクティビティを実行する責任を持つ者と、その責任が存在する理由を理解します。運用アクティビティを実行することに責任を負う者を理解すると、誰がアクティビティを実行し、結果を検証し、アクティビティの所有者にフィードバックを提供するかを通知します。

このベストプラクティスを活用するメリット: アクティビティを実行する責任を負う者を理解すると、アクションが必要になったときに誰に通知し、誰がアクションを実行し、結果を検証し、アクティビティの所有者にフィードバックを提供するかがわかります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 運用アクティビティのパフォーマンスに責任を持つ所有者の存在: 環境で使用されるプロセスと手順を実行する責任を明確にする

- プロセスと手順の特定: ワークロードのサポートにおいて実施される運用アクティビティを特定します。これらのアクティビティを検出可能な場所に文書化します。
- 各アクティビティを実行する責任者の定義: アクティビティに責任を有するチームを特定します。アクティビティの詳細と、アクティビティを実行するために必要なスキルと適切なアクセス許可、アクセス、ツールがあることを確認します。チームは、当該アクティビティが実行される条件 (イベントやスケジュールなど) を理解する必要があります。この情報を検出可能にして、組織のメンバーが、特定の二ーズに合わせて連絡する必要があるチームまたは個人を特定できるようにします。

OPS02-BP04 チームメンバーが自らの責任範囲を把握する

役割の責任と、ビジネスの成果にどのように貢献するかを理解することで、タスクの優先順位付けと役割が重要である理由を知ることができます。これにより、チームメンバーは二ーズを認識し、適切に対応できます。

このベストプラクティスを活用するメリット: 責任を理解すると、決定する事項、実行するアクション、および適切な所有者に引き渡すべきアクティビティがわかります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- チームメンバーに自らの役割と責任を確実に理解させる: チームメンバーの役割と責任を特定し、そのチームメンバーが自らの役割に期待されている事項を理解できるようにします。この情報を検出可能にして、組織のメンバーが、特定の二ーズに合わせて連絡する必要があるチームまたは個人を特定できるようにします。

OPS02-BP05 責任と所有権を特定するためのメカ二ズムが存在する

個人またはチームが特定されない場合、対処される必要がある事項についての所有権または計画を割り当てる権限を持つ者へのエスカレーションパスが定義されています。

このベストプラクティスを活用するメリット: 責任者または所有者を理解すると、適切なチームまたはチームメンバーに連絡して、リクエストし、またはタスクを移行することができます。責任や所有権を割り当て、または二ーズに対処する計画を立てる権限を持つ個人を特定することで、不作為のリスクや対応されないままとなるリスクを軽減できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 責任と所有権を特定するためのメカニズムの存在: 組織のメンバーが所有権と責任を知り、特定するために、メンバーがアクセス可能なメカニズムを提供します。これらのメカニズムにより、メンバーは、特定のニーズについて、連絡すべきチームまたは個人を特定できます。

OPS02-BP06 追加、変更、例外をリクエストするメカニズムが存在する

プロセス、手順、およびリソースの所有者にリクエストを送信できます。利点とリスクを評価した後、実行可能で適切であると判断されたリクエストを、十分な情報に基づいて承認します。

このベストプラクティスを確立するメリット: チームの活動のサポートに対する追加、変更、例外をリクエストするメカニズムを持つことは重要です。このオプションがなければ、現在の状態はイノベーションの制約になります。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 追加、変更、例外をリクエストするメカニズムの存在: 規格が硬直化すると、イノベーションが制約されます。組織のメンバーのために、ビジネスニーズをサポートするプロセス、手順、およびリソースの所有者にリクエストを行うメカニズムを提供します。

OPS02-BP07 チーム間の責任は事前定義済みまたは交渉済みである

チーム間には、チームがどのように連携し、互いにサポートするかを説明する、定義済みまたは交渉済みの契約があります (応答時間、サービスレベル目標、サービスレベルアグリーメントなど)。チームの仕事がビジネスの成果に及ぼす影響、および他のチームや組織の成果を理解することで、タスクの優先順位付けを知り、適切に対応できるようになります。

責任と所有権が未定義または不明な場合、必要な活動をタイムリーに処理できず、これらのニーズへの対応が重複し、競合する可能性のある作業が生じるリスクがあります。

このベストプラクティスを活用するメリット: チーム間の責任、目的、およびニーズを伝達する方法を確立することで、リクエストの流れがスムーズになり、必要な情報が確実に提供されます。これにより、チーム間の移行タスクによって生じる遅延が軽減され、ビジネス成果の達成をサポートします。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- チーム間の責任は事前定義済みまたは交渉済み: チームがやり取りする方法と、互いにサポートするために必要な情報を指定することで、リクエストが反復的にレビューおよび明確化されることに伴う遅延を最小限に抑えることができます。期待される事項 (応答時間や履行時間など) を定義する具体的な契約があることで、チームは効果的な計画とリソースを適切に作成することができます。

OPS 3 組織の文化はビジネスの成果をどのようにサポートしますか?

チームメンバーにサポートを提供することで、チームメンバーがより効果的に行動し、ビジネスの成果をサポートできるようにします。

ベストプラクティス

- OPS03-BP01 エグゼクティブスポンサーシップ
- OPS03-BP02 チームメンバーに、結果にリスクがあるときにアクションを実行する権限が付与されている
- OPS03-BP03 エスカレーションが推奨されている
- OPS03-BP04 タイムリーで明確、かつ実用的なコミュニケーション
- OPS03-BP05 実験の推奨
- OPS03-BP06 チームメンバーがスキルセットを維持、強化することができ、それが推奨されている
- OPS03-BP07 チームに適正なリソースを提供する
- OPS03-BP08 チーム内やチーム間でさまざまな意見が推奨され、求められる

OPS03-BP01 エグゼクティブスポンサーシップ

シニアリーダーシップは、組織に対する期待を明確に設定し、成功を評価します。シニアリーダーシップは、ベストプラクティスの採用と組織の進化の協賛者、支持者、および推進者です。

このベストプラクティスを確立するメリット: 熱心なリーダーシップ、期待事項の明確な伝達、目標の共有により、チームメンバーは、何が期待されているのかを知ることができます。成功を評価することで、成功に至るまでの障壁を特定でき、支持者や権限を与えられた者の介入を通じて対処できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- エグゼクティブスポンサーシップ: シニアリーダーシップは、組織に対する期待値を明確に設定し、成功を評価します。シニアリーダーシップは、ベストプラクティスの採用と組織の進化の協賛者、支持者、および推進者です。
  - 期待値の設定: 測定方法を含め、組織の目標を定義し、公開します。
  - 目標の達成の追跡: 目標の段階的な達成度を定期的に測定し、結果を共有して、結果にリスクがある場合に適切なアクションが講じられるようにします。
  - 目標達成に必要なリソースの提供: 新しい情報、目標、責任、ビジネス環境の変化などに基づいて、リソースが適切かどうか、また追加のリソースが必要であるかどうかを定期的に見直します。
  - チームの支援: チームと常に関わり、彼らがどのような状態にあるのか、また彼らに影響を与える外的要因があるのかを理解します。チームが外部要因の影響を受けた場合、目標を再評価し、必要に応じてターゲットを調整します。チームの進行を妨げている障害を特定します。チームのために障害に対処し、不要な負担を取り除きます。
  - ベストプラクティスの導入の促進: 定量的な利益をもたらしたベストプラクティスを認定し、その考案者と採用者を評価します。さらなる導入を推奨して、得られるメリットを拡大します。
  - チームの進化の推進: 継続的な改善の文化を生み出します。個人と組織の両者の成長と発展を奨励します。時間の経過とともに段階的な達成を求める長期的なターゲットを提供します。ニーズ、ビジネス目標、ビジネス環境の変化に応じて、これらを補完するために、このビジョンを調整します。

OPS03-BP02 チームメンバーに、結果にリスクがあるときにアクションを実行する権限が付与されている

ワークロード所有者は、結果にリスクがあるときにチームメンバーが対応できるようにするためのガイダンスと範囲を定義しています。エスカレーションメカニズムは、イベントが定義された範囲外にある場合に対応の方向性を取得するために使用されます。

このベストプラクティスを活用するメリット: 変更を早期にテストして検証することで、コストを最小限に抑えて問題に対処し、顧客への影響を抑えることができます。デプロイ前にテストすることで、エラーの発生を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 結果にリスクがあるときにアクションを実行する権限が、チームメンバーに付与されている: 効果的に対応するために必要なスキルを実践するためのアクセス許可、ツール、機会をチームメンバーに提供します。
  - 対応するために必要なスキルを訓練する機会をチームメンバーに提供する: プロセスと手順のテストとトレーニングを安全に実行できる、安全な代替環境を提供します。ゲームデーを実施して、チームメンバーがシミュレートされた安全な環境で現実世界のインシデントに対応する経験を積むことができるようにします。
  - アクションを実行するチームメンバーの権限を定義および認識する: チームメンバーがサポートするワークロードとコンポーネントにアクセス許可とアクセス権を割り当てることで、アクションを実行するチームメンバーの権限を具体的に定義します。チームメンバーが、結果にリスクがあるときにアクションを実行する権限を付与されていることを認識します。

OPS03-BP03 エスカレーションが推奨されている

チームメンバーにはメカニズムがあり、結果にリスクがあると思われる場合は、意思決定者や利害関係者に懸念をエスカレートすることが推奨されます。エスカレーションは、リスクを特定し、インシデントの発生を防ぐために、早く、頻繁に実行する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 早期かつ頻繁なエスカレーションを奨励する: 早期かつ頻繁なエスカレーションがベストプラクティスであることを組織的に認識します。エスカレーションに理由がないとされる場合があること、そして、エスカレーションしないことによって、インシデントを阻止する機会を逃すよりも、インシデントを阻止する機会が得られる方がよいことを組織的に認識し、受け入れます。
  - エスカレーションメカニズムの存在: エスカレーションをいつどのように行うかを定義する手順を文書化します。アクションを実行したり、アクションを承認したりする強い権限を持つ人々とその連絡先情報を文書化します。リスクに対処できる人物に当該リスクが引き渡されたとチームメンバーが満足するまで、またはワークロードの運用に関するリスクと責任の所有者に連絡するまで、エスカレーションを続行する必要があります。当該者がワークロードに関するすべての決定を最終的に所有します。エスカレーションには、リスクの性質、ワークロードの重要度、影響を受ける者、影響の内容、緊急性 (予想される影響の時期など) を含める必要があります。

- エスカレーションする従業員の保護: 無策の意思決定や利害関係者などにエスカレーションする場合にチームメンバーを報復から保護するポリシーを備えます。これが発生しているかどうかを特定し、適切に対応するメカニズムを備えます。

OPS03-BP04 タイムリーで明確、かつ実用的なコミュニケーション

メカニズムが存在し、 チームメンバーに既知のリスクや計画されたイベントをタイムリーに通知するために使用されます。アクションが必要かどうか、どのようなアクションが必要かを判断し、タイミングよくそのアクションを実行するために、必要なコンテキスト、詳細、および時間 (可能な場合) が提供されます。たとえば、パッチ適用を迅速に行えるようにソフトウェアの脆弱性を通知したり、サービス中断のリスクを回避するために変更のフリーズを実装できるように計画された販売プロモーションの通知を提供したりします。

計画されたイベントは変更カレンダーまたはメンテナンススケジュールに記録できるため、チームメンバーが保留中のアクティビティを特定できます。

AWS では、 AWS Systems Manager 変更カレンダー を使用してこれらの詳細を記録できます。カレンダーのステータスのプログラムによるチェックをサポートし、特定の時点でカレンダーがアクティビティに対して開いているか閉じているかを判断します。運用アクティビティは、障害となり得るアクティビティのために確保された 特定の「承認済み」の 時間枠を中心に計画できます。AWS Systems Manager メンテナンスウィンドウは、アクティビティを自動化し、それらのアクティビティを検出可能にするために、 インスタンスやその他のサポートされているリソースに対し アクティビティをスケジュールできるようにします。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- タイムリーで明確、かつ実用的なコミュニケーション: リスクや計画されたイベントの通知を明確かつ実用的な方法で提供し、適切な対応を可能にするのに十分な通知を提供するメカニズムが設けられています。
  - 計画されたアクティビティを変更カレンダーに記録し、通知する: 計画されたイベントを知ることができる、アクセス可能な情報ソースを提供します。同じシステムから計画されたイベントを通知します。
  - ワークロードに影響を与え得るイベントとアクティビティを追跡する: 脆弱性の通知とパッチ情報をモニタリングして、ワークロードコンポーネントに関連する野放し状態の脆弱性と潜在的なリスクを理解します。チームメンバーがアクションを実行できるように通知を送信します。

リソース

関連するドキュメント:

- AWS Systems Manager 変更カレンダー
- AWS Systems Manager メンテナンスウィンドウ

OPS03-BP05 実験の推奨

実験は、学習を加速し、チームメンバーが関心と当事者意識を持ち続けることの一助となります。望ましくない結果は、成功につながらないパスを特定することに成功した実験です。望ましくない結果が得られた実験が成功してもチームメンバーが罰せられることはありません。イノベーションを起こし、アイデアを成果に変えるには、実験が必要です。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 実験の推奨: 学習とイノベーションをサポートする実験を奨励します。
  - さまざまなテクノロジーを試す: ビジネス上の成果を達成するために、現在または将来的に適用可能なテクノロジーを用いた実験を奨励します。この知識は、将来のイノベーションに役立つ可能性があります。
  - 目標を念頭に置いて実験する: チームメンバーが到達すべき具体的な目標や、近い将来に適用可能となり得るテクノロジーを使って実験することを奨励します。この知識はイノベーションに役立つ可能性があります。
  - 実験のための時間を計画的に確保する: チームメンバーが実験に集中できるよう、通常の業務から解放される時間を確保します。
  - 実験をサポートするためのリソースを提供する: 実験に必要なリソース (ソフトウェアやクラウドリソースなど) に予算を割り当てます。
  - 成功を認める: 実験によって得られた価値を評価します。望ましくない結果を得られた実験は、成功につながらないパスを特定できた成功した実験であることを理解します。チームメンバーは、実験から望ましくない結果を得たことについて罰せられることはありません。

OPS03-BP06 チームメンバーがスキルセットを維持、強化することができ、それが推奨されている

チームは、ワークロードに対応するに際して、新しいテクノロジーを採用し、需要と責任の変化をサポートするために、スキルセットを強化する必要があります。新しいテクノロジーにおけるスキルの

発達は、多くの場合、チームメンバーの満足度の源となり、イノベーションをサポートします。チームメンバーが強化している自らのスキルを検証および認識し、業界認証を追求および維持できるように支援します。組織の知識とスキルを持ち、熟練したチームメンバーを失った場合は、クロストレーニングによって知識の伝達を促進し、重大な影響のリスクを緩和します。学習のために専用の時間を割り当てます。

AWS は、 AWS ご利用開始のためのリソースセンター、 AWS ブログ、 AWS オンラインテックトーク、 AWS イベントとオンラインセミナー、 AWS Well-Architected ラボなどのリソースを提供しており、チームを教育するためのガイダンス、事例、詳細なチュートリアルを提供しています。

また、AWS では、 Amazon Builders' Library で AWS の運用を通じて学んだベストプラクティスとパターン、 AWS ブログ や 公式 AWS ポッドキャストを通じて幅広い有益な教材を共有しています。.

AWS が提供する Well-Architected ラボ、 AWS Support (AWS ナレッジセンター、 AWS ディスカッションフォーラム、および AWS Support センター) および AWS ドキュメント などのリソースを活用して、チームの教育に役立ててください。AWS に関しての質問については、AWS Support センターを利用して AWS Support に連絡してください。

AWS トレーニング と認定 では、AWS の基礎に関するセルフペースデジタルコースを通じて無料のトレーニングを提供しています。また、インストラクターが実施するトレーニングに登録して、チームの AWS スキルの開発をさらにサポートすることもできます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- チームメンバーはスキルセットの維持と成長が可能で推奨されている: 新しいテクノロジーを採用し、イノベーションをサポートし、ワークロードのサポートにおける需要と責任の変化に対応するためには、継続的な教育が必要です。
  - 教育のためのリソースを提供する: 構造的に設けられた専用の時間、トレーニング資料へのアクセス、ラボリソース、カンファレンスや専門家組織への参加のサポートにより、教育者と同僚の両方から学習する機会を得ることができます。下級チームのメンバーが、上級チームのメンバーをメンターとするためにアクセスできるようにしたり、上級チームのメンバーの業務をシャドーイングして、自らの手法やスキルを評価してもらえるようにしたりします。より広い視点を持つために、仕事に直接関係しないコンテンツについて学習することを奨励します。
  - チーム教育とチーム間のエンゲージメント: チームメンバーの継続的な教育のニーズに合った計画を立てます。チームメンバーが他のチームに (一時的または永続的に) 参加し、組織全体に役立つスキルやベストプラクティスを共有する機会を提供する

- 業界認証の追求と維持をサポートする: チームメンバーが学んだことを検証し、その成果を認める業界認証を取得および維持するのをサポートします。

リソース

関連するドキュメント:

- AWS ご利用開始のためのリソースセンター
- AWS ブログ
- AWS クラウド コンプライアンス
- AWS ディスカッションフォーラム
- AWS ドキュメント
- AWS オンラインテックトーク
- AWS イベントとオンラインセミナー
- AWS ナレッジセンター
- AWS Support
- AWS トレーニング と認定
- AWS Well-Architected ラボ、
- Amazon Builders' Library
- 公式 AWS ポッドキャストを通じて幅広い有益な教材を共有しています。.

OPS03-BP07 チームに適正なリソースを提供する

チームメンバーのキャパシティを維持し、ワークロードのニーズを満たすツールとリソースを提供します。チームメンバーに過剰な負荷がかかることは、人為的ミスに起因するインシデントのリスクを高めます。ツールやリソースへの投資 (頻繁に実行されるアクティビティのオートメーションなど) によって、チームの有効性を高め、より多くの活動をサポートすることを可能にします。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- チームに適正なリソースを提供する: チームの成功、および成功または失敗の要因を確実に把握します。チームに適正なリソースを提供してサポートします。
  - チームのパフォーマンスを理解する: チームによる業務成果の達成度や アセット開発の成果を測定します。時間の経過とともに成果と エラー率の変化を追跡します。チームと協力して、チーム

に影響する業務に関する課題 (責任の増加、テクノロジーの変化、人員の喪失、サポート対象の顧客の増加など) を理解します。

- チームのパフォーマンスへの影響を理解する: チームと常に関わり、彼らがどのような状態にあるのか、また彼らに影響を与える外的要因があるのかを理解します。チームが外部要因の影響を受けた場合、目標を再評価し、必要に応じてターゲットを調整します。チームの進行を妨げている障害を特定します。チームのために障害に対処し、不要な負担を取り除きます。
- チームの成功に必要なリソースを提供する: リソースが適切かどうか、追加リソースが必要かどうかを定期的に検証し、チームをサポートするために適切な調整を行います。

OPS03-BP08 チーム内やチーム間でさまざまな意見が推奨され、求められる

組織間の多様性を活用して、複数のユニークな視点を追求します。この視点を使用して、イノベーションを高め、想定に挑み、確証バイアスに傾くリスクを軽減します。チーム内のインクルージョン、多様性、アクセシビリティを向上させ、有益な視点を得ます。

組織文化は、チームメンバーのジョブに対する満足度と定着率に直接影響します。チームメンバーのやる気と能力を引き出して、ビジネスの成功につなげます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 多様な意見や視点を求める: すべてのメンバーからの貢献を求めます。立場の弱いグループの意見に耳を傾けます。ミーティングでは、役割と責任の割り当てを定期的に変更します。
  - 役割と責任を拡張する: チームメンバーが通常引き受けることのないであろう役割を引き受ける機会を提供します。その役割から、そして、通常はやり取りしない新しいチームメンバーとのやり取りから、経験や視点を得ることができます。また自分の経験と視点を、やり取りをする新しい役割やチームメンバーにもたらします。視点が増えるにつれて、追加のビジネスの機会が得られたり、改善のための新しい機会が見出されたりすることがあります。チーム内のメンバーに、他のメンバーが通常実行する一般的なタスクを交替で担当してもらい、当該タスクの実行の需要と影響を理解してもらいます。
- 安全で温かい環境を提供する: 組織内のチームメンバーの精神的および物理的な安全を確保するポリシーと統制を備えます。チームメンバーは、報復を恐れずにやり取りできる必要があります。チームメンバーが安心し、温かい気持ちになると、当事者意識が高まり、生産性が向上する可能性が高くなります。組織がより多様になるほど、顧客を含め、サポートする人々に対する理解が深まります。チームのメンバーが快適で、自由に話し、自分の話を聞いてもらえることを確信すると、自らの貴重な洞察 (マーケティングの機会、アクセシビリティのニーズ、未開拓の市場セグメント、環境内の認識されていないリスクなど) を共有する可能性が高まります。

- チームメンバーが完全に参加できるようにする: 従業員がすべての業務関連活動に完全に参加するために必要なリソースを提供します。日々の課題に直面するチームメンバーは、それを回避するためのスキルを身に付けています。これらの独自に開発したスキルは、組織に大きなメリットをもたらします。チームメンバーに必要な配慮をしてサポートすることで、貢献から得られるメリットが大きくなります。

準備

質問

- OPS 4 ワークロードの状態を理解するために、ワークロードをどのように設計すればよいですか?
- OPS 5 欠陥を減らし、修正を簡単にし、本番環境へのフローを改善するにはどうすればよいですか?
- OPS 6 デプロイのリスクを軽減するにはどうすればよいですか?
- OPS 7 ワークロードをサポートする準備が整っていることを確認するにはどうすればよいですか?

OPS 4 ワークロードの状態を理解するために、ワークロードをどのように設計すればよいですか?

ワークロードを設計する際には、すべてのコンポーネント (メトリクス、ログ、トレースなど) にわたって内部状態を理解するために必要な情報が送出されるようにします。そうすることによって、適時に有用な返答を提供できるようになります。

ベストプラクティス

- OPS04-BP01 アプリケーションテレメトリーを実装する
- OPS04-BP02 ワークロードテレメトリーを実装して設定する
- OPS04-BP03 ユーザーアクティビティテレメトリーを実装する
- OPS04-BP04 依存関係のテレメトリーを実装する
- OPS04-BP05 トランザクショントレーサビリティを実装する

OPS04-BP01 アプリケーションテレメトリーを実装する

アプリケーションテレメトリーは、ワークロードの可観測性の基盤です。アプリケーションは、アプリケーションの状態やビジネス成果の達成に関するインサイトを提供するテレメトリーを送信する必要があります。トラブルシューティングから新しい機能の効果測定に至るまで、アプリケーションテ

レメトリーを使用することで、ワークロードの構築、運用、展開方法に関する情報を得ることができます。

アプリケーションテレメトリーは、メトリクスとログで構成されます。メトリクスは、脈や体温などの診断情報を指します。複数のメトリクスを使用することで、アプリケーションの状態を知ることができます。長期間にわたるメトリクスの収集は、ベースラインの開発や異常の検知に役立ちます。ログは、アプリケーションの内部の状態や発生したイベントに関してアプリケーションが送信するメッセージです。記録されるイベントには、エラーコード、トランザクション識別子、ユーザーアクションなどが含まれます。

期待される成果:

- アプリケーションは、アプリケーションの状態とビジネス成果の達成に関するメトリクスとログを送信します。
- ワークロードのすべてのアプリケーションのメトリクスとログは、一元的に保存されます。

一般的なアンチパターン:

- アプリケーションはテレメトリーを送出しません。何か問題が生じたときは、顧客から通知してもらう以外に方法はありません。
- お客様から、アプリケーションが応答しないと報告されました。あなたはテレメトリーを備えておらず、現在のユーザーエクスペリエンスを理解するために自らアプリケーションを使用することなく、問題が存在することを確認したり、問題の特徴を把握したりすることができません。

このベストプラクティスを活用するメリット:

- アプリケーションの状態、ユーザーエクスペリエンス、ビジネス成果の達成を理解できます。
- アプリケーションの状態の変化にすばやく対応できます。
- アプリケーションの状態の傾向を知ることができます。
- アプリケーションの改善に関する情報に基づく意思決定を行えます。
- アプリケーションの問題をすばやく検知して解決できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

アプリケーションテレメトリーを 3 つ手順で実装する: テレメトリーを保存する場所の特定、アプリケーションの状態を示すテレメトリーの特定、アプリケーションへのテレメトリー送信機能の追加によって、アプリケーションテレメトリーを実装します。

例えば、eコマースの会社はアーキテクチャベースのマイクロサービスを持っています。アーキテクチャ設計プロセスの一環として、この会社は各マイクロサービスの状態を理解するのに役立つアプリケーションテレメトリーを特定しました。例えば、ユーザーのカートサービスは、カートへの追加、カートの削除、アイテムがカートに追加されるまでの時間などのイベントに関するテレメトリーを送信します。すべてのマイクロサービスは、エラー、警告、トランザクション情報を記録します。テレメトリーは Amazon CloudWatch に送信され、保存および分析されます。

実装手順

最初の手順は、ワークロード内のアプリケーションテレメトリーを一元保存する場所を特定することです。既存のプラットフォームがない場合、 Amazon CloudWatch はテレメトリー収集、ダッシュボード、分析、およびイベント生成機能を提供します。

必要なテレメトリーを特定するには、まず以下の質問から始めます。

- アプリケーションの状態は正常か。
- アプリケーションはビジネス成果を達成しているか。

  アプリケーションは、これらの質問に対して回答を提示するログやテレメトリーを送信する必要があります。既存のアプリケーションテレメトリーがこれらの質問に答えられない場合、ビジネスおよびエンジニアリングのステークホルダーと協力して、これらの質問に答えるテレメトリーの一覧を作成します。新しいアプリケーションテレメトリーの特定と開発に関しては、AWS アカウントチームに技術的なアドバイスを求めることができます。

  新しいアプリケーションテレメトリーを特定できたら、エンジニアリングのステークホルダーと協力して、アプリケーションにテレメトリーの送信機能を追加します。 AWS Distro for OpenTelemetryは、 アプリケーションテレメトリーを収集する API、ライブラリ、エージェントを提供します。 この例は、カスタムメトリクスを使用して JavaScript アプリケーションの状態を計測する方法を示します。

  AWS が提供する可観測性サービスを知りたいお客様は、ご自身で 1 つの可観測性ワークショップを実施したり、AWS アカウント チームにガイダンスとサポートを依頼したりすることができます。このワークショップでは、AWS の可観測性ソリューションの概要を学び、使用例を実際に体験することができます。

アプリケーションテレメトリーの詳細については、 Amazon Builder's Library の 「分散システムを装備して、運用の可視性を高める」の記事をご覧ください。この記事では、Amazon によるアプリケーションの計測方法、およびお客様の計測ガイドラインの開発に Amazon がどのように役立つかを紹介しています。

計画の実装に必要な工数レベル: ミディアム

リソース

関連するベストプラクティス:

the section called "OPS04-BP02 ワークロードテレメトリーを実装して設定する" - アプリケーションテレメトリーはワークロードテレメトリーのコンポーネントの 1 つです。ワークロード全体の状態を理解するには、ワークロードを構成する個別のアプリケーションの状態を理解する必要があります。

the section called "OPS04-BP03 ユーザーアクティビティテレメトリーを実装する" - ユーザーアクティビティテレメトリーは、アプリケーションテレメトリーのサブセットになることがあります。カートへの追加イベント、ストリームのクリック、トランザクションの完了などのユーザーアクティビティは、ユーザーエクスペリエンスに関するインサイトを提供します。

the section called "OPS04-BP04 依存関係のテレメトリーを実装する" - 依存関係チェックはアプリケーションテレメトリーに関連しているため、アプリケーションに追加することができます。アプリケーションが DNS やデータベースなどの外部システムに依存している場合、アプリケーションはアクセス性、タイムアウト、および他のイベントに関するメトリクスとログを送信できます。

the section called "OPS04-BP05 トランザクショントレーサビリティを実装する" - ワークロード全体のトランザクションを追跡するには、各アプリケーションが共有したイベントをどのように処理したかについての情報を送信する必要があります。各アプリケーションがこれらのイベントを処理した方法は、アプリケーションテレメトリーを介して送信されます。

the section called "OPS08-BP02 ワークロードのメトリクスを定義する" - ワークロードメトリクスは、ワークロードの状態を示す重要なインジケーターです。主要なアプリケーションメトリクスは、ワークロードメトリクスの一部です。

関連するドキュメント:

- AWS Builders' Library - 分散システムを装備して、運用の可視性を高める
- AWS Distro for OpenTelemetry

- AWS Well-Architected 運用上の優秀性の柱のホワイトペーパー - テレメトリーの設計
- フィルターを使用したログイベントからのメトリクスの作成
- Amazon CloudWatch を使用したログ記録とモニタリングの実装
- AWS Distro for OpenTelemetry を使用したアプリケーションの状態とパフォーマンスのモニタリング
- 新規 - Amazon CloudWatch エージェントを使用してカスタムアプリケーションメトリクスをより効果的にモニタリングする方法
- AWS における可観測性
- シナリオ - CloudWatch へのメトリクスの公開
- 構築の開始 - アプリケーションを効率的にモニタリングする方法
- AWS SDK での CloudWatch の使用

関連動画:

- AWS re:Invent 2021 - オープンソースの可観測性
- CloudWatch エージェントを使用して Amazon EC2 インスタンスからメトリクスとログを収集する
- AWS ワークロードのアプリケーションモニタリングを簡単に設定する方法 - AWS オンラインテックトーク
- サーバーレスアプリケーションの可観測性をマスターする - AWS オンラインテックトーク
- AWS におけるオープンソースの可観測性 - AWS 仮想ワークショップ

関連する例:

- AWS のログ記録およびモニタリングの例に関するリソース
- AWS ソリューション: Amazon CloudWatch モニタリングフレームワーク
- AWS ソリューション: 集中ロギング
- 1 つの可観測性ワークショップ

OPS04-BP02 ワークロードテレメトリーを実装して設定する

内部状態や現在のステータスに関する情報 (API 呼び出しのボリューム、HTTP ステータスコード、スケーリングイベントなど) が送出されるよう、ワークロードを設計および設定します。この情報を使用して、応答が必要とされるタイミングを特定します。

Amazon CloudWatch などの サービスを を使用して、ワークロードコンポーネントからのログとメトリクスを集計します (例: API ログ、 AWS CloudTrail、 AWS Lambda メトリクス、 Amazon VPC フローログ、および その他のサービス)。

一般的なアンチパターン:

- 顧客が、パフォーマンスが低いことについて苦情を申し立てています。アプリケーションに最近の変更はないため、あなたは、ワークロードコンポーネントに問題があると考えています。あなたは、パフォーマンスの低さに影響しているコンポーネントを特定するために分析を行うテレメトリーを備えていません。
- あなたは、アプリケーションにアクセスできません。あなたには、ネットワーキングの問題であるかどうかを判断するためのテレメトリーがありません。

このベストプラクティスを確立するメリット: ワークロード内で何が起こっているかを理解することで、必要に応じて対応できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- ログとメトリクステレメトリーを実装する: 内部状態、ステータス、およびビジネス成果の達成に関する情報が送出されるよう、ワークロードを計測します。この情報を使用して、応答が必要とされるタイミングを特定します。
  - Gaining better observability of your VMs with Amazon CloudWatch - AWS Online Tech Talks (Amazon CloudWatch で VM の可観測性を改善する - AWS オンラインテックトーク)
  - Amazon CloudWatch の仕組み
  - Amazon CloudWatch とは
  - Amazon CloudWatch メトリクスを使用する
  - Amazon CloudWatch Logs とは
    - ワークロードテレメトリーを実装して設定する: 内部状態や現在のステータスに関する情報 (API 呼び出しのボリューム、HTTP ステータスコード、スケーリングイベントなど) が送出されるよう、ワークロードを設計および設定します。
      - Amazon CloudWatch metrics and dimensions reference (Amazon CloudWatch のメトリクスとディメンションのリファレンス)
      - AWS CloudTrail
      - AWS CloudTrail とは

- VPC フローログ

リソース

関連するドキュメント:

- AWS CloudTrail
- Amazon CloudWatch ドキュメント
- Amazon CloudWatch metrics and dimensions reference (Amazon CloudWatch のメトリクスとディメンションのリファレンス)
- Amazon CloudWatch の仕組み
- Amazon CloudWatch メトリクスを使用する
- VPC フローログ
- AWS CloudTrail とは
- Amazon CloudWatch Logs とは
- Amazon CloudWatch とは

関連動画:

- Application Performance Management on AWS (AWS でのアプリケーションのパフォーマンスメトリクス)
- Gaining Better Observability of Your VMs with Amazon CloudWatch (Amazon CloudWatchで VM の可観測性を改善する)
- Gaining better observability of your VMs with Amazon CloudWatch - AWS Online Tech Talks (Amazon CloudWatch で VM の可観測性を改善する - AWS オンラインテックトーク)

OPS04-BP03 ユーザーアクティビティテレメトリーを実装する

ユーザーアクティビティに関する情報 (クリックストリームのほか、開始、放棄、完了済みトランザクションなど) が送出されるよう、アプリケーションコードをインストルメント化します。この情報を使用して、アプリケーションの使用方法や使用パターンを理解したり、応答が必要とされるタイミングを特定したりできます。

一般的なアンチパターン:

- 開発者は、ユーザーのテレメトリーなしで新しい機能をデプロイし、使用率が増加しました。あなたは、使用率の増加が新しい機能の使用によるものなのか、あるいは新しいコードで発生した問題なのかを判別できません。
- 開発者は、ユーザーのテレメトリーなしで新しい機能をデプロイしました。あなたは、顧客に問い合わせて尋ねなければ、顧客が当該機能を使用しているかどうかがわかりません。

このベストプラクティスを活用するメリット: 顧客がアプリケーションを使用して使用パターンや予期しない動作を特定し、必要に応じてお客様が対応することを可能にする方法を理解します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ユーザーアクティビティテレメトリを実装する: ユーザーアクティビティに関する情報 (クリックストリームのほか、開始、放棄、完了済みトランザクションなど) が送出されるよう、アプリケーションコードを設計します。この情報を使用して、アプリケーションの使用方法や使用パターンを理解したり、応答が必要とされるタイミングを特定したりできます。

OPS04-BP04 依存関係のテレメトリーを実装する

依存するリソースの状態 (到達可能性や応答時間など) に関する情報を出力するようにワークロードを設計および設定します。外部依存関係の例としては、外部データベース、DNS、ネットワーク接続などがあります。この情報を使用して、応答が必要とされるタイミングを特定します。

一般的なアンチパターン:

- アプリケーションにアクセスできない理由が DNS の問題であるかどうかを判断するには、手動で DNS プロバイダーが動作しているかどうかを確認する必要があります。
- ショッピングカートアプリケーションはトランザクションを完了できません。クレジットカード処理プロバイダーの問題であるかどうかを確認するには、そのプロバイダーに連絡する必要があります。

このベストプラクティスを活用するメリット: 依存関係の状態を理解することで、必要に応じて対応できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 依存関係のテレメトリーを実装する: ワークロードが依存するシステムの状態やステータスに関する情報が送出されるよう、ワークロードを設計および設定します。たとえば、外部データベース、DNS、ネットワーク接続、外部クレジットカード処理サービスなどがあります。
  - AWS Systems Manager 統合での Amazon CloudWatch エージェント - Linux および Windows 向けの統合メトリクスおよびログ収集
  - CloudWatch エージェントを使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集

リソース

関連するドキュメント:

- AWS Systems Manager 統合での Amazon CloudWatch エージェント - Linux および Windows 向けの統合メトリクスおよびログ収集
- CloudWatch エージェントを使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集

関連する例:

- Well-Architected ラボ - 依存関係のモニタリング

OPS04-BP05 トランザクショントレーサビリティを実装する

ワークロード全体のトランザクションフローに関する情報が送出されるよう、アプリケーションコードを実装し、ワークロードコンポーネントを設定します。この情報を使用して、応答が必要とされるタイミングを特定し、問題につながる要素の特定に役立てます。

AWS では、次のような分散トレーシングサービスを使用できます。 AWS X-Rayでは、トランザクションがワークロードを通過するときにトレースを収集して記録したり、マップを生成してワークロードやサービス間でトランザクションがどのように流れるかを確認したり、コンポーネント間の関係を把握したり、さらにはリアルタイムで問題を特定して分析したりできます。

一般的なアンチパターン:

- あなたは、複数のアカウントにまたがるサーバーレスマイクロサービスアーキテクチャを実装しました。断続的なパフォーマンスの問題が顧客に発生しています。アプリケーション内でパフォーマ

ンスの問題が存在する場所と問題の原因の特定を可能にするトレースがないため、どの機能またはコンポーネントが問題を引き起こしているのかを特定できません。

- あなたは、開発を行う際に対処できるように、ワークロードのパフォーマンスのボトルネックがどこにあるかを特定しようとしています。あなたは、アプリケーションコンポーネントと、それらがやり取りするサービスとの関係を確認できず、アプリケーションのパフォーマンスに影響を及ぼす特定のサービスやパスのドリルダウンを可能にするトレースがないため、ボトルネックがどこにあるかを特定できません。

このベストプラクティスを確立するメリット: ワークロード全体のトランザクションフローを理解することで、ワークロードトランザクションの予想される動作と、ワークロード全体の予想される動作のバリエーションを理解できるため、必要に応じて対応できます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- トランザクショントレーサビリティを実装する: トランザクションステージ、アクティブコンポーネント、アクティビティ完了までの時間など、システムコンポーネント全体のトランザクションフローに関する情報を送出するよう、アプリケーションとワークロードを設計します。この情報を使用して、進行中のもの、完了しているもの、および完了したアクティビティの結果を特定できます。これは、応答が必要とされるタイミングを特定するのに役立ちます。例えば、コンポーネント内のトランザクション応答時間が予想より長い場合、そのコンポーネントに問題があることがわかります。
  - AWS X-Ray
  - AWS X-Ray とは

リソース

関連するドキュメント:

- AWS X-Ray
- AWS X-Ray とは

OPS 5 欠陥を減らし、修正を簡単にし、本番環境へのフローを改善するにはどうすればよいですか?

リファクタリング、品質についてのすばやいフィードバック、バグ修正を可能にし、本番環境への変更のフローを改善するアプローチを採用します。これらにより、本番環境に採用される有益な変更を加速させ、デプロイされた問題を制限できます。またデプロイアクティビティを通じて挿入された問題をすばやく特定し、修復できます。

ベストプラクティス

- OPS05-BP01 バージョン管理を使用する
- OPS05-BP02 変更をテストし、検証する
- OPS05-BP03 設定管理システムを使用する
- OPS05-BP04 構築およびデプロイ管理システムを使用する
- OPS05-BP05 パッチ管理を実行する
- OPS05-BP06 設計標準を共有する
- OPS05-BP07 コード品質の向上のためにプラクティスを実装する
- OPS05-BP08 複数の環境を使用する
- OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う
- OPS05-BP10 統合とデプロイを完全自動化する

OPS05-BP01 バージョン管理を使用する

変更とリリースの追跡を有効にするにはバージョン管理を使用します。

AWS の多くのサービスには、バージョン管理機能が備わっています。リビジョンまたはソース管理システム、例えば AWS CodeCommit コードや他のアーティファクト (インフラストラクチャのバージョン管理された AWS CloudFormation テンプレートなど) を管理します。

一般的なアンチパターン:

- あなたは、コードを開発し、ワークステーションに保存しました。ワークステーションで回復不可能なストレージ障害が発生し、コードが失われました。
- 既存のコードを変更で上書きした後、アプリケーションを再起動すると、操作できなくなりました。あなたは、変更を元に戻すことができません。

- あなたは、レポートファイルへの書き込みをロックされており、他の誰かが編集する必要があります。編集をしようとする者は、タスクを完了できるように、作業を停止するようにあなたに求めています。
- 研究チームは、今後の業務を形作る詳細な分析に取り組んでいます。誰かが誤って買い物リストを最終レポートに上書きして保存してしまいました。あなたは変更を元に戻すことができず、レポートを再作成する必要があります。

このベストプラクティスを確立するメリット: バージョン管理機能を使用すると、既知の良好な状態や以前のバージョンに簡単に戻すことができ、アセットが失われるリスクを低減できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- バージョン管理を使用する: バージョン管理されたレポジトリでアセットを維持します。そうすることで、変更の追跡、新しいバージョンのデプロイ、既存バージョンへの変更の検出、以前のバージョンの回復 (障害が発生する場合に、その前の良好な状態に戻すなど) をサポートします。構成管理システムのバージョン管理機能を手順に統合します。
  - AWS CodeCommit の紹介
  - AWS CodeCommit とは

リソース

関連するドキュメント:

- AWS CodeCommit とは

関連動画:

- AWS CodeCommit の紹介

OPS05-BP02 変更をテストし、検証する

エラーの制限と検出に役立てるため、変更をテスト、検証します。手動プロセスによって発生するエラーと、テストにかかる労力を減らすためにテストを自動化します。

AWS の多くのサービスには、バージョン管理機能が備わっています。AWS CodeCommit などのリビジョンまたはソース管理システムを 使用して、 コードや他のアーティファクト (インフラストラクチャのバージョン管理された AWS CloudFormation テンプレートなど) を管理します。

一般的なアンチパターン:

- あなたが新しいコードを本稼働環境にデプロイしたところ、アプリケーションが機能しなくなったため、顧客からの電話が鳴りはじめました。
- あなたは、ペリメータセキュリティを強化するために新しいセキュリティグループを適用します。機能しましたが、意図しない結果が発生し、ユーザーがアプリケーションにアクセスできなくなっています。
- あなたは、新しい機能によって呼び出されるメソッドを変更します。もう 1 つの機能もそのメソッドに依存しており、機能しなくなりました。問題は検出されず、本稼働環境に入ります。もう 1 つの当該機能はしばらくの間呼び出されず、結局、原因との相関なしに本稼働環境で失敗します。

このベストプラクティスを活用するメリット: 変更を早期にテストして検証することで、コストを最小限に抑えて問題に対処し、顧客への影響を抑えることができます。デプロイ前にテストすることで、エラーの発生を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 変更をテストし、検証する: すべてのライフサイクルステージ (開発、テスト、本番環境など) で変更をテストし、その結果を検証してください。テスト結果を使用して、新機能を確認し、失敗したデプロイのリスクと影響を緩和します。テストと検証を自動化し、レビューの一貫性を確保し、手動プロセスによって発生するエラーとそれにかかる労力を減らすことができます。
  - AWS CodeBuild とは
  - AWS CodeBuild のローカル構築サポート

リソース

関連するドキュメント:

- AWS デベロッパーツール
- AWS CodeBuild のローカル構築サポート

- AWS CodeBuild とは

OPS05-BP03 設定管理システムを使用する

設定を変更し、変更を追跡記録するには、構成管理システムを使用します。これらのシステムは、手動プロセスによって発生するエラーと、変更を導入する労力を減らします。

静的な構成管理では、ライフタイムを通じて一貫性を維持することが期待されるリソースの初期化時に値を設定します。このケースの例として、インスタンス上のアプリケーションサーバーまたはウェブサーバー用の構成を設定する場合や、 AWS Management Console 内または AWS CLI を介して AWS サービスの構成を定義する場合が挙げられます。

動的な構成管理では、ライフタイムを通じて変化する、または変化することが予測されるリソースの初期化時に値を設定します。例えば、構成変更を介してコードの機能を有効にするように機能トグルを設定したり、インシデント発生時にログの詳細レベルを変更してより多くのデータを取得し、インシデント終了時に詳細レベルを元に戻して不要なログや負荷を減らしたりすることができます。

インスタンス、コンテナ、サーバーレス機能、またはデバイスで実行されているアプリケーションで動的な構成を使用している場合、 AWS AppConfig を使用して 環境全体での管理と実装を行うことができます。

AWS では、 AWS Config を使用して アカウントおよびリージョン全体の AWS リソース構成を 継続的にモニタリングできます。そうすることで、構成履歴の追跡、構成変化の他のリソースへの影響、 AWS Config ルール および AWS Config コンフォーマンスパックを使用した期待される、または望まれる設定との比較監査を行えます。

AWS では、以下のサービスを使用して、継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを構築できます。 AWS デベロッパーツール (例: AWS CodeCommit、 AWS CodeBuild、 AWS CodePipeline、 AWS CodeDeploy、および AWS CodeStar)。

変更カレンダーを用意して、変更の実施によって影響を受ける可能性のある重要なビジネスや運用上の活動やイベントが計画されている時期を追跡します。アクティビティを調整して、これらの計画に関するリスクを管理します。 AWS Systems Manager 変更カレンダー は、変更に対して時間ブロックがオープンであるかクローズであるか、およびその理由を文書化し、 その情報を他の AWS アカウント と共有します。AWS Systems Manager Automation スクリプトは、カレンダーの変化に沿って実行されるように設定できます。

AWS Systems Manager メンテナンスウィンドウは、 AWS SSM Run Command または Automation スクリプト、AWS Lambda 呼び出し、または AWS Step Functions アクティビティの実行を指定し

た時間にスケジュールできます。これらのアクティビティを評価に含めることができるように、変更カレンダー上で印を付けます。

一般的なアンチパターン:

- あなたがフリート全体でウェブサーバー設定を手動で更新したところ、更新エラーのために多数のサーバーが応答しなくなりました。
- あなたは、何時間もかけて、アプリケーションサーバーフリートを手動で更新します。変更中の設定の不整合が、予期しない動作を引き起こします。
- 誰かがセキュリティグループを更新したため、ウェブサーバーにアクセスできなくなりました。変更内容を把握しなければ、問題の調査にかなりの時間を費やすことになり、復旧までより長くの時間を要することになります。

このベストプラクティスを活用するメリット: 設定管理システムを採用することで、変更やその追跡の労力のレベルと、手動の手順に起因するエラーの頻度を軽減できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 設定管理システムを使用する: 設定管理システムを使用して、変更を追跡、実装し、手動プロセスによって発生するエラーと労力を減らすことができます。
  - インフラストラクチャ設定管理
  - AWS Config
  - AWS Config とは
  - AWS CloudFormation の紹介
  - AWS CloudFormation とは
  - AWS OpsWorks
  - AWS OpsWorks とは
  - AWS Elastic Beanstalk の紹介
  - AWS Elastic Beanstalk とは

リソース

関連するドキュメント:

- AWS AppConfig
- AWS デベロッパーツール
- AWS OpsWorks
- AWS Systems Manager 変更カレンダー
- AWS Systems Manager メンテナンスウィンドウ
- インフラストラクチャ設定管理
- AWS CloudFormation とは
- AWS Config とは
- AWS Elastic Beanstalk とは
- AWS OpsWorks とは

関連動画:

- AWS CloudFormation の紹介
- AWS Elastic Beanstalk の紹介

OPS05-BP04 構築およびデプロイ管理システムを使用する

構築およびデプロイ管理システムを使用します。これらのシステムは、手動プロセスによって発生するエラーと、変更を導入する労力を減らします。

AWS では、以下のサービスを使用して、継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを構築できます。 AWS デベロッパーツール (例: AWS CodeCommit、 AWS CodeBuild、AWS CodePipeline、 AWS CodeDeploy、および AWS CodeStar)。

一般的なアンチパターン:

- 開発システムでコードをコンパイルした後、あなたは、実行可能ファイルを本稼働システムにコピーし、起動に失敗します。ローカルログファイルは、依存関係がないために失敗したことを示しています。
- あなたは、開発環境で新機能を使用してアプリケーションを正常に構築し、品質保証 (QA) にコードを提供します。静的アセットがないため、QA が失敗します。
- 金曜日に、多くの労力をかけて、開発環境でアプリケーションを手動で構築することができました。これには、新しくコード化された機能も含まれます。月曜日に、あなたは、アプリケーションを正常に構築することを可能にするステップを繰り返すことができません。

- あなたは、新しいリリース用に作成したテストを実行します。その後、あなたは、翌週いっぱいをかけて、テスト環境をセットアップし、すべての既存の統合テストを実行してから、パフォーマンステストを実行します。新しいコードには許容できないパフォーマンスへの影響があり、再開発してから再テストする必要があります。

このベストプラクティスを活用するメリット: ビルドとデプロイのアクティビティを管理するメカニズムを提供することで、反復的なタスクを実行するための労力の程度を減らし、チームメンバーは高価値のクリエイティブなタスクに専念し、手動の手順によるエラーの発生を抑制できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 構築およびデプロイ管理システムを使用する: ビルドおよびデプロイ管理システムを使用して、変更を追跡、実装し、手動プロセスによって発生するエラーと労力を減らすことができます。構築、テスト、デプロイ、検証を通じたコードのチェックインから統合とデプロイのパイプラインを完全自動化します。これにより、リードタイムを削減し、変更の頻度を増やすことが可能になり、それにかかわる労力のレベルを減らすことができます。
  - AWS CodeBuild とは
  - ソフトウェア開発のための継続的インテグレーションのベストプラクティス
  - Slalom: AWS のサーバーレスアプリケーション用の CI/CD
  - AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ
  - AWS CodeDeploy とは

リソース

関連するドキュメント:

- AWS デベロッパーツール
- AWS CodeBuild とは
- AWS CodeDeploy とは

関連動画:

- ソフトウェア開発のための継続的インテグレーションのベストプラクティス
- AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ

- Slalom: AWS のサーバーレスアプリケーション用の CI/CD

OPS05-BP05 パッチ管理を実行する

パッチ管理を実行し、問題を解決して、ガバナンスに準拠するようにします。パッチ管理の自動化により、手動プロセスによって発生するエラーと、パッチにかかる労力を減らすことができます。

パッチと脆弱性の管理は、メリットとリスクを管理するアクティビティの一環です。不変のインフラストラクチャを使用し、検証済みの正常な状態でワークロードをデプロイすることが推奨されます。これが不可能な場合は、残りのオプションとしてパッチの適用があります。

マシンイメージ、コンテナイメージ、または Lambda カスタムランタイムと追加ライブラリを更新して 脆弱性を取り除くことは、パッチ管理の一環です。Linux または Windows Server イメージの Amazon マシンイメージ (AMI) への更新については、 EC2 Image Builderを使用して管理する必要があります。既存のパイプラインに Amazon Elastic Container Registry を使用して、 Amazon ECS イメージの管理 および Amazon EKS イメージの管理ができます。AWS Lambda には、 バージョン 管理機能が含まれます。

最初に安全な環境でテストを実施していない状態で、パッチを本番環境のシステムに適用しないでください。パッチは運用上またはビジネス上の成果に対応している場合にのみ適用してください。AWS では、 AWS Systems Manager パッチマネージャーを使用して 管理対象システムにパッチを適用するプロセスを自動化し、 AWS Systems Manager メンテナンスウィンドウを使用してアクティビティをスケジューリングします。

一般的なアンチパターン:

- あなたには、すべての新しいセキュリティパッチを 2 時間以内に適用するために権限が付与されました。その結果、アプリケーションにパッチとの互換性がないため、複数の機能停止が発生しました。
- パッチが適用されていないライブラリは、不明な関係者がライブラリ内の脆弱性を使用してワークロードにアクセスするため、意図しない結果をもたらします。
- あなたは、開発者に通知することなく、自動的に開発者環境にパッチを適用します。あなたには、開発者から、環境が想定どおりに動作しなくなったという苦情が複数寄せられます。
- あなたは、永続的なインスタンスの商用オフザシェルフのセルフソフトウェアにパッチを適用していません。ソフトウェアに問題があり、ベンダーに連絡すると、ベンダーから、バージョンがサポートされておらず、サポートを受けるためには、特定のレベルにパッチを適用する必要があることが伝えられます。

- あなたが使用した暗号化ソフトウェアの最近リリースされたパッチにより、パフォーマンスが大幅に向上します。パッチが適用されていないあなたのシステムには、パッチを適用しない結果として、パフォーマンスの問題が残存しています。

このベストプラクティスを活用するメリット: パッチ適用の基準や環境全体への配布方法など、パッチ管理プロセスを確立することで、それらの利点を実現し、影響を制御することができます。これにより、必要な機能と能力の導入、問題の除去、ガバナンスの継続的な遵守が可能になります。パッチ管理システムと自動化を実装して、パッチをデプロイする労力を軽減し、手動プロセスに起因するエラーの発生を抑制します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- パッチ管理: 問題の修正、希望する機能や能力の取得、ガバナンスポリシーやベンダーのサポート要件への準拠継続を行うためにはシステムをパッチします。変更不可能なシステムでは、必要な成果を達成するために適切なパッチを使用してデプロイします。パッチ管理メカニズムの自動化により、パッチの経過時間、手動プロセスによって発生するエラーと、パッチにかかわる労力を減らすことができます。
  - AWS Systems Manager パッチマネージャーを使用して

リソース

関連するドキュメント:

- AWS デベロッパーツール
- AWS Systems Manager パッチマネージャー

関連動画:

- AWS のサーバーレスアプリケーション用の CI/CD
- Ops を考慮に入れて設計する

関連する例:

- Well-Architected ラボ - インベントリおよびパッチ管理

OPS05-BP06 設計標準を共有する

チーム全体でベストプラクティスを共有し、デプロイ作業における利点の認識を高め、それを最大限にします。

AWS では、アプリケーション、コンピューティング、インフラストラクチャ、運用をコードとして定義し、管理できます。これにより、リリース、共有、採用が簡単になります。

多くの AWS のサービスとリソースは、アカウント間で共有されるように設計されているので、作成されたアセットや発見をチーム間で共有できます。たとえば、 CodeCommit リポジトリ、 Lambda 関数、 Amazon S3 バケット、および AMI を特定のアカウントに共有できます。

新しいリソースやアップデートを発行するときは、Amazon SNS を使用して クロスアカウント通知を発行します。受信者は Lambda を使って新しいバージョンを入手できます。

組織内で共有された標準が適用されている場合、チームのアクティビティをサポートする標準の追加、変更、例外をリクエストするメカニズムを持つことは重要です。このオプションがなければ、標準はイノベーションの障壁になります。

一般的なアンチパターン:

- あなたは、組織内の他の各開発チームが作成したように、独自のユーザー認証メカニズムを作成しました。ユーザーは、アクセスするシステムの各部分について、個別の一連の認証情報を維持する必要があります。
- あなたは、組織内の他の各開発チームが作成したように、独自のユーザー認証メカニズムを作成しました。組織には、遵守されなければならない新しいコンプライアンス要件が与えられています。個々の開発チームには、新しい要件を実装するためにリソースに投資する必要が生じました。
- あなたは、組織内の他の各開発チームが作成したように、独自の画面レイアウトを作成しました。整合性のないインターフェイスを操作することが困難であることについて、ユーザーが苦情を申し出ています。

このベストプラクティスを活用するメリット: 標準が複数のアプリケーションや組織の要件を満たしている場合は、ベストプラクティスの採用をサポートし、開発にかける労力から得られる恩恵を最大化するために、共有された標準を使用します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 設計標準を共有する: 既存のベストプラクティス、設計標準、チェックリスト、運用手順、ガイダンス、ガバナンスの要件をチーム間で共有し、複雑になるのを防ぎながら開発努力の成果を最大化する継続的な改善とイノベーションを支援するために、設計標準の変更、追加、例外を申請する手順を設けます。チームが公開済みのコンテンツを把握していることを確認し、コンテンツを活用して、やり直しや無駄な労力を制限します。
  - AWS 環境へのアクセスの委任
  - AWS CodeCommit リポジトリを共有する
  - AWS Lambda 関数の簡単な承認
  - 特定の AWS アカウント と AMI を共有する
  - AWS CloudFormation デザイナー URL によるテンプレート共有の高速化
  - Amazon SNS での AWS Lambda の使用

リソース

関連するドキュメント:

- AWS Lambda 関数の簡単な承認
- AWS CodeCommit リポジトリを共有する
- 特定の AWS アカウント と AMI を共有する
- AWS CloudFormation デザイナー URL によるテンプレート共有の高速化
- Amazon SNS での AWS Lambda の使用

関連動画:

- AWS 環境へのアクセスの委任

OPS05-BP07 コード品質の向上のためにプラクティスを実装する

コード品質の向上のためにプラクティスを実装し、欠陥を最小限に抑えます。例えば、テスト駆動型開発、コードレビュー、標準の導入などがあります。

AWS では、 Amazon CodeGuru などのサービスを パイプラインと統合し、プログラム分析と機械学習を使用して潜在的なコードと セキュリティの問題を自動的に特定することができます。

CodeGuru は、これらの問題に対処する AWS のベストプラクティスを実装するためのレコメンデーションを提供します。

一般的なアンチパターン:

- あなたは、機能をより迅速にテストできるように、標準入力サニタイズライブラリを統合しないことに決めました。テスト後、あなたは、ライブラリの組み込みを完了することを思い出すことなく、コードをコミットします。
- あなたには、処理しているデータセットに関する経験がほとんどないため、データセット内に一連のエッジケースが存在し得ることを認識していません。これらのエッジケースには、実装したコードとの互換性がありません。

このベストプラクティスを活用するメリット: コードの品質を向上させるためのプラクティスを採用することは、本稼働環境に発生する問題を最小限に抑えることに役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- コード品質の向上のためにプラクティスを実装する: プラクティスを実装して、コード品質を向上させ、欠陥と欠陥がデプロイされるリスクを最低限に抑えます。例えば、テスト駆動型デプロイ、ペアプログラミング、コードレビュー、規約の導入などがあります。
  - Amazon CodeGuru

リソース

関連するドキュメント:

- Amazon CodeGuru

OPS05-BP08 複数の環境を使用する

ワークロードの実験、開発、テストを行うには、複数の環境を使用します。環境が本稼働環境に近づくにつれて増加するコントロールレベルを使用して、デプロイ時にワークロードが意図したとおりに運用するように確信を強化します。

一般的なアンチパターン:

- あなたは、共有開発環境で開発を実行しており、別の開発者があなたのコードの変更を上書きします。
- 共有開発環境の制限的なセキュリティ制御により、あなたは新しいサービスや機能を試すことができません。
- あなたは本稼働用システムで負荷テストを実行し、ユーザーの機能停止を引き起こします。
- データ損失につながる重大なエラーが本稼働環境で発生しました。あなたは、データ損失がどのように発生したかを特定し、これを再び発生させないようにするため、本稼働環境において、データ損失につながる条件を再現しようとします。テスト中のさらなるデータ損失を防ぐため、あなたは、ユーザーがアプリケーションを使用できないようにすることを強制されます。
- あなたは、マルチテナントサービスを運用しており、専用環境に対する顧客のリクエストをサポートできません。
- あなたは常にテストするわけではありませんが、テストする場合は本稼働環境で行います。
- あなたは、単一環境というシンプルさが、環境内での変更の影響範囲に勝ると考えています。

このベストプラクティスを活用するメリット: 複数の環境をデプロイすることで、開発者やユーザーコミュニティ間で競合を生じさせることなく、複数の同時開発、テスト、本稼働環境をサポートできます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 複数の環境を使用する: 開発者が実験できるように、最小のコントロールのサンドボックス環境を提供します。連携できるように個々の開発環境を提供し、開発の俊敏性を増します。開発者がイノベーションを試せるように、本番に近い環境でより厳格なコントロールを実装します。コードとしてインフラストラクチャを使用したり、構成管理システムを使用したりして本番環境に存在するコントロールに準拠して設定された環境をデプロイし、システムがデプロイ時に予想どおりに動作することを確認します。環境を使用しない場合は、オフにして、アイドル状態のリソース (夜間や週末の開発システムなど) に関連するコストを避けることができます。ロードテストで妥当な結果を有効にする場合、本番に相当する環境をデプロイします。
  - AWS CloudFormation とは
  - Amazon EC2 を使用して、AWS Lambda インスタンスを一定の間隔で停止および起動する方法

リソース

関連するドキュメント:

- Amazon EC2 を使用して、AWS Lambda インスタンスを一定の間隔で停止および起動する方法
- AWS CloudFormation とは

OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う

頻繁に、小さく、可逆的な変更を行うことで、変更の範囲と影響を減らします。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また変更を元に戻すこともできます。

一般的なアンチパターン:

- あなたは、四半期ごとに、アプリケーションの新しいバージョンをデプロイします。
- あなたは、データベーススキーマに対して頻繁に変更を加えます。
- あなたは、手動のインプレースアップグレードを実行し、既存のインストールと設定を上書きします。

このベストプラクティスを活用するメリット: 小さな変更を頻繁にデプロイすることで、開発にかける労力から得られる恩恵をすばやく認識できます。変更が小さい場合、意図しない結果が発生するかどうかを識別することがより容易になります。変更を元に戻すことができる場合、復旧が簡素化されるため、変更を実装するリスクが低減されます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 小規模で可逆的な変更を頻繁に行う: 頻繁に、小さく、可逆的な変更を行うことで、変更の範囲と影響を縮小します。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また変更を元に戻すこともできます。また、ビジネスに価値をもたらす速度も向上します。

OPS05-BP10 統合とデプロイを完全自動化する

ワークロードのビルド、デプロイ、テストを自動化します。これにより、手動プロセスによって発生するエラーと、変更をデプロイする労力を減らすことができます。

一貫したタグ付け戦略に従って リソースタグ および AWS Resource Groups を使用して メタデータを適用し、 リソースの識別を可能にします。組織、原価計算、アクセスコントロールのリソースにタグを付け、自動化された運用アクティビティの実行に的を絞ります。

一般的なアンチパターン:

- 金曜日に、あなたは、機能ブランチ用の新しいコードの作成を完了します。月曜日に、あなたは、コード品質テストスクリプトと各ユニットテストスクリプトを実行した後、予定された次のリリースのためにコードをチェックインします。
- あなたは、本稼働中の多数の顧客に影響を与える重要な問題の修正コードを記述するように指示されます。修正をテストした後、あなたは、コードをコミットし、変更管理部門にメールで本番環境へのデプロイの承認を依頼します。

このベストプラクティスを確立するメリット: 自動化されたビルドおよびデプロイ管理システムを実装することで、手動プロセスにより発生するエラーを削減し、変更をデプロイする労力を減らして、チームメンバーがビジネス価値の提供に注力できるようにします。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 構築およびデプロイ管理システムを使用する: ビルドおよびデプロイ管理システムを使用して、変更を追跡、実装し、手動プロセスによって発生するエラーと労力を減らすことができます。構築、テスト、デプロイ、検証を通じたコードのチェックインから統合とデプロイのパイプラインを完全自動化します。これにより、リードタイムを削減し、変更の頻度を増やすことが可能になり、それにかかわる労力のレベルを減らすことができます。
  - AWS CodeBuild とは
  - ソフトウェア開発のための継続的インテグレーションのベストプラクティス
  - Slalom: CI/CD for serverless applications on AWS (Slalom: AWS のサーバーレスアプリケーション用の CI/CD)
  - Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services (AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ)
  - AWS CodeDeploy とは

リソース

関連するドキュメント:

- AWS CodeBuild とは
- AWS CodeDeploy とは

関連動画:

- ソフトウェア開発のための継続的インテグレーションのベストプラクティス
- Introduction to AWS CodeDeploy - automated software deployment with Amazon Web Services (AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ)
- Slalom: CI/CD for serverless applications on AWS (Slalom: AWS のサーバーレスアプリケーション用の CI/CD)

OPS 6 デプロイのリスクを軽減するにはどうすればよいですか?

品質に関する迅速なフィードバックを提供し、望ましい結果をもたらさない変更から迅速に復旧できるようにするアプローチを採用します。このような手法を使用すると、変更のデプロイによって生じる問題の影響を軽減できます。

ベストプラクティス

- OPS06-BP01 変更の失敗に備える
- OPS06-BP02 変更をテストし、検証する
- OPS06-BP03 デプロイ管理システムを使用する
- OPS06-BP04 限定的なデプロイを使用してテストする
- OPS06-BP05 並列環境でデプロイする
- OPS06-BP06 小規模で可逆的な変更を頻繁にデプロイする
- OPS06-BP07 統合とデプロイを完全自動化する
- OPS06-BP08 テストとロールバックを自動化する

OPS06-BP01 変更の失敗に備える

変更が望ましい結果をもたらさない場合に、既知の良好な状態に戻すか、本番環境で修正を行うことを計画します。この準備を行うことで、迅速な対応によって復旧時間を短縮できます。

一般的なアンチパターン:

- あなたがデプロイを実行したところ、アプリケーションが不安定になりましたが、システムにはアクティブなユーザーがいるように見えます。あなたは、変更をロールバックしてアクティブなユーザーに影響を与えるか、またはユーザーが影響を受ける可能性があることを考慮して変更をロールバックするのを待つかを判断しなければなりません。

- ルーティンを変更すると、新しい環境はアクセスできますが、サブネットの 1 つにアクセスできなくなります。あなたは、すべてをロールバックするか、アクセスできないサブネットを修正するかを判断しなければなりません。その判断がなされるまでの間、サブネットはアクセスできないままとなります。

このベストプラクティスを活用するメリット: 計画を事前に立てることで、失敗からの平均復旧時間 (MTTR) を短縮し、エンドユーザーへの影響を抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 変更の失敗に備える: 変更が望ましい結果をもたらさない場合に、既知の良好な状態に戻す (変更をロールバックする) か、本番環境で修正を行う (変更をロールフォワードする) ことを計画します。失敗した変更をロールバックできないことがわかった場合は、変更をコミットする前にデューデリジェンスを適用します。

OPS06-BP02 変更をテストし、検証する

あらゆるライフサイクルステージで変更をテストし、その結果を検証することで、新しい機能を確認するとともに、デプロイの失敗のリスクと影響を最小限に抑えます。

AWS では、実験やテストのリスク、労力、コストを削減するために一時的な並列環境を作成できます。これらの環境のデプロイを AWS CloudFormation を使用して自動化し、一時環境の実装に一貫性を持たせます。

一般的なアンチパターン:

- あなたは、斬新な新機能をアプリケーションにデプロイします。動作しません。理由はわかりません。
- あなたは、証明書を更新します。あなたは、意図せずに、誤ったコンポーネントに証明書をインストールします。理由はわかりません。

このベストプラクティスを活用するメリット: デプロイ後の変更をテストして検証することで、問題を早期に特定し、顧客への影響を軽減する機会が得られます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 変更をテストし、検証する: あらゆるライフサイクルステージ (開発、テスト、本番など) で変更をテストし、その結果を検証することで、新しい機能を確認するとともに、デプロイの失敗のリスクと影響を最小限に抑えます。
  - AWS Cloud9
  - AWS Cloud9 とは
  - コードを送信する前に AWS CodeDeploy をローカルでテスト/デバッグする方法

リソース

関連するドキュメント:

- AWS Cloud9
- AWS デベロッパーツール
- コードを送信する前に AWS CodeDeploy をローカルでテスト/デバッグする方法
- AWS Cloud9 とは

OPS06-BP03 デプロイ管理システムを使用する

デプロイ管理システムを使用して変更を追跡および実装します。これにより、手動プロセスによって発生するエラーと、変更をデプロイする労力を減らすことができます。

AWS では、以下のサービスを使用して、継続的インテグレーションと継続的デプロイ (CI/CD) パイプラインを構築できます AWS デベロッパーツール (例: AWS CodeCommit、 AWS CodeBuild、AWS CodePipeline、 AWS CodeDeploy、および AWS CodeStar)。

一般的なアンチパターン:

- あなたがフリート全体でアプリケーションサーバーに対して手動で更新をデプロイしたところ、更新エラーのために多数のサーバーが応答しなくなりました。
- あなたは、何時間もかけて、アプリケーションサーバーフリートを手動でデプロイします。変更中のバージョンの不整合が、予期しない動作を引き起こします。

このベストプラクティスを活用するメリット: デプロイ管理システムを採用することで、変更のデプロイにかける労力のレベルと、手動の手順に起因するエラーの頻度を軽減できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- デプロイ管理システムを使用する: デプロイ管理システムを使用して変更を追跡および実装します。これにより、手動プロセスによって発生するエラーと、変更をデプロイする労力が減ります。テスト、デプロイ、検証を通じたコードのチェックインから統合とデプロイのパイプラインを自動化します。これにより、リードタイムが減り、変更の頻度を増やすことが可能になるとともに、必要な労力がさらに減ります。
  - AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ
  - AWS CodeDeploy とは
  - AWS Elastic Beanstalk とは
  - Amazon API Gateway とは

リソース

関連するドキュメント:

- AWS CodeDeploy ユーザーガイド
- AWS デベロッパーツール
- AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す
- AWS CodeDeploy とは
- AWS Elastic Beanstalk とは
- Amazon API Gateway とは

関連動画:

- AWS を使用した高度な継続的デリバリーテクニックの詳細
- AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ

OPS06-BP04 限定的なデプロイを使用してテストする

完全なデプロイを行う前に、既存のシステムと並行して限定的なデプロイを実施してテストを行い、望ましい結果が得られるかどうか確認します。例えば、デプロイ Canary テストまたはワンボックスデプロイを使用します。

一般的なアンチパターン:

- あなたは、失敗した変更を一度にすべての本稼働環境にデプロイします。あなたにはわかりません。

このベストプラクティスを確立するメリット: 制限されたデプロイ後の変更をテストして検証することで、顧客への影響を最小限に抑えながら、問題を早期に特定し、顧客への影響をさらに軽減する機会が得られます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 限定的なデプロイを使用してテストする: 本格的なデプロイの前に、既存のシステムと一緒に限定的にデプロイしてテストを行い、期待される結果が得られるかどうかを確認します。例えば、デプロイ Canary テストまたはワンボックスデプロイを使用します。
  - AWS CodeDeploy ユーザーガイド
  - AWS Elastic Beanstalk を使用したブルー/グリーンデプロイ
  - API Gateway Canary リリースデプロイの設定
  - Try a Sample Blue/Green Deployment in AWS CodeDeploy (AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す)
  - Working with deployment configurations in AWS CodeDeploy (AWS CodeDeploy でのデプロイ構成の操作)

リソース

関連するドキュメント:

- AWS CodeDeploy ユーザーガイド
- AWS Elastic Beanstalk を使用したブルー/グリーンデプロイ
- API Gateway Canary リリースデプロイの設定
- Try a Sample Blue/Green Deployment in AWS CodeDeploy (AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す)
- Working with deployment configurations in AWS CodeDeploy (AWS CodeDeploy でのデプロイ構成の操作)

OPS06-BP05 並列環境でデプロイする

並列環境に変更を実装し、その後、新しい環境に移行します。デプロイの成功を確認するまで、以前の環境を維持します。こうすることで、以前の環境へのロールバックが可能になり、復旧時間を最小限に抑えることができます。

一般的なアンチパターン:

- あなたは、既存のシステムを変更することにより、変更可能なデプロイを実行します。変更が失敗したことを検出した後、あなたは、システムを再度変更して古いバージョンを復元するよう命じられ、これにより、復旧までの時間がより長くかかります。
- あなたは、メンテナンスウィンドウ中に、古い環境を停止してから、新しい環境の構築を開始します。この手順には何時間もかかり、あなたは、デプロイに復旧できない問題を検出します。非常に疲れていますが、あなたは、以前のデプロイの手順を探し、古い環境の再構築を開始するように命じられます。

このベストプラクティスを活用するメリット: 並列環境を使用することで、新しい環境を事前にデプロイし、必要に応じて環境に移行できます。新しい環境が失敗した場合は、元の環境に戻すことで、すばやく復旧できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 並列環境でデプロイする: 並列環境に変更を実装し、その後、新しい環境に移行またはカットオーバーします。デプロイの成功を確認するまで、以前の環境を維持します。こうすることで、以前の環境へのロールバックが可能になり、復旧時間を最小限に抑えることができます。例えば、ブルー/グリーンデプロイでイミュータブルインフラストラクチャを使用します。
  - AWS CodeDeploy でのデプロイ構成の操作
  - AWS Elastic Beanstalk を使用したブルー/グリーンデプロイ
  - API Gateway Canary リリースデプロイの設定
  - AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す

リソース

関連するドキュメント:

- AWS CodeDeploy ユーザーガイド

- AWS Elastic Beanstalk を使用したブルー/グリーンデプロイ
- API Gateway Canary リリースデプロイの設定
- AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す
- AWS CodeDeploy でのデプロイ構成の操作

関連動画:

- AWS を使用した高度な継続的デリバリーテクニックの詳細

OPS06-BP06 小規模で可逆的な変更を頻繁にデプロイする

小規模で可逆的な変更を頻繁に行うことで、変更の範囲を減らします。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また、変更をロールバックすることもできます。

一般的なアンチパターン:

- あなたは、四半期ごとに、アプリケーションの新しいバージョンをデプロイします。
- あなたは、データベーススキーマに対して頻繁に変更を加えます。
- あなたは、手動のインプレースアップグレードを実行し、既存のインストールと設定を上書きします。

このベストプラクティスを活用するメリット: 小さな変更を頻繁にデプロイすることで、開発にかける労力から得られる恩恵をすばやく認識できます。変更が小さい場合、意図しない結果が発生するかどうかを識別することがより容易になります。変更を元に戻すことができる場合、復旧が簡素化されるため、変更を実装するリスクが低減されます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 小規模で可逆的な変更を頻繁にデプロイする: 頻繁に、小さく、可逆的な変更を使用することで、変更の範囲を縮小します。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また、変更をロールバックすることもできます。

OPS06-BP07 統合とデプロイを完全自動化する

ワークロードのビルド、デプロイ、テストを自動化します。これにより、手動プロセスによって発生するエラーと、変更をデプロイする労力を減らすことができます。

一貫したタグ付け戦略に従って リソースタグ および AWS Resource Groups を使用して メタデータを適用し、 リソースの識別を可能にします。組織、原価計算、アクセスコントロールのリソースにタグを付け、自動化された運用アクティビティの実行に的を絞ります。

一般的なアンチパターン:

- 金曜日に、あなたは、機能ブランチ用の新しいコードの作成を完了します。月曜日に、あなたは、コード品質テストスクリプトと各ユニットテストスクリプトを実行した後、予定された次のリリースのためにコードをチェックインします。
- あなたは、本稼働中の多数の顧客に影響を与える重要な問題の修正コードを記述するように指示されます。修正をテストした後、あなたは、コードと E メールの変更管理をコミットして、本稼働環境にデプロイするための承認をリクエストします。

このベストプラクティスを活用するメリット: 自動化されたビルドおよびデプロイ管理システムを実装することで、手動プロセスにより発生するエラーを削減し、変更をデプロイする労力を減らして、チームメンバーがビジネス価値の提供に注力できるようにします。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 構築およびデプロイ管理システムを使用する: ビルドおよびデプロイ管理システムを使用して、変更を追跡、実装し、手動プロセスによって発生するエラーと労力を減らすことができます。構築、テスト、デプロイ、検証を通じたコードのチェックインから統合とデプロイのパイプラインを完全自動化します。これにより、リードタイムを削減し、変更の頻度を増やすことが可能になり、それにかかわる労力のレベルを減らすことができます。
  - AWS CodeBuild とは
  - ソフトウェア開発のための継続的インテグレーションのベストプラクティス
  - Slalom: AWS のサーバーレスアプリケーション用の CI/CD
  - AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ
  - AWS CodeDeploy とは
  - AWS を使用した高度な継続的デリバリーテクニックの詳細

リソース

関連するドキュメント:

- AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す
- AWS CodeBuild とは
- AWS CodeDeploy とは

関連動画:

- ソフトウェア開発のための継続的インテグレーションのベストプラクティス
- AWS を使用した高度な継続的デリバリーテクニックの詳細
- AWS CodeDeploy の紹介 - Amazon Web Services を使用したソフトウェアの自動デプロイ
- Slalom: AWS のサーバーレスアプリケーション用の CI/CD

OPS06-BP08 テストとロールバックを自動化する

デプロイした環境のテストを自動化し、望ましい結果が得られるかどうか確認します。結果が得られなかった場合に、以前の正常な状態へのロールバックを自動化し、復旧時間を最小限に抑え、手動プロセスによるエラーを低減します。

一般的なアンチパターン:

- あなたは、ワークロードに変更をデプロイします。変更が完了したことを確認した後、あなたは、デプロイ後のテストを開始します。テストが完了したことを確認した後、あなたは、ワークロードが操作不可であり、顧客の接続が切断されたことに気づきます。その後、あなたは、以前のバージョンへのロールバックを開始します。問題を検出するのに長い時間をかけた後、復旧にかかる時間は、手動による再デプロイによってさらに長くなります。

このベストプラクティスを確立するメリット: デプロイ後の変更をテストして検証することで、問題をすぐに特定できます。以前のバージョンに自動的にロールバックすることで、顧客への影響を最小限に抑えることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- テストとロールバックを自動化する: デプロイした環境のテストを自動化し、望ましい結果が得られるかどうか確認します。結果が得られなかった場合に、以前の正常な状態へのロールバックを自動化し、復旧時間を最小限に抑え、手動プロセスによるエラーを低減します。例えば、デプロイ後に詳細な合成ユーザートランザクションを実施し、その結果を確認して、失敗した場合にはロールバックします。
  - Redeploy and roll back a deployment with AWS CodeDeploy (AWS CodeDeploy を使用した再デプロイとロールバック)

リソース

関連するドキュメント:

- Redeploy and roll back a deployment with AWS CodeDeploy (AWS CodeDeploy を使用した再デプロイとロールバック)

OPS 7 ワークロードをサポートする準備が整っていることを確認するにはどうすればよいですか?

ワークロード、プロセス、手順、従業員の運用準備状況を評価し、ワークロードに関連する運用上のリスクを理解するようにします。

ベストプラクティス

- OPS07-BP01 人材能力の確保
- OPS07-BP02 運用準備状況の継続的な確認を実現する
- OPS07-BP03 ランブックを使用して手順を実行する
- OPS07-BP04 プレイブックを使用して問題を調査する
- OPS07-BP05 システムや変更をデプロイするために十分な情報に基づいて決定を下す

OPS07-BP01 人材能力の確保

運用上のニーズに対応できるようにトレーニングを受けた、適切な人数の従業員が配置されていることを検証するメカニズムを導入します。効果的なサポートを継続できるように必要に応じて従業員のトレーニングを実施し、従業員の対応力を調整します。

すべてのアクティビティ (オンコールを含む) に対応できる十分なチームメンバーが必要になります。ワークロード、運用ツール、AWS に関するトレーニングを成功させるために必要なスキルがチームにあることを確認します。

AWS は、 AWS ご利用開始のためのリソースセンター、 AWS ブログ、 AWS オンラインテックトーク、 AWS イベントスケジュール、および AWS Well-Architected ラボなどのリソースを提供しており、チームを教育するためのガイダンス、事例、詳細なチュートリアルを提供しています。さらに、 AWS トレーニング と認定 では、AWS の基礎に関する自習用デジタルコースによる無料のトレーニングを提供しています。また、インストラクターが実施するトレーニングに登録して、チームの AWS スキルの開発をさらにサポートすることもできます。

一般的なアンチパターン:

- 使用中のプラットフォームとサービスをサポートするスキルがあるチームメンバーなしでワークロードをデプロイする。
- サポート時間中に対応可能なチームメンバーなしでワークロードをデプロイする。
- 退職したチームメンバーまたは病気で休職中のチームメンバーがいる場合、それをサポートするために十分なチームメンバーなしでワークロードをデプロイする。
- 追加のワークロードおよび他のワークロードをサポートするチームメンバーに対する追加の影響を確認することなく、追加のワークロードをデプロイする。

このベストプラクティスを確立するメリット: スキルのあるチームメンバーを持つことで、ワークロードを効果的にサポートできます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- 人材能力: ワークロードを効果的にサポートするために、十分にトレーニングを受けた人材がいることを確認します。
  - チームの規模: オンコールを含め、運用アクティビティに対応できるだけの十分なチームのメンバーを確保します。
  - チームのスキル: チームのメンバーが、AWS、ワークロード、業務を遂行するために使用する運用ツールに関して十分なトレーニングを受けていることを確認します。
    - AWS イベントスケジュール
    - AWS トレーニング と認定にようこそ

- 能力の見直し: 運用状況や作業量の変化に応じてチームの規模やスキルを見直し、運用上の優秀性を維持するために十分な能力を確保します。調整を行い、チームの規模とスキルが、そのチームでサポートするワークロードの運用要件に一致するようにします。

リソース

関連するドキュメント:

- AWS ブログ
- AWS イベントスケジュール
- AWS ご利用開始のためのリソースセンター
- AWS オンラインテックトーク
- AWS トレーニング と認定にようこそ

関連する例:

- Well-Architected ラボ

OPS07-BP02 運用準備状況の継続的な確認を実現する

運用準備状況レビュー (ORR) を使用して、組織のワークロードを運用できることを検証します。ORR は Amazon が開発した仕組みの 1 つで、チームがワークロードを安全に運用できることを検証します。ORR は、要件のチェックリストを使用したレビューおよび検証プロセスです。ORR は、ワークロードの検証をチームが自分たちで行うことができるセルフサービスエクスペリエンスです。ORR には、Amazon がソフトウェアを開発する中で学んだ知識や経験に基づくベストプラクティスが含まれます。

ORR チェックリストは、アーキテクチャレコメンデーション、運用プロセス、イベント管理、リリース品質によって構成されます。Amazon のエラーの修正 (CoE) プロセスは、主にこれらの項目によって推進されます。組織の ORR の発展を推進するには、独自のインシデント後の分析を使用する必要があります。ORR はベストプラクティスに従うためだけでなく、過去に経験したイベントの再発を防ぐためのものです。また、セキュリティ、ガバナンス、コンプライアンスの各要件も ORR に含めることができます。

ワークロードの一般提供前に ORR を実施し、その後はソフトウェア開発ライフサイクルをとおして実施し続けます。ワークロードのローンチ前に ORR を実施することで、ワークロードをより安全に運用することができます。ORR をワークロードで定期的に実施することで、ベストプラクティスか

らの逸脱を検知することができます。ORR チェックリストは、新しいサービスのローンチや、ORR の定期的なレビューに使用できます。そうすることで、新しいベストプラクティスに沿って更新したり、インシデント後の分析で学んだ知識や経験を反映したりできます。クラウドの使用に慣れていくにしたがって、組織のアーキテクチャのデフォルトの要件として ORR を組み込むことができます。

期待される成果: 組織にはベストプラクティスを含む ORR チェックリストがあります。ORR はワークロードのローンチ前に実施されます。ORR はワークロードライフサイクルを通じて定期的に実施されます。

一般的なアンチパターン:

- 運用できるかどうか不明なままワークロードをローンチする。
- ガバナンスおよびセキュリティ要件は、ワークロードのローンチ要件に含まれていない。
- ワークロードは定期的に評価されていない。
- 必要な手続きなしでワークロードがローンチされる。
- 複数のワークロードで同じ根本原因の故障が繰り返される。

このベストプラクティスを活用するメリット:

- 組織のワークロードには、アーキテクチャ、プロセス、および管理のベストプラクティスが含まれる。
- 学んだ知識や経験は ORR プロセスに反映される。
- 必要な手続きでワークロードがローンチされる。
- ORR はワークロードのソフトウェアライフサイクルを通じて実施される。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ORR は、プロセスとチェックリストの 2 つの要素で構成されます。ORR プロセスは組織で採用され、エグゼクティブスポンサーによってサポートされる必要があります。ORR は少なくともワークロードの一般提供前に実施する必要があります。ソフトウェア開発ライフサイクルを通じて ORR を実施し、ベストプラクティスや新しい要件を反映して更新します。ORR チェックリストは、構成可能な項目、セキュリティおよびガバナンスの要件、組織のベストプラクティスを含める必要があります。時間の経過とともに、 AWS Config、 AWS Security Hub、 AWS Control Tower ガードレールなどのサービスを使用して、ORR のベストプラクティスをガードレールに変換し、ベストプラクティスの検出の自動化を行います。

顧客の事例

いくつかの製造インシデントが発生した後、AnyCompany Retail は ORR プロセスを導入することを決めました。彼らはベストプラクティス、ガバナンスおよびコンプライアンスの要件、故障から学んだ知識や経験で構成されたチェックリストを作成しました。新しいワークロードのローンチ前には、ORR が実施されます。すべてのワークロードでは、ベストプラクティスのサブセットを使用して年次 ORR が実施され、ORR チェックリストに追加されたベストプラクティスや要件が反映されます。時間の経過とともに、AnyCompany Retail は AWS Config を使用して、ベストプラクティスを検出し ORR プロセスを迅速化しました。

実装手順

ORR の詳細については、 運用準備状況の確認 (ORR) に関するホワイトペーパーをご覧ください。このドキュメントでは、ORR プロセスの歴史、独自の ORR プラクティスの構築方法、ORR チェックリストの作成方法に関する詳細な情報を提供しています。以下の手順は、このドキュメントからの抜粋です。ORR および独自の ORR の構築方法の詳細については、このホワイトペーパーをご覧ください。

1. セキュリティ、運用、開発の代表者を含む、主要な関係者を集めます。
2. 各関係者に少なくとも 1 つの要件を提供してもらいます。初回に提供される要件は、30 項目以下に制限します。
   - 付録 B: 運用準備状況の確認 (ORR) に関する ホワイトペーパーの ORR 質問の例には、使用できるいくつかの質問の例が含まれています。
3. 要件をスプレッドシートにまとめます。
   - ここでは AWS Well-Architected Tool の カスタムレンズを 使用して ORR を作成し、 アカウントや AWS 組織全体で共有することができます。
4. ORR を実施するワークロードを 1 つ選びます。ローンチ前のワークロード、または内部ワークロードが理想的です。
5. ORR チェックリストを実施し、発見した事柄をメモします。定められた緩和がある場合、発見は問題になる可能性があります。緩和が定められていない発見については、対応予定の項目に追加して、ローンチ前に対応を実施します。
6. 時間の経過とともに、ベストプラクティスや要件を ORR に継続的に追加します。

エンタープライズサポートのある AWS Support の顧客は、 運用準備状況の確認に関するワークショップ をテクニカルアカウントマネージャーからリクエストできます。このワークショップでは、 顧客の視点から ORR チェックリストの作成を行います。

実装計画に必要な工数レベル: 高。組織で ORR プラクティスを採用するには、エグゼクティブスポンサーと関係者の同意が必要です。組織全体からのインプットを含めてチェックリストを作成し更新します。

リソース

関連するベストプラクティス:

- OPS01-BP03 ガバナンス要件を評価する - ガバナンス要件は ORR チェックリストに適しています。
- OPS01-BP04 コンプライアンス要件を評価する - コンプライアンス要件は ORR チェックリストに含まれることがあります。別のプロセスに含まれる場合もあります。
- OPS03-BP07 チームに適正なリソースを提供する - チームキャパシティは ORR 要件の良い候補です。
- OPS06-BP01 変更の失敗に備える - ワークロードをローンチする前に、ロールバックプランまたはロールフォワードプランを確立する必要があります。
- OPS07-BP01 人材能力の確保 - ワークロードをサポートするために、必要な人材を確保する必要があります。
- SEC01-BP03 管理目標を特定および検証する - セキュリティ管理目標は ORR 要件に最適の項目です。
- REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する - ディザスタリカバリプランは ORR 要件に適しています。
- COST02-BP01 組織の要件に基づいてポリシーを策定する - コスト管理ポリシーは ORR チェックリストの項目として適しています。

関連するドキュメント:

- AWS Control Tower - AWS Control Tower のガードレール
- AWS Well-Architected Tool - カスタムレンズ
- Adrian Hornsby による運用準備状況レビューテンプレート
- 運用準備状況の確認 (ORR) に関するホワイトペーパー

関連動画:

- あなたをサポートする AWS | 効果的な運用準備状況レビュー (ORR) の構築

関連サンプル:

- 運用準備状況レビュー (ORR) レンズの例

関連サービス:

- AWS Config
- AWS Control Tower
- AWS Security Hub
- ORR を作成し、

OPS07-BP03 ランブックを使用して手順を実行する

A ランブック は、特定の成果を達成するために文書化されたプロセスです。ランブックは一連のステップから成り、それをたどることでプロセスを完了できます。ランブックは、飛行機の黎明期から運用に使用されてきました。クラウド運用では、ランブックを使用してリスクを削減し、望ましい成果を達成します。端的に言うと、ランブックはタスクを完了するためのチェックリストです。

ランブックは、ワークロードを運用するための不可欠の一部です。新しいチームメンバーのオンボーディングからメジャーリリースのデプロイまで、ランブックは、使用者に関係なく、一定の成果をもたらすように成文化されたプロセスです。ランブックの更新は変更管理プロセスの重要な要素であるため、ランブックは一箇所で公開し、プロセスの進化に合わせて更新する必要があります。また、エラー処理、ツール、アクセス許可、例外、問題発生時のエスカレーションに関するガイダンスを含める必要があります。

組織が成熟してきたら、ランブックの自動化を始めましょう。短く、頻繁に使用されるランブックから始めます。スクリプト言語を使用して、ステップを自動化するか、ステップを実行しやすくします。最初のいくつかのランブックを自動化したら、より複雑なランブックを自動化するために時間を割くようにします。やがて、ほとんどのランブックが何らかの方法で自動化されるはずです。

期待される成果: チームには、ワークロードのタスクを実行するためのステップバイステップのガイド集があります。ランブックには、期待される成果、必要なツールとアクセス許可、エラー処理に関する指示が含まれています。一箇所に保管され、頻繁に更新されます。

一般的なアンチパターン:

- プロセスの各ステップの完了を記憶に頼る。
- チェックリストなしで、変更を手動でデプロイする。

- 異なるチームメンバーが同じプロセスを実行しても、手順や結果が異なる。
- システムの変更や自動化に伴い、ランブックの同期がとれなくなる

このベストプラクティスを活用するメリット:

- 手動タスクのエラー率を削減します。
- 運用が一貫した方法で実行されます。
- 新しいチームメンバーがタスクの実行をすぐに始められます。
- ランブックの自動化により、苦労を減らすことができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

ランブックは、組織の成熟度に応じて、いくつかの形態をとります。少なくとも、ステップバイステップのテキスト文書で構成されている必要があります。期待される成果が明確に示されている必要があります。必要な特殊なアクセス許可やツールを明確に文書化します。問題発生時にエラー処理とエスカレーションに関する詳細なガイダンスを提供します。ランブックの所有者をリストアップし、一元的な場所で公開します。ランブックが文書化されたら、チームの別のメンバーに使用してもらって検証します。プロセスの進化につれて、変更管理プロセスに従ってランブックを更新します。

組織が成熟するにつれて、テキストのランブックは自動化されるはずです。例えば、AWS Systems Manager オートメーションなどのサービスを使用すると、フラットなテキストを、ワークロードに対して実行可能なオートメーションに変換できます。これらのオートメーションはイベントに反応して実行でき、ワークロードを保守する運用上の負担が軽減されます。

顧客の事例

AnyCompany Retail は、ソフトウェアのデプロイ時にデータベーススキーマの更新を行う必要があります。クラウド運用チームはデータベース管理チームと協力して、これらの変更を手動でデプロイするためのランブックを作成しました。ランブックには、プロセスの各ステップがチェックリスト形式で記載されました。問題発生時のエラー処理のセクションも含まれています。このランブックは、他のランブックとともに社内 Wiki で公開されました。クラウド運用チームは、将来のスプリントでランブックを自動化する予定です。

実装手順

既存のドキュメントリポジトリがない場合、バージョン管理リポジトリはランブックライブラリの構築を始める場所として最適です。ランブックは Markdown を使用して作成できます。ランブック作成の開始に使用できるサンプルのランブックテンプレートを提供しています。

```
# ######### ## ####### | ##### ID | ## | ##### | ######### | ######## | ##### | #######
# POC | |-------|-------|-------|-------|-------|-------|-------| | RUN001 | ##########
 ####### | ### | ###### | ##### | 2022 # 9 # 21 # | ######### | ## #### 1##### 1 2#####
 2
```

1. 既存のドキュメントリポジトリや Wiki がない場合は、バージョン管理システムに新しいバージョン管理リポジトリを作成します。
2. ランブックがないプロセスを特定します。理想的なプロセスは、半定期的に実施され、ステップ数が少なく、失敗の影響が少ないプロセスです。
3. ドキュメントリポジトリに、テンプレートを使用して新しいドラフト Markdown ドキュメントを作成します。その際、 ########## と ######################.
4. 最初のステップから開始して、ランブックの #### 部分を入力します。
5. ランブックをチームメンバーに渡します。ランブックを使用してもらって、ステップを検証します。不足しているものや明確化が必要なものがあれば、ランブックを更新します。
6. ランブックを社内ドキュメントストアに公開します。公開したら、チームや他の関係者に伝えましょう。
7. 時間が経てば、ランブックのライブラリが構築されますライブラリが大きくなったら、ランブックを自動化する作業を開始します。

実装計画に必要な工数レベル: 低。ランブックの最低基準は、ステップバイステップのテキストガイドです。ランブックの自動化は、導入の手間を増やす可能性があります。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する: ランブックには、保守を担当する所有者が必要です。
- OPS07-BP04 プレイブックを使用して問題を調査する: ランブックとプレイブックは似ていますが、1 つだけ違うのは、ランブックには期待される成果があることです。多くの場合、プレイブックが根本原因を特定すると、ランブックがトリガーさ れます。

- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用するランブックは、適切なイベント、インシデント、および問題管理の実践の一部です。
- OPS10-BP02 アラートごとにプロセスを用意する: ランブックやプレイブックは、アラートに対応するために使用する必要があります。時間の経過とともに、これらの対応を自動化する必要があります。
- OPS11-BP04 ナレッジ管理を実施する: ランブックの保守は、ナレッジマネジメントの重要な一部です。

関連するドキュメント:

- 自動化されたプレイブックとランブックを使用して運用上の優秀性を達成する
- AWS Systems Manager: ランブックの操作
- AWS の大規模移行のための移行プレイブック - タスク 4: 移行ランブックの改良
- AWS Systems Manager Automation ランブックを使用して、運用タスクを解決する

関連動画:

- AWS re:Invent 2019: ランブック、インシデントレポート、インシデント対応の DIY ガイド (SEC318-R1)
- AWS | Amazon Web Services での IT 運用を自動化する方法
- スクリプトを AWS Systems Manager に統合する

関連する例:

- AWS Systems Manager: オートメーションチュートリアル
- AWS Systems Manager: 最新のスナップショットランブックからルートボリュームを復元する
- Jupyter Notebook と CloudTrail Lake を使用して、AWS インシデント対応ランブックを作成する
- Gitlab - ランブック
- Rubix - Jupyter Notebook でランブックを作成するための Python ライブラリ
- Document Builder を使用してカスタムランブックを作成する
- Well-Architected ラボ: プレイブックとランブックによるオペレーションの自動化

関連サービス:

- AWS Systems Manager Automation

OPS07-BP04 プレイブックを使用して問題を調査する

プレイブックは、インシデントの調査に使用するステップバイステップガイドです。インシデントが発生した際は、プレイブックを使用して調査を行い、影響の範囲と根本原因を特定します。プレイブックは、デプロイの失敗からセキュリティインシデントに至るまで、さまざまなシナリオで使用されます。ランブックを使用して緩和する根本原因は、多くの場合プレイブックによって特定します。プレイブックは、組織のインシデント対応計画の基幹的なコンポーネントです。

優れたプレイブックには、いくつかの重要な特徴があります。プレイブックは検出プロセスにおける各手順をユーザーに示します。外側から内側への思考を使って、インシデントの診断に必要な手順を示します。 特別なツールやより高い権限が必要な場合は、プレイブックで明確に定義します。インシデント調査のステータスを関係者と共有するためのコミュニケーションプランの策定は重要なコンポーネントです。根本原因を特定できない場合に備え、プレイブックにはエスカレーションプランが必要です。根本原因を特定できる場合、プレイブックは問題の解決方法が記載されているランブックを示す必要があります。プレイブックは一元的に保管し、定期的に更新する必要があります。特定のアラートにプレイブックを使用する場合、使用すべきプレイブックをアラート内でチームに示します。

組織が成熟するにしたがって、プレイブックを自動化します。最初に、低リスクインシデント用のプレイブックを作成します。スクリプトを使用して検出手順を自動化します。一般的な根本原因を緩和するための関連するランブックも作成します。

期待される成果: 組織には一般的なインシデントに対するプレイブックがあります。プレイブックは一元的に保管され、チームメンバーに提供されます。プレイブックは頻繁に更新されます。既知の根本原因については、関連するランブックが作成されています。

一般的なアンチパターン:

- インシデントを調査する標準的な方法はない。
- チームメンバーは過去の経験や社内で蓄積した知識に基づいて、失敗したデプロイの問題を解決している。
- 新しいチームメンバーは、トライアンドエラーを通じて問題の調査方法を学んでいる。
- 問題調査のベストプラクティスは、チーム間で共有されていない。

このベストプラクティスを活用するメリット:

- プレイブックはインシデント緩和の工数を削減します。
- さまざまなチームメンバーが同じプレイブックを使って、一貫した方法で根本原因の特定を行えます。
- 既知の根本原因にはランブックが用意されており、復旧時間を短縮できます。
- プレイブックによって、新しいチームメンバーはすぐにチームに貢献できるようになります。
- 繰り返し使用可能なプレイブックを持つことで、チームはプロセスをスケールすることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

プレイブックの作成方法と使用方法は、組織の成熟度によって異なります。組織がクラウドに慣れていない場合、文章によるプレイブックを作成し、中央ドキュメントリポジトリに保管します。組織が成熟するにしたがって、Python などのスクリプト言語を使用して、プレイブックを半自動化できます。これらのスクリプトは Jupyter Notebook 内で実行でき、復旧を迅速化します。高度な組織では、一般的な問題のプレイブックを完全に自動化し、ランブックを使用して自動的に問題を緩和します。

プレイブックの作成は、組織のワークロードで発生する一般的なインシデントを一覧化することから始めます。最初に、根本原因がいくつかの問題に絞られている、低リスクインシデント用のプレイブックを作成します。シンプルなシナリオ用のプレイブックの作成後、高リスクシナリオや根本原因があまり知られていないシナリオ用のプレイブックを作成します。

組織が成熟するにつれて、文章によるプレイブックを自動化します。例えば、 AWS Systems Manager Automations などのサービスを使用して、テキストを自動化に変換することができます。これらの自動化を組織のワークロードで実行し、調査を迅速化できます。これらの自動化はイベントへの応答としてアクティブ化され、インシデントの検出と解決の平均時間を短縮します。

顧客は AWS Systems Manager Incident Manager を使用して インシデントに対応できます。このサービスは、インシデントのトリアージを行い、インシデントの検出中および緩和中に関係者に情報を提供し、インシデントをとおしてコラボレーションを行うための単一のインターフェイスを提供します。このサービスは AWS Systems Manager Automations を使用して検出と復旧を迅速化します。

顧客の事例

AnyCompany Retail で製造上の問題が発生しました。オンコールエンジニアは、プレイブックを使用して問題を調査しました。調査を進める中で、AnyCompany Retail はプレイブックに記載さ

れている主要な関係者と情報を共有し続けました。エンジニアは、根本原因がバックエンドサービス内の競合状態であることを特定しました。エンジニアはランブックを使用してサービスを再起動し、AnyCompany Retail をオンライン状態に戻しました。

実装手順

既存のドキュメントリポジトリがない場合、プレイブックライブラリ用のバージョン管理リポジトリを作成することをお勧めします。プレイブックは Markdown を使用して作成できます。Markdown は、ほとんどのプレイブック自動化システムとの互換性を持っています。プレイブックを一から作成する場合、以下のプレイブックテンプレートの例を使用します。

```
# ########## ## ######## | ###### ID | ## | ######## | #####
 | ########## | ##### | ######## POC | ### | ############ |
 |-------|-------|-------|-------|-------|-------|-------|-------|-------| | RUN001 | #
###########? ##############? | ### | ## | ##### | 2022-09-21 | ######### | #### | #####
#########? | ## ## 1.## 1 2## 2
```

1. 既存のドキュメントリポジトリや Wiki がない場合は、バージョン管理システムにプレイブック用の新しいバージョン管理リポジトリを作成します。
2. 調査が必要な一般的な問題を特定します。根本原因がいくつかの問題に絞られており、解決策が低リスクであるシナリオを選んでください。
3. Markdown テンプレートを使用して、 ############## プレイブック情報以下の ########## #。
4. トラブルシューティング手順を入力します。実行すべきアクション、または調査すべき領域をできるだけ明確に記載します。
5. プレイブックをチームメンバーに渡して、内容を確認してもらいます。記載漏れや不明瞭な記載がある場合、プレイブックを更新します。
6. プレイブックをドキュメントリポジトリに公開し、チームと関係者に通知します。
7. このプレイブックライブラリは、追加のプレイブックによって拡大します。いくつかのプレイブックを作成したら、AWS Systems Manager Automations などのツールを使用して自動化を開始し、自動化とプレイブックの同期を維持します。

実装計画に必要な工数レベル: 低。プレイブックは、一元的に保管されるテキストドキュメントとして作成します。組織が成熟するにしたがって、プレイブックの自動化に移行します。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する: プレイブックには、保守を担当する所有者が必要です。
- OPS07-BP03 ランブックを使用して手順を実行する: ランブックとプレイブックは似ていますが、重要な違いが 1 つあり、それはランブックには期待される成果があることです。多くの場合、ランブックは、プレイブックで根本原因を特定したときに使用されます。
- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する: プレイブックは、イベント、インシデント、および問題管理の適切な実践の一部です。
- OPS10-BP02 アラートごとにプロセスを用意する: ランブックとプレイブックは、アラートへの応答として使用されます。時間の経過とともに、これらの応答を自動化します。
- OPS11-BP04 ナレッジ管理を実施する: プレイブックの保守は、ナレッジマネジメントの重要な一部です。

関連するドキュメント:

- 自動化されたプレイブックとランブックを使用して運用上の優秀性を達成する
- AWS Systems Manager: ランブックの操作
- AWS Systems Manager Automation ランブックを使用して、運用タスクを解決する

関連動画:

- AWS re:Invent 2019: ランブック、インシデントレポート、インシデント対応の DIY ガイド (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS 仮想ワークショップ
- スクリプトを AWS Systems Manager に統合する

関連サンプル:

- AWS カスタムプレイブックフレームワーク
- AWS Systems Manager: オートメーションチュートリアル
- Jupyter Notebook と CloudTrail Lake を使用して、AWS インシデント対応ランブックを作成する
- Rubix - Jupyter Notebook でランブックを作成するための Python ライブラリ
- Document Builder を使用してカスタムランブックを作成する
- Well-Architected ラボ: プレイブックとランブックによるオペレーションの自動化

- Well-Architected ラボ: Jupyter を使用したインシデント対応プレイブック

関連サービス:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager を使用して

OPS07-BP05 システムや変更をデプロイするために十分な情報に基づいて決定を下す

ワークロードと、ワークロードのガバナンスとのコンプライアンスをサポートするためにチームの能力を評価します。システムを移行するか、本番環境に変更するかどうかを判断する際に、これらをデプロイの利点に対して評価します。メリットとリスクを理解し、十分な情報に基づく決定を下します。

プレモータムは、チームが行う演習で、ここでは軽減戦略を策定するために障害のシミュレーションを行います。プレモータムを使用して、障害を予測し、必要に応じて手順を作成します。ワークロードを評価するために使用するチェックリストに変更を加える場合は、もう準拠していない本番システムで行うことを計画します。

一般的なアンチパターン:

- ワークロードに存在するセキュリティリスクを理解することなく当該ワークロードのデプロイを決定する。
- ワークロードがガバナンスと標準に準拠しているかどうかを理解することなく当該ワークロードのデプロイを決定する。
- チームがサポートできるかどうかを理解することなくワークロードのデプロイを決定する。
- 組織にどのようなメリットがあるかを理解することなくワークロードのデプロイを決定する。

このベストプラクティスを活用するメリット: スキルのあるチームメンバーを持つことで、ワークロードを効果的にサポートできます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- ワークロードや変更をデプロイするために十分な情報に基づいて決定を下す: ワークロードと、ワークロードのガバナンスとのコンプライアンスをサポートするためにチームの能力を評価しま

す。システムを移行するか、本番環境に変更するかどうかを判断する際に、これらをデプロイの利点に対して評価します。メリットとリスクを理解し、十分な情報に基づく決定を下します。

運用

質問

- OPS 8 ワークロードの正常性はどのように把握するのですか?
- OPS 9 オペレーションの正常性をどのように把握するのですか?
- OPS 10 ワークロードと運用イベントはどのように管理するのですか?

OPS 8 ワークロードの正常性はどのように把握するのですか?

ワークロードメトリクスの定義、キャプチャ、分析をすると、適切なアクションを取れるようにワークロードイベントの可視性を高めることができます。

ベストプラクティス

- OPS08-BP01 主要業績評価指標 (KPI) を特定する
- OPS08-BP02 ワークロードのメトリクスを定義する
- OPS08-BP03 ワークロードメトリクスを収集および分析する
- OPS08-BP04 ワークロードメトリクスのベースラインを設定する
- OPS08-BP05 ワークロードに対して予想されるアクティビティのパターンを知る
- OPS08-BP06 ワークロードの結果にリスクがある場合に警告する
- OPS08-BP07 ワークロードの異常が検出された場合に警告する
- OPS08-BP08 KPI とメトリクスの成果の達成度と有効性を検証する

OPS08-BP01 主要業績評価指標 (KPI) を特定する

希望するビジネス上の業績 (注文率、顧客定着率、営業費用に対する利益など) と顧客に関する成果 (顧客満足度など) に基づいて、主要業績評価指標 (KPI) を特定します。KPI を評価して、ワークロードの成功を判別します。

一般的なアンチパターン:

- あなたは、ビジネスリーダーから、ワークロードがビジネスニーズにどのように寄与したかを尋ねられますが、寄与度を判断するための基準となる枠組みがありません。

- あなたは、組織で運用している商用オフザシェルフのアプリケーションの費用対効果が高いかどうかを判断することはできません。

このベストプラクティスを活用するメリット: ワークロードの正常性と成功のテストとして、主要業績評価指標を特定することで、ビジネス成果を達成できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 主要業績評価指標 (KPI) を特定する: ビジネスおよび顧客が求める成果に基づいて、主要業績評価指標 (KPI) を特定します。KPI を評価して、ワークロードの成功を判別します。

OPS08-BP02 ワークロードのメトリクスを定義する

KPI の達成度を測定するワークロードメトリクスを定義します (たとえば、中止されたショッピングカート、注文数、コスト、価格、配分されたワークロード費用)。ワークロードの正常性 (インターフェイスの応答時間、エラー率、リクエスト数、完了したリクエスト、使用率など) を測定するワークロードのメトリクスを定義します。メトリクスを評価して、ワークロードが必要な成果に達しているかを判定し、ワークロードの正常性を把握します。

CloudWatch Logs などのサービスにログデータを送信し、必要なログコンテンツの観察からメトリクスを生成する必要があります。

CloudWatch には、 .NET や SQL Server のための Amazon CloudWatch Insights および Container Insights などの専門的な機能があり、 サポートされている特定のアプリケーションリソースやテクノロジースタック全体で重要なメトリクス、ログ、アラームを特定して設定するのに役立ちます。

一般的なアンチパターン:

- あなたは、KPI に関連付けられていない、またはワークロードに合わせて調整されていない「標準」のメトリクスを定義しました。
- メトリクスの計算にエラーがあり、無効な結果が生成されます。
- あなたは、ワークロードに対して定義されたメトリクスを備えていません。
- あなたは可用性のみを測定します。

このベストプラクティスを活用するメリット: ワークロードメトリクスを定義して評価することで、ワークロードの状態を判断し、ビジネス成果の達成を測定できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ワークロードのメトリクスを定義する: KPI の達成度を測定するため、ワークロードのメトリクスを定義します。ワークロードとその個々のコンポーネントの正常性を測定するため、ワークロードのメトリクスを定義します。メトリクスを評価して、ワークロードが必要な成果に達しているかを判定し、ワークロードの正常性を把握します。
  - カスタムメトリクスをパブリッシュする
  - ログデータの検索およびフィルタリング
  - Amazon CloudWatch のメトリクスとディメンションのリファレンス

リソース

関連するドキュメント:

- Amazon CloudWatch のメトリクスとディメンションのリファレンス
- カスタムメトリクスをパブリッシュする
- ログデータの検索およびフィルタリング

OPS08-BP03 ワークロードメトリクスを収集および分析する

メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。

アプリケーション、ワークロードコンポーネント、サービス、および CloudWatch Logs などのサービスへの API 呼び出しのログデータを集約する必要があります。必要なログコンテンツの観測からメトリクスを生成して、運用アクティビティのパフォーマンスを把握できるようにします。

AWS では、 Amazon DevOps Guru の機械学習機能を使用して、ワークロードのメトリクスを分析し、運用の問題を特定できます。AWS DevOps Guru は、問題を解決しアプリケーションの状態を良好に保つための 対象を定めたプロアクティブな推奨事項を含む 運用の問題に関する通知を提供します。

AWS の責任共有モデルでは、モニタリングの一部が AWS Health Dashboard を通じて提供されます。このダッシュボードは、お客様に影響を与える可能性があるイベントが AWS で発生した場合に、アラートと修正ガイダンスを提供します。ビジネスサポートとエンタープライズサポートのサブ

スクリプションをご利用のお客様は、 AWS Health API にアクセスすることもでき、イベント管理システムとの統合が可能になります。

AWS では、 ログデータを Amazon S3 に エクスポートしたり、 ログ を Amazon S3 に直接送信して、 長期保存したりできます。分析のために、 AWS Glueを使用すると、Amazon S3 でログデータを検出して準備し、関連するメタデータを AWSAWS Glue Data Catalog に保存できます。 Amazon Athena では、AWS Glue とのネイティブな統合により、ログデータを分析し、標準 SQL を使用してクエリを実行できます。Amazon QuickSight などのビジネスインテリジェンスツールを 使用して、データを可視化、調査、分析することができます。

代替 ソリューション は、 Amazon OpenSearch Service および OpenSearch Dashboards を使用して、複数のアカウントと AWS リージョン にわたる AWS のログを収集、分析、表示することです。

一般的なアンチパターン:

- あなたは、ネットワーク設計チームから現在のネットワーク帯域幅使用率について尋ねられています。あなたは、現在のメトリクスを提供します。ネットワーク使用率は 35% です。当該チームは、コスト削減手段として回路容量を削減します。あなたのポイントインタイム測定では利用率の傾向が反映されず、接続に関する問題が広がってしまいます。
- ルーターに障害が発生しました。これまでに、重大ではないメモリエラーがログ記録されていました。その頻度はますます多くなり、ついには完全な障害となりました。あなたは、この傾向に気付かなかったため、ルーターがサービスの中断を引き起こす前に、障害のあるメモリを交換しませんでした。

このベストプラクティスを確立するメリット: ワークロードメトリクスを収集して分析することで、ワークロードの状態を把握し、ワークロードやビジネス成果の達成に影響を与える可能性のある傾向について洞察を得ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- ワークロードのメトリクスを収集および分析する: メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。
  - Amazon CloudWatch メトリクスを使用する
  - Amazon CloudWatch のメトリクスとディメンションのリファレンス
  - CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する

リソース

関連するドキュメント:

- Amazon Athena
- Amazon CloudWatch のメトリクスとディメンションのリファレンス
- Amazon DevOps Guru
- AWS Glue
- AWSAWS Glue Data Catalog に保存できます
- Amazon OpenSearch Service
- AWS Health Dashboard
- Amazon QuickSight
- CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する
- Amazon CloudWatch メトリクスを使用する

OPS08-BP04 ワークロードメトリクスのベースラインを設定する

メトリクスにベースラインを設定し、パフォーマンスの低いコンポーネントと高いコンポーネントを比較、特定するためのベースラインとして期待値を提供します。改善、調査、および介入のしきい値を特定します。

一般的なアンチパターン:

- サーバーが 95% の CPU 使用率で実行されており、あなたは、それが良いのか悪いのかを尋ねられています。そのサーバーの CPU 使用率がベースライン化されていないため、あなたは、それが良いのか悪いのかがわかりません。

このベストプラクティスを確立するメリット: ベースラインメトリクス値を定義することで、現在のメトリクス値とメトリクスの傾向を評価し、アクションが必要かどうかを判断できます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- ワークロードメトリクスのベースラインを設定する: ワークロードメトリクスのベースラインを設定し、比較の基準となる期待値を提供します。

- Amazon CloudWatch でのアラームの作成

リソース

関連するドキュメント:

- Amazon CloudWatch でのアラームの作成

OPS08-BP05 ワークロードに対して予想されるアクティビティのパターンを知る

ワークロードアクティビティのパターンを確立して異常な動作を識別し、必要に応じて適切に対応できるようにします。

CloudWatch Anomaly Detection 機能を使用した CloudWatch は、 統計および機械学習アルゴリズムを適用して、通常のメトリクスの動作を表す予想値の範囲を生成します。

Amazon DevOps Guru を使用して、 イベントの関連性、ログ分析、ワークロードテレメトリー分析への機械学習の適用によって、異常な動作を検出できます。予期しない動作が検出された場合、 当該動作に対処するためのレコメンデーションを含む 関連メトリクスとイベントを提供します。

一般的なアンチパターン:

- あなたは、ネットワーク使用率のログを確認しています。このとき、ネットワーク使用率が午前 11 時 30 分から午後 1 時 30 分まで増加し、午後 4 時 30 分から午後 6 時 00 分まで再び増加していることに気づきます。あなたには、これを正常とみなすべきかどうかがわかりません。
- ウェブサーバーは、毎日午前 3 時に再起動します。あなたには、これが予期される動作であるかどうかがわかりません。

このベストプラクティスを活用するメリット: 動作パターンを学習することで、予期しない動作を認識し、必要に応じてアクションを実行できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ワークロードに対して予想されるアクティビティのパターンを知る: ワークロードアクティビティのパターンを確立すると、行動が期待値から外れるタイミングを把握し、必要に応じて適切に対応できるようになります。

リソース

関連するドキュメント:

- Amazon DevOps Guru
- CloudWatch Anomaly Detection

OPS08-BP06 ワークロードの結果にリスクがある場合に警告する

ワークロードの結果にリスクがある場合、必要に応じて適切な対応ができるよう、アラートを発生させます。

理想的には、警告の対象となるメトリクスのしきい値、または自動応答をトリガーするために使用できるイベントを前もって指定しておきます。

AWS では、 Amazon CloudWatch Synthetics を使用して 顧客と同じアクションを実行することで、エンドポイントと API をモニタリングするための Canary スクリプトを作成できます。生成されたテレメトリーと 得られたインサイトを使用して、 顧客が影響を受ける前に問題を特定できます。

また、 CloudWatch Logs Insights を使用して、 専用のクエリ言語により ログデータをインタラクティブに検索および分析することもできます。CloudWatch Logs Insights は、 AWS のサービスおよび JSON のカスタムログイベントからの ログのフィールドを自動的に検出します。ログボリュームとクエリの複雑さに応じてスケールし、数秒で回答が得られるため、インシデントの原因となる要因の検索に役立ちます。

一般的なアンチパターン:

- ネットワーク接続がありません。誰も気づいていません。理由を特定しようとしたり、接続を復元するためのアクションを採ろうとしたりする人はいません。
- パッチの適用後、永続的なインスタンスが使用できなくなり、ユーザーの操作を中断します。ユーザーがサポートケースをオープンしました。誰にも通知されていません。アクションを採ろうとしている人はいません。

このベストプラクティスを活用するメリット: ビジネス上の成果にリスクがあることを特定し、アクションが実行されるべきことをアラートすることで、インシデントの影響を防止または軽減する機会を得られます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ワークロードの結果にリスクがある場合にアラートを出す: ワークロードの結果にリスクがある場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
  - Amazon CloudWatch Events とは
  - Amazon CloudWatch アラームの作成
  - Amazon SNS 通知を使用した Lambda 関数の呼び出し

リソース

関連するドキュメント:

- Amazon CloudWatch Synthetics
- CloudWatch Logs Insights
- Amazon CloudWatch アラームの作成
- Amazon SNS 通知を使用した Lambda 関数の呼び出し
- Amazon CloudWatch Events とは

OPS08-BP07 ワークロードの異常が検出された場合に警告する

ワークロードの異常が検出された場合、必要に応じて適切な対応ができるよう、アラートを発生させます。

時間をかけてワークロードメトリクスを分析すると、イベントを定義したり、それに応じてアラームを発生させるために十分に定量化できる行動パターンが確立されることがあります。

トレーニングが完了すると、 CloudWatch Anomaly Detection 機能を使用して、 検出された異常を警告したり、 メトリクスデータのグラフに重ねて 予想値を渡して 継続的な比較を行うことができます。

一般的なアンチパターン:

- 小売業のウェブサイトの売上が急激かつ劇的に増加しました。誰も気づいていません。誰もこの急増の原因を特定しようとしていません。負荷が増える中で、質の高いカスタマーエクスペリエンスを確保するための措置を講じている人はいません。
- パッチの適用後、永続的なサーバーが頻繁に再起動し、ユーザーの操作を中断します。サーバーは通常、最大 3 回まで再起動しますが、それを超えては再起動しません。誰も気づいていません。これが生じている理由を誰も特定しようとしていません。

このベストプラクティスを活用するメリット: ワークロードの動作パターンを理解することで、予期しない動作を特定し、必要に応じてアクションを実行できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- ワークロードの異常が検出された場合にアラートを出す: ワークロードの異常が検出された場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
  - Amazon CloudWatch Events とは
  - Amazon CloudWatch アラームの作成
  - Amazon SNS 通知を使用した Lambda 関数の呼び出し

リソース

関連するドキュメント:

- Amazon CloudWatch アラームの作成
- CloudWatch Anomaly Detection
- Amazon SNS 通知を使用した Lambda 関数の呼び出し
- Amazon CloudWatch Events とは

OPS08-BP08 KPI とメトリクスの成果の達成度と有効性を検証する

ワークロードオペレーションに対するビジネスレベルの視点を確立すると、ニーズを満足しているかどうかを判断したり、ビジネス目標を達成するために改善が必要な領域を特定したりできます。KPI とメトリクスの有効性を検証し、必要に応じて修正します。

また、AWS は、AWS のサービス API や SDK を介してサードパーティーのログ分析システム (Grafana、Kibana、Logstash など) やビジネスインテリジェンスツールもサポートしています。

一般的なアンチパターン:

- これまで、ページの応答時間は、顧客満足度に対する貢献要因であると考えられたことはありません。あなたは、ページの応答時間のメトリクスまたはしきい値を確立したことがありません。顧客は、「遅さ」について苦情を申し立てています。

- あなたは、最小応答時間の目標を達成したことはありません。あなたは、応答時間を短縮するために、アプリケーションサーバーをスケールアップしました。応答時間の目標を大幅に超過することになり、また、支払っている容量の未使用部分も大幅に増加しました。

このベストプラクティスを活用するメリット: KPI とメトリクスを確認して修正することで、ワークロードがビジネス成果の達成をどのようにサポートしているかを理解し、ビジネス目標を達成するために改善が必要な場所を特定できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- KPI とメトリクスの成果の達成度と有効性を検証する: ワークロード運用に対するビジネスレベルの視点を確立すると、ニーズを満たしているかどうかを判断したり、ビジネス目標を達成するために改善が必要な領域を特定したりできます。KPI とメトリクスの有効性を検証し、必要に応じて修正します。
  - Amazon CloudWatch ダッシュボードの使用
  - ログ分析とは

リソース

関連するドキュメント:

- Amazon CloudWatch ダッシュボードの使用
- ログ分析とは

OPS 9 オペレーションの正常性をどのように把握するのですか?

オペレーションメトリクスを定義し、キャプチャし、分析することで、オーペレーションイベントの可視性を高め、適切なアクションがとれるようになります。

ベストプラクティス

- OPS09-BP01 主要業績評価指標 (KPI) を特定する
- OPS09-BP02 運用メトリクスを定義する
- OPS09-BP03 運用メトリクスを収集し、分析する
- OPS09-BP04 運用メトリクスの基準値を設定する

- OPS09-BP05 運用に対して予想されるアクティビティのパターンを知る
- OPS09-BP06 オペレーションの結果にリスクがある場合に警告する
- OPS09-BP07 運用の異常が検出された場合に警告する
- OPS09-BP08 KPI とメトリクスの成果の達成度と有効性を検証する

OPS09-BP01 主要業績評価指標 (KPI) を特定する

希望するビジネスの成果 (新機能など) とお客様の成果 (カスタマーサポートケースなど) に基づいて、主要業績指標 (KPI) を特定します。KPI を評価して、オペレーションの成功を判別します。

一般的なアンチパターン:

- あなたは、ビジネスリーダーから、オペレーションがビジネス目標をどのように達成したかを尋ねられますが、寄与度を判断するための基準となる枠組みがありません。
- あなたは、メンテナンスウィンドウがビジネスの成果に影響しているかどうかを判断できません。

このベストプラクティスを活用するメリット: オペレーションの正常性と成功のテストとして、主要業績評価指標を特定することで、ビジネス成果を達成できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 主要業績評価指標 (KPI) を特定する: ビジネスおよび顧客が求める成果に基づいて、主要業績評価指標 (KPI) を特定します。KPI を評価して、オペレーションの成功を判別します。

OPS09-BP02 運用メトリクスを定義する

運用メトリクスを定義して、KPI の達成度 (デプロイの成功、失敗したデプロイなど) を測定します。運用アクティビティの正常性を測定する運用メトリクスを定義します (たとえば、インシデントを検出する平均時間 (MTTD)、インシデントからの平均復旧時間 (MTTR) など)。メトリクスを評価して、運用アクティビティが必要な成果に達しているかを判定し、運用の正常性を把握します。

一般的なアンチパターン:

- 運用メトリクスは、チームが合理的であると考える内容に基づいています。
- メトリクスの計算にエラーがあり、誤った結果が生成されます。
- あなたは、運用アクティビティに対して定義されたメトリクスを備えていません。

このベストプラクティスを活用するメリット: 運用メトリクスを定義して評価することで、運用アクティビティの状態を判断し、ビジネス成果の達成を測定できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 運用メトリクスを定義する: KPI の達成度を測定するため、運用メトリクスを定義します。運用メトリクスを定義して、運用とそのアクティビティの正常性を測定します。メトリクスを評価して、オペレーションが必要な成果に達しているかを判定し、オペレーションの正常性を把握します。
  - カスタムメトリクスをパブリッシュする
  - ログデータの検索およびフィルタリング
  - Amazon CloudWatch のメトリクスとディメンションのリファレンス

リソース

関連するドキュメント:

- AWS Answers: 集中ログ記録
- Amazon CloudWatch のメトリクスとディメンションのリファレンス
- Amazon CloudWatch Events を使用してパイプラインの状態変化を検出し対処する
- カスタムメトリクスをパブリッシュする
- ログデータの検索およびフィルタリング

関連動画:

- モニタリング計画を立てる

OPS09-BP03 運用メトリクスを収集し、分析する

メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。

運用アクティビティと運用 API コールの実行から、CloudWatch Logs などのサービスにログデータを集約する必要があります。必要なログコンテンツの観測からメトリクスを生成して、運用アクティビティのパフォーマンスのインサイトを得られるようにします。

AWS では、 ログデータを Amazon S3 に エクスポートしたり、 ログ を Amazon S3 に直接送信して、 長期保存したりできます。分析のために、 AWS Glueを使用すると、Amazon S3 でログデータを検出して準備し、関連するメタデータを AWSAWS Glue Data Catalog に保存できます。 Amazon Athena では、AWS Glue とのネイティブな統合により、ログデータを分析し、標準 SQL を使用してクエリを実行できます。Amazon QuickSight などのビジネスインテリジェンスツール を使用して、データを可視化、調査、分析することができます。

一般的なアンチパターン:

- 新機能の継続的な提供は、主要業績評価指標とみなされます。あなたは、デプロイの発生頻度を測定する方法を備えていません。
- あなたは、デプロイ、ロールバックされたデプロイ、パッチ、およびロールバックされたパッチをログに記録して、運用アクティビティを追跡しますが、メトリクスを確認する人はいません。
- あなたは、システムがデプロイされ、ユーザーがいないときに、15 分間の復旧時間以内に失われたデーベースを復旧することを目標として課せられました。このシステムは 2 年間運用されており、現在 10,000 人のユーザーがいます。最近の復旧には 2 時間以上かかりました。これは記録されておらず、誰も認識していません。

このベストプラクティスを活用するメリット: 運用メトリクスを収集して分析することで、運用の状態を把握し、運用やビジネス成果の達成に影響を与える可能性のある傾向について洞察を得ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 運用のメトリクスを収集および分析する: メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。
  - Amazon CloudWatch メトリクスの使用
  - Amazon CloudWatch のメトリクスとディメンションのリファレンス
  - CloudWatch エージェントを使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集

リソース

関連するドキュメント:

- Amazon Athena

- Amazon CloudWatch のメトリクスとディメンションのリファレンス
- Amazon QuickSight
- AWS Glue
- AWSAWS Glue Data Catalog に保存できます
- CloudWatch エージェントを使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集
- Amazon CloudWatch メトリクスの使用

OPS09-BP04 運用メトリクスの基準値を設定する

運用アクティビティのパフォーマンスを比較し、過不足を特定する基準となる期待値として、メトリクスに対する基準値を設定します。

一般的なアンチパターン:

- あなたは、デプロイに必要な時間について尋ねられました。あなたは、デプロイにかかる時間を測定しておらず、予想時間を判断できません。
- あなたは、アプリケーションサーバーの問題から復旧するまでにどれくらいの時間がかかるかを尋ねられました。あなたは、顧客の最初の連絡から復旧までの時間についての情報を持っていません。あなたは、モニタリングを通じて問題を最初に特定した時点から復旧までの時間についての情報を持っていません。
- あなたは、週末に必要となるサポート担当者の数を尋ねられました。あなたは、週末に発生するサポートケースの一般的な数がわからず、予測を示すことができません。
- あなたは、システムがデプロイされ、ユーザーがいないときに、15 分間の復旧時間以内に失われたデーベースを復旧することを目標として課せられました。このシステムは 2 年間運用されており、現在 10,000 人のユーザーがいます。あなたは、データベースの時間がどのように変化してきているかについての情報を持っていません。

このベストプラクティスを活用するメリット: ベースラインメトリクス値を定義することで、現在のメトリクス値とメトリクスの傾向を評価し、アクションが必要かどうかを判断できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 運用に対して予想されるアクティビティのパターンを知る: 運用アクティビティのパターンを確立すると、動作が期待値から外れるタイミングを把握し、必要に応じて適切に対応できるようになります。

OPS09-BP05 運用に対して予想されるアクティビティのパターンを知る

運用アクティビティのパターンを確立して異常なアクティビティを識別し、必要に応じて適切に対応できるようにします。

一般的なアンチパターン:

- あなたのデプロイ失敗率は、最近大幅に増加しました。あなたは、各障害に個別に対処しています。あなたは、失敗が、デプロイ管理システムに慣れていない新入社員によるデプロイに原因があることに気づいていません。

このベストプラクティスを活用するメリット: 動作パターンを学習することで、予期しない動作を認識し、必要に応じてアクションを実行できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 運用に対して予想されるアクティビティのパターンを知る: 運用アクティビティのパターンを確立すると、動作が期待値から外れるタイミングを把握し、必要に応じて適切に対応できるようになります。

OPS09-BP06 オペレーションの結果にリスクがある場合に警告する

オペレーションの結果にリスクがある場合は、アラートを送信し、それに対処する必要があります。オペレーションの結果とは、本番稼働のワークロードをサポートするすべてのアクティビティを指します。これには、アプリケーションの新しいバージョンのデプロイから障害の復旧に至るまで、すべてのアクティビティが含まれます。オペレーションの結果は、ビジネスの成果と同等の重要度で対応する必要があります。

ソフトウェアチームは、主要なオペレーションメトリクスとアクティビティを特定し、それらに対するアラートを構築する必要があります。適切なタイミングでアラートを送信し、アラートは実行可能なものである必要があります。アラートを送信する際は、対応するランブックまたはプレイブックへ

の参照を含める必要があります。対応するアクションがないアラートは、以下につながる可能性があります。

期待される成果: オペレーションのアクティビティにリスクがある場合、アクションを促すためのアラートが送信されます。このアラートにはアラートが送信された理由のコンテキストが含まれます。また、調査を行うためのプレイブック、またはアラートを緩和するためのランブックへの参照が含まれます。可能な場合、ランブックは自動化し、通知を送信します。

一般的なアンチパターン:

- インシデントの調査が行われ、サポートケースが作成されています。サポートケースはサービスレベルアグリーメント (SLA) に違反していますが、アラートは送信されていません。
- 本番稼働へのデプロイは深夜に予定されていましたが、直前の変更によりデプロイが遅れました。アラートは送信されず、デプロイは完了しませんでした。
- 本番稼働に障害が発生しましたが、アラートは送信されませんでした。
- デプロイ時間には頻繁に遅延が生じています。調査は行われていません。

このベストプラクティスを活用するメリット:

- オペレーションの結果にリスクがある場合にアラートを送信することで、問題が発生する前にワークロードをサポートする能力を高めることができます。
- 正常なオペレーションの結果によって、ビジネス成果を改善できます。
- オペレーションの問題の検出と問題への対応を改善できます。
- 全体的なオペレーションの正常性が向上します。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

オペレーションの結果に関するアラートを送信する前に、オペレーションの結果を定義する必要があります。組織にとって最も重要なオペレーションアクティビティを定義することから始めます。それは、本番稼働へのデプロイを 2 時間以内に終えることでしょうか？それとも、決められた時間内にサポートケースに対応することでしょうか？ 組織は、主要なオペレーションアクティビティを監視、改善、およびそれらに関するアラートを送信するために、主要なオペレーションアクティビティとそれらを計測する方法を定義する必要があります。ワークロードとオペレーションのテレメトリを保存し分析するための、一元的な場所が必要です。同じ仕組みを使用して、オペレーションの結果にリスクがある場合に、アラートを送信します。

顧客の事例

AnyCompany Retail のルーティンデプロイ中に CloudWatch アラームがトリガーされました。デプロイのリードタイムは予定を超えてしまい、Amazon EventBridge によって AWS Systems Manager OpsCenter で OpsItem が作成されました。クラウドオペレーションチームは、プレイブックを使用して問題を調査し、スキーマの変更が想定よりも長くかかっていることを特定しました。彼らはオンコール開発者にアラートを送信し、デプロイの監視を続けました。デプロイの完了後、クラウドオペレーションチームは OpsItem を解決しました。チームは事後分析でインシデントを分析する予定です。

実装手順

1. オペレーションの KPI、メトリクス、およびアクティビティを特定していない場合、この質問に対するベストプラクティスを採用します (OPS09-BP01 から OPS09-BP05)。
   - エンタープライズサポートのある AWS Support カスタマーは、テクニカルアカウントマネージャーから オペレーション KPI ワークショップを リクエストできます 。このコラボレーティブワークショップは、ビジネス目標に沿ったオペレーション KPI やメトリクスの定義に役立ちます。追加の費用はありません。詳細については、担当のテクニカルアカウントマネージャーにお問い合わせください。
2. オペレーションアクティビティ、KPI、メトリクスの確立後、可観測性プラットフォームでアラートを構成します。プレイブックやランブックと同様に、アラートには関連するアクションが必要です。アクションを伴わないアラートは避けてください。
3. 時間の経過とともに、オペレーションメトリクス、KPI、アクティビティを評価し、改善の領域を特定します。フィードバックをオペレーターからのランブックとプレイブックに反映し、アラートへの対応を改善できる領域を特定します。
4. アラートには、アラートを誤検出としてフラグを付けるための仕組みを含める必要があります。これは、メトリクスのしきい値のレビューにつながります。

実装計画に必要な工数レベル: 中。このベストプラクティスを採用する前に、採用が必要ないくつかのベストプラクティスがあります。オペレーターアクティビティを特定し、オペレーション KPI を確立したら、アラートを作成します。

リソース

関連するベストプラクティス:

- OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する: すべてのオペレーションアクティビティおよび結果に対して責任を持つ所有者を特定する必要があります。オペレーションの結果にリスクがある場合、アラートは所有者に送信されます。
- OPS03-BP02 チームメンバーに、結果にリスクがあるときにアクションを実行する権限が付与されている: チームは、アラートが送信された際に問題に対応するエージェンシーを用意する必要があります。
- OPS09-BP01 主要業績評価指標 (KPI) を特定する: オペレーションの結果に関するアラートは、オペレーション KPI を特定することから開始されます。
- OPS09-BP02 運用メトリクスを定義する: アラートを生成し始める前に、このベストプラクティスを確立します。
- OPS09-BP03 運用メトリクスを収集し、分析する: アラートを構築するには、オペレーションメトリクスを一元的に収集する必要があります。
- OPS09-BP04 運用メトリクスの基準値を設定する: オペレーションメトリクスは、アラートを調整しアラート疲れを防ぐための基準を提供します。
- OPS09-BP05 運用に対して予想されるアクティビティのパターンを知る: オペレーションイベントのアクティビティパターンを理解することで、アラートの精度を高められます。
- OPS09-BP08 KPI とメトリクスの成果の達成度と有効性を検証する: オペレーションの結果の成果を評価して、KPI とメトリクスが正しいことを確認します。
- OPS10-BP02 アラートごとにプロセスを用意する: すべてのアラートには関連するランブックまたはプレイブックが必要で、アラートを受け取る担当者にコンテキストを提供する必要があります。
- OPS11-BP02 インシデント後の分析を実行する: アラート後の事後分析を実施し、改善の領域を特定します。

関連するドキュメント:

- AWS デプロイパイプラインリファレンスアーキテクチャ: アプリケーションパイプラインアーキテクチャ
- GitLab: アジャイル/DevOps メトリクスの使用を開始する

関連動画:

- AWS Systems Manager OpsCenter を使用したオペレーションの問題の収集と解決
- AWS Systems Manager OpsCenter と Amazon CloudWatch アラームの統合
- Amazon EventBridge を使用して AWS Systems Manager OpsCenter にデータソースを統合する

関連サンプル:

- Automate remediation actions for Amazon EC2 notifications and beyond using Amazon EC2 Systems Manager Automation and AWS Health
- AWS 管理とガバナンスツールワークショップ - オペレーション 2022
- AWS の DevOps モニタリングダッシュボードでメトリクスの取り込み、分析、視覚化を行う

関連サービス:

- Amazon EventBridge
- AWS Support プロアクティブサービス - オペレーション KPI ワークショップ
- AWS Systems Manager OpsCenter
- CloudWatch イベント

OPS09-BP07 運用の異常が検出された場合に警告する

運用の異常が検出された場合、必要に応じて適切な対応ができるよう、アラートを発生させます。

時間をかけて運用メトリクスを分析すると、イベントを定義したり、それに応じてアラームを発生させるために十分に定量化できる動作パターンが確立される可能性があります。

トレーニングが完了すると、 CloudWatch Anomaly Detection 機能を使用して、 検出された異常を警告したり、 メトリクスデータのグラフに重ねて 予想値を渡して 継続的な比較を行うことができます。

Amazon DevOps Guru を使用して、 イベントの関連性、ログ分析、ワークロードテレメトリー分析への機械学習の適用によって、異常な動作を検出できます。取得した インサイトは、 関連データとレコメンデーションとともに表示されます。

一般的なアンチパターン:

- あなたは、インスタンスのフリートにパッチを適用しようとしています。テスト環境では、パッチが正常にテストされました。フリート内のインスタンスの大部分でパッチが失敗しています。あなたは、何らのアクションも行っていません。
- あなたは、金曜日の終わりから始まるデプロイがあることに気づいています。組織は、火曜日と木曜日のメンテナンスウィンドウを事前定義しています。あなたは、何らのアクションも行っていません。

このベストプラクティスを活用するメリット: 運用の動作パターンを理解することで、予期しない動作を特定し、必要に応じてアクションを実行できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 運用の異常が検出された場合にアラートを出す: 運用の異常が検出された場合、必要に応じて適切な対応ができるよう、警告を発生させます。
  - Amazon CloudWatch Events とは
  - Amazon CloudWatch アラームの作成
  - Amazon SNS 通知を使用した Lambda 関数の呼び出し

リソース

関連するドキュメント:

- Amazon DevOps Guru
- CloudWatch Anomaly Detection
- Amazon CloudWatch アラームの作成
- Amazon CloudWatch Events を使用してパイプラインの状態変化を検出し対処する
- Amazon SNS 通知を使用した Lambda 関数の呼び出し
- Amazon CloudWatch Events とは

OPS09-BP08 KPI とメトリクスの成果の達成度と有効性を検証する

オペレーションアクティビティに対するビジネスレベルの視点を確立すると、ニーズを満足しているかどうかを判断したり、ビジネス目標を達成するために改善が必要な領域を特定したりできます。KPI とメトリクスの有効性を検証し、必要に応じて修正します。

また、AWS は、AWS のサービス API や SDK を介してサードパーティーのログ分析システム (Grafana、Kibana、Logstash など) やビジネスインテリジェンスツールもサポートしています。

一般的なアンチパターン:

- 開発チームの数が増えるにつれて、デプロイの頻度が増加しています。定義された予想デプロイ数は 1 週間に 1 回です。あなたは、毎日定期的にデプロイしています。デプロイシステムに問題があり、デプロイが不可能な場合、それが検出されるのは数日後です。

- 以前、あなたの会社では、月曜日から金曜日までの営業時間中にのみサポートを提供していました。あなたは、インシデントについて、「翌営業日」の応答時間の目標を設定しました。最近、あなたは、2 時間の応答時間の目標で 24 時間年中無休のサポートを提供し始めました。夜勤のスタッフは対応しきれておらず、顧客は不満を感じています。報告は「翌営業日」のターゲットに対して行われているので、インシデントへの応答時間に問題があることは示唆されていません。

このベストプラクティスを活用するメリット: KPI とメトリクスを確認して修正することで、ワークロードがビジネス成果の達成をどのようにサポートしているかを理解し、ビジネス目標を達成するために改善が必要な場所を特定できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- KPI とメトリクスの成果の達成度と有効性を検証する: 運用に対するビジネスレベルの視点を確立すると、ニーズを満たしているかどうかを判断したり、ビジネス目標を達成するために改善が必要な領域を特定したりできます。KPI とメトリクスの有効性を検証し、必要に応じて修正します。
  - Amazon CloudWatch ダッシュボードの使用
  - ログ分析とは

リソース

関連するドキュメント:

- Amazon CloudWatch ダッシュボードの使用
- ログ分析とは

OPS 10 ワークロードと運用イベントはどのように管理するのですか?

イベントに対応するための手順を準備、検証してワークロードの中断を最小限にします。

ベストプラクティス

- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する
- OPS10-BP02 アラートごとにプロセスを用意する
- OPS10-BP03 ビジネスへの影響に基づいて運用上のイベントの優先度を決定する
- OPS10-BP04 エスカレーション経路を決定する
- OPS10-BP05 プッシュ通知を有効にする

- OPS10-BP06 ダッシュボードでステータスを知らせる
- OPS10-BP07 イベントへの対応を自動化する

OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する

組織には、イベント、インシデント、問題を扱うためのプロセスがあります。イベント は、ワークロードで発生しますが、必ずしも介入を必要としない出来事です。インシデント は、介入を必要とするイベントです。 問題 は、介入しなければ解決できない、繰り返し発生するイベントです。これらのイベントがビジネスに与える影響を軽減し、適切に対応できるようにするためのプロセスが必要です。

ワークロードにインシデントや問題が発生したとき、それらを処理するためのプロセスが必要です。イベントの状況を関係者にどのように伝えますか。 対応をだれが監督しますか。 イベントの緩和に使用するツールは何ですか。 これらは、確実な対応策を講じるために必要な質問の一例です。

プロセスは一元的に文書化し、ワークロードに関わる誰もが利用できるようにする必要があります。もし、一元的な Wiki や ドキュメントの保管場所がない場合は、バージョン管理リポジトリを使用することができます。プロセスの進化につれて、これらのプランを最新に保ちます。

問題は、オートメーションの候補です。これらのイベントが発生すると、イノベーションにかける時間が奪われます。問題を軽減するための繰り返し可能なプロセスから始めましょう。時間をかけて、軽減策のオートメーション化または根本的な問題の解決に注力します。そうすることで、ワークロードの改善に充てる時間を確保することができます。

期待される成果: 組織には、イベント、インシデント、問題を扱うためのプロセスがあります。これらのプロセスは文書化され、一元的に保管されます。プロセスの変化につれて更新されます。

一般的なアンチパターン:

- インシデントが週末に発生すると、オンコールエンジニアはどうしてよいかわかりません。
- 顧客からアプリケーションがダウンしたという E メールが送られてきます。修正するためにサーバーを再起動します。これが頻繁に起きます。
- 複数のチームが解決のために個別に取り組んでいるインシデントがあります。
- ワークロードで、記録されることなくデプロイが行われます。

このベストプラクティスを活用するメリット:

- ワークロードのイベントの監査証跡ができます。

- インシデントからの復旧時間が短縮されます。
- チームメンバーは一貫した方法でインシデントと問題を解決できます。
- インシデントを調査するときに、より総合的に取り組むことができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

このベストプラクティスを実装すると、ワークロードイベントを追跡することになります。インシデントと問題を扱うためのプロセスができます。プロセスは文書化され、共有され、頻繁に更新されます。問題が特定され、優先順位が付けられ、修正されます。

顧客の事例

AnyCompany Retail では、社内 Wiki の一部がイベント、インシデント、問題管理のためのプロセス専用になっています。 すべてのイベントは以下に送信されます : Amazon EventBridge.問題は AWS Systems Manager OpsCenter で OpsItem として特定され、優先的に修正されるため、未分化な労力を削減することができます。プロセスが変化すると、社内 Wiki で更新されます。同社では、 AWS Systems Manager Incident Manager を使用してインシデントを管理し、緩和の取り組みを調整しています。

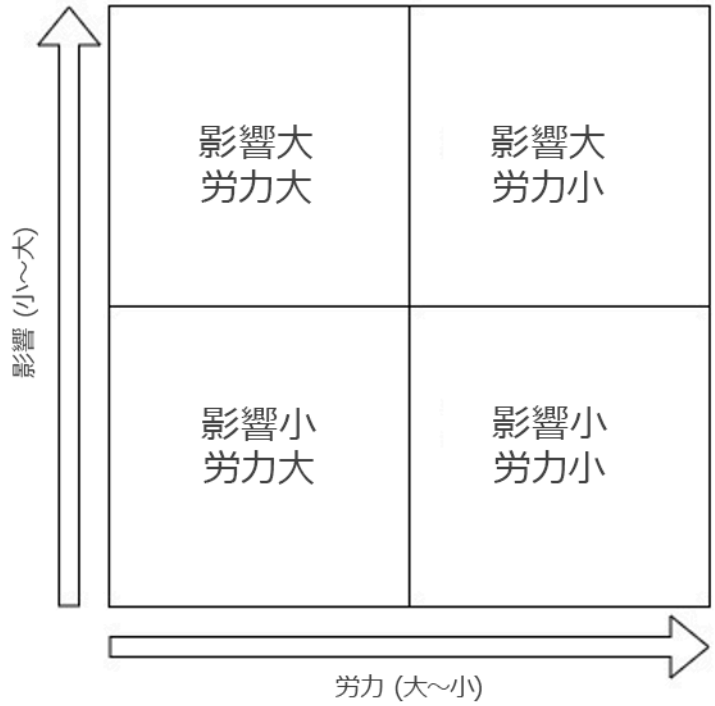
実装手順

1. イベント
   - 人間の介入を必要としない場合でも、ワークロードで発生するイベントを追跡します。
   - ワークロードの関係者と協力して、追跡すべきイベントのリストを作成します。例えば、デプロイの完了やパッチ適用の成功などです。
   - また、 Amazon EventBridge または Amazon Simple Notification Service などのサービスを使用して、追跡するカスタムイベントを生成することができます。
2. インシデント
   - インシデント発生時の情報伝達プランを明確にすることから始めましょう。どのような関係者に通知する必要がありますか。 どのようにして情報を共有しますか。 誰が調整作業を監督しますか。 コミュニケーションと調整のために、社内チャットチャネルを立ち上げることをお勧めします。
   - 特にオンコールのローテーションがないチームの場合は、ワークロードをサポートするチームのエスカレーションパスを定義しておきましょう。サポートレベルに基づいて、AWS Support に申請を行うことも可能です。

- インシデントを調査するためのプレイブックを作成します。これには、情報伝達プランや詳細な調査手順が含まれている必要があります。調査には、 AWS Health Dashboard の確認を含めます。
- インシデント対応計画を文書化します。インシデント管理計画を伝達し、社内外の顧客がエンゲージメントのルールと何を期待されているのかを理解できるようにします。使用方法について、チームメンバーをトレーニングします。
- 顧客は Incident Manager を使用してインシデント対応プランを設定し、管理できます。
- エンタープライズサポートのお客様は インシデント管理ワークショップ をテクニカルアカウントマネージャーからリクエストできます。このガイド付きワークショップでは、既存のインシデント対応計画をテストし、改善すべき点を明らかにすることができます。

3. 問題

- ITSM システムで問題を特定して、追跡する必要があります。
- 既知の問題をすべて特定し、修正作業とワークロードへの影響から優先順位を付けます。



- 影響が大きく、労力が少なくて済む問題から解決します。そのような問題が解決したら、影響が少なく、労力が少なくて済む象限の問題に移ります。
- 専用のインフラストラクチャで Systems Manager OpsCenter を使用して、これらの問題を特定し、ランブックをアタッチして、追跡します。

実装計画に必要な工数レベル: 中程度。このベストプラクティスを実装するには、プロセスとツールの両方が必要です。プロセスを文書化し、ワークロードに関わる誰もが使用できるようにします。頻繁に更新します。問題を管理し、問題を緩和または修正するためのプロセスがあります。

リソース

関連するベストプラクティス:

- OPS07-BP03 ランブックを使用して手順を実行する: 既知の問題には、緩和作業に一貫性を持たせるために、関連するランブックが必要です。
- OPS07-BP04 プレイブックを使用して問題を調査する: インシデントはプレイブックを使用して調査する必要があります。
- OPS11-BP02 インシデント後の分析を実行する: インシデントからの復旧後は、必ず事後分析を実施します。

関連するドキュメント:

- Atlassian - DevOps 時代のインシデント管理
- AWS セキュリティインシデント対応ガイド
- DevOps および SRE 時代のインシデント管理
- PagerDuty - インシデント管理とは

関連動画:

- AWS re:Invent 2020: 分散組織におけるインシデント管理
- AWS re:Invent 2021: イベント駆動型アーキテクチャによる次世代アプリケーションの構築
- AWS Supports You | インシデント管理の机上演習を試す
- AWS Systems Manager Incident Manager - AWS 仮想ワークショップ
- AWS What's Next ft. Incident Manager | AWS イベント

関連する例:

- AWS 管理とガバナンスツールワークショップ - OpsCenter
- AWS プロアクティブサービス - インシデント管理ワークショップ
- Amazon EventBridge によるイベント駆動型アプリケーションの構築

- AWS でのイベント駆動型アーキテクチャの構築

関連サービス:

- Amazon EventBridge
- Amazon SNS
- AWS Health Dashboard
- AWS Systems Manager Incident Manager
- AWS Systems Manager OpsCenter

OPS10-BP02 アラートごとにプロセスを用意する

アラートを発生させるイベントすべてに対して具体的な対応策 (ランブックやプレイブック) を定め、所有者を明確に指定しておくようにします。こうすることで、運用上のイベントに対する効果的で迅速な対応が可能になり、アクションの必要なイベントが重要度の低い通知に埋もれてしまうことを避けられます。

一般的なアンチパターン:

- モニタリングシステムは、他のメッセージとともに、承認された接続のストリームを表示します。メッセージの量が非常に大きいため、あなたは、介入を必要とする定期的な工ラーメッセージを見逃します。
- あなたは、ウェブサイトがダウンしている旨のアラートを受け取ります。このような状況が発生した場合のプロセスは定義されていません。あなたは、アドホックのアプローチで問題を診断して解決しなければなりません。対応に当たりながらこのプロセスを開発すると、復旧までの時間が長くなります。

このベストプラクティスを確立するメリット: アクションが必要な場合にのみアラートを出すことで、低い価値のアラートによって高い価値のアラートが見過ごされるのを防ぎます。アクション可能なアラートごとにプロセスを設定することで、環境内のイベントに対して一貫した迅速な対応を可能にします。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- アラートを発するイベントには、明確に定義された対応策 (ランブックまたはプレイブック) が必要であり、成功裏に完了する責任を負う所有者 (例えば、個人、チーム、役割) が明確に特定されていなければなりません。対応策が実際には自動で、または別のチームによって実行される場合でも、プロセスによって期待される成果を実現させる責任は所有者が持ちます。こうしたプロセスによって、運用上のイベントに対する効果的で迅速な対応が可能になり、アクションの必要なイベントが重要度の低い通知に埋もれてしまうことを避けられます。例えば、ウェブのフロントエンドをスケールする際にスケーリングが自動的に適用される場合でも、自動スケーリングのルールや制限をワークロードのニーズに適したものにすることは運用チームの責任になります。

リソース

関連するドキュメント:

- Amazon CloudWatch の特徴
- Amazon CloudWatch Events とは

関連動画:

- モニタリング計画を立てる

OPS10-BP03 ビジネスへの影響に基づいて運用上のイベントの優先度を決定する

介入を必要とする複数のイベントが発生したときに、ビジネスにとって最重要なものから対応できるようにしておきます。影響の例として、死亡や傷害、経済的損失、評判や信用の低下などがあります。

一般的なアンチパターン:

- あなたは、ユーザーのプリンター設定を追加するサポートリクエストを受け取ります。問題に対応している間に、あなたは、小売サイトがダウンしている旨のサポートリクエストを受け取ります。ユーザーのプリンター設定が完了したら、あなたは、ウェブサイトの問題の対応を開始します。
- あなたには、小売ウェブサイトと給与システムの両方が停止していることが通知されます。あなたは、どちらを優先すべきかわかりません。

このベストプラクティスを活用するメリット: ビジネスに最も大きな影響を与えるインシデントへの対応に優先順位を付けることで、その影響を管理できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ビジネスへの影響に基づき、運用イベントの優先度を決定する: 介入を必要とする複数のイベントが発生したときに、ビジネスにとって最も重要なものから対応できるようにしておきます。影響の例として、死亡や傷害、経済的損失、規定違反、評判や信用の低下などがあります。

OPS10-BP04 エスカレーション経路を決定する

ランブックとプレイブックで、エスカレーションをトリガーするものとエスカレーションの手順を含むエスカレーション経路を決定します。特に、各アクションの所有者を特定し、運用イベントに効果的かつすばやく対応できるようにします。

アクションを行う前に、人間による決定が必要なタイミングを特定します。意思決定者と協力して事前に決定を行い、アクションを事前に承認することで、MTTR が応答を長時間待機しないようにします。

一般的なアンチパターン:

- あなたの小売サイトがダウンしています。あなたは、サイトを復旧するためのランブックを理解していません。あなたは、誰かがサポートしてくれることを願いながら、同僚に電話をかけ始めます。
- あなたは、アクセス不能なアプリケーションのサポートケースを受け取ります。あなたには、システムを管理するためのアクセス許可がありません。あなたは、誰が管理しているかを知りません。ケースを開いたシステム所有者に連絡しようとしましたが、応答がありません。あなたはシステムに関する問い合わせ先を知らず、同僚はそれになじみがありません。

このベストプラクティスを活用するメリット: エスカレーション、エスカレーションのトリガー、エスカレーションの手順を定義することで、影響に対して適切な割合で、インシデントへの体系的なリソースの追加が可能となります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- エスカレーションパスを定義する: ランブックとプレイブックで、何がエスカレーションをトリガーするかやエスカレーションの手順を含む、エスカレーション経路を決定します。例えば、ランブックで問題が解決できない場合や、一定期間が経過した場合にサポートエンジニアからシニアサポートエンジニアに向けた問題のエスカレーションがあります。また、プレイブックでは修正経路が特定できない場合や、一定期間が経過した場合に、ワークロードについてシニアサポートエンジニアから開発チームに向けたエスカレーションなども例として挙げられます。特に、各アクションの所有者を特定し、運用イベントに効果的かつすばやく対応できるようにします。エスカレーションには第三者が入る場合があります。例えば、ネットワーク接続プロバイダーまたはソフトウェアベンダーです。エスカレーションには、影響するシステムについて承認を受けた特定の意思決定者を含めることができます。

OPS10-BP05 プッシュ通知を有効にする

ユーザーの使用するサービスに影響が生じたときに、ユーザーと直接通信し (E メールや SMS など)、再び通常運用状態に復帰したときに再度通信し、ユーザーが適切な対応アクションを起こせるようにします。

一般的なアンチパターン:

- アプリケーションで分散型サービス拒否のインシデントが発生し、数日間応答がない状態になっています。エラーメッセージはありません。あなたは、通知 E メールを送信していません。あなたは、テキスト通知を送信していません。あなたは、ソーシャルメディアで情報を共有していません。お客様は不満を抱いており、サポートしてくれる他のベンダーを探しています。
- 月曜日に、アプリケーションでパッチ適用後に問題が生じ、数時間停止しました。火曜日に、コードのデプロイ後にアプリケーションに問題が発生し、数時間にわたり信頼性が低下しました。水曜日に、失敗したパッチに関連するセキュリティの脆弱性を軽減するためコードをデプロイした後に、アプリケーションで問題があり、数時間使用できませんでした。木曜日に、不満を募らせたお客様が、サポートしてくれる他のベンダーを探し始めました。
- アプリケーションは、今週末にメンテナンスのために停止します。あなたは、顧客に通知しません。一部の顧客は、アプリケーションの使用を要するアクティビティを予定していました。顧客は、アプリケーションが利用できないことを知ると、不満を爆発させます。

このベストプラクティスを確立するメリット: 通知、通知のトリガー、および通知の手順を定義することで、ワークロードに関する問題が顧客に影響した場合に、顧客に当該事実を伝え、対応できるようになります。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- プッシュ通知を有効にする: ユーザーが利用しているサービスに影響があった場合、またサービスが正常な動作状態に戻った場合、ユーザーが適切なアクションを起こせるように、E メールや SMS などで直接ユーザーに伝えます。
  - Amazon SES の特徴
  - Amazon SES とは
  - Amazon SNS 通知の設定

リソース

関連するドキュメント:

- Amazon SES の特徴
- Amazon SNS 通知の設定
- Amazon SES とは

OPS10-BP06 ダッシュボードでステータスを知らせる

対象となる利用者 (内部技術チーム、指導部、顧客など) に合わせたダッシュボードを用意して、現在の業務の運用状況と、相手が関心を持つメトリクスを知らせます。

コンソールのカスタマイズ可能なホームページで Amazon CloudWatch ダッシュボードを使用して コンソール CloudWatch でダッシュボードを作成できます。Amazon QuickSight のようなビジネスインテリジェンスサービスを 使用すると、 ワークロードと運用状態 (注文率、接続ユーザー、トランザクション時間など) のインタラクティブなダッシュボードを作成して公開できます。メトリクスのシステムレベルおよびビジネスレベルのビューを表示するダッシュボードを作成します。

一般的なアンチパターン:

- リクエストに応じて、あなたは、管理のためのアプリケーションの現在の使用状況に関するレポートを実行します。
- インシデント中、あなたには、心配なシステム所有者から「修正状況」を知りたいという連絡が 20 分ごとにあります。

このベストプラクティスを活用するメリット: ダッシュボードを作成することで、情報へのセルフサービスアクセスが可能になり、顧客自身が情報を確認し、アクションが必要かどうかを判断できるようになります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ダッシュボードで状態を知らせる: 対象となる利用者 (内部技術チーム、経営陣、顧客など) に合わせたダッシュボードを用意して、現在の業務の運用状況と、相手が関心を持つメトリクスを知らせます。ステータス情報のセルフサービスオプションによって、ステータスのリクエスト処理による運用チームの中断を減らすことができます。例として、Amazon CloudWatch ダッシュボードと AWS Health Dashboard が挙げられます。
  - CloudWatch ダッシュボードを使ったカスタムメトリクスビューの作成と使用

リソース

関連するドキュメント:

- Amazon QuickSight
- CloudWatch ダッシュボードを使ったカスタムメトリクスビューの作成と使用

OPS10-BP07 イベントへの対応を自動化する

イベントへの対応を自動化し、手動プロセスによって発生するエラーを減らして、迅速かつ一貫した対応を実現します。

AWS には、ランブックやプレイブックのアクションを自動的に実行する複数の方法があります。AWS リソースの状態変化や独自のカスタムイベントからのイベントに対応するには、CloudWatch Events ルールを作成し、 CloudWatch のターゲット (Lambda 関数、Amazon Simple Notification Service (Amazon SNS) トピック、Amazon ECS タスク、AWS Systems Manager Automation など) を通じて対応を起動します。

リソースのしきい値を超えるメトリクス (待機時間など) に応答するには、 CloudWatch アラームを作成して Amazon EC2 アクションや Auto Scaling アクションを使用して 1 つ以上のアクションを実行するか、Amazon SNS トピックに通知を送信します。アラームへの応答でカスタムアクションを実行する必要がある場合は、Amazon SNS 通知を通じて Lambda を呼びだします。Amazon SNS を使用して、イベント通知とエスカレーションメッセージを発行し、ユーザーに情報を提供します。

また、AWS は、AWS のサービス API と SDK を通じてサードパーティーシステムもサポートしています。AWS パートナーやサードパーティーでは、モニタリング、通知、応答を可能にするモニタリングツールを多数提供しています。これらのツールには、New Relic、Splunk、Loggly、SumoLogic、Datadog などがあります。

自動化された手順が失敗した場合に、手動でも重要な手順を実施できるようにしておく必要があります。

一般的なアンチパターン:

- 開発者が自分のコードをチェックインします。このイベントは、ビルドを開始し、テストを実行するために使用することもできましたが、結局、何にも使用されていません。
- アプリケーションが動作を停止する前に、特定のエラーをログ記録します。アプリケーションを再起動する手順はよく理解されており、スクリプト化することもできました。あなたは、ログイベントを使用して、スクリプトを呼び出し、アプリケーションを再起動することもできました。しかし、それらの対応を行わなかったため、日曜日の午前 3 時にエラーが発生し、あなたは、オンコールでシステムを修正する責任者として起こされます。

このベストプラクティスを確立するメリット: イベントへの対応を自動化することで、対応にかかる時間を短縮し、手作業によるエラーの発生を抑制することができます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- イベントへの対応を自動化する: イベントへの対応を自動化し、手動プロセスによって発生するエラーを減らして、迅速かつ一貫した対応を実現します。
  - Amazon CloudWatch Events とは
  - イベントでトリガーする CloudWatch Events のルールの作成
  - AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成
  - サポートされているサービスからの CloudWatch Events イベントの例

リソース

関連するドキュメント:

- Amazon CloudWatch の特徴
- サポートされているサービスからの CloudWatch Events イベントの例

- AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成
- イベントでトリガーする CloudWatch Events のルールの作成
- Amazon CloudWatch Events とは

関連動画:

- モニタリング計画を立てる

関連する例:

進化

質問

- OPS 11 オペレーションをどのように進化させるのですか?

OPS 11 オペレーションをどのように進化させるのですか?

漸進的な継続的改善に時間とリソースを費やすことで、オペレーションを効果的かつ効率的に進化させることができます。

ベストプラクティス

- OPS11-BP01 継続的改善のプロセスを用意する
- OPS11-BP02 インシデント後の分析を実行する
- OPS11-BP03 フィードバックループを実装する
- OPS11-BP04 ナレッジ管理を実施する
- OPS11-BP05 改善の推進要因を定義する
- OPS11-BP06 インサイトを検証する
- OPS11-BP07 オペレーションメトリクスのレビューを実行する
- OPS11-BP08 教訓を文書化して共有する
- OPS11-BP09 改善を行うための時間を割り当てる

OPS11-BP01 継続的改善のプロセスを用意する

最も大きな利益をもたらす取り組みに集中できるように、改善の機会を定期的に評価し、優先順位を設定します。

一般的なアンチパターン:

- これまでに、開発環境またはテスト環境の作成に必要な手順を文書化しました。あなたは、CloudFormation を使用してプロセスを自動化することもできますが、代わりにコンソールから手動で行います。
- テストでは、アプリケーション内の CPU 使用率の大部分が、非効率的な機能の小さな集まりの状態にあることが示されています。あなたは、改善とコスト削減に注力することもできますが、新しいユーザビリティ機能を作成するタスクが割り当てられています。

このベストプラクティスを活用するメリット: 継続的な改善は、改善の機会を定期的に評価し、機会に優先順位を付けて、最大のメリットを提供できる領域に注力するメカニズムを提供します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 継続的改善のプロセスを定義する: 最も大きな利益をもたらす取り組みに集中できるように、改善の機会を定期的に評価し、優先順位を設定します。改善のための変更を加えて結果を評価し、成功を判断します。結果が目標に達しておらず、今後も改善が優先事項である場合は、代わりの一連のアクションを使用して繰り返します。オペレーションプロセスには、漸進的な継続的改善を可能にする時間とリソースを含める必要があります。

OPS11-BP02 インシデント後の分析を実行する

顧客に影響を与えるイベントを確認し、寄与する要因と予防措置を特定します。この情報を使用して、再発を制限または回避するための緩和策を開発します。迅速で効果的な対応のための手順を開発します。対象者に合わせて調整された、寄与因子と是正措置を必要に応じて伝えます。

一般的なアンチパターン:

- あなたは、アプリケーションサーバーを管理しています。約 23 時間 55 分ごとに、すべてのアクティブなセッションが終了します。あなたは、アプリケーションサーバーで何が問題なのかを特定しようとしました。あなたは、これがネットワークの問題である可能性があることを疑っていますが、ネットワークチームが忙しすぎてサポートを提供できないため、当該チームから協力を得ることができません。あなたには、サポートを得て、何が起こっているかを判断するために必要な情報を収集するための事前定義されたプロセスがありません。
- あなたは、ワークロード内でデータを失ってしまいました。このような問題が発生したのはこれが最初であり、原因は明らかではありません。あなたは、データを再作成できるため、これが重要

ではないと判断しています。データ損失は、顧客に影響するほどの高い頻度で発生し始めます。また、これにより、失われたデータの復元に際して、追加の運用上の負担も発生します。

このベストプラクティスを活用するメリット: インシデントの原因となったコンポーネント、条件、アクション、イベントを決定する事前定義されたプロセスを持つことで、改善の機会を把握できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- プロセスを使用して寄与した要因を判断する: 顧客に影響を与えるすべてのインシデントを確認します。インシデントに寄与した要因を特定してドキュメント化するためのプロセスを用意しておき、再発を抑制または防止する緩和策と、迅速で効果的な対応手順を展開できるようにしておきます。必要に応じ、対象者に合わせて根本原因を通知します。

OPS11-BP03 フィードバックループを実装する

フィードバックループは、意思決定を推進するための実行可能なインサイトを提供します。フィードバックループを手順やワークロードに組み込みます。そうすることで、問題および改善すべき領域を特定することができます。またフィードバックループは、改善への投資を検証することもできます。これらのフィードバックループは、ワークロードの継続的な改善の基盤となります。

フィードバックループは、 即時フィードバック および 遡及分析の 2 つのカテゴリに分類されます。即時フィードバックは、オペレーションアクティビティのパフォーマンスと結果のレビューをとおして収集されます。このフィードバックは、チームメンバー、顧客、またはアクティビティの自動出力から得られます。即時フィードバックは A/B テストや新機能のリリースなどからも得ることができ、フェイルファストにおいて不可欠なものです。

遡及分析は定期的に実行され、オペレーションの結果とメトリクスの長期間にわたるレビューからフィードバックを取得します。これらの遡及分析は、スプリント、サイクル、またはメジャーリリースやイベントの完了時に行われます。このタイプのフィードバックループは、オペレーションまたはワークロードへの投資を検証でき、成果と戦略の計測に役立ちます。

期待される成果: 即時フィードバックと遡及分析を使用して、改善を推進します。ユーザーやチームメンバーからのフィードバックを取得する仕組みがあります。遡及分析を使用して、改善を推進する傾向を特定します。

一般的なアンチパターン:

- 新しい機能をローンチしたが、顧客からのフィードバックを得る方法はない。
- オペレーションの改善に投資した後、遡及分析を行って投資を検証していない。
- 顧客からのフィードバックを収集しているが、定期的にレビューしていない。
- フィードバックループに基づいて提案されたアクション項目があるが、それらはソフトウェア開発プロセスに含まれていない。
- 顧客からの改善提案に対するフィードバックを行っていない。

このベストプラクティスを活用するメリット:

- 顧客の視点から新しい機能を推進することができる。
- 組織の文化をより迅速に変化させることができる。
- 傾向をレビューすることで、改善の機会を特定できる。
- 遡及分析によって、ワークロードやオペレーションへの投資を検証できる。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

このベストプラクティスを採用すると、即時フィードバックと遡及分析の両方を使用することになります。これらのフィードバックループによって改善を推進します。即時フィードバックには、調査、顧客へのアンケート、フィードバックフォーラムなど、さまざまな仕組みがあります。また組織は、遡及分析を使用して改善の機会を特定し、取り組みを検証できます。

顧客の事例

AnyCompany Retail は、顧客がフィードバックを投稿したり、問題を報告したりすることができるウェブフォーラムを作成しました。週次会議では、ソフトウェア開発チームがユーザーからのフィードバックを評価します。プラットフォームの改善方針の決定のために、フィードバックは定期的に使用されます。各スプリントの完了時に遡及分析を実施して、改善する項目を特定します。

実装手順

1. 即時フィードバック
    - 顧客やチームメンバーからフィードバックを得るための仕組みが必要です。また、オペレーションアクティビティを構成して、自動的にフィードバックを受信することもできます。
    - 組織にはフィードバックをレビューし、改善点を決定して、改善のスケジュールを策定するプロセスが必要です。

- フィードバックはソフトウェア開発プロセスに追加する必要があります。
- 改善を進めるとともに、改善の提案者にフォローアップのフィードバックを行います。
  - AWS Systems Manager OpsCenterを使用して、 これらの改善を OpsItems として作成し 追跡できます。

2. 遡及分析
   - 開発サイクル、定められたサイクル、またはメジャーリリースの完了時に遡及分析を実施します。
   - ワークロードの関係者を集めて、遡及分析会議を行います。
   - ホワイトボードまたはスプレッドシートに、停止、開始、維持の 3 つの列を作成します。
     - 停止は、 チームの活動を停止する項目を指します。
     - 開始は、 アイデアへの取り組みを開始する項目を指します。
     - 維持は、 取り組みを維持する項目を指します。
   - 会議室内の関係者からフィードバックを収集します。
   - フィードバックに優先順位を付けます。アクションと関係者を開始項目または維持項目に割り当てます。
   - アクションをソフトウェア開発プロセスに追加し、改善を進めながら更新されたステータスを関係者に通知します。

実装計画に必要な工数レベル: 中。このベストプラクティスを採用するには、即時フィードバックを収集し分析するプロセスが必要です。また、遡及分析プロセスを確立する必要もあります。

リソース

関連するベストプラクティス:

- OPS01-BP01 外部顧客のニーズを評価する: フィードバックループは、外部顧客のニーズを収集する仕組みです。
- OPS01-BP02 内部顧客のニーズを評価する: 内部関係者は、フィードバックループを使用して、ニーズや要件を伝えることができます。
- OPS11-BP02 インシデント後の分析を実行する: 事後分析は、インシデント後に実施される重要な遡及分析の 1 つです。
- OPS11-BP07 オペレーションメトリクスのレビューを実行する: オペレーションメトリクスレビューでは、傾向および改善の領域を特定します。

関連するドキュメント:

- CCOE を構築するときに回避すべき 7 つの落し穴
- Atlassian チームプレイブック - 振り返り
- E メールの定義: フィードバックループ
- AWS Well-Architected フレームワークレビューに基づくフィードバックループの確立
- IBM ガレージメソドロジー - 振り返りの保留
- Investopedia - PDCS サイクル
- 開発者の有効性を最大化する (Tim Cochran 著)
- 運用準備状況の確認 (ORR) に関するホワイトペーパー - イテレーション
- TIL CSI - 継続的なサービスの改善
- トヨタでの e コマースの採用: Amazon での無駄のない管理

関連動画:

- 効果的な顧客フィードバックループの構築

関連サンプル:

- Astuto - オープンソースの顧客フィードバックツール
- AWS ソリューション - AWS の QnABot
- Fider - 顧客フィードバックの管理プラットフォーム

関連サービス:

- これらの改善を

OPS11-BP04 ナレッジ管理を実施する

チームメンバーが探している情報をタイムリーに検出し、アクセスして、最新かつ完全であることを確認するメカニズムが存在しています。必要なコンテンツ、更新が必要なコンテンツ、および今後参照されることのないようにアーカイブする必要があるコンテンツを特定するためのメカニズムが存在しています。

一般的なアンチパターン:

- 不満のある顧客が、認識された問題への対応を求めて、新しい製品機能のリクエストのサポートケースをオープンします。これは、優先的な改善のリストに追加されます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ナレッジ管理を実施する: チームメンバーが、探している情報をタイムリーに入手し、アクセスして、最新かつ完全であることを識別するための仕組みを確立します。必要なコンテンツ、更新が必要なコンテンツ、および今後参照されることのないようにアーカイブする必要があるコンテンツを特定するためのメカニズムを維持します。

OPS11-BP05 改善の推進要因を定義する

機会を評価して優先順位を設定できるよう、改善の推進要因を特定します。

AWS では、すべての運用アクティビティ、ワークロード、インフラストラクチャのログを集約して詳細なアクティビティ履歴を作成できます。AWS ツールを使用して長期的に運用とワークロードの状態を分析し (トレンドの特定、イベントやアクティビティの成果への関連付け、環境間やシステム全体の比較や対比など)、推進要因に基づいて改善の機会を見つけることができます。

CloudTrail を使用し、AWS Management Console、CLI、SDK、API を介して API アクティビティを追跡し、アカウント全体で何が起きているかを把握する必要があります。CloudTrail および CloudWatch を使用して、AWS デベロッパーツールのデプロイアクティビティを追跡します。これにより、デプロイの詳細なアクティビティ履歴と結果が CloudWatch Logs Logs のログデータに追加されます。

長期保管用の Amazon S3 にログデータを エクスポートします。分析のために、 AWS Glueを使用して、Amazon S3 のログデータを検出して準備します。AWS Glue とネイティブで統合された Amazon Athena を使用して、ログデータを分析します。Amazon QuickSight のような ビジネスインテリジェンスツール を使用して、データを可視化、調査、分析することができます。

一般的なアンチパターン:

- あるスクリプトは、機能はするものの、洗練されてはいません。あなたは、書き換えに時間を費やします。現在は素晴らしいスクリプトです。
- スタートアップは、ベンチャーキャピタリストから別の資金を調達しようとしています。そのスタートアップは、PCI DSS へのコンプライアンスを実証することをあなたに求めています。あな

たは、ベンチャーキャピタリストを満足させたいと考え、コンプライアンスを文書化し、ある顧客の納期に遅れ、その顧客を失います。それをするのは間違ったことではありませんでしたが、今では正しいことだったのかどうかを疑問に思います。

このベストプラクティスを活用するメリット: 改善に使用する基準を決定することで、イベントベースのモチベーションや感情的投資の影響を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 改善の推進要因を理解する: システムに変更を加えるのは、望まれている成果がサポートされているときだけにしてください。
  - 望まれている機能: 改善の機会を評価する際は、望まれている機能を評価してください。
    - AWS の最新情報
  - 許容できない問題: 改善の機会を評価する際は、許容できない問題、バグ、脆弱性を評価してください。
    - AWS セキュリティ速報
    - AWS Trusted Advisor
  - コンプライアンスの要件: 改善の機会を確認する際は、規制/ポリシー遵守の維持、またはサードパーティーによるサポートの維持に必要な更新と変更を評価します。
    - AWS コンプライアンス
    - AWS コンプライアンスプログラム
    - AWS コンプライアンスの最新情報

リソース

関連するドキュメント:

- Amazon Athena
- Amazon QuickSight
- AWS コンプライアンス
- AWS コンプライアンスの最新情報
- AWS コンプライアンスプログラム
- AWS Glue

- AWS セキュリティ速報
- AWS Trusted Advisor
- ログデータを Amazon S3 にエクスポートする
- AWS の最新情報

OPS11-BP06 インサイトを検証する

分析結果を確認してクロスな役割を持つチームやビジネスオーナーで応答します。これらのレビューに基づいて共通の理解を確立し、追加的な影響を特定するとともに、一連のアクションを決定します。必要に応じて対応を調整してください。

一般的なアンチパターン:

- あなたは、CPU 使用率がシステムで 95% であることを確認し、システムの負荷を軽減する方法を見つけることを優先します。あなたは、最適なアクションがスケールアップであると判断します。システムはトランスコーダーであり、いつでも 95% の CPU 使用率で実行するようにスケールされています。あなたがシステム所有者に連絡していれば、状況を説明してもらえたかもしれません。時間を無駄にしてしまいました。
- システム所有者は、システムがミッションクリティカルであると主張しています。システムは強固なセキュリティ環境に配置されていませんでした。セキュリティの向上のため、あなたは、ミッションクリティカルなシステムに要求される追加の発見的統制および予防統制を実装します。あなたは、作業が完了し、追加のリソースに対して課金される旨をシステム所有者に通知します。この通知の後の話し合いにおいて、システム所有者は、ミッションクリティカルという用語について正式な定義があり、自身のシステムがこれに適合していないことを知ります。

このベストプラクティスを活用するメリット: ビジネスオーナーや各分野の エキスパートとインサイトを検証することで、共通の理解を確立し、より効果的に改善につなげることができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- インサイトを検証する: ビジネスオーナーや各分野の エキスパートと協力して、収集したデータの意味について共通の理解と合意があることを確認します。追加の懸念事項や潜在的な影響を特定し、一連のアクションを判断します。

OPS11-BP07 オペレーションメトリクスのレビューを実行する

ビジネスのさまざまな分野のチームメンバー間でオペレーションメトリクスの遡及分析を定期的に実施します。これらのレビューに基づいて、改善の機会と取り得る一連のアクションを特定するとともに、教訓を共有します。

すべての環境 (開発、テスト、生産など) で改善する機会を探します。

一般的なアンチパターン:

- 大々的な販促活動が行われていましたが、メンテナンスウィンドウによって中断されました。ビジネスに影響する他のイベントがある場合、標準メンテナンスウィンドウが延期される可能性があることが認識されていません。
- あなたは、組織で一般的に使用されているバグのあるライブラリを使用しているため、停止時間が長くなり、困っていました。その後、あなたは、信頼性の高いライブラリに移行しました。組織内の他のチームは、自身がリスクにさらされているかはわかっていません。あなたが定期的にミーティングを行い、このインシデントを確認していれば、彼らはリスクを認識していたでしょう。
- トランスコーダーのパフォーマンスは着実に低下しており、メディアチームに影響を及ぼしています。まだひどい状態であるとまでは言えません。インシデントの原因となるほど悪くなるまで気付く機会はありません。メディアチームと一緒にオペレーションメトリクスを見直すことで、メトリクスの変化や彼らの経験を認識し、問題に対処する機会が生まれるはずです。
- あなたは、顧客の SLA の満足度を確認していません。あなたは、顧客の SLA に適合しない傾向があります。顧客の SLA に適合しない場合は、金銭的ペナルティが発生します。これらの SLA のメトリクスを確認するためのミーティングを定期的に開催していれば、問題を認識して対処する機会が得られたはずです。

このベストプラクティスを確立するメリット: 定期的にミーティングを行い、オペレーションメトリクス、イベント、インシデントを確認することで、チーム全体で共通の理解を維持し、学んだ教訓を共有し、改善を優先順位付けして目標を設定することができます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- オペレーションメトリクスのレビュー: 定期的に、さまざまな分野のチームメンバーとともに、オペレーションメトリクスの遡及分析を行います。ビジネス、開発、オペレーションチームを含む関係者を参加させて、即時フィードバックと遡及分析から得られた結果を検証し、教訓を共有します。それらの洞察に基づいて、改善の機会と取り得る一連のアクションを特定します。

- Amazon CloudWatch
- Amazon CloudWatch メトリクスを使用する
- カスタムメトリクスをパブリッシュする
- Amazon CloudWatch のメトリクスとディメンションのリファレンス

リソース

関連するドキュメント:

- Amazon CloudWatch
- Amazon CloudWatch のメトリクスとディメンションのリファレンス
- カスタムメトリクスをパブリッシュする
- Amazon CloudWatch メトリクスを使用する

OPS11-BP08 教訓を文書化して共有する

運用アクティビティから学んだ教訓を文書化して共有し、社内とチーム全体で利用できるようにします。

チームが学んだことを共有して、組織全体のメリットを増やす必要があります。情報とリソースを共有して、回避可能なエラーを防止し、開発作業を容易にする必要があります。これにより、望まれる機能の提供に集中できます。

AWS Identity and Access Management (IAM) を使用して、アカウント内またはアカウント間で共有するリソースへのコントロールされたアクセスを可能にするアクセス許可を定義します。その後、バージョン管理された AWS CodeCommit リポジトリを使用して、アプリケーションライブラリ、スクリプト化された手順、手順のドキュメント、その他のシステムドキュメントを共有する必要があります。AMI へのアクセスを共有し、Lambda 関数の使用をアカウント間で許可することで、コンピューティング標準を共有します。また、インフラストラクチャ標準を AWS CloudFormation のテンプレートとして共有する必要もあります。

AWS API と SDK を使用すると、外部ツールやサードパーティーのツールやリポジトリ (GitHub、BitBucket、SourceForge など) を統合できます。学んだことや開発したことを共有するときは、共有リポジトリの完全性を保証するためにアクセス許可を構造化することに注意してください。

一般的なアンチパターン:

- あなたは、組織で一般的に使用されているバグのあるライブラリを使用しているため、停止時間が長くなり、困っていました。その後、あなたは、信頼性の高いライブラリに移行しました。組織内の他のチームは、自身がリスクにさらされているかはわかっていません。このライブラリの経験を文書化および共有することとしていれば、これらのチームはリスクを認識していたでしょう。
- あなたは、セッションがドロップする原因となる内部共有マイクロサービスのエッジケースを特定しました。このエッジケースを回避するために、サービスへの呼び出しを更新しました。組織内の他のチームは、自身がリスクにさらされているかはわかっていません。このライブラリの経験を文書化および共有することとしていれば、これらのチームはリスクを認識していたでしょう。
- マイクロサービスの 1 つについて、CPU 使用率要件を大幅に削減する方法を見つけました。あなたは、他のチームがこの手法を利用できるかどうかはわかりません。このライブラリで経験を文書化および共有することとしていれば、これらのチームは利用する機会を得ていたでしょう。

このベストプラクティスを活用するメリット: 教訓を共有して、改善をサポートし、経験から得られる恩恵を最大化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 教訓を文書化して共有する: 運用アクティビティと遡及分析の実行から学習した教訓を文書化する手順を決めて、ほかのチームが使用できるようにします。
  - 教訓を共有する: 教訓と関連するアーティファクトをチーム全体で共有する手順を決めます。たとえば、アクセス可能な Wiki を使用して手順の更新、ガイダンス、ガバナンス、ベストプラクティスを共有します。共通のリポジトリを使用してスクリプト、コード、ライブラリを共有します。
    - AWS 環境へのアクセスの委任
    - AWS CodeCommit リポジトリを共有する
    - AWS Lambda 関数の簡単な承認
    - 特定の AWS アカウントと AMI を共有する
    - AWS CloudFormation デザイナー URL によるテンプレート共有の高速化
    - Amazon SNS での AWS Lambda の使用

リソース

関連するドキュメント:

- AWS Lambda 関数の簡単な承認
- AWS CodeCommit リポジトリを共有する
- 特定の AWS アカウントと AMI を共有する
- AWS CloudFormation デザイナー URL によるテンプレート共有の高速化
- Amazon SNS での AWS Lambda の使用

関連動画:

- AWS 環境へのアクセスの委任

OPS11-BP09 改善を行うための時間を割り当てる

漸進的な継続的改善を可能にする時間とリソースをプロセス内に設けます。

AWS では、一時的に重複する環境を作成することで、実験やテストのリスク、労力、コストを削減できます。こうした重複する環境を使用して、分析、実験からの結論をテストし、計画した改善を開発してテストできます。

一般的なアンチパターン:

- アプリケーションサーバーに既知のパフォーマンスの問題があります。この問題は、計画されたすべての機能実装の後に実施されるバックログに追加されます。計画的に追加される機能の割合が一定であると、パフォーマンスの問題が解決されることはありません。
- 継続的な改善をサポートするために、管理者と開発者が改善の選択と実装にすべての余分な時間を費やすことを承認します。改善は完了しません。

このベストプラクティスを確立するメリット: 時間とリソースをプロセス内に設けることで、漸進的な継続的改善が可能となります。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 改善を行うための時間を割り当てる: 継続的な漸進的改善を可能にするために、プロセス内に時間とリソースを割り当てます。改善のための変更を加えて結果を評価し、成功を判断します。結果が目標に達しておらず、今後も改善が優先事項である場合は、アクションの代替案を追及してください。

セキュリティ

トピック

セキュリティの基礎

質問

SEC 1 ワークロードを安全に運用するには、どうすればよいですか?

ワークロードを安全に運用するには、セキュリティのすべての領域に包括的なベストプラクティスを適用する必要があります。組織レベルおよびワークロードレベルにおいて、「運用上の優秀性」で定義した要件とプロセスを抽出し、それらをすべての領域に適用します。AWS や業界のレコメンデーションおよび脅威インテリジェンスを最新に保つことで、脅威モデルと管理目標を進化させることができます。セキュリティプロセス、テスト、検証を自動化することで、セキュリティオペレーションを拡張できます。

ベストプラクティス

SEC01-BP01 アカウントを使用してワークロードを分ける:

セキュリティとインフラストラクチャを念頭に置いて、ワークロードが増大するにつれて組織が共通のガードレールを設定できるようにします。このアプローチによって、ワークロード間の境界と制御が確立します。開発環境およびテスト環境から本番環境を分離する場合、または外部コンプライアンス要件 (PCI-DSS や HIPAA など) で定義されている機密レベルの異なるデータを処理するワークロードとそうでないワークロードとの間に強力な論理的境界を設ける場合は、アカウントレベルの分離を強くお勧めします。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- AWS Organizations を使用する: AWS Organizations を使用し、複数の AWS アカウント にポリシーベースの管理を一元的に適用します。
  - AWS Organizations の開始方法
  - サービスコントロールポリシーを使用して、AWS Organization のアカウント間にアクセス許可ガードレールを設定する方法
- AWS Control Tower を考慮する: AWS Control Tower では、ベストプラクティスに基づいて、新しいセキュアなマルチアカウントの AWS 環境を容易にセットアップおよび管理できます。
  - AWS Control Tower

リソース

関連するドキュメント:

- IAM ベストプラクティス
- セキュリティ速報
- AWS セキュリティ監査のガイドライン

関連動画:

- Managing Multi-Account AWS Environments Using AWS Organizations (AWS Organizations を使用したマルチアカウント AWS 環境の管理)
- Well-Architected の手法によるセキュリティのベストプラクティス
- Using AWS Control Tower to Govern Multi-Account AWS Environments (AWS Control Tower を使用したマルチアカウント AWS 環境の統制)

SEC01-BP02 AWS アカウント をセキュリティ保護する

AWS アカウント のセキュリティ保護には、 ルートユーザーの保護および使用の回避、連絡先情報を最新の状態に保つなど、さまざまな側面があります。専用のインフラストラクチャで AWS Organizations AWS を使用すれば、ワークロードの拡大やスケーリングに合わせて、アカウントを一元管理できます。AWS Organizations は、アカウント全体の管理、制御の設定、サービスの構築に役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- AWS Organizations を使用する: AWS Organizations を使用し、複数の AWS アカウント にポリシーベースの管理を一元的に適用します。
  - AWS Organizations の開始方法
  - サービスコントロールポリシーを使用して、AWS Organization のアカウント間に許可ガードレールを設定する方法
- AWS ルートユーザーの使用を制限する: 特定のルートユーザーを必要とするタスクについては、当該ルートユーザーのみを使用します。
  - AWS アカウントのルートユーザー認証情報を必要とする AWS タスク
- ルートユーザー用の AWS Multi-Factor Authentication (AWS MFA): AWS Organizations が代理でルートユーザーを管理していない場合、AWS アカウント ルートユーザーで MFA を有効化します。
  - ルートユーザー
- ルートユーザーパスワードを定期的に変更する: ルートユーザーのパスワードを変更することにより、保存したパスワードが使用できる状態となっていることによるリスクが軽減されます。これは、AWS Organizations を使用しておらず、あらゆるユーザーが物理的にアクセスできる場合に特に重要です。
  - AWS アカウント のルートユーザーのパスワードの変更
- AWS アカウント のルートユーザーが使用された場合に通知を有効化する: 通知を受け取ることで、リスクは自動的に軽減されます。
  - AWS アカウント のルートアクセスキーを使用した場合の通知方法
- 新しく追加されたリージョンへのアクセスを制限する: 新しい AWS リージョン について、ユーザーやロールなどの IAM リソースは、有効にしたリージョンのみに伝播されます。
  - 今後の AWS リージョン に対してアカウントを有効にするアクセス許可の設定

- AWS CloudFormation StackSets を検討する: CloudFormation StackSets を使用すると、IAM ポリシー、ロール、グループなどのリソースをさまざまな AWS アカウント とリージョンに承認されたテンプレートからデプロイできます。
  - CloudFormation StackSets を使用する

リソース

関連するドキュメント:

- AWS Control Tower
- AWS セキュリティ監査ガイドライン
- IAM ベストプラクティス
- セキュリティ速報

関連動画:

- 自動化とガバナンスにより AWS の大規模な採用を可能にする
- Well-Architected の手法によるセキュリティのベストプラクティス

関連する例:

- ラボ: AWS アカウント およびルートユーザー

SEC01-BP03 管理目標を特定および検証する:

脅威モデルから特定されたコンプライアンス要件とリスクに基づいて、ワークロードに適用する必要がある管理目標および管理を導き出し、検証します。管理目標と制御を継続的に検証することは、リスク軽減の効果測定に役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- コンプライアンス要件を特定する: ワークロードが準拠する必要のある組織要件、法的要件、規制要件を確認します。
- AWS コンプライアンスリソースを特定する: コンプライアンスを支援するために使用できる AWS のリソースを特定します。

- https://aws.amazon.com/compliance/
- https://aws.amazon.com/artifact/

リソース

関連するドキュメント:

- AWS セキュリティ監査のガイドライン
- セキュリティ速報

関連動画:

- AWS Security Hub: Manage Security Alerts and Automate Compliance
- Well-Architected の手法によるセキュリティのベストプラクティス

SEC01-BP04 セキュリティ脅威に関する最新情報を入手する:

適切な制御を定義して実装するために、最新のセキュリティ脅威を常に把握して攻撃ベクトルを認識します。AWS Managed Services を利用することで、AWS アカウントにおける予期しない動作や異常な動作の通知を簡単に受けることができます。セキュリティ情報フローの一環として、AWS Partner ツールまたはサードパーティーの脅威情報フィードの利用を検討します。それらの 共通脆弱性識別子 (CVE) リスト には、一般に公開されているサイバーセキュリティの脆弱性が含まれており、最新の情報を入手するために利用することができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- 脅威インテリジェンスソースを購読する: ワークロードで使用しているテクノロジーに関連する、複数のソースからの脅威インテリジェンス情報を定期的に確認します。
  - 共通脆弱性識別子リスト
- 検討 AWS Shield Advanced サービスを検討する: ワークロードがインターネットに接続できる環境であれば、インテリジェンスソースをほぼリアルタイムで可視化することができます。

リソース

関連するドキュメント:

- AWS セキュリティ監査のガイドライン
- AWS Shield
- セキュリティ速報

関連動画:

- Well-Architected の手法によるセキュリティのベストプラクティス

SEC01-BP05 セキュリティに関する推奨事項を常に把握する

AWS と業界の両方のセキュリティの推奨事項を常に最新に保ち、ワークロードのセキュリティ体制を進化させます。AWS セキュリティ速報 は、セキュリティおよびプライバシー通知に関する重要な情報を含みます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- AWS のアップデートをフォローする: 購読または定期的にチェックして、新しい推奨事項や ヒント、テクニックを確認しましょう。
  - AWS Well-Architected ラボ
  - AWS セキュリティブログ
  - AWS のサービスドキュメント
- 業界ニュースを購読する: 複数のソースから、ワークロードで使用しているテクノロジーに関連するニュースフィードを定期的に確認します。
  - 例: 共通脆弱性識別子リスト

リソース

関連するドキュメント:

- セキュリティ速報

関連動画:

- Well-Architected の手法によるセキュリティのベストプラクティス

SEC01-BP06 パイプラインのセキュリティコントロールのテストと検証を自動化する

ビルド、パイプライン、プロセスの一環としてテストおよび検証されるセキュリティメカニズムの安全なベースラインとテンプレートを確立します。ツールとオートメーションを使用して、すべてのセキュリティコントロールの継続的なテストと検証を実施します。たとえば、マシンイメージやインフラストラクチャなどの項目をコードテンプレートとしてスキャンして、セキュリティの脆弱性、不規則性、ドリフトを各ステージで確立されたベースラインから確認します。AWS CloudFormation Guard を使用すると、CloudFormation が安全なことを検証し、時間を節約し、設定エラーのリスクを低減するのに役立ちます。

本番環境に取り込まれるセキュリティの誤設定の数を減らすことが非常に重要です。ビルドプロセスでより適切な品質管理をより多く実行し、欠陥の数を減らすことができれば、より優れたものになります。継続的インテグレーションおよび継続的デプロイ (CI/CD) のパイプラインは、可能な限りセキュリティの問題をテストできるように設計する必要があります。CI/CD パイプラインは、ビルドと配信の各段階でセキュリティを強化する機会を提供します。CI/CD セキュリティツールも更新して、進化する脅威を軽減する必要があります。

ワークロード設定への変更をトラッキングして、監査、変更管理、および該当する可能性がある調査へのコンプライアンスに役立てます。AWS Config を使用して、AWS およびサードパーティーリソースを記録および評価できます。これにより、ルールやコンフォーマンスパックへの全体的なコンプライアンスを継続的に監査および評価できます。コンフォーマンスパックとは、是正措置に関する一連のルールのことです。

変更トラッキングには、組織の変更管理プロセス (MACD-Move/Add/Change/Delete とも呼ばれる) の一部である計画されていた変更、予定外の変更、インシデントなどの予期しない変更を含める必要があります。変更はインフラストラクチャで発生する場合もあれば、コードリポジトリの変更、マシンイメージおよびアプリケーションインベントリの変更、プロセスとポリシーの変更、ドキュメントの変更などの他のカテゴリに関連するものである場合もあります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 設定管理を自動化する: 設定管理サービスまたはツールを使用して、リモートでアクションを実行し、安全な設定を自動的に適用および検証します。
  - AWS Systems Manager
  - AWS CloudFormation
  - AWS で CI/CD パイプラインを設定する

リソース

関連するドキュメント:

- How to use service control policies to set permission guardrails across accounts in your AWS Organization

関連動画:

- Managing Multi-Account AWS Environments Using AWS Organizations
- Well-Architected の手法によるセキュリティのベストプラクティス

SEC01-BP07 脅威モデルを使用してリスクを特定し、優先順位を付ける

脅威モデルを使用して潜在的な脅威を特定し、その登録を最新の状態に維持します。脅威に優先順位を付け、セキュリティコントロールを調整して防止、検出、対応を行います。進化するセキュリティ環境の状況に応じてセキュリティコントロールを再確認および維持します。

脅威のモデル化は、設計プロセスの早い段階でセキュリティ上の問題を発見して対処するのを支援する体系的なアプローチを提供します。ライフサイクルの後半に緩和策を講じるよりも、早い段階で緩和策を講じた方がコストがかからないからです。

脅威のモデル化の典型的なコアステップは次のとおりです。

1. アセット、アクター、エントリポイント、コンポーネント、ユースケース、信頼レベルを特定し、これらを設計図に記載する。
2. 脅威のリストを特定する。
3. 各脅威について、セキュリティコントロールの実装を含む緩和策を特定する。
4. リスクマトリックスを作成し、脅威が適切に緩和されているかどうかを確認する。

脅威のモデル化は、ワークロード (またはワークロードの機能) レベルで行うのが最も効果的であり、すべてのコンテキストが評価に利用できることを保証します。セキュリティ状況の変化に応じて、このマトリックスを見直し、維持してください。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 脅威モデルを作成する: 脅威モデルは、潜在的なセキュリティ脅威を特定して対処するのに役立ちます。
  - NIST: Guide to Data-Centric System Threat Modeling (データ中心システム脅威のモデル化へのガイド)

リソース

関連するドキュメント:

- AWS セキュリティ監査のガイドライン
- セキュリティ速報

関連動画:

- Well-Architected の手法によるセキュリティのベストプラクティス

SEC01-BP08 新しいセキュリティサービスと機能を定期的に評価および実装する

ワークロードのセキュリティ体制を進化させることができる、AWS および AWS パートナーのセキュリティサービスと機能を評価および実装します。AWS セキュリティブログは、新しい AWS サービスおよび機能、実装ガイド、および一般的なセキュリティガイダンスを取り上げます。「AWS の最新情報」は、すべての AWS 機能、サービス、および発表に関する最新情報を確認する優れた方法です。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 定期的なレビューを計画する: コンプライアンス要件、AWS の新しいセキュリティ機能とセキュリティサービスの評価、業界の最新ニュースの入手を含むレビューアクティビティのカレンダーを作成します。
- AWS のサービスと機能について調べる: 使用中のサービスで利用可能なセキュリティ機能について調べ、新しい機能がリリースされた時には、それについて確認します。
  - AWS セキュリティブログ
  - AWS セキュリティ速報

  - AWS のサービスドキュメント
- AWS のサービスの導入プロセスを定義する: 新しい AWS サービスの導入プロセスを定義します。新しい AWS のサービスの機能とワークロードのコンプライアンス要件を評価する方法を含めます。
- 新しいサービスと機能をテストする: 新しいサービスと機能がリリースされたら、本稼働環境に近いかたちで複製する本稼働環境ではない環境でテストします。
- その他の防御メカニズムを実装する: ワークロードを保護するための自動化されたメカニズムを実装し、利用可能なオプションを確認します。
  - AWS Config ルール による非準拠 AWS リソースの修復

リソース

関連動画:

- Well-Architected の手法によるセキュリティのベストプラクティス

ID とアクセス管理

質問

- SEC 2 人とマシンの認証の管理はどのようにすればよいですか?
- SEC 3 人とマシンのアクセス許可はどのように管理すればよいでしょうか?

SEC 2 人とマシンの認証の管理はどのようにすればよいですか?

AWS ワークロードを安全に運用するには、2 種類の ID を管理する必要があります。管理およびアクセス権を付与する必要があるアイデンティティのタイプを理解することで、適切な ID が適切な条件下で適切なリソースにアクセスできるようになります。

ユーザー ID: 管理者、開発者、オペレーター、エンドユーザーは、AWS 環境とアプリケーションにアクセスするために ID を必要とします。これらは、組織のメンバー、または共同作業を行う外部ユーザーであり、ウェブブラウザ、クライアントアプリケーション、またはインタラクティブなコマンドラインツールを介して AWS リソースを操作する人たちです。

マシン ID: サービスアプリケーション、運用ツール、ワークロードには、データの読み取りなどのために、AWS のサービスにリクエストを送信するための ID が必要です。このような ID に

は、Amazon EC2 インスタンスや AWS Lambda 関数など、AWS 環境で実行されているマシンが含まれます。また、アクセスを必要とする外部関係者のマシン ID を管理することもできます。さらに、AWS 環境にアクセスする必要があるマシンが AWS 外にある場合もあります。

ベストプラクティス

- SEC02-BP01 強力なサインインメカニズムを使用する
- SEC02-BP02 一時的な認証情報を使用する
- SEC02-BP03 シークレットを安全に保存して使用する
- SEC02-BP04 一元化された ID プロバイダーを利用する
- SEC02-BP05 定期的に認証情報を監査およびローテーションする
- SEC02-BP06 ユーザーグループと属性を活用する

SEC02-BP01 強力なサインインメカニズムを使用する

パスワードに最小の長さを設定し、よくあるパスワードや再利用を避けるようにユーザーを教育します。ソフトウェアまたはハードウェアのメカニズムを使用した Multi-Factor Authentication (MFA) を強制して、検証を追加します。例えば、IAM アイデンティティセンターをアイデンティティソースとして使用する場合、MFA の「コンテキストアウェア」または「常時オン」設定でユーザーに独自の MFA デバイスの登録を許可すると、すばやく使用開始できます。外部の ID プロバイダー (IdP) を使用する場合は、IdP を MFA 用に設定します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- MFA サインインを強制するアクセス管理 (IAM) ポリシーを作成する: ユーザーが [マイセキュリティ資格情報] ページでロールを引き受け、自分の認証情報を変更し、MFA デバイスを管理できるようにするものを除いて、すべての IAM アクションを禁止するカスタマー管理 IAM ポリシーを 作成します.
- ID プロバイダーで MFA を有効にする: MFA を、アイデンティティプロバイダーや使用する AWS IAM Identity Centerなどのシングルサインオンサービスで有効にします。
- 強力なパスワードポリシーを設定する: 強力なパスワードポリシーを IAM やフェデレーテッド ID システムで設定して、総当たり攻撃から守ります。

- 認証情報を定期的にローテーションする: ワークロードの管理者が、パスワードとアクセスキー (使用されている場合) を定期的に変更するようにします。

リソース

関連するドキュメント:

- AWS Secrets Manager の開始方法
- IAM のベストプラクティス
- ID プロバイダーとフェデレーション
- AWS アカウントのルートユーザー
- AWS Secrets Manager の開始方法
- 一時的なセキュリティ認証情報
- セキュリティパートナーソリューション: アクセスおよびアクセスコントロール
- 一時的なセキュリティ認証情報
- AWS アカウントのルートユーザー

関連動画:

- シークレットを大規模に管理、取得、変更するためのベストプラクティス
- IAM Identity Center を使用した大規模なユーザー権限の管理
- すべての層での ID の把握

SEC02-BP02 一時的な認証情報を使用する

ID を要求して一時的な認証情報を 動的に取得します.ワークフォース ID の場合、AWS IAM Identity Center または AWS Identity and Access Management (IAM) ロールとのフェデレーションを使用して AWS アカウント にアクセスします。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスや AWS Lambda 関数などのマシン ID の場合、長期的なアクセスキーを持つ IAM ユーザーではなく IAM ロールを使用する必要があります。

AWS Management Console を使用するユーザー ID の場合、ユーザーは一時的な認証情報の取得と AWS へのフェデレーションが必要です。これは、AWS IAM Identity Center ユーザーポータルを使用して実行できます。CLI アクセスを必要とするユーザーの場合は、 AWS CLI v2(IAM Identity Center との直接統合をサポートする) を使用していることを確認してくださいユーザーは、IAM アイデンティティセンターアカウントおよびロールにリンクされた CLI プロファイルを作成できます。CLI は AWS の認証情報を IAM Identity Center から自動的に取得し、ユーザーに代わってその情報を更新します。これにより、 コンソールから一時的な AWS の認証情報をコピーして貼り付ける作業を省略できます。SDK については、ユーザーは AWS Security Token Service (AWS STS) を使用して、一

時的な認証情報を取得するロールを引き受ける必要があります。場合によって、一時的な認証情報が使用できないこともあります。アクセスキーを保存するリスクに注意しながら頻繁に更新し、可能なら多要素認証 (MFA) を条件として設定する必要があります。最後に評価した情報を使って、アクセスキーを変更または削除するタイミングを決定します。

AWS リソースへのアクセス権を一般ユーザーに付与する必要がある場合は、 Amazon Cognito ID プールを使用して、AWS リソースにアクセスするための、権限が制限されている一時的な認証情報のセットを割り当てます。各ユーザーの権限は、作成した IAM ロール で制御されます。ユーザーのロールを選択するルールは、ユーザーの ID トークンの登録に基づいて定義します。認証済みユーザーにはデフォルトのロールを定義します。認証されていないゲストユーザーには、制限付きのアクセス権限を持つ IAM ロールを個別に定義できます。

マシン ID に AWS へのアクセス許可を付与するには IAM ロールが必要です。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの場合は、 Amazon EC2 用のロールを使用できます.IAM ロールを Amazon EC2 インスタンスにアタッチすると、Amazon EC2 で実行されているアプリケーションは、AWS がインスタンスメタデータサービス (IMDS) を通して自動的に作成、配布、ローテーションする一時的なセキュリティ認証情報を使用できるようになります。それらの IMDS の最新バージョンを 使用すると、臨時の認証情報を公開する脆弱性から保護できるため、実装する必要があります。キーまたはパスワードを使用して Amazon EC2 インスタンスにアクセスする場合、AWS Systems Manager は、保存されたシークレットなしで、インストール済みエージェントを使用してインスタンスにアクセスし管理するためのより安全な方法として使用できます。さらに、AWS Lambda などの AWS のサービスでは、IAM サービスロールを設定して、一時的な認証情報でサービスに AWS アクションを実行する権限を与えることができます。臨時の認証情報を使用できない状況では、 AWS Secrets Managerのようなプログラムツールを使って、認証情報の変更と管理を自動化します。

定期的に認証情報を監査およびローテーションする: 正しい制御が実施されていることを確認するには、定期的な検証、できれば自動化されたツールによる検証が必要です。ユーザー ID の場合、ユーザーにはパスワードの定期的な変更と、一時的な認証情報を優先したアクセスキーの削除を要求する必要があります。IAM ユーザーから一元化されたアイデンティティに移行しているため、 認証情報レポートを生成して IAM ユーザーを監査できます。また、ID プロバイダーの MFA 設定を矯正することを推奨します。次の AWS Config ルール をセットアップして、これらの設定をモニタリングできます。マシン ID の場合、IAM ロールを使用した一時的な認証情報を使用する必要があります。それが不可能なときは、アクセスキーの監査および更新の頻度を高めることが重要です。

シークレットを安全に保存して使用する: IAM に関連せず、臨時認証情報を利用できない認証情報 (データベースのログインなど) については、シークレット管理用に設計されたサービス Secrets Manager。Secrets Manager では、サポートされているサービスを使用して、暗号化されたシーク

レットの管理、ローテーション、安全な保管を簡単に 行うことができます.シークレットにアクセスするための呼び出しは、監査用に AWS CloudTrail に記録されます。IAM のアクセス許可を使用すれば、それに最小権限のアクセス許可を付与することが可能です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 最小権限ポリシーを実装する: IAM グループおよびロールに最小権限のアクセスポリシーを割り当てて、定義したユーザーのロールまたは機能を反映します。
  - 最小権限を付与する
- 必要でないアクセス許可を削除する: 不要なアクセス許可を削除して、最小権限を実装します。
  - ユーザーアクティビティを確認してポリシーの範囲を削減する
  - ロールアクセスを表示する

- アクセス許可の境界を考慮する: アクセス許可の境界は、アイデンティティベースのポリシーが IAM エンティティに付与できるアクセス許可の上限を設定する管理ポリシーを使用するための高度な機能です。エンティティのアクセス許可の境界では、アイデンティティベースのポリシーとそのアクセス許可の境界の両方で許可されているアクションのみを実行できます。
  - ラボ: ロールの作成を委任する IAM アクセス許可境界
- アクセス許可のリソースタグを検討する: タグを使用して、タグ付けをサポートする AWS リソースへのアクセスを制御できます。また、IAM ユーザーとロールにタグ付けして、ユーザーがアクセスできる内容を制御することもできます。
  - ラボ: EC2 の IAM タグベースのアクセスコントロール
  - Attribute-based access control (ABAC)

リソース

関連するドキュメント:

- AWS Secrets Manager の開始方法
- IAM のベストプラクティス
- ID プロバイダーとフェデレーション
- セキュリティパートナーソリューション: アクセスおよびアクセスコントロール
- 一時的なセキュリティ認証情報

- AWS アカウントのルートユーザー

関連動画:

- シークレットを大規模に管理、取得、変更するためのベストプラクティス
- AWS IAM Identity Center を使用した大規模なユーザー権限の管理
- すべての層での ID の把握

SEC02-BP03 シークレットを安全に保存して使用する

サードパーティー製アプリケーションのパスワードなど、シークレットを必要とするユーザー ID とマシン ID については、最新の業界標準を用いた自動ローテーションで保管します。IAM に関連せず、データベースのログインなど一時的な認証情報を活用できないものについては、AWS Secrets Manager のようなシークレットの管理に対応したサービスを使用します。Secrets Manager では、サポートされているサービスを使用して、暗号化されたシークレットの管理、ローテーション、安全な保管を簡単に行うことができます。 シークレットにアクセスするための呼び出しは、監査用に AWS CloudTrail に記録されます。IAM 権限により、最小限の権限でアクセスできるようにすることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- AWS Secrets Manager を使用する: AWS Secrets Manager は、機密情報の管理を容易にする AWS のサービスです。シークレットとは、データベース認証情報、パスワード、サードパーティ API キー、任意のテキストなどです。

リソース

関連するドキュメント:

- AWS Secrets Manager の開始方法
- ID プロバイダーとフェデレーション

関連動画:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale (シークレットを大規模に管理、取得、変更するためのベストプラクティス)

SEC02-BP04 一元化された ID プロバイダーを利用する

ユーザー ID の場合、ID を一元管理できる ID プロバイダーを利用します。一つの場所から権限の作成、管理、取り消しを行うため、複数のアプリケーションおよびサービスに影響する権限を効率的に管理できます。たとえば誰かが組織を離れる場合、すべてのアプリケーションとサービス (AWS を含む) へのアクセスを一つの場所で取り消すことができます。これにより、複数の認証情報を用意する必要性がなくなり、既存の人事 (HR) プロセスと統合できる可能性が生まれます。

AWS の個別アカウントのフェデレーションでは、AWS Identity and Access Management を使った SAML 2.0 ベースのプロバイダーで AWS の一元化された ID を使用できます。SAML 2.0 プロトコルと互換性のあるプロバイダーであればいずれも使用できます。AWS でホストされているかどうか、AWS 外部にあるかどうか、AWS Partner パートナーネットワーク (APN) から提供されているかどうかは問いません SAML 2.0 ベースのプロバイダーで、 。AWS アカウントと選択したプロバイダーのフェデレーションを使用して、SAML アサーションで一時的なセキュリティ認証情報を取得すれば、ユーザーまたはアプリケーションに AWS API オペレーションを呼び出すアクセス権限を付与できます。ウェブベースのシングルサインオンもサポートされており、ユーザーはサインインウェブサイトから AWS Management Consoleにサインインできます。

AWS Organizations の複数のアカウントへのフェデレーションの場合は、 AWS IAM Identity Center (IAM Identity Center) で ID ソースを設定して、ユーザーとグループの保存場所を指定できます。設定が完了すると、ID プロバイダーが信頼できる ソースになり、 System for Cross-domain Identity Management (SCIM) v2.0プロトコルを利用して情報を同期できます。その後ユーザーまたはグループを検索し、AWS アカウントやクラウドアプリケーションへの IAM Identity Center アクセスを付与できます。

IAM Identity Center は AWS Organizations と統合され、ID プロバイダーを一度設定してから、 組織で管理している既存および新規のアカウントへの アクセス権を付与できます。IAM Identity Center には、ユーザーとグループの管理に使用できるデフォルトストアがあります。IAM Identity Center ストアを使用する場合は、ユーザーとグループを作成してから、最小権限のベストプラクティスに基づきそのアクセスレベルを必要な AWS アカウントとアプリケーションに割り当てます。または、 SAML 2.0 を利用して 外部の ID プロバイダーに 接続するか、 AWS Directory Service を使用して Microsoft AD ディレクトリに接続するかを選択できます。設定が完了したら、一元化された ID プロバイダーで認証すれば、AWS Management Console、AWS モバイルアプリにサインインできるようになります。

モバイルアプリなどのワークロードのエンドユーザー管理には、 Amazon Cognito。このサービスには、ウェブおよびモバイルアプリケーションの認証、承認、ユーザー管理の機能があります。ユーザーは、ユーザー名とパスワードを使用して直接サインインするか、Amazon、Apple、Facebook、Google などのサードパーティーを通じてサインインできます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- 管理アクセスを一元化する: Identity and Access Management (IAM) アイデンティティプロバイダーエンティティを作成して、AWS アカウント とアイデンティティプロバイダー (IdP) 間に信頼される関係を確率します。IAM は、OpenID Connect (OIDC) または SAML 2.0 (Security Assertion Markup Language 2.0) と互換性のある IdP をサポートします。
  - ID プロバイダーとフェデレーション
- アプリケーションアクセスを一元化する: アプリケーションアクセスの一元化には Amazon Cognito を考慮します。ユーザーのサインアップやサインイン、アクセスコントロールをモバイルアプリやウェブアプリに簡単に追加できます。 Amazon Cognito は、数百万人のユーザーに対応し、Facebook、Google、Amazon などのソーシャル ID プロバイダーや、SAML 2.0 によるエンタープライズ ID プロバイダーとのサインインをサポートします。

- 旧 IAM ユーザーおよびグループを削除する: ID プロバイダー (IdP) の使用を開始したら、不要になった IAM ユーザーとグループを削除します。
  - 未使用の認証情報の検索
  - IAM グループの削除

リソース

関連するドキュメント:

- IAM のベストプラクティス
- Security Partner Solutions: Access and Access Control (セキュリティパートナーソリューション: アクセスおよびアクセスコントロール)
- 一時的なセキュリティ認証情報
- AWS アカウントのルートユーザー

関連動画:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale (シークレットを大規模に管理、取得、変更するためのベストプラクティス)
- Managing user permissions at scale with AWS IAM Identity Center (AWS IAM Identity Center を使用した大規模なユーザー権限の管理)

- すべての層での ID の把握

SEC02-BP05 定期的に認証情報を監査およびローテーションする

一時的な認証情報に頼れず、長期的な認証情報が必要な場合は、認証情報を監査して、定義された管理方法、たとえば多要素認証 (MFA) が実施され、定期的にローテーションされ、アクセスレベルが適切であることを確認する必要があります。正しい制御が実施されていることを確認するには、定期的な検証、できれば自動化されたツールによる検証が必要です。ユーザー ID の場合、ユーザーにはパスワードの定期的な変更と、一時的な認証情報を優先したアクセスキーの削除を要求する必要があります。AWS Identity and Access Management (IAM) ユーザーから一元化されたアイデンティティに移行しているため、 認証情報レポートを生成して IAM ユーザーを監査できます。また、ID プロバイダーで MFA 設定を実施することをお勧めします。次の AWS Config ルール をセットアップして、これらの設定をモニタリングできます。マシン ID の場合、IAM ロールを使用した一時的な認証情報を使用する必要があります。それが不可能なときは、アクセスキーの監査および更新の頻度を高めることが重要です。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 認証情報を定期的に監査する: 認証情報レポートと Access Management (IAM) Access Analyzer を使用して、IAM 認証情報とアクセス許可を監査します。
  - IAM Access Analyzer
  - 認証情報レポートの取得
  - ラボ: IAM ユーザーの自動クリーンアップ
- アクセスレベルを使用して IAM アクセス許可を確認する: AWS アカウント のセキュリティを向上させるには、各 IAM ポリシーを定期的に確認してモニタリングします。ポリシーが、必要なアクションのみを実行するために必要な最小権限を付与していることを確認します。
  - アクセスレベルを使用して IAM アクセス許可を確認する
- IAM リソースの作成と更新の自動化を検討する: AWS CloudFormation を使用すると、テンプレートを検証してバージョンを管理できるため、ロールやポリシーを含む IAM リソースのデプロイを自動化して、人為的ミスを減らすことができます。
  - ラボ: IAM グループとロールの自動デプロイ

リソース

関連するドキュメント:

- AWS Secrets Manager の開始方法
- IAM のベストプラクティス
- ID プロバイダーとフェデレーション
- Security Partner Solutions: Access and Access Control (セキュリティパートナーソリューション: アクセスおよびアクセスコントロール)
- 一時的なセキュリティ認証情報

関連動画:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale (シークレットを大規模に管理、取得、変更するためのベストプラクティス)
- Managing user permissions at scale with AWS IAM Identity Center (AWS IAM Identity Center を使用した大規模なユーザー権限の管理)
- すべての層での ID の把握

SEC02-BP06 ユーザーグループと属性を活用する

管理対象のユーザー数が増えるにつれて、大規模な管理ができるユーザー管理方法が必要となります。一般的なセキュリティ要件を持つユーザーを ID プロバイダーで定義したグループに分け、アクセスコントロールに使用される可能性のあるユーザー属性 (部署や場所など) を最新で正確な状態に保つメカニズムを導入します。アクセス制御には、個々のユーザーではなくこのグループと属性を使用します。こうすると、アクセス許可セットを使用してユーザーのグループメンバーシップや属性を一度変更するだけで アクセスを一元管理でき、ユーザーのアクセスに変更が必要なときに多数のポリシーを個別に更新せずに済みます。ユーザーグループや属性の管理に AWS IAM Identity Center (IAM Identity Center)を使用できます。IAM Identity Center は、一般的に使用されている属性に対応しています。ユーザー作成時の手動入力も、クロスドメイン ID 管理システム (SCIM) 仕様などで定義された同期エンジンを使用した自動プロビジョニングも可能です。

一般的なセキュリティ要件を持つユーザーを ID プロバイダーで定義したグループに分け、アクセスコントロールに使用される可能性のあるユーザー属性 (部署や場所など) を最新で正確な状態に保つメカニズムを導入します。アクセスを制御するには、個々のユーザーではなくこれらのグループと属性を使用します。これにより、ユーザーのアクセスニーズが変化したときに多くの個別のポリシーを

更新することなく、ユーザーのグループメンバーシップや属性を 1 回変更することで、アクセスを一元管理できます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- AWS IAM Identity Center (IAM Identity Center)を使用している場合、グループを設定します: IAM Identity Center では、ユーザーのグループを設定し、必要なレベルのアクセス許可をグループに割り当てることができます。
  - AWS シングルサインオン - アイデンティティの管理
- 属性ベースのアクセスコントロール (ABAC) について学ぶ: ABAC は、属性に基づいてアクセス許可を定義する認証戦略です。
  - AWS の ABAC とは
  - ラボ: EC2 の IAM タグベースのアクセスコントロール

リソース

関連するドキュメント:

- AWS Secrets Manager の開始方法
- IAM のベストプラクティス
- ID プロバイダーとフェデレーション
- AWS アカウントのルートユーザー

関連動画:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale (シークレットを大規模に管理、取得、変更するためのベストプラクティス)
- Managing user permissions at scale with AWS IAM Identity Center (AWS IAM Identity Center を使用した大規模なユーザー権限の管理)
- すべての層での ID の把握

関連する例:

- ラボ: EC2 の IAM タグベースのアクセスコントロール

SEC 3 人とマシンのアクセス許可はどのように管理すればよいでしょうか?

アクセス許可を管理して、AWS とワークロードへのアクセスを必要とするユーザー ID やマシン ID へのアクセスを制御します。権限を分けることで、どのような条件で誰が何にアクセスできるかを制御します。

ベストプラクティス

- SEC03-BP01 アクセス要件を定義する
- SEC03-BP02 最小特権のアクセスを付与します
- SEC03-BP03 緊急アクセスのプロセスを確立する
- SEC03-BP04 アクセス許可を継続的に削減する
- SEC03-BP05 組織のアクセス許可ガードレールを定義する
- SEC03-BP06 ライフサイクルに基づいてアクセスを管理する
- SEC03-BP07 パブリックおよびクロスアカウントアクセスの分析
- SEC03-BP08 リソースを安全に共有する

SEC03-BP01 アクセス要件を定義する

ワークロードの各コンポーネントまたはリソースには、管理者、エンドユーザー、またはその他のコンポーネントからアクセスする必要があります。各コンポーネントにアクセスできるユーザーや内容を明確に定義し、適切な ID タイプと認証および承認の方法を選択します。

一般的なアンチパターン:

- シークレットをハードコーディングする、またはアプリケーション内に格納する
- 各ユーザーにカスタムのアクセス許可を付与する
- 永続的な認証情報を使用する

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードの各コンポーネントまたはリソースには、管理者、エンドユーザー、またはその他のコンポーネントからアクセスする必要があります。各コンポーネントにアクセスできるユーザーや内容を明確に定義し、適切な ID タイプと認証および承認の方法を選択します。

組織内の AWS アカウントへの通常のアクセスは、 フェデレーションアクセス または一元化された ID プロバイダーを使用して提供する必要があります。また、アイデンティティ管理を一元化し、AWS へのアクセスを従業員のアクセスライフサイクルに統合するための確立されたプラクティスを整備する必要があります。例えば、従業員がアクセスレベルの異なる職種に異動するときは、そのグループメンバーシップも新しいアクセス要件を反映するように変更される必要があります。

非人間アイデンティティのアクセス要件を定義するときは、どのアプリケーションとコンポーネントがアクセスを必要としているか、またアクセス許可をどのように付与するかを決定します。お勧めのアプローチは、最小特権アクセスモデルで構築された IAM ロールを使用する方法です。AWS マネージドポリシー は、最も一般的なユースケースをカバーする定義済みの IAM ポリシーを提供します。

AWS のサービス (AWS Secrets Manager や AWS Systems Manager パラメータストア) を使用すると、IAM ロールの使用が不可能なケースで、シークレットをアプリケーションやワークロードから安全に切り離すことができます。Secrets Manager では、認証情報の自動ローテーションを確立できます。Systems Manager でパラメータの作成時に指定した一意の名前を使用することで、スクリプト、コマンド、SSM ドキュメント、設定、オートメーションワークフロー内のパラメータを参照できます。

AWS Identity and Access Management Roles Anywhere を使用すると、 IAM 内の一時的なセキュリティ認証情報を取得して、 AWS の外部で実行されるワークロードに使用できます。ワークロードに使用できる IAM ポリシー および IAM ロール は、AWS アプリケーションが AWS リソースにアクセスするために使用するものと同じです。

可能な場合は、長期の静的な認証情報よりも、短期の一時的な認証情報を優先します。IAM ユーザーに、プログラムによるアクセスと長期の認証情報を付与する必要がある場合は、 最後に使用されたアクセスキーの情報を使用し、 アクセスキーのローテーションと削除を行います。

リソース

関連するドキュメント:

- Attribute-based access control (ABAC)
- AWS IAM Identity Center
- IAM Roles Anywhere
- AWS Managed policies for IAM Identity Center (IAM アイデンティティセンター用の AWS マネージドポリシー)
- AWS IAM policy conditions (AWS IAM ポリシー条件)
- IAM ユースケース

- 必要でない認証情報を削除する
- 「IAM ポリシーを管理する」
- How to control access to AWS resources based on AWS アカウント, OU, or organization (AWS アカウント、OU、または組織に基づいて AWS リソースへのアクセスを制御する方法)
- Identify, arrange, and manage secrets easily using enhanced search in AWS Secrets Manager (AWS Secrets Manager の拡張検索を使用してシークレットを容易に特定、調整、管理する)

関連動画:

- Become an IAM Policy Master in 60 Minutes or Less (60 分以内に IAM ポリシーマスターになる)
- Separation of Duties, Least Privilege, Delegation, and CI/CD (職務分離、最小特権、委任、および CI/CD)
- Streamlining identity and access management for innovation (アイデンティティとアクセスの管理を合理化してイノベーションを実現)

SEC03-BP02 最小特権のアクセスを付与します

特定の条件下で特定の AWS リソースに対する特定のアクションを許可して、アイデンティティに必要なアクセスのみを付与します。グループと ID 属性を利用して、個々のユーザーのアクセス許可を定義するのではなく、規模に応じてアクセス許可を動的に設定します。たとえば、開発者のグループに、扱うプロジェクトのリソースのみを管理することを許可できます。これにより、開発者がグループから削除されると、アクセスポリシーに変更を加えることなく、そのグループがどこでアクセスコントロールに使用されたかを問わず、開発者のアクセスが取り消されます。

一般的なアンチパターン:

- デフォルトでユーザーに管理者アクセス許可を付与する
- ルートアカウントを日常業務に使用する

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

最小権限の 原則を設定することで、 特定のタスクを実行するための必要最小限の機能セットのみを実行する許可が ID に与えられるようになり、使いやすさと効率のバランスを取ることができます。この原則を適用すると、意図しないアクセスは制限され、誰がどのリソースにアクセス権限があるか

を監査できます。AWS では、ルートユーザー以外のアイデンティティは、デフォルトではアクセス許可を持ちません。ルートユーザーの認証情報は、厳重に制御し、 特定のタスクにのみ使用する必要があります。

ポリシーを使用して、フェデレーティッド ID、マシン、リソース (S3 バケットなど) が使用する IAM ロールなどの IAM またはリソースエンティティにアタッチされたアクセス許可を明示的に付与できます。ポリシーの作成およびアタッチでは、AWS がアクセスを許可するために必要なサービスアクション、リソース、条件を指定できます。AWS では、アクセスを最小限にするために役立つさまざまな条件を用意しています。例えば、 `PrincipalOrgID` 条件キーを使用すると、AWS Organizations の識別子が検証されるため、AWS 組織内でアクセスを付与できます。

AWS のサービスがユーザーに代わって行うリクエスト (AWS CloudFormation による AWS Lambda 関数の作成など) を制御するには、 `CalledVia` 条件キーを使用します。さまざまなポリシータイプを積み重ねて、アカウント内でアクセス許可すべてを効果的に制限する必要があります。例えば、アプリケーションチームが独自の IAM ポリシーを作成できるようにする一方、 アクセス許可の境界を使用して、そのチームが付与できる最大限のアクセス許可を制限できます。

アクセス許可の管理をスケールして最小特権の原則を遵守するのに役立つ AWS の機能がいくつかあります。属性ベースのアクセスコントロール では、リソースの タグ に基づいてアクセス許可を制限できます。リソースに適用されるタグと呼び出し側の IAM プリンシパルに基づいた認可の決定が可能となります。それにより、タグ付けとアクセス許可ポリシーを組み合わせて、多数のカスタムポリシーを必要としない、きめ細かいリソースアクセスを実現できます。

また、最小特権ポリシーの作成を迅速に行うために、アクティビティの実行後に CloudTrail のアクセス許可に基づいてポリシーを作成することもできます。IAM Access Analyzer は、アクティビティに基づいて IAM ポリシーを自動生成できます。また、IAM アクセスアドバイザーを組織レベルまたは個々のアカウントレベルで使用して 最終アクセスの情報を追跡し、特定のポリシーを適用できます。

これらの詳細を確認して不必要なアクセス許可を削除する頻度を定めます。AWS 組織内でアクセス許可のガードレールを確立し、すべてのメンバーアカウント内で最大限のアクセス許可を制御する必要があります。例えば AWS Control Tower などのサービスには、規範に基づいて管理される予防的コントロールが含まれており、 独自のコントロールを定義できます。

リソース

関連するドキュメント:

- IAM エンティティのアクセス許可の境界

- Techniques for writing least privilege IAM policies (最小特権の IAM ポリシーを作成するテクニック)
- IAM Access Analyzer makes it easier to implement least privilege permissions by generating IAM policies based on access activity (IAM Access Analyzer は、アクセスアクティビティに基づいて IAM ポリシーを生成することにより、最小特権のアクセス許可の実装を容易にする)
- 最終アクセス情報を使用した AWS のアクセス許可の調整
- IAM でのポリシータイプと使用する状況
- IAM Policy Simulator を使用した IAM ポリシーのテスト
- AWS Control Tower のガードレール
- Zero Trust architectures: An AWS perspective (ゼロトラストアーキテクチャ: AWS の視点)
- How to implement the principle of least privilege with CloudFormation StackSets (CloudFormation StackSets を使用して最小特権の原則を実装する方法)

関連動画:

- Next-generation permissions management (次世代のアクセス許可管理)
- Zero Trust: An AWS perspective (ゼロトラスト: AWS の視点)
- How can I use permissions boundaries to limit IAM users and roles to prevent privilege escalation? (アクセス許可の境界を使用して IAM ユーザーとロールの範囲を限定し、権限の昇格を防ぐにはどうすればよいですか?)

関連サンプル:

- ラボ: ロールの作成を委任する IAM アクセス許可の境界

SEC03-BP03 緊急アクセスのプロセスを確立する

自動プロセスまたはパイプラインの問題が発生した場合に、ワークロードへの緊急アクセスを許可するプロセス。これにより、最小権限のアクセスを利用しながら、ユーザーは必要なときに適切なレベルのアクセスを取得できます。例えば、アクセス用の緊急 AWS クロスアカウントロール、または管理者が緊急リクエストの検証と承認を行う際の特定のプロセスなど、管理者がリクエストを確認して承認するプロセスを確立します。

一般的なアンチパターン:

- 既存の ID 設定を使用して停止状態から復旧するための緊急プロセスを整備していない。

- トラブルシューティングや復旧の目的で長期昇格のアクセス許可を付与する。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

緊急アクセスの確立では、複数のケースに備える必要があります。まず、プライマリ ID プロバイダーの障害です。このケースでは、復旧のためのアクセス許可が必須となる第 2 のアクセス方法を用いる必要があります。この方法では、バックアップ ID プロバイダーまたは IAM ユーザーを使用できます。第 2 の方法が用いられる場合には、 厳格に制御、監視され、通知する 必要があります。緊急アクセス ID は、この目的に固有のアカウントに属し、復旧に特化したロールを引き受けるためのアクセス許可のみを持つ必要があります。

また、緊急アクセスのために管理アクセス権の一時的な昇格が求められるケースにも備える必要があります。一般的なシナリオでは、変更のデプロイに使用される自動プロセスへのアクセス許可に変更を加えることを制限します。このプロセスで問題が発生した場合、ユーザーはアクセス許可を復元機能に昇格させることをリクエストしなければならない可能性があります。このケースでは、ユーザーがアクセス権の昇格をリクエストし、管理者が検証して承認することができるプロセスを確立します。アクセスの事前プロビジョニングと緊急の break-glass ロールの設定に関して、ベストプラクティスのガイダンスを説明する実装計画が含まれています SEC10-BP05 アクセスを事前プロビジョニングする。

リソース

関連するドキュメント:

- Monitor and Notify on AWS (AWS アカウントのルートユーザーアクティビティの監視と通知)
- Managing temporary elevated access (アクセス権の一時的な昇格の管理)

関連動画：

- Become an IAM Policy Master in 60 Minutes or Less (60 分以内に IAM ポリシーマスターになる)

SEC03-BP04 アクセス許可を継続的に削減する

チームとワークロードが必要とするアクセスを決定したら、不要になったアクセス許可を削除し、最小権限のアクセス許可を達成するためのレビュープロセスを確立します。未使用の ID とアクセス許可を継続的にモニタリングし、削減します。

チームやプロジェクトが開始した直後には、イノベーションと俊敏性を引き出すため、幅広いアクセス権 (開発またはテスト環境で) の付与を選択する場合があります。アクセス権を継続的に評価し、必要な許可のみにアクセスを制限し、最小権限を付与することをお勧めします。AWS では、未使用のアクセス権を特定するのに役立つアクセス分析機能を提供しています。未使用のユーザー、ロール、アクセス許可、および認証情報を特定しやすくするため、AWS はアクセスアクティビティを分析し、最後に使用されたアクセスキーとロールの情報を提供します。最終アクセスタイムスタンプを使用する と、 未使用のユーザーとロールを識別し、 それらを削除できます。さらに、サービスとアクションの最終アクセス時間情報を確認し、 特定のユーザーおよびロールのアクセス許可を厳密に識別できます.たとえば、最終アクセス時間情報を使用して、アプリケーションロールが必要とする特定の Amazon Simple Storage Service (Amazon S3) アクションを特定し、それらのアクションのみにアクセスを制限できます。これらの機能は、AWS Management Console およびプログラムで使用でき、インフラストラクチャワークフローや自動化ツールに組み込むことができます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- AWS Identity and Access Management (IAM) Access Analyzer を設定する: AWS IAM Access Analyzer は、Amazon Simple Storage Service (Amazon S3) バケットや IAM ロールなど、外部エンティティと共有されている組織のリソースとアカウントを特定するのに役立ちます。
  - AWS IAM Access Analyzer

リソース

関連するドキュメント:

- Attribute-based access control (ABAC)
- 最小権限を付与する
- 必要でない認証情報を削除する
- 「IAM ポリシーを管理する」

関連動画:

- Become an IAM Policy Master in 60 Minutes or Less (60 分以内に IAM ポリシーマスターになる)
- Separation of Duties, Least Privilege, Delegation, and CI/CD (職務分離、最小特権、委任、および CI/CD)

SEC03-BP05 組織のアクセス許可ガードレールを定義する

組織内のすべての ID へのアクセスを制限する共通コントロールを確立します。例えば、特定の AWS リージョン へのアクセスを制限したり、中央セキュリティチームが使用する IAM ロールなどの一般的なリソースをオペレータが削除できないようにしたりできます。

一般的なアンチパターン:

- ワークロードを組織の管理者アカウントで実行する
- 本番稼動のワークロードと非本番稼動のワークロードを同じアカウントで実行する

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

AWS で管理するワークロードの増加に伴い、アカウントを使用してワークロードを分離し、AWS Organizations を使用してそのアカウントを管理する必要があります。組織内のすべての ID へのアクセスを制限するために、共通のアクセス許可ガードレールの確立を推奨しています。例えば、特定の AWS リージョンへのアクセスを制限したり、中央セキュリティチームが使用する IAM ロールなどの共通リソースをチームのメンバーが削除できないようにしたりできます。

これを実行するには、ユーザーによるキーサービスの無効化を防止するなどの、サービスコントロールポリシーの例を実装します。SCP は IAM ポリシー言語を使用し、すべての IAM プリンシパル (ユーザーとロール) が遵守するコントロールを確立します。特定の条件に基づいて、特定のサービスアクションおよびリソースへのアクセスを制限することによって、組織のアクセスコントロールのニーズを満たすことができます。ガードレールには、必要に応じて例外を定義できます。例えば、アカウント内の特定の管理者ロールを除くすべての IAM エンティティに対して、サービスアクションを制限します。

管理アカウントでのワークロードの実行は避けることをお勧めします。管理アカウントは、メンバーアカウントに影響を及ぼすセキュリティガードレールを統制およびデプロイするために使用する必要があります。一部の AWS サービスでは、委任された管理者アカウントの使用がサポートされています。この委任アカウントを使用できる場合は、管理アカウントの代わりに使用する必要があります。組織の管理者アカウントへのアクセスは、厳しく制限する必要があります。

マルチアカウント戦略を用いると、ワークロードにガードレールを適用する際の柔軟性が大幅に向上します。AWS Security Reference Architecture では、アカウント構造の設計方法に関する規範ガイダンスが提供されます。AWS Control Tower などの AWS サービスは、組織全体で予防的コントロールと発見的コントロールの両方を一元管理する機能を提供します。組織内の各アカウントまたは OU の明確な目的を定義し、その目的に沿ってコントロールを制限します。

リソース

関連するドキュメント:

- AWS Organizations
- サービスコントロールポリシー (SCP)
- Get more out of service control policies in a multi-account environment (マルチアカウント環境でサービスコントロールポリシーをさらに活用する)
- AWS Security Reference Architecture (AWS SRA)

関連動画:

- Enforce Preventive Guardrails using Service Control Policies (サービスコントロールポリシーを使用して予防的ガードレールを適用する)
- Building governance at scale with AWS Control Tower (AWS Control Tower を使用したガバナンスの大規模な構築)
- AWS Identity and Access Management deep dive (AWS Identity and Access Management の深堀り)

SEC03-BP06 ライフサイクルに基づいてアクセスを管理する

アクセスコントロールをオペレーター、アプリケーションのライフサイクル、一元化されたフェデレーションプロバイダーと統合します。たとえば、ユーザーが組織を離れるとき、またはロールを変更するときに、ユーザーのアクセス権を削除します。

複数のアカウントでワークロードを管理する場合、それらのアカウント間でリソースを共有するケースがあります。リソースの共有には、 AWS Resource Access Manager (AWS RAM) を使用することをお勧めします.このサービスを使用すると、AWS Organizations 組織および組織単位内で AWS リソースを簡単かつ安全に共有できます。AWS RAM を使用すると、共有されている組織または組織単位内外へのアカウントの移動に伴い、共有リソースへのアクセスの許可または取り消しが自動的に行われます。これで、意図したアカウントのみとのリソースの共有を確実に行えます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ユーザーアクセスのライフサイクル: 新しいユーザーの参加、職務の変更、退職するユーザーに対するユーザーアクセスライフサイクルポリシーを実装して、現在のユーザーのみがアクセスできるようにします。

リソース

関連するドキュメント:

- Attribute-based access control (ABAC)
- 最小権限を付与する
- IAM Access Analyzer
- 必要でない認証情報を削除する
- 「IAM ポリシーを管理する」

関連動画:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD

SEC03-BP07 パブリックおよびクロスアカウントアクセスの分析

パブリックおよびクロスアカウントアクセスに焦点を当てた結果を継続的にモニタリングします。パブリックアクセスとクロスアカウントアクセスを減らして、このタイプのアクセスを必要とするリソースのみへのアクセスに限定します。

一般的なアンチパターン:

- クロスアカウントのアクセスとリソースへのパブリックアクセスを統制するためのプロセスを遵守しない。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

AWS では、別のアカウントにあるリソースへのアクセス権を許可できます。直接的なクロスアカウントアクセスを許可するには、リソース (Amazon Simple Storage Service (Amazon S3) バケットポ

リシーなど) にアタッチされたポリシーを使用するか、アイデンティティが別のアカウントの IAM ロールを引き受けることを許可します。リソースポリシーを使用するときは、組織内のアイデンティティにアクセスが許可されており、リソースをパブリックアクセス可能にする意図があることを確認します。パブリックアクセス可能にする必要があるすべてのリソースを承認するプロセスを定義します。

IAM Access Analyzer は 証明可能セキュリティ を使用して、アカウントの外部からリソースへのすべてのアクセスパスを識別します。また、リソースポリシーの継続的な確認と、パブリックおよびクロスアカウントアクセスの結果の報告により、広範囲なアクセス権の分析をしやすくします。すべてのアカウントについて表示できることを確認するために、AWS Organizations で IAM Access Analyzer を設定することを検討します。IAM Access Analyzer では、 リソースのアクセス許可をデプロイする前に Access Analyzer の調査結果をプレビューすることもできます。これにより、ポリシー変更によって、意図されたパブリックアクセスおよびクロスアカウントアクセスのみがリソースに付与されていることを検証できます。マルチアカウントアクセスについて設計するときは、 ロールを引き受け可能なケースを制御するために信頼ポリシーを使用できます。例えば、ロールの引き受けを特定の送信元 IP 範囲に限定できます。

また、 AWS Config を使用して、 AWS Config ポリシーチェックで、リソースに意図しないパブリックアクセス設定があればレポートを生成し、修復することができます。例えば、 AWS Control Tower および AWS Security Hub などのサービスでは、AWS Organizations 全体でチェックとガードレールのデプロイが簡素化され、パブリックにアクセスできるリソースを特定および修復できます。例えば、AWS Control Tower にはマネージド型のガードレールが含まれており、 Amazon EBS スナップショットがすべての AWS アカウントで復元可能かどうかを検出できます。

リソース

関連するドキュメント:

- AWS Identity and Access Management Access Analyzer を使用する
- AWS Control Tower のガードレール
- AWS Foundational Security Best Practices 標準
- AWS Config マネージドルール
- AWS Trusted Advisor チェックリファレンス

関連動画:

- Best Practices for securing your multi-account environment (マルチアカウント環境を守るためのベストプラクティス)

- Dive Deep into IAM Access Analyzer (IAM Access Analyzer を深堀りする)

SEC03-BP08 リソースを安全に共有する

アカウント間または AWS Organizations 内の共有リソースの消費を管理します。共有リソースをモニタリングし、共有リソースへのアクセスを確認します。

一般的なアンチパターン:

- 第三者にクロスアカウントアクセスを許可する際に、デフォルトの IAM 信頼ポリシーを使用する

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

複数の AWS アカウントを使用してワークロードを管理するとき、アカウント間でのリソースの共有が必要になる場合があります。多くの場合、これは AWS Organizations 内でのクロスアカウント共有です。AWS の複数のサービス (AWS Security Hub、Amazon GuardDuty、 AWS Backup など) には Organizations と統合されたクロスアカウント機能があります。そのため、 AWS Resource Access Manager を使用して、その他の一般的なリソースを共有します。例えば、 VPC サブネットや Transit Gateway アタッチメント、AWS Network Firewall、 Amazon SageMaker Runtime パイプラインなどです。アカウントで Organizations 内のリソースのみを共有する場合は、 サービスコントロールポリシー (SCP) を使用して外部プリンシパルへのアクセスを禁止することをお勧めします。

リソースを共有するとき、意図しないアクセスから保護するための手段を講じる必要があります。アイデンティティベースのコントロールとネットワークコントロールを組み合わせて 組織のデータ境界を作成することをお勧めします。これらのコントロールは、どのリソースが共有可能かを厳格に制限し、共有や公開が許可されるべきでないリソースについてはそれを禁止する必要があります。例えば、データ境界の一部として、VPC エンドポイントポリシーと `aws:PrincipalOrgId` 条件を使用すると、組織に属しているアイデンティティのみが Amazon S3 バケットにアクセスするように設定できます。

場合により、Organizations 外部でリソース共有を許可したり、アカウントへのアクセス権を第三者に付与したりする必要があるかもしれません。例えば、パートナーが提供する監視ソリューションが、貴社のアカウント内のリソースにアクセスする必要があるかもしれません。そのような場合、第三者にとって必要な権限のみを含む IAM クロスアカウントロールを作成する必要があります。また、外部 ID 条件を使用して信頼ポリシーを作成する 必要もあります。外部 ID を使用する際は、第三者それぞれに固有の ID を生成する必要があります。固有の ID は第三者によって提供、制御されるべきではありません。第三者が貴社の環境にアクセスする必要がなくなった場合は、ロールを削除

する必要があります。また、いずれの場合も、第三者に長期的な IAM 認証情報を提供することは避ける必要があります。共有をネイティブにサポートする他の AWS サービスを継続的に把握しておきます。例えば、AWS Well-Architected Tool では、 他の AWS アカウントとワークロードを共有できます。

Amazon S3 などのサービスを使用する際は、 Amazon S3 バケットの ACL を無効にし、 IAM ポリシーを使用してアクセスコントロールを定義することをお勧めします。Amazon S3 オリジンが Amazon CloudFront からアクセスされることを制限するには、オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) に移行します。OAC では AWS KMS を使用したサーバー側暗号化を含む追加機能がサポートされています。を使用します。

リソース

関連するドキュメント:

- バケット所有者が所有権のないオブジェクトへのクロスアカウントアクセス許可を付与する
- How to use Trust Policies with IAM (IAM ロールと信頼ポリシーを使用する方法)
- Building Data Perimeter on AWS (AWS でのデータ境界の構築)
- AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法

関連動画:

- Granular Access with AWS Resource Access Manager (AWS Resource Access Manager を使用したきめ細かいアクセス)
- Securing your data perimeter with VPC endpoints (VPC エンドポイントを使用したデータ境界の保護)
- Establishing a data perimeter on AWS (AWS でのデータ境界の確立)

検知

質問

- SEC 4 セキュリティイベントは、どのように検出して調査するのですか?

SEC 4 セキュリティイベントは、どのように検出して調査するのですか?

ログやメトリクスからイベントを可視化して把握し、分析します。セキュリティイベント、および潜在的な脅威に対する措置を講じて、ワークロードの保護に役立てます。

ベストプラクティス

- SEC04-BP01 サービスとアプリケーションのログ記録を設定する
- SEC04-BP02 ログ、結果、メトリクスを一元的に分析する
- SEC04-BP03 イベントへの応答を自動化する
- SEC04-BP04 実用的なセキュリティイベントを実装する

SEC04-BP01 サービスとアプリケーションのログ記録を設定する

アプリケーションログ、リソースログ、AWS のサービスログなど、ワークロード全体でログ記録を設定します。例えば、組織内のすべてのアカウントで AWS CloudTrail、Amazon CloudWatch Logs、Amazon GuardDuty および AWS Security Hub が有効になっていることを確認します。

基本的なプラクティスは、アカウントレベルで一連の検出メカニズムを確立することです。この基本的なメカニズムセットは、アカウント内のすべてのリソースに対する幅広いアクションを記録および検出することを目的としています。これらを使用すると、自動修復を含むオプションを備えた包括的な検出機能、および機能を追加するためのパートナー統合を構築できます。

AWS では、この基本セットを実装できるサービスには以下が含まれます。

- AWS CloudTrail では、AWS Management Console、AWS SDK、コマンドラインツールなどの AWS のサービスを通じて実行されたアクションを含む、AWS アカウントアクティビティのイベント履歴を提供します。
- AWS Config では、AWS リソース構成のモニタリングと記録が行われ、目標の構成に対する評価と修復が自動化できます。
- Amazon GuardDuty は脅威検出サービスです。悪意のある動作や不正な動作を継続的にモニタリングし、AWS のアカウントとワークロードを保護できるようにします。
- AWS Security Hub では、複数の AWS のサービスや任意のサードパーティー製品からのセキュリティアラートまたは検出結果の集約、整理、優先順位付けが一元的に行われ、セキュリティアラートとコンプライアンスステータスを包括的に把握できます。

アカウントレベルの基盤上に構築されている多くの中心的なAWSのサービスである Amazon Virtual Private Cloud Console (Amazon VPC)サービスレベルのログ記録機能を提供します。Amazon VPC フローログ を使用すると、ネットワークインターフェイスを出入りする IP トラフィックに関する情報をキャプチャし、接続履歴に関する貴重なインサイトを得て、異常な動作に基づいた自動アクションをトリガーできます。

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスや AWS のサービスから生成されないアプリケーションベースのログ記録の場合、ログは を使用して保存、分析できます。 Amazon CloudWatch Logs。クラウド エージェント は、オペレーティングシステムと実行中のアプリケーションからログを収集し、自動的に保存します。ログがCloudWatch Logsで利用可能になったら、リアルタイムで処理したり、 (CloudWatch Logs Insights) を使用して分析したりできます。

ログの収集や集約と同様に重要な機能が、複雑なアーキテクチャによって生成される大量のログとイベントデータから意味のある情報を抽出する機能です。詳細については、 モニタリング セクションにある 信頼性の柱に関するホワイトペーパーを参照してください 。CloudWatch Logs エージェントがキャプチャするログファイルにアプリケーションデータが誤って入ってきた場合や、クロスリージョンロギングがログ集約用に設定されていて、特定の種類の情報を国境を越えて送付することに関する法的な考慮事項がある場合など、ログ自体に機密とみなされるデータが含まれる可能性があります。

1 つのアプローチとして、ログの配信時にイベントでトリガーされる AWS Lambda 関数を使用して、Amazon Simple Storage Service (Amazon S3) バケットなどの中央ログ記録の場所に転送する前にログデータをフィルタリングおよび編集することがあります。未編集のログは、法律および法務チームが定める「妥当な時間」が経過するまでローカルバケットに保持できます。経過した時点で、Amazon S3 ライフサイクルルールが自動的にログを削除できます。S3 Object Lock を使用して、Amazon S3 でログの保護を強化できます。 Amazon S3 Object LockWrite-Once-Read-Many (WORM) モデルを使用してオブジェクトを保存できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- AWS のサービスのログ記録を有効にする: 要求事項を遵守するため AWS のサービスのログ記録を有効にします。ログ記録機能には次のようなものがあります: Amazon VPC Flow Logs、Elastic Load Balancing (ELB) ログ、Amazon S3 バケットログ、CloudFront アクセスログ、Amazon Route 53 クエリログ、および Amazon Relational Database Service (Amazon RDS) ログ。
  - AWS 回答: AWS ネイティブのセキュリティロギング機能
- オペレーティングシステムとアプリケーションごとのログ機能を評価して有効にし、不審な動作を検出します。
  - CloudWatch Logs の開始方法
  - デベロッパー用ツールおよびログ分析
- ログに適切なコントロールを適用する: ログには機密情報が含まれている場合があるため、承認されたユーザーにのみログへのアクセス権を与えるようにします。Amazon S3 バケットと CloudWatch Logs のロググループに対するアクセス権を制限することを検討します。

  - Amazon CloudWatch のための認証とアクセスコントロール
  - Amazon S3 のアイデンティティとアクセスの管理
- 設定 Amazon GuardDuty: GuardDuty は脅威検出サービスです。悪意のある動作や不正な動作を継続的に探し、AWS アカウント とワークロードを保護できるようにします。GuardDuty を有効にし、ラボを使用して E メールの自動アラートを設定します。
- CloudTrail でカスタマイズされた証跡を設定する: 証跡を設定するとデフォルトの期間よりも長くログを保存し、後で分析できます。
- 実現 AWS Config: AWS Config は、AWS アカウント アカウントの AWS リソースの設定を詳細に表示します。このビューには、リソース間の関係と設定の履歴が含まれるため、時間の経過とともに設定と関係がどのように変わるかを確認できます。
- 実現 AWS Security Hub: Security Hub では、AWS のセキュリティ状態の包括的なビューが提供され、セキュリティ業界の標準とベストプラクティスへの準拠を確認するのに役立ちます。Security Hub を使用すると、AWS アカウント、サービス、およびサポートしているサードパーティーのパートナー製品全体からセキュリティデータを収集し、セキュリティの傾向を分析して、最も優先度の高いセキュリティ問題を特定できます。

リソース

関連するドキュメント:

- Amazon CloudWatch
- Amazon EventBridge
- 開始方法: Amazon CloudWatch Logs
- セキュリティパートナーのソリューション: ログ記録とモニタリング

関連動画:

- リソースの設定とコンプライアンスを一元的にモニタリングする
- Amazon GuardDuty および AWS Security Hub の調査結果の修復
- クラウドにおける変更管理: Amazon GuardDuty および AWS Security Hub

関連する例:

- ラボ: 発見的統制の自動デプロイ

SEC04-BP02 ログ、結果、メトリクスを一元的に分析する

セキュリティ運用チームは、ログを収集し、検索ツールを使用することによって、不正なアクティビティや意図しない変更の可能性がある、潜在的に関心のあるイベントを発見します。ただし、収集されたデータを分析して手動で情報を処理するだけでは、複雑なアーキテクチャから流れる大量の情報に対応するには不十分です。分析とレポートだけでは、適切なリソースを割り当てて、イベントをタイミング良く実行する作業が容易になる訳ではありません。

熟練したセキュリティオペレーションチームを構築するには、セキュリティイベントと調査結果の流れを、チケットシステム、バグまたは問題システム、その他のセキュリティ情報とイベント管理 (SIEM) システムなどの、通知およびワークフローシステムに深く統合することをお勧めします。これにより、メールや静的レポートからワークフローが排除され、イベントや調査結果のルーティング、エスカレート、管理が可能になります。多くの組織はセキュリティアラートをチャットまたはコラボレーションや開発者の生産性プラットフォームに統合しています。自動化に着手している組織は、API 主導の、低レイテンシーのチケット発行システムによって、「何を最初に自動化するか」を計画する際にかなりの柔軟性が得られます。

このベストプラクティスは、ユーザーアクティビティやネットワークイベントを示すログメッセージから生成されたセキュリティイベントだけでなく、インフラストラクチャ自体で検出された変更から生成されたセキュリティイベントにも適用できます。変更による悪影響が小さく、AWS Identity and Access Management (IAM) と AWS Organizations の設定の組み合わせではその実行を阻止できないような状況では、変更を検出し、変更が適切かどうかを判断し、その情報を正しい修復ワークフローにルーティングする機能が、安全なアーキテクチャを維持、検証するうえで不可欠です。

Amazon GuardDuty と AWS Security Hub は、他の AWS のサービスでも利用できるログレコードの集約、重複排除、分析メカニズムを提供します。GuardDuty は、AWS CloudTrail 管理やデータイベント、VPC DNS ログ、および VPC Flow Logs などのソースからの情報を取込み、集計し、分析します。Security Hub は、GuardDuty、AWS Config、Amazon Inspector、Amazon Macie、AWS Firewall Manager、および AWS Marketplace で利用できるかなりの数のサードパーティーセキュリティ製品、そして適切にビルドした場合は独自のコードからの出力を取込み、集計、分析できます。GuardDuty と Security Hub のどちらにも、複数のアカウントにわたって調査結果とインサイトを集約できるマスターメンバーモデルがあります。Security Hub は、オンプレミスの SIEM を導入しているお客様に AWS 側のログ/アラートのプリプロセッサ/アグリゲータとしてよく使用され、お客様はそこから AWS Lambda ベースのプロセッサとフォワーダーを介して Amazon EventBridge を取り込むことができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ログ処理機能を評価する: ログの処理に使用できるオプションを評価します。
  - Amazon OpenSearch Service を使用して (ほぼ) あらゆる対象をログ記録およびモニタリングする
  - ログ記録および分析ソリューションを専門とするパートナーを探す
- CloudTrail ログの分析の最初の作業として Amazon Athena をテストする
  - CloudTrail ログを分析するように Athena を設定する
- AWS での集中ロギングを実装する: 複数のソースからのログ記録を一元化する次の AWS のサンプルソリューションを参照してください。
  - 集中ロギングソリューション
- パートナーで集中ロギングを実装する: APN パートナーは、ログを一元的に分析するためのソリューションを提供しています。
  - ログ記録とモニタリング

リソース

関連するドキュメント:

- AWS の回答: 集中ログ記録
- AWS Security Hub
- Amazon CloudWatch
- Amazon EventBridge
- 開始方法: Amazon CloudWatch Logs
- セキュリティパートナーのソリューション: ログ記録とモニタリング

関連動画:

- リソースの設定とコンプライアンスを一元的にモニタリングする
- Amazon GuardDuty および AWS Security Hub の調査結果の修復
- クラウドにおける変更管理: Amazon GuardDuty および AWS Security Hub

SEC04-BP03 イベントへの応答を自動化する

自動化を使用してイベントを調査および修正することで、人為的な労力やエラーが軽減され、調査機能をスケールできます。定期的なレビューは、自動化ツールを調整するのに役立ちます。継続して繰り返し行います。

AWS では、Amazon EventBridge を使用して、関心のあるイベントと予期しない変更についての情報の調査を自動化されたワークフローに組み込むことができます。このサービスには、スケーラブルなルールエンジンが備わっており、ネイティブの AWS イベント形式 (AWS CloudTrail イベントなど)と、独自のアプリケーションから生成できるカスタムイベントの両方を仲介できます。Amazon GuardDuty では、インシデントレスポンスシステムを構築するワークフローシステム (AWS Step Functions) や、中央のセキュリティアカウントにイベントをルーティングできます。また、バケットにルーティングして詳細分析を実行することもできます。

変更を検出してこの情報を正しいワークフローにルーティングするには、AWS Config ルール と コンフォーマンスパックを使用できます。AWS Config は、(EventBridge より高いレイテンシーを通して) スコープ内サービスへの変更を検出し、AWS Config ルール ルールを使用して分析できるイベントを生成します。分析されたイベントは、ロールバックやコンプライアンスポリシーの適用のほか、変更管理プラットフォームや運用チケット発行システムなどのシステムに対する情報の転送に使用できます。AWS Config イベントに対応する独自の Lambda 関数を作成するだけでなく、 AWS Config ルール 開発キットおよび オープンソースライブラリの AWS Config ルール も利用できます。コンフォーマンスパックとは、YAML テンプレートとして作成される単一エンティティとしてデプロイする一連の AWS Config ルール および修正アクションです。A サンプルコンフォーマンスパックテンプレートは Well-Architected セキュリティの柱で利用できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- GuardDuty で自動アラートを実装する: GuardDuty は、脅威検出サービスです。悪意のあるアクティビティや不正な動作を継続的にモニタリングし、AWS アカウント とワークロードを保護します。GuardDuty を有効にし、自動アラートを設定します。
- 調査プロセスを自動化する: 時間を節約するため、イベントを調査して情報を管理者に報告する自動化されたプロセスを開発します。
  - ラボ: Amazon GuardDuty ハンズオン

リソース

関連するドキュメント:

- AWS の回答: 集中ログ記録
- AWS Security Hub
- Amazon CloudWatch
- Amazon EventBridge
- 開始方法: Amazon CloudWatch Logs
- セキュリティパートナーのソリューション: ログ記録とモニタリング
- Amazon GuardDuty の設定

関連動画:

- リソースの設定とコンプライアンスを一元的にモニタリングする
- Amazon GuardDuty および AWS Security Hub の調査結果の修復
- クラウドにおける変更管理: Amazon GuardDuty および AWS Security Hub

関連する例:

- ラボ: 発見的統制の自動デプロイ

SEC04-BP04 実用的なセキュリティイベントを実装する

チームに送信され、チームによるアクションが可能なアラートを作成します。チームがアクションを実行するための関連情報がアラートに含まれていることを確認します。使用する検知メカニズムごとに、ランブック または プレイブック形式の調査プロセスも用意する必要があります。例えば、Amazon GuardDutyを有効にすると、さまざまな調査結果が生成されます。。調査結果タイプごとにランブックエントリが必要です。例えば、トロイの木馬 が検出された場合、調査して修復するよう指示する簡単な説明をランブックに記載する必要があります。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- AWS のサービスで利用可能なメトリクスを検出する: Amazon CloudWatch で利用可能な、利用中のサービスのメトリクスを確認することができます。
  - AWS のサービスドキュメント
  - Using Amazon CloudWatch Metrics
- Amazon CloudWatch アラームを設定します。

- Amazon CloudWatch でのアラームの使用

リソース

関連するドキュメント:

- Amazon CloudWatch
- Amazon EventBridge
- Security Partner Solutions: Logging and Monitoring (セキュリティパートナーのソリューション: ログ記録とモニタリング)

関連動画:

- Centrally Monitoring Resource Configuration and Compliance (リソースの設定とコンプライアンスを一元的にモニタリングする)
- Remediating Amazon GuardDuty and AWS Security Hub Findings (Amazon GuardDuty および AWS Security Hub の調査結果の修復)
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub (クラウドにおける変更管理: Amazon GuardDuty および AWS Security Hub)

インフラストラクチャ保護

質問

- SEC 5 ネットワークリソースをどのように保護しますか?
- SEC 6 どのようにコンピューティングリソースを保護するのですか?

SEC 5 ネットワークリソースをどのように保護しますか?

何らかの形式のネットワーク接続があるワークロードは、インターネットでもプライベートネットワークでも、外部および内部ネットワークベースの脅威から保護するために、複数の防御レイヤーが必要です。

ベストプラクティス

- SEC05-BP01 ネットワークレイヤーを作成する
- SEC05-BP02 すべてのレイヤーでトラフィックを制御する

- SEC05-BP03 ネットワーク保護を自動化する
- SEC05-BP04 検査と保護を実装する

SEC05-BP01 ネットワークレイヤーを作成する

到達可能性要件をレイヤーに共有するコンポーネントをグループ化します。例えば、インターネットアクセスを必要としない仮想プライベートクラウド (VPC) 内のデータベースクラスターは、インターネットへのルート、またはインターネットからのルートがないサブネットに配置する必要があります。VPC を使用せずに稼働するサーバーレスワークロードでは、マイクロサービスを使用した同様の階層化とセグメント化でも同じ目標を達成できます。

共通の達成可能要件を持つ Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Amazon Relational Database Service (Amazon RDS) データベースクラスター、AWS Lambda 関数などのコンポーネントは、サブネットで形成されるレイヤーにセグメント化できます。例えば、インターネットアクセスを必要としない VPC 内の Amazon RDS データベースクラスターは、インターネットへのルート、またはインターネットからのルートがないサブネットに配置する必要があります。このコントロールに対する階層的なアプローチは、意図しないアクセスを許可する可能性がある単一レイヤーの誤設定の影響を軽減します。Lambda の場合は、VPC 内で関数を実行して、VPC ベースのコントロールを利用できます。

数千の VPC、AWS アカウント、オンプレミスネットワークを含むネットワーク接続の場合は、AWS Transit Gateway を使用する必要があります。 AWS Transit Gateway.AWS Transit Gateway は、スポークのように機能するすべての接続されたネットワーク間でトラフィックがどのようにルーティングされるかを制御するハブとして機能します。Amazon Virtual Private Cloud と AWS Transit Gateway の間のトラフィックは、AWS プライベートネットワーク上にとどまります。これにより、分散型サービス妨害 (DDoS) 攻撃や一般的な脆弱性攻撃 (SQL インジェクション、クロスサイトスクリプティング、クロスサイトリクエストフォージェリ、破損した認証コードの不正使用など) といった外部からの脅威ベクトルが軽減されます。AWS Transit Gateway のリージョン間ピアリングはまた、リージョン間トラフィックを単一障害点や帯域幅のボトルネックなしで暗号化します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- VPC にサブネットを作成する: (複数のアベイラビリティーゾーンを含むグループで) 各レイヤーのサブネットを作成し、ルートテーブルを関連付けてルーティングを制御します。
  - VPC とサブネット

- ルートテーブル

リソース

関連するドキュメント:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC セキュリティ
- AWS WAF の開始方法

関連動画:

- 多くの VPC 用の AWS Transit Gateway リファレンスアーキテクチャ
- Amazon CloudFront、AWS WAF、AWS Shield によるアプリケーションの高速化と保護

関連する例:

- ラボ: VPC の自動デプロイ

SEC05-BP02 すべてのレイヤーでトラフィックを制御する

ネットワークトポロジを設計する際には、各コンポーネントの接続要件を調べる必要があります。たとえば、コンポーネントがインターネットアクセス (インバウンドおよびアウトバウンド) や、VPC、エッジサービス、外部データセンターへの接続を必要とする場合です。

VPC では、設定したプライベート IPv4 アドレス範囲または AWS によって選択された IPv6 アドレス範囲を使用して、AWS リージョン にまたがるネットワークトポロジを定義できます。インバウンドトラフィックとアウトバウンドトラフィックの両方に、多層防御アプローチを用いた複数のコントロールを適用する必要があります。これには、セキュリティグループ (ステートフルインスペクションファイアウォール)、ネットワーク ACL、サブネット、ルートテーブルの使用などが含まれます。VPC 内では、アベイラビリティーゾーンにサブネットを作成できます。各サブネットには、トラフィックがサブネット内でたどるパスを管理するためのルーティングルールを定義するルートテーブルを関連付けることができます。インターネットまたは VPC にアタッチされた NAT あるいは他の VPC ゲートウェイを経由するルートを設定することで、インターネットルーティングが可能なサブネットを定義できます。

インスタンス、Amazon Relational Database Service (Amazon RDS) データベース、またはその他のサービスが VPC 内で起動されると、ネットワークインターフェイスごとに独自のセキュリティグループが設定されます。このファイアウォールはオペレーティングシステムレイヤーの外側にあり、許可されるインバウンドトラフィックとアウトバウンドトラフィックのルールを定義するために使用できます。また、セキュリティグループ間の関係も定義できます。たとえば、データベース層のセキュリティグループ内のインスタンスは、関連するインスタンスに適用されるセキュリティグループを参照して、アプリケーション層のインスタンスからのトラフィックのみを受け入れます。TCP 以外のプロトコルを使用している場合を除き、ロードバランサーや CloudFront なしでインターネットから Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに直接アクセスできるようにする必要はありません (セキュリティグループによって制限されているポートでも)。 これにより、オペレーティングシステムやアプリケーションの問題による意図しないアクセスから保護できます。サブネットには、ステートレスファイアウォールとして機能する、サブネットにアタッチされたネットワーク ACL を設定することもできます。レイヤー間で許可されるトラフィックの範囲を絞り込むようにネットワーク ACL を設定する必要があります。インバウンドルールとアウトバウンドルールの両方を定義する必要があることに注意してください。

一部の AWS サービスは、インターネットにアクセスして API 呼び出しをする (AWS API エンドポイント がある) ためのコンポーネントが必要です。その他の AWS サービスは VPC エンドポイント を Amazon VPC 内で使用します。Amazon S3 や Amazon DynamoDB を含む多くの AWS サービスは VPC エンドポイントをサポートしており、このテクノロジーは次で一般化されています。 AWS PrivateLink.AWS のサービス、サードパーティーのサービス、および他の VPC セキュリティでホストされる独自のサービスにアクセスするには、このアプローチを使用することが推奨されます。AWS PrivateLink のすべてのネットワークトラフィックは、グローバルな AWS バックボーンにとどまり、インターネットにトラバースすることはありません。接続を開始できるのは、サービスのプロバイダーではなくサービスのコンシューマーのみです。外部サービスアクセスに AWS PrivateLink を使用することにより、インターネットなしで エアギャップ VPC を作成することができるため、外部の脅威ベクトルから VPC を保護するのに役立ちます。サードパーティーのサービスは AWS PrivateLink を使用して、プライベート PI アドレス経由で顧客が VPC からサービスに接続できるようにします。インターネットへのアウトバウンド接続を必要とする VPC アセットでは、これらは、AWS が管理する NAT ゲートウェイ、アウトバウンド専用のインターネットゲートウェイ、ユーザーが作成して管理するウェブプロキシを経由するアウトバウンド (一方向) でのみ可能です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- VPC 内のネットワークトラフィックを制御する: VPC ベストプラクティスを実装してトラフィックを制御する

  - Amazon VPC セキュリティ
  - VPC エンドポイント
  - Amazon VPC セキュリティグループ
  - ネットワーク ACL
- エッジでのトラフィックを制御する: Amazon CloudFront などのエッジサービスを実装して、追加の保護レイヤーやその他の機能を提供します。
  - Amazon CloudFront ユースケース
  - AWS Global Accelerator
  - AWS Web Application Firewall (AWS WAF)
  - Amazon Route 53
  - Amazon VPC Ingress Routing
- プライベートネットワークトラフィックを制御する: ワークロードのプライベートトラフィックを保護するサービスを実装します。
  - Amazon VPC ピアリング
  - Amazon VPC エンドポイントサービス (AWS PrivateLink)
  - Amazon VPC トランジットゲートウェイ
  - AWS Direct Connect
  - AWS サイト間 VPN
  - AWS クライアント VPN
  - Amazon S3 Access Points

リソース

関連するドキュメント:

- AWS Firewall Manager
- Amazon Inspector
- AWS WAF の開始方法

関連動画:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, andAWS Shield

関連する例:

- Lab: Automated Deployment of VPC

SEC05-BP03 ネットワーク保護を自動化する

保護メカニズムを自動化し、脅威インテリジェンスと異常検出に基づく自己防御型ネットワークを提供します。たとえば、現在の脅威に適応し、その影響を軽減できる侵入検知および防止ツールなどです。ウェブアプリケーションファイアウォールは、ネットワーク保護を自動化できる例の 1 つです。たとえば、AWS WAF セキュリティの自動化ソリューション (https://github.com/awslabs/aws-waf-security-automationsを使用して、既知の脅威アクターに関連付けられた IP アドレスからのリクエストを自動的にブロックします。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- ウェブベースのトラフィックの保護を自動化する: AWS では、AWS CloudFormation を使用して、一般的なウェブベースの攻撃をフィルタリングするために設計された AWS WAF ルールセットを自動的にデプロイするソリューションを提供しています。ユーザーは、AWS WAF ウェブアクセスコントロールリスト (ウェブ ACL) に含まれるルールを定義する、あらかじめ設定された保護機能から選択することができます。
  - AWS WAF のセキュリティオートメーション
- AWS Partner ソリューションを検討する: AWS パートナーは、お客様のオンプレミス環境にある既存のコントロールと同等または統合された、業界をリードする何百もの製品を提供しています。これらの製品は、既存の AWS サービスを補完し、包括的なセキュリティアーキテクチャの導入と、クラウドとオンプレミス環境におけるよりシームレスなエクスペリエンスを実現します。
  - インフラストラクチャのセキュリティ

リソース

関連するドキュメント:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC のセキュリティ
- AWS WAF の開始方法

関連動画:

- AWS Transit Gateway reference architectures for many VPCs (多くの VPC 用の AWS Transit Gateway リファレンスアーキテクチャ)
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield (Amazon CloudFront、AWS WAF、AWS Shield によるアプリケーションの高速化と保護)

関連する例:

- ラボ: VPC の自動デプロイ

SEC05-BP04 検査と保護を実装する

各レイヤーでトラフィックを検査し、フィルタリングします。VPC の設定に潜在的な意図しないアクセスの可能性がないかを検査するには、 VPC Network Access Analyzer を使用できます。. ネットワークアクセス要件を指定して、それを満たさない潜在的なネットワークパスを特定できます。HTTP ベースのプロトコルを介してトランザクションを実行するコンポーネントの場合、一般的な攻撃からの保護にはウェブアプリケーションファイアウォールが役立ちます。 AWS WAF は、Amazon API Gateway API、Amazon CloudFront、または Application Load Balancer に転送される設定可能なルールに一致する HTTP リクエストを監視してブロックできるウェブアプリケーションファイアウォールです。AWS WAF の使用を開始するには 、 AWS マネージドルール を独自のルールと組み合わせて使用するか、既存の パートナー統合を使用できます。 。

AWS Organizations 全体にわたって AWS WAF、AWS Shield Advanced による保護、Amazon VPC セキュリティグループを管理するには、AWS Firewall Manager を使用できます。AWS Firewall Manager を使用すると、アカウントとアプリケーション全体にわたってファイアウォールルールを一元的に設定および管理できるため、一般的なルールの適用を簡単に拡張できます。また、 AWS Shield Advanced、または ソリューション を使用して、攻撃に迅速に対応できます。これらは、ウェブアプリケーションへの不要なリクエストを自動的にブロックします。Firewall Manager は、 AWS ネットワークファイアウォールとも併用できます。.AWS ネットワークファイアウォールは、ルールエンジンを使用して、ステートフルとステートレスの両方のネットワークトラフィックを細かくコントロールするマネージドサービスです。ルールに対しては Suricata 対応の オープンソース侵入防止システム (IPS) 仕様がサポートされており、ワークロードの保護に役立ちます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- Amazon GuardDuty を設定する: GuardDuty は、脅威検出サービスです。悪意のあるアクティビティや不正な動作を継続的にモニタリングし、AWS アカウント とワークロードを保護します。GuardDuty を有効にし、自動アラートを設定します。
  - Amazon GuardDuty
  - ラボ: 発見的統制の自動デプロイ
- 仮想プライベートクラウド (VPC) フローログを設定する: VPC フローログは、VPC のネットワークインターフェイス間を行き来する IP トラフィックに関する情報をキャプチャできるようにする機能です。フローログデータは Amazon CloudWatch Logs および Amazon Simple Storage Service (Amazon S3) にパブリッシュできます。フローログを作成した後、選択した送信先でデータを取得したり表示したりできます。
- VPC トラフィックのミラーリングを検討する: トラフィックミラーリングは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの Elastic Network Interface からネットワークトラフィックをコピーし、コンテンツ検査、脅威のモニタリング、トラブルシューティングのために帯域外セキュリティおよびモニタリングアプライアンスに送信するために使用できる Amazon VPC の機能です。
  - VPC トラフィックミラーリング

リソース

関連するドキュメント:

- AWS Firewall Manager
- Amazon Inspector
- Amazon VPC セキュリティ
- AWS WAF の開始方法

関連動画:

- AWS Transit Gateway reference architectures for many VPCs (多くの VPC 用の AWS Transit Gateway リファレンスアーキテクチャ)
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield (Amazon CloudFront、AWS WAF、AWS Shield によるアプリケーションの高速化と保護)

関連する例:

- ラボ: VPC の自動デプロイ

SEC 6 どのようにコンピューティングリソースを保護するのですか?

ワークロード内のコンピューティングリソースを内外の脅威から守るには、複数の防御レイヤーを設ける必要があります。コンピューティングリソースには、EC2 インスタンス、コンテナ、AWS Lambda 関数、データベースサービス、IoT デバイスなどがあります。

ベストプラクティス

- SEC06-BP01 脆弱性管理を実行する
- SEC06-BP02 攻撃対象領域を縮小する
- SEC06-BP03 マネージドサービスを活用する
- SEC06-BP04 コンピューティング保護を自動化する
- SEC06-BP05 ユーザーがリモートからアクションを実行できるようにする
- SEC06-BP06 ソフトウェアの整合性を検証する

SEC06-BP01 脆弱性管理を実行する

コード、依存関係、インフラストラクチャ内の脆弱性のスキャンとパッチ適用を頻繁に実施し、新しい脅威から保護します。

コンピューティングインフラストラクチャの設定から始め、AWS CloudFormation を使用してリソースの作成と更新を自動化できます。CloudFormation を使うと、AWS の例を使用するか、または自分で記述することにより、YAML または JSON で書かれたテンプレートを作成できます。これにより、 CloudFormation Guardで検証できるデフォルトで保護されたインフラストラクチャを作成できるため、時間を節約して設定エラーのリスクを低減できます。インフラストラクチャをビルドして、 AWS CodePipelineなどの継続的デリバリーを使ってアプリケーションをデプロイすることで、ビルド、テスト、およびリリースを自動化できます。

Amazon Elastic Compute Cloud(Amazon EC2) インスタンス、Amazon マシンイメージ (AMI)、およびその他多くのコンピューティングリソースなど、AWS リソースのパッチ管理を行う責任があります。Amazon EC2 インスタンスの場合、AWS Systems Manager Patch Manager は、セキュリティ関連および他のタイプの更新の両方を使用して、マネージドインスタンスにパッチを適用するプロセスを自動化します。Patch Manager を使用して、オペレーティングシステムとアプリ

ケーションの両方にパッチを適用できます。(Windows サーバーでは、アプリケーションサポートは Microsoft アプリケーションの更新に限定されます)。 Patch Manager を使用して、Service Packs on Windows インスタンスをインストールし、Linux インスタンスのマイナーバージョンアップグレードを実行します。オペレーティングシステムのタイプ別に、Amazon EC2 インスタンス、オンプレミスのサーバー、仮想マシン (VM) のフリートにパッチを適用できます。これには、Windows Server、Amazon Linux、Amazon Linux 2、CentOS、Debian Server、Oracle Linux、Red Hat Enterprise Linux (RHEL)、SUSE Linux Enterprise Server (SLES)、および Ubuntu Server に対応するバージョンが含まれます。インスタンスをスキャンして、不足しているパッチのレポートのみを表示したり、不足しているすべてのパッチをスキャンして自動的にインストールしたりできます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- Amazon Inspector を設定する: Amazon Inspector は、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのネットワークアクセシビリティと、それらのインスタンスで実行されるアプリケーションの状態をテストします。Amazon Inspector は、アプリケーションの露出、脆弱性、ベストプラクティスからの逸脱を評価します。
  - Amazon Inspector とは何ですか?
- ソースコードをスキャンする: ライブラリや依存関係をスキャンして脆弱性に対応します。
  - Amazon CodeGuru
  - OWASP: Source Code Analysis Tools

リソース

関連するドキュメント:

- AWS Systems Manager
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

関連動画:

- Running high-security workloads on Amazon EKS
- サーバーレスおよびコンテナサービスを保護する
- Security best practices for the Amazon EC2 instance metadata service

関連する例:

- Lab: Automated Deployment of Web Application Firewall

SEC06-BP02 攻撃対象領域を縮小する

オペレーティングシステムを強化し、使用するコンポーネント、ライブラリ、外部から利用可能なサービスを最小限に抑えることで、意図しないアクセスへの露出を減らします。まずオペレーティングシステムパッケージやアプリケーション (Amazon Elastic Compute Cloud (Amazon EC2) ベースのワークロード)、あるいはコード内の外部ソフトウェアモジュールなどの、未使用のコンポーネント (すべてのワークロード) を減らします。一般的なオペレーティングシステムやサーバーソフトウェア向けの強化およびセキュリティ設定ガイドが多数あります。例えば、 Center for Internet Security から始めて、反復できます。

Amazon EC2 では、パッチしたり強化したりした自身の Amazon マシンイメージ (AMI) を作成して、組織の具体的なセキュリティ要件を満たすのに役立てることができます。AMI に適用するパッチやその他のセキュリティコントロールは、作成された時点では効果的です。起動後、例えば AWS Systems Manager で変更しない限り動的ではありません。

EC2 Image Builder を使って、安全な AMI をビルドするプロセスを簡素化できます。EC2 Image Builder は、自動化を記述して維持せずにゴールデンイメージを作成して維持するための作業を大幅に軽減します。ソフトウェアアップデートが利用可能になると、ユーザーがイメージビルドを手動で開始しなくても、新しいイメージが自動作成されます。EC2 Image Builder では、AWS 提供のテストと自分のテストの本番で使用する前に、イメージの機能とセキュリティを簡単に検証できます。また AWS 提供のセキュリティ設定を適用して、イメージをさらにセキュリティ保護し、内部セキュリティ条件を満たすことができます。例えば、AWS を使って、セキュリティテクニカル実装ガイド (STIG) に準拠するイメージを作成できます。

サードパーティー製の静的コード分析ツールを使用して、チェックされていない関数入力境界や、該当する共有脆弱性および露出 (CVE) などの一般的なセキュリティ問題を特定できます。専用のインフラストラクチャで Amazon CodeGuru を、サポートされる言語に対して使用できます。コードがリンクしているライブラリが最新バージョンであるかどうか、ライブラリ自体に CVE が含まれていないかどうか、ライブラリにソフトウェアポリシー要件を満たすライセンス条件があるかどうかを判断するために依存関係チェックツールを使用することもできます。

Amazon Inspectorを使用すると、インスタンスに対する設定評価を実行して既知の CVE を確認したり、セキュリティベンチマークに対して評価したり、欠陥の通知を自動化したりすることができます。Amazon Inspector は本番環境インスタンス上またはビルドパイプライン上で実行され、調査結果があるとデベロッパーとエンジニアに通知します。調査結果にはプログラムを使用してアクセス

し、バックログやバグ追跡システムにチームを誘導することができます。EC2 Image Builder は、自動パッチ適用、AWS が提供するセキュリティポリシーの適用、その他のカスタマイズにより、サーバーイメージ (AMI) を保持するために使用できます。コンテナを使用する場合は、ビルドパイプラインの ECR イメージスキャン をイメージリポジトリに対して定期的に実行し、コンテナ内の CVE を探します。

Amazon Inspector やその他のツールは、設定や CVE の有無を特定するのには効果的ですが、アプリケーションレベルでワークロードをテストするには他の方法が必要になります。ファジング は、オートメーションを使用して不正な形式のデータを入力フィールドやアプリケーションの他の領域に挿入するバグを見つけるためのよく知られた手法です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- オペレーティングシステムを強化する: ベストプラクティスを満たすようにオペレーティングシステムを設定します。
  - Amazon Linux のセキュリティ保護
  - Microsoft Windows Server のセキュリティ保護
- コンテナ化されたリソースを強化する: セキュリティのベストプラクティスを満たすよう、コンテナ化されたリソースを設定します。
- AWS Lambda のベストプラクティスを導入する
  - AWS Lambda のベストプラクティス

リソース

関連するドキュメント:

- AWS Systems Manager
- 要塞ホストを Amazon EC2 Systems Manager と置換する
- AWS Lambda のセキュリティ概要

関連動画:

- Amazon EKS で高セキュリティワークロードを実行する
- サーバーレスおよびコンテナサービスを保護する

- Amazon EC2 インスタンスメタデータサービスのセキュリティに関するベストプラクティス

関連する例:

- ラボ: ウェブアプリケーションファイアウォールの自動デプロイ

SEC06-BP03 マネージドサービスを活用する

Amazon Relational Database Service (Amazon RDS)、AWS Lambda、Amazon Elastic Container Service (Amazon ECS) などのリソースを管理するサービスを実装し、共有責任モデルの一部としてのセキュリティメンテナンスタスクを減らします。例えば、Amazon RDS は、リレーショナルデータベースのセットアップ、運用、スケーリングを支援し、ハードウェアのプロビジョニング、データベースのセットアップ、パッチ適用、バックアップなどの管理タスクを自動化します。つまり、「AWS Well-Architected フレームワーク」で説明されているその他の方法でアプリケーションを保護することに集中できる時間が増加します。Lambda では、サーバーのプロビジョニングや管理を行わずにコードを実行できるため、インフラストラクチャやオペレーティングシステムではなく、コードレベルの接続、呼び出し、セキュリティに集中するだけで済みます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 利用可能なサービスを調べる: Amazon RDS、AWS Lambda、Amazon ECS などのリソースを管理するサービスを調査、テスト、実装します。

リソース

関連するドキュメント:

- AWS ウェブサイト
- AWS Systems Manager
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

関連動画:

- Running high-security workloads on Amazon EKS
- サーバーレスおよびコンテナサービスを保護する

- Security best practices for the Amazon EC2 instance metadata service

関連する例:

- Lab: AWS Certificate Manager Request Public Certificate

SEC06-BP04 コンピューティング保護を自動化する

脆弱性管理、攻撃対象領域削減、リソース管理などのコンピューティング保護メカニズムを自動化します。自動化により、ワークロードの他の側面の保護に時間を使えるようになり、人為的ミスを犯すリスクを軽減できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 設定管理を自動化する: 設定管理サービスまたはツールを使用して、リモートでアクションを実行し、安全な設定を自動的に適用および検証します。
  - AWS Systems Manager
  - AWS CloudFormation
  - ラボ: VPC の自動デプロイ
  - ラボ: EC2 ウェブアプリケーションの自動デプロイ

- Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのパッチを自動化する: インスタンスの場合、AWS Systems Manager Patch Manager は、セキュリティ関連および他のタイプの更新の両方を使用して、マネージドインスタンスにパッチを適用するプロセスを自動化します。Patch Manager を使用して、オペレーティングシステムとアプリケーションの両方にパッチを適用できます。
  - AWS Systems Manager パッチマネージャーを使用して
  - AWS Systems Manager オートメーションを使った一元化されたマルチアカウントおよびマルチリージョンパッチ適用

- 侵入検知と防止ツールを実装する: 侵入検知と防止ツールを実装することで、インスタンス上の悪意のあるアクティビティをモニタリングし、停止できます。
- AWS Partner ソリューションを検討する: AWS パートナーは、オンプレミス環境の既存のコントロールと同等、同一、またはそれらと統合される、業界をリードする多くの製品を提供していま

す。これらの製品は、AWS の既存のサービスを補完し、クラウド環境とオンプレミス環境にわたって包括的なセキュリティアーキテクチャと、よりシームレスなエクスペリエンスをデプロイできるようにします。

- インフラストラクチャのセキュリティ

リソース

関連するドキュメント:

- AWS CloudFormation
- AWS Systems Manager
- AWS Systems Manager パッチマネージャーを使用して
- AWS Systems Manager オートメーションを使った一元化されたマルチアカウントおよびマルチリージョンパッチ適用
- インフラストラクチャのセキュリティ
- 要塞ホストを Amazon EC2 Systems Manager と置換する
- AWS Lambda のセキュリティ概要

関連動画:

- Amazon EKS で高セキュリティワークロードを実行する
- サーバーレスおよびコンテナサービスを保護する
- Amazon EC2 インスタンスメタデータサービスのセキュリティに関するベストプラクティス

関連する例:

- ラボ: ウェブアプリケーションファイアウォールの自動デプロイ
- ラボ: EC2 ウェブアプリケーションの自動デプロイ

SEC06-BP05 ユーザーがリモートからアクションを実行できるようにする

インタラクティブアクセスの機能を排除すると、人為的ミスのリスクが軽減され、設定や管理が手動で行われる可能性が低くなります。たとえば、直接アクセスや踏み台ホスト経由のアクセスを許可する代わりに、infrastructure-as-codeを使って Amazon Elastic Compute Cloud (Amazon EC2) インスタンスをデプロイし、次に AWS Systems Manager などのツールを使って Amazon EC2 を管理し

ます。AWS Systems Manager は、 オートメーション ワークフロー、io1ドキュメント (プレイブック)、 Run Commandなどの機能を使用して、さまざまなメンテナンスおよびデプロイタスクを自動化できます。AWS CloudFormation スタックは、パイプラインから構築され、AWS Management Console や API を直接使用することなく、インフラストラクチャのデプロイおよび管理タスクを自動化できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- コンソールアクセスを置き換える: インスタンスへのコンソールアクセス (SSH または RDP) を AWS Systems Manager Run Command に置き換えて、管理タスクを自動化します。

- AWS Systems Manager Run Command

リソース

関連するドキュメント:

- AWS Systems Manager
- AWS Systems Manager Run Command
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Security Overview of AWS Lambda

関連動画:

- Running high-security workloads on Amazon EKS
- サーバーレスおよびコンテナサービスを保護する
- Security best practices for the Amazon EC2 instance metadata service

関連する例:

- Lab: Automated Deployment of Web Application Firewall

SEC06-BP06 ソフトウェアの整合性を検証する

ワークロードで使用されるソフトウェア、コード、ライブラリが信頼できるソースからのものであり、改ざんされていないことを検証するメカニズム (コード署名など) を実装します。たとえば、バイナリとスクリプトのコード署名証明書を検証して作成者を確認し、作成者が作成してから改ざんされていないことを確認する必要があります。AWS Signer は、署名証明書や パブリックキー、プライベートキーを含むコード署名のライフサイクルを一元管理することで、お客様のコードの信頼性と完全性を確保することができます。コード署名のための高度なパターンとベストプラクティスは、以下で学ぶことができます: AWS Lambda.さらに、ダウンロードするソフトウェアのチェックサムをプロバイダーからのチェックサムと比較し、改ざんされていないことを確認できます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- メカニズムを検証する: コード署名は、ソフトウェアの整合性を検証するために使用できるメカニズムの 1 つです。
  - NIST: Security Considerations for Code Signing (コード署名の考慮事項)

リソース

関連するドキュメント:

- AWS Signer
- New – Code Signing, a Trust and Integrity Control for AWS Lambda (New – コード署名、AWS Lambda の信頼性および整合性のコントロール)

データ保護

質問

- SEC 7 データをどのように分類すればよいですか?
- SEC 8 保管時のデータをどのように保護すればよいですか?
- SEC 9 転送時のデータをどのように保護すればよいですか?

SEC 7 データをどのように分類すればよいですか?

分類方法を確立すると、重要度と機密性に基づいてデータをカテゴリ別に分類して、各カテゴリに適した保護と保持方法でデータを管理できるようになります。

ベストプラクティス

- SEC07-BP01 ワークロード内のデータを特定する
- SEC07-BP02 データ保護コントロールを定義する
- SEC07-BP03 識別および分類を自動化する
- SEC07-BP04 データのライフサイクル管理を定義する

SEC07-BP01 ワークロード内のデータを特定する

ワークロードで処理しているデータの種類と分類、関連するビジネスプロセス、データ所有者、適用される法律· コンプライアンス上の要件、保存場所、結果として実行が必要な統制について理解する必要があります。これには、データが一般公開されることを意図しているかどうか、データが顧客個人識別情報 (PII) などの内部使用のみかどうか、またはデータが知的財産である、法的な秘匿特権がある、機密性が高いと特記されているなど、より制限されたアクセス用であるかどうかを示す分類が含まれます。適切なデータ分類システムを、各ワークロードの保護要件レベルとともに慎重に管理することで、データに適したコントロールとアクセスまたは保護のレベルをマッピングすることができます。たとえば、パブリックコンテンツは誰でもアクセスできますが、重要なコンテンツは暗号化され、コンテンツを復号するためのキーには承認アクセスを要求することで保護しながら保存されます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- Amazon Macie を使用したデータの検出を検討する: Macie は、個人識別情報 (PII) や知的財産などの機密データを認識します。
  - Amazon Macie

リソース

関連するドキュメント:

- Amazon Macie
- データ分類に関するホワイトペーパー

- Amazon Macie の開始方法

関連動画:

- Introducing the New Amazon Macie (新しい Amazon Macie の紹介)

SEC07-BP02 データ保護コントロールを定義する

分類レベルに従ってデータを保護します。たとえば、関連するレコメンデーションを使用してパブリックとして分類されたデータを保護すると同時に、追加のコントロールで機密データを保護します。

リソースタグ、機密性ごと (および注意事項、エンクレーブ、関心のあるコミュニティごと) の個別の AWS アカウント 、IAM ポリシー、AWS Organizations SCP、AWS Key Management Service (AWS KMS)、AWS CloudHSM を使用することで、暗号化によるデータ分類と保護のポリシーを定義および実装できます。たとえば、非常に重要なデータを含む S3 バケット、または、秘密データを処理する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを含むプロジェクトがある場合、それらに `#Project=ABC#` を付けることができます。直属のチームのみがこのプロジェクトコードの意味を知っていて、属性ベースのアクセス統制手段を使用する方法を提供します。キーポリシーと許可を使用して AWS KMS 暗号化キーへのアクセスレベルを定義し、安全なメカニズムを通じて適切なサービスだけが機密コンテンツにアクセスできるようにします。タグに基づいて承認決定を判断する場合、AWS Organizations 内のタグポリシーを使用して、タグの許可が適切に定義されていることを確認する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- データの識別および分類スキーマを定義する: データの識別と分類は、保存するデータの潜在的な影響とタイプ、およびデータにアクセスできるユーザーを評価するために実行されます。
  - AWS ドキュメント
- 利用可能な AWS のコントロールを確認する: 使用しようとしているか、使用を計画している AWS サービスについて、セキュリティコントロールを確認します。多くのサービスには、ドキュメントにセキュリティセクションがあります。
  - AWS ドキュメント
- AWS コンプライアンスリソースを特定する: 支援のために使用できる AWS のリソースを特定します。

- https://aws.amazon.com/compliance/

リソース

関連するドキュメント:

- AWS ドキュメント
- データ分類に関するホワイトペーパー
- Amazon Macie の開始方法
- 欠落テキスト

関連動画:

- Introducing the New Amazon Macie

SEC07-BP03 識別および分類を自動化する

データの識別と分類を自動化すると、適切な統制を実装するのに役立ちます。人が直接アクセスするよりも自動化した方が、人為的ミスや開示リスクは小さくなります。など、 Amazon Macieなどの、機械学習を使用して AWS の機密データを自動的に検出、分類、保護するツールの利用を評価する必要があります。Amazon Macie は個人識別情報 (PII) や知的財産などの機密データを 認識し、このデータへのアクセスや移動の状況を可視化するダッシュボードやアラートを提供します。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- Amazon Simple Storage Service (Amazon S3) インベントリを使用する: Amazon S3 インベントリは、オブジェクトのレプリケーションと暗号化ステータスの監査とレポートに使用できるツールの 1 つです。
  - Amazon S3 インベントリ
- Amazon Macie を検討する: Amazon Macie は、機械学習を使用して Amazon S3 内に保存されているデータを自動的に検出、分類します。
  - Amazon Macie

リソース

関連するドキュメント:

- Amazon Macie
- Amazon S3 インベントリ
- データ分類に関するホワイトペーパー
- Amazon Macie の開始方法

関連動画:

- Introducing the New Amazon Macie (新しい Amazon Macieの紹介)

SEC07-BP04 データのライフサイクル管理を定義する

定義されるライフサイクル戦略は、機密性レベル、また法的および組織の要件に基づいている必要があります。データを保持する期間、データ破壊プロセス、データアクセス管理、データ変換、データ共有などの側面を考慮する必要があります。データ分類方法を選択するときは、可用性とアクセスのバランスを取ります。また、各レベルにとって安全でありながら使いやすい方式を採用するために、複数レベルのアクセスと微妙な差異も実装する必要がります。常に多層防御方式を採用し、データおよびデータの変換、削除、コピーのメカニズムに人間がアクセスする機会を減らします。例えば、アプリケーション認証を厳格にし、遠距離操作を実行するために必要なアクセス許可をユーザーでなくアプリケーションに付与します。さらに、ユーザーが信頼できるネットワークパスからアクセスしていることを確認して、復号鍵へのアクセスを要求します。ユーザーにデータへの直接アクセス権を付与するのではなく、ダッシュボードや自動レポートなどのツールを使用して、データからの情報をユーザーに提供します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- データタイプを識別する: ワークロードに保存または処理するデータのタイプを特定します。そのデータは、テキスト、イメージ、バイナリデータベースなどが考えられます。

リソース

関連するドキュメント:

- データ分類に関するホワイトペーパー
- Amazon Macie の開始方法

関連動画:

- 新しい Amazon Macie の導入

SEC 8 保管時のデータをどのように保護すればよいですか?

複数のコントロールを実装して保管中のデータを保護し、不正アクセスや不正処理のリスクを低減します。

ベストプラクティス

- SEC08-BP01 安全なキー管理を実装する
- SEC08-BP02 保管中に暗号化を適用する
- SEC08-BP03 保管時のデータの保護を自動化する
- SEC08-BP04 アクセスコントロールを適用する
- SEC08-BP05 人をデータから遠ざけるメカニズムを使用する

SEC08-BP01 安全なキー管理を実装する

キーの保存、ローテーション、アクセス制御を含む暗号化アプローチを定義することで、不正ユーザーからのコンテンツの保護や、正規ユーザーへの不必要な公開を防止することができます。AWS Key Management Service (AWS KMS) は暗号化キーの管理をサポートして 多数の AWS のサービスと統合します.このサービスでは、AWS KMS キーのための、耐久性と安全性が高く、冗長なストレージを利用できます。キーのエイリアスのほか、キーレベルのポリシーも定義できます。ポリシーは、キー管理者やキーユーザーを定義するのに役立ちます。さらに、AWS CloudHSM はクラウドベースのハードウェアセキュリティモジュール (HSM) であり、AWS クラウド 上で独自の暗号化キーを簡単に生成して使用できます。FIPS 140-2 レベル 3 検証済みの HSM を使用することで、データセキュリティに関する企業、契約、規制のコンプライアンス要件を満たすことができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- AWS KMS を実装する: AWS KMS は、キーを作成および管理し、さまざまな AWS のサービスおよびアプリケーションで暗号化の使用を制御することを容易にします。AWS KMS は、FIPS

140-2 で検証されたハードウェアセキュリティモジュールを使用してキーを保護する、安全で弾力性のあるサービスです。

  - Getting started: AWS Key Management Service (AWS KMS) (AWS Key Management Service (AWS KMS) の使用を開始)
- AWS Encryption SDK を検討する: アプリケーションでクライアント側でのデータ暗号化が必要な場合、AWS KMS が統合された AWS Encryption SDK を使用します。
  - AWS Encryption SDK

リソース

関連するドキュメント:

- AWS Key Management Service
- AWS cryptographic services and tools (AWS 暗号化サービスとツール)
- Getting started: AWS Key Management Service (AWS KMS) (AWS Key Management Service (AWS KMS) の使用を開始)
- 暗号化を使用した Amazon S3 データの保護

関連動画:

- How Encryption Works in AWS (AWS での暗号化のしくみ)
- Securing Your Block Storage on AWS (AWS でブロックストレージを保護する)

SEC08-BP02 保管中に暗号化を適用する

データを保存する唯一の方法は、暗号化を使用することだということを確実にする必要があります。AWS Key Management Service (AWS KMS) は、保管中のすべてのデータをより簡単に暗号化できるように、多数の AWS のサービスとシームレスに統合します。例えば Amazon Simple Storage Service (Amazon S3) では、 デフォルトの暗号化を バケットに設定して、すべての新しいオブジェクトが自動的に暗号化されるようにすることができます。さらに、Amazon Elastic Compute Cloud (Amazon EC2) および Amazon S3 は、デフォルト暗号化を設定することにより、暗号化の適用をサポートしています。専用のインフラストラクチャで AWS Config ルール を使用して、例えば次の項目に対して暗号化を使用していることを自動的に確認できます: Amazon Elastic Block Store (Amazon EBS) ボリュームio1 Amazon Relational Database Service (Amazon RDS) インスタンス、および Amazon S3 バケット.

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- Amazon Simple Storage Service (Amazon S3) に対して保管中に暗号化を適用する: Amazon S3 バケットのデフォルト暗号化を実施します。
  - S3 バケットに対してデフォルトの暗号化を有効にするにはどうすればよいですか。
- AWS Secrets Manager を使用する: Secrets Manager は、機密情報の管理を容易にする AWS のサービスです。シークレットとは、データベース認証情報、パスワード、サードパーティ API キー、任意のテキストなどです。
  - AWS Secrets Manager
- 新しい EBS ボリュームのデフォルトの暗号化を設定する: 新しく作成したすべての EBS ボリュームを暗号化形式で作成することを指定します。AWS が提供するデフォルトキーを使用するか、作成したキーを使用するかを選択できます。
  - EBS ボリュームのデフォルトの暗号化
- 暗号化された Amazon Machine Image (AMI) を設定する: 暗号化を有効化して既存の AMI をコピーすると、自動的にルートボリュームとスナップショットが暗号化されます。
  - 暗号化されたスナップショットを持つ AMI
- Amazon Relational Database Service (Amazon RDS) 暗号化を設定する: 暗号化オプションを有効化して、保管中の Amazon RDS データベースクラスターとスナップショットに対して暗号化を設定します。
  - Amazon RDS リソースの暗号化
- 追加の AWS サービス: 使用する AWS のサービスについて、暗号化機能を決定します。
  - AWS ドキュメント

リソース

関連するドキュメント:

- 暗号化されたスナップショットを持つ AMI
- AWS Crypto Tools
- AWS ドキュメント
- AWS Encryption SDK
- AWS KMS 暗号化の詳細についてのホワイトペーパー
- AWS Key Management Service

- AWS Secrets Manager
- AWS 暗号化サービスとツール
- Amazon EBS 暗号化
- EBS ボリュームのデフォルトの暗号化
- Amazon RDS リソースの暗号化
- S3 バケットに対してデフォルトの暗号化を有効にするにはどうすればよいですか。
- 暗号化を使用して Amazon S3 データを保護する

関連動画:

- AWS での暗号化のしくみ
- AWS でブロックストレージを保護する

SEC08-BP03 保管時のデータの保護を自動化する

自動化ツールを使用して保管中のデータの制御を継続的に検証し、強化します。例えば、すべてのストレージリソースが暗号化されていることを確認します。また、 すべての EBS ボリューム が AWS Config ルール。AWS Security Hub は、セキュリティ標準に対する自動チェック機能を通じて、いくつかの制御を検証することもできます。さらに AWS Config ルール は、自動的に 非準拠のリソースを修復できます.

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

保管中のデータ とは、ワークロードの任意の期間に永続的ストレージに保持されるすべてのデータを指します。たとえば、ブロックストレージ、オブジェクトストレージ、データベース、アーカイブ、IoT デバイス、データが保持されているその他のストレージ媒体などがあります。暗号化と適切なアクセスコントロールが実装されている場合は、保管中のデータを保護することで不正アクセスのリスクを軽減できます。

保管中に暗号化を適用する: データを保存する唯一の方法は、暗号化を使用することだということを確実にする必要があります。AWS KMS は、保管中のすべてのデータをより簡単に暗号化できるように、多数の AWS のサービスとシームレスに統合します。例えば、Amazon Simple Storage Service (Amazon S3) では、 デフォルトの暗号化を バケットに設定して、すべての新しいオブジェクトが自動的に暗号化されるようにすることができます。さらに、Amazon EC2 および Amazon S3 は、デフォルト暗号化を設定することにより、暗号化の適用をサポートしています。専用のインフラスト

ラクチャで AWS マネージド Config ルール を使用して、例えば次の項目に対して暗号化を使用していることを自動的に確認できます: EBS ボリュームio1Amazon Relational Database Service (Amazon RDS) インスタンス、および Amazon S3 バケット.

リソース

関連するドキュメント:

- AWS Crypto Tools
- AWS Encryption SDK

関連動画:

- AWS での暗号化のしくみ
- AWS でブロックストレージを保護する

SEC08-BP04 アクセスコントロールを適用する

最低限の権限によるアクセスコントロールや、バックアップ、分離、バージョニングなどのメカニズムを適用することは、保管中のデータの保護に役立ちます。オペレーターがデータへのパブリックアクセスを許可しないようにします。

アクセス (最小特権を使用)、バックアップ (信頼性ホワイトペーパーを参照)、隔離およびバージョニングなどのさまざまなコントロールはすべて、保管中のデータを保護するのに役立ちます。データへのアクセスは、CloudTrail などのこのホワイトペーパーで前述した探査メカニズムと、Amazon Simple Storage Service (Amazon S3) アクセスログなどのサービスレベルログを使用して監査する必要があります。パブリックにアクセス可能なデータをインベントリし、時間の経過とともに利用可能なデータ量の削減を可能にする方法を計画する必要があります。Amazon S3 Glacier のボールトロックと Amazon S3 オブジェクトロックは、必須のアクセス制御を提供する機能です。ボールトポリシーがコンプライアンスオプションを使用してロックされると、ロックの有効期限が切れるまではルートユーザーでも変更できません。このメカニズムは、SEC、CFTC、FINRA の帳簿および記録管理要件を満たしています。詳細については、 このホワイトペーパーを参照してください.

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- アクセスコントロールを適用する: 暗号キーへのアクセスを含め、最小権限を用いたアクセスコントロールを適用します。

  - Amazon S3 リソースへのアクセス許可の管理の導入
- さまざまな分類レベルに基づいてデータを分離する: AWS Organizations によって管理されるデータ分類レベルには、さまざまな AWS アカウント アカウントを使用します。
  - AWS Organizations
- AWS KMS ポリシーをレビューする: AWS KMS ポリシーで付与されるアクセスのレベルを確認します。
  - AWS KMS リソースへのアクセス管理の概要
- Amazon S3 バケットとオブジェクトアクセス許可をレビューする: Amazon S3 バケットのポリシーで付与されるアクセスのレベルを定期的に確認します。ベストプラクティスは、バケットを公開で読み取りまたは書き込み可能にしないことです。AWS Config を使用して公開されているバケットを検出し、Amazon CloudFront を使用して Amazon S3 からコンテンツを提供することを検討します。
  - AWS Config ルール
  - Amazon S3 + Amazon CloudFront: 理想的な組み合わせ
- Amazon S3 バージョニングとオブジェクトロックを有効にします。
  - バージョニングの使用
  - Amazon S3 Object Lock を使ってオブジェクトをロックする
- Amazon S3 インベントリを使用する: Amazon S3 インベントリは、オブジェクトのレプリケーションと暗号化ステータスの監査とレポートに使用できるツールの 1 つです。
  - Amazon S3 インベントリ
- Amazon EBS および AMI 共有アクセス許可をレビューする: 共有アクセス許可は、イメージとボリュームをワークロード外の AWS アカウント に共有することを可能にします。
  - Amazon EBS スナップショットの共有
  - 共有 AMI

リソース

関連するドキュメント:

- AWS KMS 暗号化の詳細についてのホワイトペーパー

関連動画:

- AWS でブロックストレージを保護する

SEC08-BP05 人をデータから遠ざけるメカニズムを使用する

通常の運用状況で、すべてのユーザーが機密データおよびシステムに直接アクセスできないようにします。たとえば、変更管理ワークフローを使用して、直接アクセスや踏み台ホストを許可する代わりに、ツールを使用して Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを管理します。これは、タスクを実行する手順を含むオートメションドキュメントを使用する AWS Systems Manager Automation をトリガーしてを通じて 達成できます 。これらのドキュメントはソース管理に保存し、実行前にピアレビューを行い、シェルアクセスと比較してリスクを最小限に抑えるために徹底的にテストできます。ビジネスユーザーは、データストアに直接アクセスする代わりにダッシュボードを使用し、クエリを実行できます。CI/CD パイプラインを使用しない場合は、通常無効になっている特権アクセスメカニズムを適切に提供するために必要な制御とプロセスを決定します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 人をデータから遠ざけるメカニズムを実装する: メカニズムには、Amazon QuickSight などのダッシュボードを使用して、直接クエリを実行する代わりにユーザーにデータを表示することが含まれます。
  - Amazon QuickSight
- 設定管理を自動化する: 設定管理サービスまたはツールを使用して、リモートでアクションを実行し、安全な設定を自動的に適用および検証します。踏み台ホストを使用したり、EC2 インスタンスに直接アクセスしたりすることを回避します。
  - AWS Systems Manager
  - AWS CloudFormation
  - AWS の AWS CloudFormation テンプレートの CI/CD パイプライン

リソース

関連するドキュメント:

- AWS KMS 暗号化の詳細についてのホワイトペーパー

関連動画:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC 9 転送時のデータをどのように保護すればよいですか?

複数のコントロールを実装して、転送中のデータを保護し、不正アクセスや損失のリスクを軽減します。

ベストプラクティス

- SEC09-BP01 安全な鍵および証明書管理を実装する
- SEC09-BP02 伝送中に暗号化を適用する
- SEC09-BP03 意図しないデータアクセスの検出を自動化する
- SEC09-BP04 ネットワーク通信を認証する:

SEC09-BP01 安全な鍵および証明書管理を実装する

暗号化キーと証明書を安全に保存し、厳格なアクセスコントロールによって適切な時間間隔でローテーションします。これを実現する最善の方法として、 AWS Certificate Manager (ACM).これにより、AWS のサービスおよび内部接続リソースで使用するためのパブリックおよびプライベートの Transport Layer Security (TLS) 証明書のプロビジョニング、管理、デプロイが容易になります。TLS 証明書は、ネットワーク通信を保護し、プライベートネットワーク上のリソースだけでなく、インターネット上のウェブサイトのアイデンティティを確立するために使用されます。ACM は、Elastic Load Balancers (ELB)、AWS ディストリビューション、API Gateway の API などの AWS リソースと統合し、証明書の自動更新も処理します。Amazon Elastic Compute Cloud を使用してプライベートルート CA をデプロイする場合、証明書とプライベートキーの両方を ACM (Amazon EC2) インスタンス、コンテナなどで使用するために提供できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 安全な鍵および証明書管理を実装する: 定義された安全なキーおよび証明書管理ソリューションを実装します。
  - AWS Certificate Manager
  - AWS でプライベート証明書インフラストラクチャをホストおよび管理する方法
- 安全なプロトコルを実装する: 認証と機密性を提供する安全なプロトコル (Transport Layer Security (TLS) や IPsec など) を使用し、データの改ざんや損失のリスクを軽減します。使用しているサービスに関連するプロトコルとセキュリティについては、AWS ドキュメントを参照してください。

リソース

関連するドキュメント:

- AWS ドキュメント

SEC09-BP02 伝送中に暗号化を適用する

組織、法律、コンプライアンスの要件を満たすために、適切な基準や 推奨事項に基づいて、定義された暗号化要件を実施します。AWS サービスは、通信に TLS を使用して HTTPS エンドポイントを提供するため、AWS API と通信する際には伝送中に暗号化されます。HTTP などの安全でないプロトコルは、セキュリティグループを使用して VPC で監査およびブロックできます。HTTP リクエストは 自動的に Amazon CloudFront で HTTPS または Application Load Balancer.コンピューティングリソースを完全に制御して、サービス全体に伝送中データの暗号化を実装できます。また、外部ネットワークからお使いの VPC に VPN で接続して、トラフィックの暗号化を促進できます。特別な要件がある場合は、AWS Marketplace でサードパーティーのソリューションを入手できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- 伝送中に暗号化を適用する: 暗号化の要件は、最新の標準とベストプラクティスに基づき、安全なプロトコルのみを許可する必要があります。たとえば、Application Load Balancer または Amazon Elastic Compute Cloud (Amazon EC2) インスタンスには、HTTPS プロトコルを許可するセキュリティグループのみを設定します。
- エッジサービスで安全なプロトコルを設定する: Amazon CloudFront と必要な暗号で HTTPS を設定します。
  - CloudFront で HTTPS を使用する
- 外部接続に VPN を使用する: ポイントツーポイント接続やネットワーク間接続を IPsec 仮想プライベートネットワーク (VPN) で保護し、データのプライバシーと整合性の両方を提供することを検討してください。
  - VPN 接続
- ロードバランサーで安全なプロトコルを設定する: ロードバランサーへの接続を保護するために、HTTPS リスナーを有効にします。
  - Application Load Balancer 用の HTTPS リスナー
- インスタンスの安全なプロトコルを設定する: インスタンスに HTTPS 暗号化を設定することを検討します。

  - チュートリアル: Amazon Linux 2 で SSL/TLS を使用できるように Apache ウェブサーバーを設定する
- Amazon Relational Database Service (Amazon RDS) で安全なプロトコルを設定する: Secure Socket Layer (SSL) または Transport Layer Security (TLS) を使って、データベースインスタンスへの接続を暗号化します。
  - SSL を使用した DB インスタンスへの接続の暗号化
- Amazon Redshift で安全なプロトコルを設定する: クラスターで Secure Socket Layer (SSL) または Transport Layer Security (TLS) 接続が必要となるよう設定します。
  - 接続のセキュリティオプションを設定する
- 追加の AWS のサービスで安全なプロトコルを設定する: 使用する AWS のサービスについて、転送中の暗号化機能を決定します。

リソース

関連するドキュメント:

- AWS のドキュメント

SEC09-BP03 意図しないデータアクセスの検出を自動化する

Amazon GuardDuty などのツールを使用して、疑わしい活動や定義された境界外にデータを移動させようとする試みを自動的に検出します。例えば、GuardDuty は Amazon Simple Storage Service (Amazon S3) 読み取りアクティビティを検出できますが、それには Exfiltration:S3/AnomalousBehavior 調査結果を使用します.GuardDuty に加えて、ネットワークトラフィック情報をキャプチャする Amazon VPC フローログを Amazon EventBridge とともに使用して、異常な接続 (成功と拒否の両方) の検出をトリガーできます。Amazon S3 Access Analyzer は Amazon S3 バケット内で誰がどのデータにアクセス可能かを評価するのに役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 意図しないデータアクセスの検出を自動化する: ツールまたは検出メカニズムを使用し、定義された境界の外側にデータを移動する試みを自動的に検出します。例えば、認識できないホストにデータをコピーしているデータベースシステムを検出します。
  - VPC フローログ

- Amazon Macie を検討する: Amazon Macie は、機械学習とパターンマッチングを使用して AWS の機密データを検出および保護する、フルマネージドのデータセキュリティおよびデータプライバシーサービスです。
  - Amazon Macie

リソース

関連ドキュメント:

- VPC フローログ
- Amazon Macie

SEC09-BP04 ネットワーク通信を認証する:

Transport Layer Security (TLS) や IPsec など、認証をサポートするプロトコルを使用して、通信の ID を検証します。

認証をサポートするネットワークプロトコルを使用すると、当事者間で信頼を確立できます。これにより、プロトコルで使用される暗号化が追加され、通信が変更または傍受されるリスクが軽減します。認証を実装する一般的なプロトコルには、多くの AWS のサービスで使用される Transport Layer Security (TLS) と、 AWS Virtual Private Network (AWS VPN) で使用される IPsecがあります.

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 安全なプロトコルを実装する: 認証と機密性を提供する安全なプロトコル (Transport Layer Security (TLS) や IPsec など) を使用し、データの改ざんや損失のリスクを軽減します。使用しているサービスに関連するプロトコルとセキュリティについては、 AWS ドキュメントを 参照してください。

リソース

関連するドキュメント:

- AWS ドキュメント

インシデント対応

質問

- SEC 10 インシデントの予測、対応、復旧はどのように行いますか?

SEC 10 インシデントの予測、対応、復旧はどのように行いますか?

組織に支障をきたすことを最小限に抑えるために、セキュリティインシデントのタイムリーで効果的な調査、対応、復旧に備えることが重要です。

ベストプラクティス

- SEC10-BP01 重要な人員と外部リソースを特定する:
- SEC10-BP02 インシデント管理計画を作成する
- SEC10-BP03 フォレンジック機能を備える:
- SEC10-BP04 封じ込め機能を自動化する
- SEC10-BP05 アクセスを事前プロビジョニングする
- SEC10-BP06 ツールを事前デプロイする
- SEC10-BP07 ゲームデーを実施する

SEC10-BP01 重要な人員と外部リソースを特定する:

組織がインシデントに対応するのに役立てるため、社内外の担当者、リソース、法的義務を特定します。

クラウド上でのインシデントレスポンスへのアプローチを他のチーム (顧問弁護士、経営陣、ビジネスステークホルダー、AWS サポートサービスなど) と連携して定義する場合、重要な人物、ステークホルダー、関連する担当者を特定する必要があります。依存性を減らし、応答時間を短縮するために、チームや専門のセキュリティチーム、応答者が利用するサービスについて教育を受け、実践的な演習をする機会を持つようにしてください。

対応能力を強化するために、外部の専門知識および異なる視点を備えた社外の AWS セキュリティパートナーを探しておくことをお勧めします。信頼できるセキュリティパートナーは、馴染みのない潜在的なリスクや脅威を特定するのに役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 組織内の主要な人員を特定する: インシデント対応と復旧に必要な組織内の人員の連絡先リストを保持します。
- 外部パートナーを特定する: 必要に応じて、インシデント対応と復旧を支援できる外部パートナーと連携します。

リソース

関連するドキュメント:

- AWS インシデント対応ガイド

関連動画:

- AWS 環境のセキュリティインシデントの準備と対応

関連する例:

SEC10-BP02 インシデント管理計画を作成する

インシデントへの応答、インシデント時の伝達、インシデントからの復旧に役立つ計画を作成します。たとえば、ワークロードと組織にとって起こる可能性が最も高いシナリオで、インシデント応答計画を作成してみましょう。内部および外部に伝達およびエスカレーションする方法を含めます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

インシデント管理計画は、セキュリティインシデントの潜在的な影響への対応、復旧、軽減に不可欠です。インシデント管理計画は、セキュリティインシデントをタイムリーに特定し、修復、対応するための体系的なプロセスです。

クラウドには、オンプレミス環境と同じオペレーション上のロールと要件があります。インシデント管理計画を作成する際は、ビジネス成果とコンプライアンス要件と最も合致する対応および復旧戦略を組み込むことが重要です。例えば、米国の FedRAMP 準拠のワークロードを AWS で運用している場合は、『NIST SP 800-61 Computer Security Handling Guide』を遵守することが役に立ちます。同様に、ヨーロッパの PII (個人を特定できる情報) データを含むワークロードを運用しているときは、EU 一般データ保護規則 (GDPR) で義務付けられているようにデータレジデンシー関連の問題に対してどのように防御、対応するかといったシナリオを考慮します。

AWS で運用するワークロードについてインシデント管理計画を策定する際は、 AWS 責任共有モデルから始め、インシデント対応に向けた多層防御アプローチを構築することを目指します。このモデルでは、AWS はクラウドのセキュリティを管理します。クラウド内のセキュリティについてはお客様の責任です。つまり、お客様はコントロールを保持するとともに、実装しようとするセキュリティコントロールに責任を持つということです。『 AWS Security Incident Response Guide』(AWS セキュリティインシデント対応ガイド) には、クラウド中心のインシデント管理計画を策定するための重要なコンセプトと基本的なガイダンスが記載されています。

効果的なインシデント管理計画は、クラウド運用の目標に沿って継続的に繰り返し、最新の状態に保つ必要があります。インシデント管理計画を作成して進化させるにあたり、以下に記載の実装計画を使用することを検討してください。

- インシデント対応の教育とトレーニング: 定義されたベースラインからの逸脱 (間違ったデプロイ、設定ミスなど) が発生した場合は、対応と調査が必要になる可能性があります。これを適切に行うために、自社の AWS 環境内でセキュリティインシデント対応に使用できるコントロールと機能を理解するとともに、インシデント対応に関与するクラウドチームの準備を整え、教育とトレーニングを行うプロセスを把握する必要があります。
  - プレイブック および ランブック は、インシデントへの対応方法のトレーニングで一貫性を確保するのに効果的なメカニズムです。まず、インシデント対応中に頻繁に実行する手順の最初のリストを作成し、継続的に繰り返しながら新しい手順の学習や採用を行います。
  - スケジュールされた ゲームデーをとおして、プレイブックとランブックを広めます。ゲームデーの期間中、制御された環境でインシデント対応をシミュレーションします。それにより、チームは対応方法を思い出し、インシデント対応に関与する各チームがワークフローを熟知していることを検証できます。シミュレーションされたイベントの結果をレビューし、改善点を特定し、さらにトレーニングや追加ツールが必要か判断します。
  - セキュリティは全員の任務と見なす必要があります。普段、ワークロードを運用するすべてのスタッフを関与させることで、インシデント管理プロセスの集合知を構築します。これにはビジネスのすべての側面、つまり、運用、テスト、開発、セキュリティ、ビジネスオペレーション、ビジネスリーダーが含まれます。
- インシデント管理計画をドキュメント化する: アクティブなインシデントの記録、対応、進捗に関するコミュニケーション、通知の提供のためのツールとプロセスをドキュメント化します。インシデント管理計画の目標は、通常のオペレーションができるだけ迅速に復旧され、ビジネスへの影響が最小限に抑えられ、すべての関係者に情報が提供されることを検証することです。インシデントの例としては、ネットワーク接続の損失やパフォーマンス低下、応答しないプロセスまたは API、スケジュール済みだが実行されないタスク (パッチ適用の失敗など)、アプリケーションデータまた

はサービスの利用不可、セキュリティイベントに起因する計画外のサービス中断、認証情報の漏洩、設定ミスによるエラーがあります (ただし、これらに限定されません)。

- インシデント解決に責任を持つ主な所有者 (ワークロード所有者など) を特定します。誰がインシデントを管理するか、またどのようにコミュニケーションを取るかについて、明確なガイダンスを用意します。外部ベンダーなど複数の当事者をインシデント解決プロセスに関与させる場合は、インシデント解決で求められる、さまざまなチームやスタッフの役割と責任を記述した 責任分担 (RACI) マトリックスを作成することを検討します。

  RACI マトリックスには以下を記述します。

  - R: Responsible - 作業を行いタスクを完了する実行責任者。
  - A: Accountable - 指定されたタスクの正常な完了に対して最終的な権限を持つ説明責任者またはステークホルダー。
  - C: Consulted - 意見を求められる相談先。一般的には対象分野の専門家。
  - I: Informed - 進捗について通知を受ける情報提供先。多くの場合、タスクの完了や成果物についてのみ通知される。

- インシデントを分類する: インシデントを定義し、重大度と影響度のスコアに基づき分類することで、インシデントをトリアージして解決するための体系的なアプローチが可能となります。次の推奨事項は、インシデントを数値化するための 影響と解決の緊急マトリックス を説明するものです。例えば、影響度: 低、緊急度: 低のインシデントは、重大度: 低のインシデントと見なされます。

  - 高 (H): ビジネスは多大な影響を受けます。AWS リソースに関連するアプリケーションのクリティカルな機能は使用できません。本番システムに影響を及ぼすほとんどのクリティカルイベントのために用意されています。インシデントの影響は急速に拡大し、修復には時間的制約があります。
  - 中 (M): AWS リソースに関連するビジネスサービスまたはアプリケーションは、適度に影響を受けますが、パフォーマンスが低下した状態で機能します。サービスレベル目標 (SLO) に寄与するアプリケーションは、サービスレベルアグリーメント (SLA) の範囲内で影響を受けます。システムは、能力が低下した状態で機能することができ、財務的影響および評判上の影響はあまりありません。
  - 低 (L): AWS リソースに関連するビジネスサービスまたはアプリケーションの非クリティカルな機能が影響を受けます。システムは、能力が低下した状態で機能することができ、財務的影響および評判上の影響は最小限にとどまります。

- セキュリティコントロールを標準化する: セキュリティコントロールを標準化する目的は、オペレーションの結果に関する一貫性、トレーサビリティ、再現性を実現することです。インシデント対応に欠かせない重要なアクティビティについて標準化を推進します。以下に例を挙げます。

- アイデンティティとアクセスの管理: データへのアクセスを制御し、人間とマシンアイデンティティの両方の権限を管理するためのメカニズムを確立します。シングルサインオンとロールベースの権限を含むフェデレーテッドセキュリティを使用して、貴社独自のアイデンティティとアクセスの管理をクラウドに拡張し、アクセス管理を最適化します。アクセス管理を標準化するためのベストプラクティスの推奨事項と改善計画については、 セキュリティの柱に関するホワイトペーパーの「ID とアクセス管理」のセクション を参照してください
- 脆弱性管理: AWS 環境で、攻撃者がシステムを侵害、悪用するために用いる可能性のある脆弱性を特定するためのメカニズムを確立します。予防的コントロールと発見的コントロールの両方をセキュリティメカニズムとして実装し、セキュリティインシデントの潜在的な影響に対応し、その影響を緩和します。脅威のモデル化などのプロセスを、インフラストラクチャ構築およびアプリケーションデリバリーライフサイクルの一環として標準化します。
- 構成管理: 標準構成を定義し、AWS クラウド にリソースをデプロイする手順を自動化します。インフラストラクチャとリソースの両方のプロビジョニングを標準化すると、間違ったデプロイによる設定ミスや意図しない人的な設定ミスのリスクの軽減に役立ちます。このコントロールを実装するためのガイダンスと改善計画については、『運用上の優秀性の柱』のホワイトペーパーの 「設計原則」のセクション を参照してください。
- 監査コントロールのためのログ記録と監視: リソースの障害、パフォーマンス低下、セキュリティの問題を監視するためのメカニズムを実装します。これらのコントロールを標準化すると、システムで発生したアクティビティの監査証跡が提供され、問題のタイムリーなトリアージと修復に役立ちます。SEC04 (「セキュリティイベントは、どのように検出して調査するのですか?」) のベストプラクティスは、 このコントロールを実装するためのガイダンスを提供しています。

- オートメーションを使用する: オートメーションにより、インシデントのタイムリーで大規模な解決が可能となります。AWS は、インシデント対応戦略の枠組みの中で自動化を行うサービスを複数提供しています。オートメーションと人の介入との適切なバランスを見つけることに重点を置きます。プレイブックとランブックでインシデント対応を構築しながら、繰り返し可能なステップを自動化します。AWS Systems Manager Incident Manager などの AWS サービスを使用して IT インシデントの解決を迅速化します。デベロッパーツールを 使用して、 バージョン管理を提供し、Amazon Machine Images (AMI) と Infrastructure as Code (IaC) のデプロイを自動化し、人間の介入を排除します。該当する場合は、Amazon GuardDuty、Amazon Inspector、AWS Security Hub、AWS Config、Amazon Macie などのマネージドサービスを使用して検出とコンプライアンス評価を自動化します。Amazon DevOps Guru などの機械学習を使用して検出機能を最適化し、異常な動作パターンの問題が発生する前に検出します。
- 根本原因分析と、教訓から得られたアクションを実施します。 過去のインシデント対応レビューの一環として、教訓を取り込むためのメカニズムを実装します。インシデントの根本原因により、

より大規模な検出、設計上の欠陥、設定ミス、再発の可能性が明らかになった場合、問題として分類されます。そのような場合、問題を分析および解決して、通常のオペレーションの中断を最小限に抑えます。

リソース

関連するドキュメント:

- AWS Security Incident Response Guide (AWS セキュリティインシデント対応ガイド)
- NIST: Computer Security Incident Handling Guide

関連動画:

- AWS のインシデント対応とフォレンジックの自動化
- ランブック、インシデントレポート、インシデント対応の DIY ガイド
- AWS 環境のセキュリティインシデントの準備と対応

関連サンプル:

- ラボ : Jupyter を使用したインシデント対応プレイブック - AWS IAM
- ラボ: Incident Response with AWS Console and CLI (AWS コンソールと CLI を使用したインシデント対応)

SEC10-BP03 フォレンジック機能を備える:

インシデントレスポンダーがフォレンジック調査が対応計画に適合する時期と方法を理解しておくことが重要です。組織は収集される証拠と、プロセスで使用されるツールを定義する必要があります。外部のスペシャリスト、ツール、オートメーションなど、適切なフォレンジック調査能力を特定し、準備します。前もって下す重要な決定は、ライブシステムからデータを収集するかどうかです。システムの電源を切ったり再起動したりすると、揮発性メモリの内容やアクティブなネットワーク接続などの一部のデータが失われます。

対応チームは、AWS Systems Manager、Amazon EventBridge、および AWS Lambda などのツールを組み合わせて、オペレーティングシステムおよびトラフィックミラーリング内でフォレンジックツールを実行し、ネットワークパケットキャプチャを取得し、非永続的証拠を収集できます。ログ分析やディスクイメージの分析などその他のアクティビティは、レスポンダーがアクセスできるカスタ

マイズされたフォレンジックワークステーションとツールを備えた専用のセキュリティアカウントで実行します。

関連ログは、高い耐久性と整合性を提供するデータストアに定期的に送信します。レスポンダーは、それらのログにアクセスできる必要があります。AWS には、Amazon Athena、Amazon OpenSearch Service (OpenSearch Service)、および Amazon CloudWatch Logs Insights など ログ調査をやりやすくするツールがいくつかあります。さらに、Amazon Simple Storage Service (Amazon S3) Object Lock を使って安全に証拠を保持します。このサービスは WORM (write-once- read-many) モデルに従っており、定義済みの期間オブジェクトが検出されたり上書きされたりするのを防ぎます。フォレンジック調査技術には専門的なトレーニングが必要なため、外部のスペシャリストとの連携が必要になる場合があります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- フォレンジック機能を確認する: 組織のフォレンジック調査機能、利用可能なツール、外部スペシャリストを調査します。
- インシデント対応とフォレンジックの自動化

リソース

関連するドキュメント:

- How to automate forensic disk collection in AWS

SEC10-BP04 封じ込め機能を自動化する

インシデントの封じ込めおよび復旧を自動化し、対応時間を短縮するとともに組織的影響を軽減します。

プレイブックからプロセスやツールを作成して実践したら、ロジックをコードベースのソリューションに分解します。そうすることによって、多くの応答者が応答を自動化し、応答者によるばらつきや推測作業を取り除くためのツールとして使用することができます。これにより、対応のライフサイクルを高速化できます。次の目標は、このコードを人間の対応者ではなく、アラートやイベント自体によって呼び出すことで完全な自動化を実現し、イベント駆動型の対応を有効にすることです。これらのプロセスではまた、セキュリティシステムに関連データを自動的に追加する必要があります。たとえば、希望しない IP アドレスからのトラフィックが関与するインシデントが起こると、AWS WAF

ブロックリストまたは Network Firewall ルールグループに自動的に入力されて、それ以上のアクティビティが防止されます。

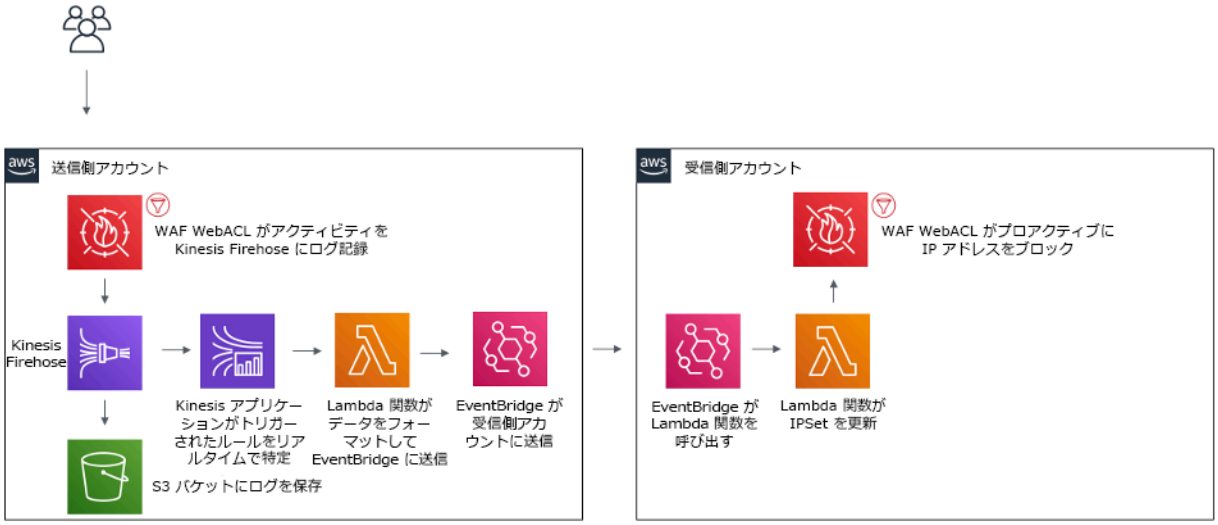


図 3: AWS WAF が既知の悪意ある IP アドレスのブロックを自動化します。

イベント駆動型の対応システムでは、検出メカニズムによって対応メカニズムがトリガーされ、自動的にイベントが修正されます。イベント駆動型の応答機能を使用すれば、検出メカニズムから応答メカニズムが動作するまでの時間を短縮できます。このイベント駆動型アーキテクチャを作成するには、AWS Lambda を使用できます。AWS Lambda は、イベントに応答してコードを実行し、基盤となるコンピューティングリソースを自動的に管理するサーバーレスコンピューティングサービスです。例えば、AWS CloudTrail サービスが有効な AWS アカウントがあるとします。AWS CloudTrail が無効になっている場合は (`cloudtrail:StopLogging API ########`)# Amazon EventBridge を使用して特定の (`cloudtrail:StopLogging API ########`)# イベントをモニタリングし、AWS Lambda 関数を起動して `cloudtrail:StartLogging` を呼び出してログを再開できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

封じ込め機能を自動化します。

リソース

関連するドキュメント:

- AWS インシデント対応ガイド

関連動画:

- AWS 環境のセキュリティインシデントの準備と対応

SEC10-BP05 アクセスを事前プロビジョニングする

インシデント対応者が AWS に事前プロビジョニングされた正しいアクセス権を持っていることを検証しておき、調査から復旧までに必要な時間を短縮します。

一般的なアンチパターン:

- ルートアカウントをインシデント対応に使用する
- 既存のユーザーアカウントに変更を加える
- ジャストインタイムの権限昇格を提供する際に IAM アクセス許可を直接操作する

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

AWS は、可能であれば長期的な認証情報への依存を削減または排除し、一時的な認証情報と ジャストインタイム の権限昇格メカニズムを優先することを推奨します。長期的な認証情報は、セキュリティリスクにさらされやすく、オペレーションのオーバーヘッドを増大させます。ほとんどの管理タスクと、インシデント対応タスクについては、管理アクセスの一時的な昇格と併せて ID フェデレーション を実装することを お勧めします。このモデルでは、ユーザーはより高いレベルの権限 (インシデント対応ロールなど) への昇格をリクエストします。ユーザーに昇格の資格がある場合、リクエストは承認者に送信されます。リクエストが承認された場合、ユーザーは、一時的な AWS 認証情報 のセットを受け取り、これを使用してタスクを完了できます。これらの認証情報の期限が切れたら、ユーザーは新たな昇格リクエストを送信する必要があります。

インシデント対応の大半のケースでは、一時的な権限昇格を使用することをお勧めします。そのための適切な方法は、 AWS Security Token Service および セッションポリシー を使用してアクセスのスコープを定義することです。

ID フェデレーションを使用できないケースがあります。例えば次のケースです。

- 侵害を受けた ID プロバイダー (IdP) に関連する停止状態
- 設定ミスや人的エラーに起因する、フェデレーションアクセス管理システムの障害
- 分散型サービス拒否 (DDoS) イベントやシステムがレンダリング不可となるなどの悪意あるアクティビティ

上記のケースでは、緊急 break glass アクセス設定により、インシデントの調査とタイムリーな修復を許可する必要があります。AWS は、適切なアクセス許可を持つ IAM ユーザー を使用することをお勧めします。IAM ユーザーがタスクを実行し AWS のリソースにアクセスするための適切な許可を付与します。ルート認証情報は、ルートユーザーアクセスが必要なタスクのみに使用します。インシデント対応者が AWS と他の関連システムへの適切なレベルのアクセス権を持っていることを検証するには、専用のユーザーアカウントへの事前プロビジョニングをお勧めします。このユーザーアカウントには特権アクセスが必要で、アカウントは厳格に制御、監視されなければなりません。このアカウントは、必要なタスクの実行で要求される最小特権で構成しなければなりません。アクセス権のレベルは、インシデント管理計画の一環として作成されたプレイブックに基づいている必要があります。

ベストプラクティスとして、特定の目的のための専用のユーザーとロールを使用します。IAM ポリシーの追加によりユーザーまたはロールアクセスを一時的に昇格させると、インシデント対応中にユーザーがどのアクセス権を持っていたかが明確でなくなり、昇格された権限が取り消されないリスクが生じます。

できるだけ多くの依存関係を削除し、できるだけ多くの障害シナリオでアクセスが可能になることを検証することが重要です。そのためには、インシデント対応ユーザーが、専用のセキュリティアカウントで AWS Identity and Access Management ユーザーとして作成されており、既存のフェデレーションまたはシングルサインオン (SSO) ソリューションにより管理されていないことを検証するためのプレイブックを作成します。個々のインシデント対応者は、自分の名前が付いたアカウントを持つ必要があります。アカウント設定では、強力なパスワードポリシー および多要素認証 (MFA) を適用する必要があります。インシデント対応プレイブックで AWS Management Console へのアクセスのみが要求されている場合、そのユーザーのアクセスキーが設定されてはならず、アクセスキー作成を明示的に禁止する必要があります。これは IAM ポリシーまたはサービスコントロールポリシー (SCP) で設定できます。詳細は、『AWS Security Best Practices for AWS Organizations SCPs』(AWS Organizations SCP のための AWS セキュリティベストプラクティス) に記載されています。ユーザーは、他のアカウントのインシデント対応ロールを引き受ける以外の権限を持つべきではありません。

インシデント対応中、調査、修復、または復旧アクティビティをサポートするためのアクセス権を社内または社外の他の個人に付与する必要が生じる可能性があります。この場合、前述のプレイブックメカニズムを使用します。また、インシデント完結後直ちに追加のアクセス権を取り消すためのプロセスが必要です。

インシデント対応ロールの使用が適切に監視および監査されていることを検証するには、この目的のために作成された IAM ユーザーアカウントが個人間で共有されないようにすること、および特定のタスクで必要な場合を除き、AWS アカウント ルートユーザーが使用されないようにすることが 不

可欠です。ルートユーザーが必要な場合 (例えば、特定のアカウントへの IAM アクセスが利用できない場合) は、用意されたプレイブックに従って別個のプロセスを使用し、ルートユーザーのパスワードと MFA トークンの使用の可否を検証します。

インシデント対応ロールのための IAM ポリシーを設定するには、 IAM Access Analyzer を使用し AWS CloudTrail ログに基づいてポリシーを生成することを検討します。そのためには、非本番アカウントのインシデント対応ロールに管理者アクセス権を付与し、プレイブックを一通り実行します。完了したら、実行されたアクションのみを許可するポリシーを作成できます。このポリシーは、すべてのアカウントのすべてのインシデント対応ロールに適用できます。各プレイブックについて個別の IAM ポリシーを作成すると、管理と監査が容易になるでしょう。プレイブックの例には、ランサムウェア、データ侵害、本番環境へのアクセス不可、その他のシナリオについての対応計画が含まれています。

インシデント対応ユーザーアカウントを使用して、 別の AWS アカウント アカウントのインシデント対応専用の IAM ロールを引き受けます。これらのロールは、セキュリティアカウントのユーザーのみが引き受け可能なように設定する必要があります。信頼関係では、呼び出しプリンシパルが MFA を使用して認証されたことを要求する必要があります。ロールは、スコープが厳密に定義された IAM ポリシーを使用してアクセスを制御する必要があります。これらのロールに対するすべての `AssumeRole` リクエストが CloudTrail ログに記録され、アラートが送信されるようにします。また、これらのロールを使用して実行されたアクションがログに記録されるようにします。

IAM ユーザーアカウントと IAM ロールの両方を CloudTrail ログで見つけやすくするために、これらに明快な名前を付けることを強くお勧めします。例えば、IAM アカウントに *<USER_ID>*`break-glass` 、IAM ロールに `BREAK-GLASS-ROLE` という名前を付けます。

CloudTrail を使用して、AWS アカウントの API アクティビティを口グに記録します。また、 インシデント対応ロールの使用状況に関するアラートを設定する必要があります。ルートキーを使用する際のアラートの設定に関するブログ記事を参照してください。インストラクションに変更を加えて、Amazon CloudWatch メトリクスフィルターを `AssumeRole` イベント (インシデント対応 IAM ロールに関連する) に対して設定できます。

```
{ $.eventName = "AssumeRole" && $.requestParameters.roleArn =
 "<INCIDENT_RESPONSE_ROLE_ARN>" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !
= "AwsServiceEvent" }
```

インシデント対応ロールは高いレベルのアクセス権を持っている可能性があるため、これらのアラートは幅広いグループに送信され、速やかに対応が取られることが重要です。

インシデント対応中、対応者は、IAM によって直接保護されていないシステムへのアクセスが必要となる可能性があります。これには Amazon Elastic Compute Cloud インスタンス、Amazon Relational Database Service データベース、Software-as-a-Service (SaaS) プラットフォームが含まれます。SSH や RDP などのネイティブプロトコルではなく、AWS Systems Manager Session Manager を使用して Amazon EC2 インスタンスへの管理アクセスを行うことを強くお勧めします。このアクセスは、IAM を使用して制御できます。それにより安全が確保され、監査が行われます。また、 AWS Systems Manager Run Command ドキュメントを使用してプレイブックの一部を自動化することも可能です。それにより、ユーザーのエラーを減らし、復旧にかかる時間を短縮できます。データベースとサードパーティーツールへのアクセスでは、アクセス認証情報を AWS Secrets Manager に保管し、インシデント対応者ロールにアクセス権を付与することをお勧めします。

最後に、インシデント対応 IAM ユーザーアカウントの管理は、 Joiners、Movers、および Leavers プロセス に追加し、定期的にテストして、意図されたアクセスのみが許可されていることを検証する必要があります。

リソース

関連するドキュメント:

- Managing temporary elevated access to your AWS environment (AWS 環境へのアクセスの一時的な昇格の管理)
- AWS Security Incident Response Guide (AWS セキュリティインシデント対応ガイド)
- AWS Elastic Disaster Recovery
- AWS Systems Manager Incident Manager
- IAM ユーザー用のアカウントパスワードポリシーの設定
- AWS での多要素認証 (MFA) の使用
- Configuring Cross-Account Access with MFA (MFA を使用したクロスアカウントアクセスの設定)
- Using IAM Access Analyzer to generate IAM policies (IAM Access Analyzer を使用した IAM ポリシーの設定)
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment (マルチアカウント環境の AWS Organizations サービスコントロールポリシーのためのベストプラクティス)
- How to Receive Notifications When Your AWS Account's Root Access Keys Are Used (AWS アカウントのルートアクセスキーを使用した場合の通知の受信方法)
- Create fine-grained session permissions using IAM managed policies(AWS マネージドポリシーを使用して、きめ細かいセッション許可を作成する)

関連動画:

- AWS のインシデント対応とフォレンジックの自動化
- ランブック、インシデントレポート、インシデント対応の DIY ガイド
- AWS 環境のセキュリティインシデントの準備と対応

関連サンプル:

- ラボ: AWS Account Setup and Root User (AWS アカウントのセットアップとルートユーザー)
- ラボ: Incident Response with AWS Console and CLI (AWS コンソールと CLI を使用したインシデント対応)

SEC10-BP06 ツールを事前デプロイする

復旧までの調査時間を短縮できるように、セキュリティ担当者は適切なツールを AWS に事前にデプロイしておきます。

セキュリティエンジニアリングとオペレーションの機能を自動化するために、AWS の包括的な API とツールセットを使用できます。ID 管理、ネットワークセキュリティ、データ保護、モニタリング機能を完全に自動化し、すでに導入されている一般的なソフトウェア開発方法を使用して提供できます。セキュリティオートメーションを構築すれば、担当者がセキュリティ上の位置づけを監視し、イベントに手動で応答する代わりに、システムが監視、レビューを行い応答を開始できます。AWS サービス間で検索可能で関連性の高いログデータをインシデント対応者に自動的に提供する効果的な方法は、次を有効にすることです: Amazon Detective.

インシデント対応チームが同じ方法でアラートに対応し続けると、アラート疲れになるリスクがあります。時間の経過とともに、チームはアラートに対する感度が鈍くなり、通常の状況の処理で間違いを犯したり、異常なアラートを見逃したりする可能性があります。自動化を利用すれば、繰り返し発生する通常のアラートを処理する機能を使用してアラート疲れを回避し、機密性の高いインシデントや独自のインシデントの処理を人間に任せることができます。Amazon GuardDuty, AWS CloudTrail Insights、および Amazon CloudWatch Anomaly Detection などの異常の検出システムを統合することで、よくあるしきい値ベースのアラートの負担を減らすことができます。

プロセス内のステップをプログラムで自動化すれば、手動プロセスを改善できます。イベントに対する修復パターンを定義したら、そのパターンを実行可能なロジックに分解して、そのロジックを実行するコードを記述できます。その後、対応者は、そのコードを実行して問題を修正します。時間の経過とともに、より多くのステップを自動化し、最終的には一般的なインシデントのクラス全体を自動的に処理できるようになります。

Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのオペレーティングシステム内で実行されるツールでは、AWS Systems Manager Run Command の使用を評価する必要があります。このコマンドを使うと、Amazon EC2 インスタンスのオペレーティングシステムにインストールしたエージェントを使用して、インスタンスをリモートで安全に管理できます。その際、Systems Manager Agent (SSM Agent) が必要です。これは多くの Amazon マシンイメージ (AMI) にデフォルトでインストールされています。ただし、一度インスタンスが侵害されると、そのインスタンス上で実行されているツールやエージェントからの応答は信頼できる応答とみなされません。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- ツールを事前デプロイする: セキュリティ担当者がインシデント発生時に適切な対応ができるよう、AWS に適切なツールをあらかじめ配備しておきます。
  - ラボ : AWS Management Console と CLI を使用したインシデント対応
  - Jupyter を使用したインシデント対応プレイブック - AWS IAM
  - AWS セキュリティオートメーション
- リソースのタグ付けを実施する: インシデント発生時にリソースを特定できるように、調査中のリソースのコードなどの情報をリソースにタグ付けします。
  - AWS のタグ付け戦略

リソース

関連するドキュメント:

- AWS Incident Response Guide (AWS インシデント対応ガイド)

関連動画:

- ランブック、インシデントレポート、インシデント対応の DIY ガイド

SEC10-BP07 ゲームデーを実施する

ゲームデーは、シミュレーションや演習とも呼ばれ、現実的なシナリオでインシデント管理計画や手順を練習するための体系的な機会を提供する内部イベントです。これらのイベントは、実際のシナリオで使用されるのと同じツールやテクニックを使って、レスポンダーを訓練するものでなければなり

ません。ゲームデーは基本的に、準備をすることで対応能力を反復的に高めていくものです。ゲームデーのアクティビティを実行すべき理由は、次のとおりです。

- 準備態勢を検証する
- 自信の向上 - シミュレーションやトレーニングスタッフから学ぶ
- コンプライアンスまたは契約上の義務に準拠する
- 認定のためのアーティファクトを生成する
- 俊敏性 - 段階的な改善
- 高速化とツールの改善
- コミュニケーションとエスカレーションを詳細化する
- まれで予期外の事態に備える

このような理由から、シミュレーションアクティビティへの参加は、ストレスの多いイベント時の組織の有効性を高めるという価値があります。現実的で有益なシミュレーションアクティビティを開発するのは簡単な作業ではありません。すでに把握しているイベントを処理する手順や自動化のテストには一定のメリットがありますが、予想外の事象に対して自身をテストして継続的に改善するクリエイティブな セキュリティインシデント対応のシミュレーション (SIRS) アクティビティへの参加も重要です。

環境、チーム、ツールに合わせたカスタムのシミュレーションを作成します。問題を見つけて、それに関するシミュレーションを設計します。これは、漏洩した認証情報、不要なシステムと通信しているサーバー、または不正な露出をもたらす設定ミスなどが考えられます。組織に精通したエンジニアを特定して、シナリオと参加する別のグループを作成してもらいます。シナリオは現実的で、十分に価値のある挑戦的なものであるべきです。ロギング、通知、エスカレーション、ランブックまたは自動化の実行を実践する機会も含まれているはずです。シミュレーション中、レスポンダーは技術的および組織的スキルを発揮し、リーダーはインシデント管理スキルを高めるために参加する必要があります。シミュレーションの終わりには、チームの努力を称え、さらなるシミュレーションの反復、繰り返し、拡張の方法を探します。

AWS は、インシデント対応ランブックのテンプレートを作成しました。 これは、対応策の準備だけでなく、シミュレーションのベースとしても活用できます。計画時、シミュレーションは 5 段階に分けられます。

証拠の収集: この段階では、内部チケッティングシステム、モニタリングツールからのアラート、匿名のヒント、または公共のニュースなどさまざまな手法を使ってアラートを取得します。次にチームはインフラストラクチャとアプリケーションログのレビューを開始して、侵害のソースを特定しま

す。このステップでは、内部エスカレーションとインシデントリーダーシップも関与する必要があります。特定されたら、チームがインシデントの封じ込めに取り掛かります。

インシデントを封じ込める: チームはインシデント発生を特定し、侵害のソースを突き止めます。チームは次に、侵害された認証情報の無効化、コンピューティングリソースの隔離、またはロールのアクセス許可の取り消しなど、封じ込めるためのアクションを取る必要があります。

インシデントを根絶する これでインシデントが封じ込められたため、チームは侵害を受けやすいアプリケーションやインフラストラクチャ設定の脆弱性を軽減する作業に取り掛かります。これには、ワークロードに使用された認証情報のローテーション、アクセスコントロールリスト (ACL) の修正、またはネットワーク設定の変更などが含まれます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 実行 本番環境で実行する: 主なスタッフやマネジメントが関与するさまざまな脅威に対してシミュレーションされた インシデント 対応 イベントを 実行します (ゲームデー)。
- 教訓を把握する: 本番環境で実行する から得た教訓は、プロセスを改善するためのフィードバックループの一部であるべきです。

リソース

関連するドキュメント:

- AWS Incident Response Guide (AWS インシデント対応ガイド)
- AWS Elastic Disaster Recovery

関連動画:

- ランブック、インシデントレポート、インシデント対応の DIY ガイド

信頼性

トピック

- 基盤
- ワークロードアーキテクチャ

- 変更管理
- 障害管理

基盤

質問

- REL 1 サービスクォータと制約はどのように管理しますか?
- REL 2 ネットワークトポロジをどのように計画しますか?

REL 1 サービスクォータと制約はどのように管理しますか?

クラウドベースのワークロードアーキテクチャには、サービスクォータ (サービスの制限とも呼ばれます) というものがあります。このようなクォータは、誤って必要以上のリソースをプロビジョニングするのを防ぎ、サービスを不正使用から保護することを目的として API 操作のリクエスト頻度を制限するために存在します。リソースにも制約があります。たとえば、光ファイバーケーブルのビットレートや、物理ディスクの記憶容量などです。

ベストプラクティス

- REL01-BP01 サービスクォータと制約を認識する
- REL01-BP02 アカウントおよびリージョンをまたいでサービスクォータを管理する
- REL01-BP03 アーキテクチャを通じて、固定サービスクォータと制約に対応する
- REL01-BP04 クォータをモニタリングおよび管理する
- REL01-BP05 クォータ管理を自動化する
- REL01-BP06 フェイルオーバーに対応するために、現在のクォータと最大使用量の間に十分なギャップがあることを確認する

REL01-BP01 サービスクォータと制約を認識する

あなたは、ワークロードアーキテクチャに対するデフォルトのクォータとクォータ引き上げリクエストを認識しています。さらに、ディスクやネットワークなど、どのリソースの制約が潜在的に大きな影響を与えるかを知っておきましょう。

Service Quotas は、100 を超える AWS のサービスのクォータを一元的に管理するのに役立つ AWS のサービスです。クォータ値の検索に加えて、Service Quotas コンソールから、または AWS SDK 経由でクォータ増加をリクエスト、追跡することもできます。AWS Trusted Advisor には、あるサー

ビスの一部の要素に関する使用状況とクォータを表示するサービスクォータチェックが用意されています。サービスごとのデフォルトのサービスクォータは、それぞれのサービスごとの AWS ドキュメントにも記載されています。例えば、次を参照してください。 Amazon VPC クォータ。スロットルされた API のレート制限は、API Gateway 内で使用量プランを変更することで設定できます。それぞれのサービス上の構成として設定される他の制限には、プロビジョンド IOPS、割り当てられた RDS ストレージ、EBS ボリューム割り当てなどがあります。Amazon Elastic Compute Cloud (Amazon EC2) には、インスタンス、Amazon Elastic Block Store (Amazon EBS)、および Elastic IP アドレスの制限を管理するのに役立つ独自のサービスの制限ダッシュボードがあります。サービスクォータがアプリケーションのパフォーマンスに影響を及ぼし、ニーズに合わせて調整できないような事例が発生した場合は、AWS Support に連絡し、緩和策の有無についてお問い合わせください。

一般的なアンチパターン:

- 使用される AWS のサービスに対するサービスクォータを考慮に入れることなく、ワークロードをデプロイする。
- AWS のサービスの設計上の制約を調査、対応することなく、ワークロードを設計する。
- 必要なクォータを設定することなく、また AWS Support に事前に連絡することなく、既知の既存のワークロードを置き換えるような、使用量の多いワークロードを導入すること。
- ワークロードへのトラフィックを促進するイベントを計画したが、必要なクォータを設定せず、事前に AWS Support に連絡しない。

このベストプラクティスを確立するメリット: サービスクォータ、API スロットリング制限、および設計制約を認識することで、ワークロードの設計、実装、およびオペレーションでこれらを考慮することができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- 公開されているドキュメントと Service Quotas で AWS のサービスクォータを確認します
  - AWS Service Quotas (旧称は「Liimits (制限)」)
- デプロイコードを見て、ワークロードに必要なすべてのサービスを決定します。
- AWS Config を使用して、AWS アカウント で使用するすべての AWS リソースを検索します。
  - AWS Config によってサポートされている AWS リソースタイプとリソース関係
- AWS CloudFormation を使用して、使用する AWS リソースを決めることもできます。AWS Management Console で、または list-stack-resources CLI コマンドで作成されたリソースを確認します。テンプレート自体にデプロイされるように設定されたリソースも確認できます。

  - AWS Management Console での AWS CloudFormation スタックデータとリソースの表示
  - AWS CLI for CloudFormation: list-stack-resources
- 適用されるサービスクォータを決定します。Trusted Advisor および Service Quotas を通じて、プログラムでアクセスできる情報を使用します。

リソース

関連するドキュメント:

- AWS Marketplace: 制限の追跡に役立つ CMDB 製品
- AWS Service Quotas (旧称は「Liimits (制限)」)
- AWS Trusted Advisor ベストプラクティスチェックリスト (「サービスの制限」セクションを参照)
- AWS Answers の AWS Limit Monitor
- Amazon EC2 サービスの制限
- Service Quotas とは何ですか?

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP02 アカウントおよびリージョンをまたいでサービスクォータを管理する

複数の AWS アカウント または AWS リージョン をご利用の場合は、必ず本番ワークロードを実行するすべての環境で適切なクォータをリクエストしてください。

サービスクォータの追跡はアカウントごとに行います。特に明記されていない限り、各クォータは AWS リージョン 固有です。テストと開発が妨げられないように、本番環境に加えて、該当するすべての非本番環境でもクォータを管理します。

一般的なアンチパターン:

- 1 つの分離ゾーンでのリソース使用率の増加を許可し、他のゾーンにはキャパシティーを維持するメカニズムがない。
- 分離ゾーン内のすべてのクォータを手動で設定する。
- デプロイが失われた場合に別のリージョンからのトラフィック増加に対応できるように、リージョン別に分離されたデプロイが適切なサイズとなっていない。

このベストプラクティスを活用するメリット: 分離ゾーンが使用できない場合に現在の負荷を処理できることを確認することで、フェイルオーバー中に発生するエラーの数を減らすことができ、顧客にサービス拒否を発生させないようにします。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- サービスの要件、レイテンシー、およびディザスタリカバリ (DR) 要件に基づいて、関連するアカウントとリージョンを選択します。
- 関連するすべてのアカウント、リージョン、アベイラビリティーゾーン全体のサービスクォータを特定します。制限の対象範囲はアカウントとリージョンです。
- Service Quotas とは何ですか?

リソース

関連するドキュメント:

- AWS Marketplace: CMDB products that help track limits
- AWS Service Quotas (旧称は「サービスの制限」)
- AWS Trusted Advisor ベストプラクティスチェックリスト (「サービスの制限」セクションを参照)
- AWS Answers の AWS Limit Monitor
- Amazon EC2 サービスの制限
- Service Quotas とは何ですか?

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP03 アーキテクチャを通じて、固定サービスクォータと制約に対応する

サービスクォータと物理リソースには変更できないものもあることに注意し、これらが信頼性に影響を及ぼさないように設計します。

例としては、ネットワーク帯域幅、AWS Lambda ペイロードサイズ、API Gateway のスロットルバーストレート、Amazon Redshift クラスターへの同時ユーザー接続などがあります。

一般的なアンチパターン:

- バースト制限を利用しながらベンチマークをごく短期間実行するが、継続する期間にわたってそのキャパシティーでサービスが実行されることが予想される。
- ユーザーまたは顧客ごとに 1 つのサービスリソースを使用する設計を選択するが、この設計の制約 (スケーリング時にこの設計の失敗を引き起こすもの) があることを認識していない。

このベストプラクティスを活用するメリット: AWS のサービスの固定クォータと、接続の制約、IP アドレスの制約、サードパーティーのサービスの制約など、ワークロードの他の部分の制約を追跡することで、クォータに対する傾向を検出し、クォータを超える前にそのクォータに対応できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 固定されたサービスクォータと制約のほか、これらに関するアーキテクトを把握します。
  - AWS Service Quotas

リソース

関連するドキュメント:

- AWS Marketplace: 制限の追跡に役立つ CMDB 製品
- AWS Service Quotas (旧称は「サービスの制限」)
- AWS Trusted Advisor ベストプラクティスのチェック (「サービスの制限」セクションを参照)
- AWS Answers の AWS Limit Monitor
- Amazon EC2 サービスの制限
- 「What Is Service Quotas?」

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP04 クォータをモニタリングおよび管理する

予想される使用量を評価し、クォータを必要に応じて引き上げて、使用量を予定通り増やせるようにします。

サポートされているサービスの場合、CloudWatch アラームを設定して使用量をモニタリングし、クォータのしきい値に近づいていることのアラームを受けることで、クォータを管理できます。このアラームは、Service Quotas または Trusted Advisor からトリガーできます。CloudWatch Logs のメトリクスフィルターを使用して、ログのパターンを検索および抽出し、使用量がクォータのしきい値に近づいているかどうかを判断することもできます。

一般的なアンチパターン:

- Service Quotas に近づいている場合のアラートを設定するが、アラートへの対応方法に関するプロセスがない。
- Service Quotas でサポートされているサービスのアラームのみを設定し、他のサービスのモニタリングを行わない。

このベストプラクティスを活用するメリット: AWS のサービスクォータの自動追跡と、それらのクォータに対する使用率のモニタリングにより、クォータの限界に近づいていることを確認できます。また、このモニタリングデータを使用して、コストを節約するためにクォータを下げるタイミングを評価することもできます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- AWS で予想される使用量を評価し、リージョンごとのサービスクォータを適切に増やして、計画的な使用量の増加を可能にします。
  - 現在のリソース消費 (バケットやインスタンスなど) を把握します。現在のリソース消費を収集するには、Amazon EC2 DescribeInstances API などのサービス API オペレーションを使用します。
  - 現在のクォータを把握する AWS Service Quotas、AWS Trusted Advisor、AWS のドキュメントを使用します。
    - AWS Service Quotas を使用します。これは、100 を超える AWS のサービスのクォータを一元的に管理するのに役立つ AWS のサービスです。
    - Trusted Advisor のサービスの制限を使用して、現在のサービスの制限を決定します。
    - サポートされている場合は、サービス API オペレーションを使用して、現在のサービスクォータを決定します。
    - リクエストしたクォータの増加とそれらのステータスを記録する クォータの増加が承認された後、クォータの変更を反映するように記録を更新します。

リソース

関連するドキュメント:

- AWS Marketplace: CMDB products that help track limits
- AWS Service Quotas (旧称は「サービスの制限」)
- サービスの制限に関する AWS Trusted Advisor ベストプラクティスのチェック
- AWS Answers の AWS Limit Monitor
- Amazon EC2 サービスの制限
- 「What Is Service Quotas?」
- Amazon CloudWatch を使用して Service Quotas をモニタリングする

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP05 クォータ管理を自動化する

しきい値に近づいたときに警告するツールを実装します。AWS Service Quotas API を使用すると、クォータの引き上げリクエストを自動化できます。

お使いの Configuration Management Database (CMDB) またはチケット発行システムを Service Quotas と統合すると、クォータの引き上げリクエストと現在のクォータに関する情報のトラッキングを自動化できます。AWS SDK のほかに、Service Quotas も AWS Command Line Interface (AWS CLI) を使用した自動化を提供しています。

一般的なアンチパターン:

- スプレッドシートでクォータと使用状況を追跡する。
- 毎日、毎週、または毎月の使用状況に関するレポートを実行し、使用量とクォータを比較する。

このベストプラクティスを活用するメリット: AWS のサービスクォータの自動追跡と、そのクォータに対する使用量のモニタリングにより、クォータに近づいていることを確認できます。必要に応じてクォータの引き上げをリクエストできるように、オートメーションを設定できます。使用量の傾向が反対の方向にある場合は、(認証情報が漏えいした場合の) リスクの低下とコスト削減のメリットを実現するために、クォータの削減を検討することをお勧めします。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 自動モニタリングをセットアップする しきい値に近づいたときにアラートを発行するため、SDK を使用するツールを実装します。
  - Service Quotas を使用し、AWS Limit Monitor や AWS Marketplace からのサービスなど、自動クォータモニタリングソリューションでサービスを補強します。
    - Service Quotas とは何ですか?
    - AWS でのクォータモニタリング - AWS ソリューション
  - Amazon SNS および AWS Service Quotas API を使用して、クォータしきい値に基づいてトリガーされるレスポンスをセットアップします。
  - 自動化をテストします。
    - 制限のしきい値を設定します。
    - AWS Config、デプロイパイプライン、Amazon EventBridge、またはサードパーティーからの変更イベントと統合します。
    - 応答をテストするために、人為的に低いクォータしきい値を設定します。
    - 通知に対して適切なアクションを取り、必要に応じて AWS Support に問い合わせるトリガーをセットアップします。
    - 変更イベントを手動でトリガーします。
    - ゲームデーを実行して、クォータ引き上げの変更プロセスをテストします。

リソース

関連するドキュメント:

- APN パートナー: 設定管理を支援できるパートナー
- AWS Marketplace: 制限の追跡に役立つ CMDB 製品
- AWS Service Quotas (旧称は「サービスの制限」)
- AWS Trusted Advisor ベストプラクティスのチェック (「サービスの制限」セクションを参照)
- AWS でのクォータモニタリング - AWS ソリューション
- Amazon EC2 サービスの制限
- Service Quotas とは何ですか?

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas

REL01-BP06 フェイルオーバーに対応するために、現在のクォータと最大使用量の間に十分なギャップがあることを確認する

リソースに障害が発生したときには、リソースが正常に終了されるまで、クォータにカウントされることがあります。エラーが生じたリソースが停止されるまで、エラーが生じたすべてのリソースと代替リソースの合計リソース数がクォータ内に収まるようにします。この開きを計算するときは、アベイラビリティーゾーンの不具合を考慮する必要があります。

一般的なアンチパターン:

- フェイルオーバーシナリオを考慮せずに、現在のニーズに基づいてサービスクォータを設定する。

このベストプラクティスを活用するメリット: イベントが可用性に影響する恐れがあるとき、クラウドでは、これらのイベントを緩和またはイベントから復旧するための戦略を実装できます。そのような戦略には、多くの場合、障害が発生したリソースに置き換わる追加リソースの作成が含まれます。クォータ戦略は、これらの追加リソースに対応する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- フェイルオーバーに対応するため、サービスクォータと最大使用量の間に十分なギャップがあることを確認します。
  - デプロイのパターン、可用性の要件、消費の増加を考慮して、サービスクォータを決定します。
  - 必要に応じてクォータの引き上げをリクエストします。クォータの引き上げリクエストの実行に必要な時間を計画します。
    - 信頼性の要件 (「9 の数」としても知られる) を決定します。
    - 障害シナリオ (コンポーネント、アベイラビリティーゾーン、リージョンの損失など) を確立します。
    - デプロイ手法 (例えば、Canary、ブルー/グリーン、レッド/ブラック、ローリングなど) を確立します。
    - 現在の制限に適切なバッファ (例えば、15%) を含めます。
    - 消費の増加を計画します (例えば、消費の傾向をモニタリングする)。

リソース

関連するドキュメント:

- AWS Marketplace: CMDB products that help track limits
- AWS Service Quotas (旧称は「サービスの制限」)
- AWS Trusted Advisor ベストプラクティスチェックリスト (「サービスの制限」セクションを参照)
- Amazon EC2 サービスの制限
- 「What Is Service Quotas?」

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas

REL 2 ネットワークトポロジをどのように計画しますか?

多くの場合、ワークロードは複数の環境に存在します。このような環境には、複数のクラウド環境 (パブリックにアクセス可能なクラウド環境とプライベートの両方) と既存のデータセンターインフラストラクチャなどがあります。計画する際は、システム内およびシステム間の接続、パブリック IP アドレスの管理、プライベート IP アドレスの管理、ドメイン名解決といったネットワークに関する諸点も考慮する必要があります。

ベストプラクティス

- REL02-BP01 ワークロードのパブリックエンドポイントに高可用性ネットワーク接続を使用する
- REL02-BP02 クラウド環境とオンプレミス環境のプライベートネットワーク間の冗長接続をプロビジョニングする
- REL02-BP03 拡張性と可用性を考慮した IP サブネットの割り当てを確実に行う
- REL02-BP04 多対多メッシュよりもハブアンドスポークトポロジを優先する
- REL02-BP05 接続されているすべてのプライベートアドレス空間において、重複しないプライベート IP アドレス範囲を指定する

REL02-BP01 ワークロードのパブリックエンドポイントに高可用性ネットワーク接続を使用する

これらのエンドポイントとそれらへのルーティングは、可用性が高い必要があります。これを実現するには、可用性の高い DNS、コンテンツ配信ネットワーク (CDN)、API Gateway、ロードバランシング、またはリバースプロキシを使用します。

Amazon Route 53、AWS Global Accelerator、Amazon CloudFront、Amazon API Gateway、Elastic Load Balancing (ELB) はすべて、高可用性パブリックエンドポイントを提供します。また、ロードバランシングとプロキシ処理を行うために、AWS Marketplace のソフトウェアアプライアンスを評価することもできます。

ワークロードで提供されるサービスの消費者は、エンドユーザーであろうと他のサービスのユーザーであろうと、これらのサービスエンドポイントでリクエストを行います。高可用性のエンドポイントを提供できるように、いくつかの AWS リソースが利用できます。

Elastic Load Balancing は、アベイラビリティーゾーン間のロードバランシングを提供し、レイヤー 4 (TCP) またはレイヤー 7 (http/https) のルーティングを実施し、AWS WAF と統合します。また、AWS Auto Scaling との統合により、自己回復型のインフラストラクチャを構築し、トラフィックの増加を吸収しながら、トラフィックが減少したときにリソースを解放することができます。

Amazon Route 53 は、スケーラブルで可用性の高いドメインネームシステム (DNS) サービスであり、Amazon EC2 インスタンス、Elastic Load Balancing ロードバランサー、Amazon S3 バケットなど、AWS で実行するインフラストラクチャとユーザーリクエストを接続します。これはユーザーを AWS の外部のインフラストラクチャにルーティングするためにも使用できます。

AWS Global Accelerator は、AWS グローバルネットワーク上で最適なエンドポイントにトラフィックを誘導するために使用できるネットワークレイヤーサービスです。

分散型サービス拒否 (DDoS) 攻撃は、正当なトラフィックをシャットアウトして、ユーザーの可用性の低下を招きかねません。AWS Shield は、ワークロードにおける AWS サービスエンドポイントに対して、追加コストなしでこれらの攻撃に対する自動保護を提供します。これらの機能は、APN Partners や AWS Marketplace の仮想アプライアンスを使って、ニーズに合わせて拡張することができます。

一般的なアンチパターン:

- インスタンスまたはコンテナでパブリックインターネットアドレスを使用し、DNS 経由でそれらのアドレスへの接続を管理する。
- サービスの検索のために、ドメイン名ではなく、インターネットプロトコルアドレスを使用する。
- 大規模な地理的領域にコンテンツ (ウェブページ、静的アセット、メディアファイル) を提供し、コンテンツ配信ネットワークを使用しない。

このベストプラクティスを確立するメリット: ワークロードに可用性の高いサービスを実装することで、ユーザーがワークロードを利用できるようになることがわかります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードのユーザーのために高可用性接続を確保する Amazon Route 53、AWS Global Accelerator、Amazon CloudFront、Amazon API Gateway、Elastic Load Balancing (ELB) はすべて、高可用性パブリックエンドポイントを提供します。また、ロードバランシングとプロキシ処理を行うために、AWS Marketplace のソフトウェアアプライアンスを評価することもできます。

- ユーザーへの高可用性接続を確保します。
- 高可用性 DNS を使用して、アプリケーションエンドポイントのドメイン名を管理していることを確認します。
  - ユーザーがインターネット経由でアプリケーションにアクセスする場合は、サービス API オペレーションを使用して、インターネットゲートウェイの正しい使用状況を確認します。また、アプリケーションエンドポイントをホストするサブネットのルートテーブルのエントリが正しいことを確認します。
    - DescribeInternetGateways
    - DescribeRouteTables
- アプリケーションの前に高可用性リバースプロキシまたはロードバランサーを使用していることを確認します。
  - ユーザーがオンプレミス環境を通じてアプリケーションにアクセスする場合は、AWS とオンプレミス環境の間の接続が高可用性であることを確認します。
  - Route 53 を使用してドメイン名を管理します。
    - Amazon Route 53 とは?
  - 要件を満たすサードパーティーの DNS プロバイダーを使用します。
  - Elastic Load Balancing を使用します。
    - Elastic Load Balancing とは?
  - 要件を満たす AWS Marketplace アプライアンスを使用します。

リソース

関連するドキュメント:

- APN パートナー: ネットワークの計画を支援できるパートナー
- AWS Direct Connect の回復性に関する推奨事項

- ネットワークインフラストラクチャ向け AWS Marketplace
- Amazon Virtual Private Cloud の接続オプションホワイトペーパー
- Multiple data center HA network connectivity
- Direct Connect Resiliency Toolkit を使用して開始する
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- AWS Global Accelerator とは?
- 「Amazon VPC とは?」
- Transit Gateway とは?
- 「Amazon CloudFront とは」
- Amazon Route 53 とは?
- Elastic Load Balancing とは?
- Direct Connect ゲートウェイの操作

関連動画:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303) (高度な VPC 設計と Amazon VPC の新機能)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1) (多くの VPC に対応した AWS Transit Gateway のリファレンスアーキテクチャ)

REL02-BP02 クラウド環境とオンプレミス環境のプライベートネットワーク間の冗長接続をプロビジョニングする

個別にデプロイされたプライベートネットワーク間で複数の AWS Direct Connect 接続または VPN トンネルを使用します。高可用性のために複数の Direct Connect ロケーションを使用します。複数の AWS リージョン を使用している場合は、少なくとも 2 つのリージョンで冗長性を確保してください。VPN を終端する AWS Marketplace アプライアンスを評価したい場合があります。AWS Marketplace アプライアンスを使用する場合は、さまざまなアベイラビリティーゾーンで高可用性を実現するために冗長インスタンスをデプロイします。

AWS Direct Connect は、オンプレミス環境から AWS への専用ネットワーク接続を簡単に確立できるようにするクラウドサービスです。Direct Connect ゲートウェイを使用すると、オンプレミスのデータセンターを複数の AWS リージョン にまたがる複数の AWS VPC に接続できます。

この冗長性は、接続の回復力に影響を与える可能性のある障害に対処します。

- トポロジにおける障害に対する弾力性を高めるにはどうすればよいでしょうか?
- 間違った設定を行い、接続が停止したらどうなりますか?
- トラフィックとサービス利用量が予想外に増加した場合に対応できますか?
- 分散型サービス拒否 (DDoS) 攻撃の影響を緩和できますか?

VPC をVPN 経由でオンプレミスのデータセンターに接続するときは、ベンダーとそのアプライアンスを実行する際に必要となるインスタンスサイズを選択する際に、弾力性と必要とする帯域幅の要件を考慮する必要があります。使用するVPN アプライアンスがその実装において十分な弾力性がない場合、2 つ目のアプライアンスを通じて冗長接続を設定する必要があります。すべてのシナリオにおいて、許容可能な復旧時間を定義し、その要件を満たすかどうかをテストする必要があります。

Direct Connect 接続を使用して VPC をデータセンターに接続することにし、この接続を高可用性にする必要がある場合は、各データセンターから冗長化した Direct Connect 接続を使用します。冗長接続では、最初の接続とは異なる場所から 2 番目の Direct Connect 接続を使用する必要があります。複数のデータセンターがある場合は、接続が異なる場所で終端するようにしてください。このセットアップに役立てるため Direct Connect Resiliency Toolkit を使用します。

AWS VPN を使用してインターネット経由で VPN にフェイルオーバーする場合、VPN トンネルごとに最大 1.25 Gbps のスループットはサポートしますが、同じ VGW 上で終端する AWS マネージド VPN トンネルが複数ある場合、アウトバウンドトラフィック用の Equal Cost Multi Path (ECMP) はサポートしないということを理解しておくことが重要です。フェイルオーバー中に 1 Gbps 未満の速度を許容できないのであれば、Direct Connect 接続のバックアップとして AWS マネージド VPN を使用することはお勧めしません。

VPC エンドポイントを使用して、VPC を、パブリックインターネットを経由せずに、サポートされている AWS のサービスおよび AWS PrivateLink が提供する VPC エンドポイントサービスにプライベートに接続することもできます。エンドポイントは仮想デバイスです。これは、水平にスケーリングされ、冗長性と可用性に優れた VPC コンポーネントです。仮想デバイスにより、ネットワークトラフィックに可用性のリスクや帯域幅の制約を課すことなく、VPC 内のインスタンスとサービス間の通信が可能になります。

一般的なアンチパターン:

- オンサイトネットワークと AWS の間に接続プロバイダを 1 つだけ持つ。
- AWS Direct Connect 接続の接続機能を消費するが、接続は 1 つのみである。
- VPN 接続用のパスは 1 つだけである。

このベストプラクティスを活用するメリット: クラウド環境と企業/オンプレミス環境の間に冗長接続を実装することで、2 つの環境間で依存するサービスが確実に通信できるようになります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- AWS とオンプレミス環境との間に高可用性接続を確保します。個別にデプロイされたプライベートネットワーク間で複数の AWS Direct Connect 接続または VPN トンネルを使用します。高可用性のために複数の Direct Connect ロケーションを使用します。複数の AWS リージョン を使用している場合は、少なくとも 2 つのリージョンで冗長性を確保してください。VPN を終端する AWS Marketplace アプライアンスを評価したい場合があります。AWS Marketplace アプライアンスを使用する場合は、さまざまなアベイラビリティーゾーンで高可用性を実現するために冗長インスタンスをデプロイします。
  - オンプレミス環境への冗長接続を確保する可用性ニーズを達成するには、複数の AWS リージョン への冗長接続が必要な場合があります。
    - AWS Direct Connect の回復性に関する推奨事項
    - 冗長な Site-to-Site VPN 接続を使用してフェイルオーバーを提供する
      - サービス API オペレーションを使用して Direct Connect 回線の適切な使用方法を明確にします。
        - DescribeConnections
        - DescribeConnectionsOnInterconnect
        - DescribeDirectConnectGatewayAssociations
        - DescribeDirectConnectGatewayAttachments
        - DescribeDirectConnectGateways
        - DescribeHostedConnections
        - DescribeInterconnects
      - Direct Connect 接続が 1 つだけあるか、1 つもない場合は、仮想プライベートゲートウェイへの冗長 VPN トンネルを設定します。
        - AWS Site-to-Site VPN とは?
  - 現在の接続をキャプチャします (例えば、Direct Connect、仮想プライベートゲートウェイ、AWS Marketplace アプライアンス)。
    - サービス API オペレーションを使用して、Direct Connect 接続の設定をクエリします。
      - DescribeConnections

    - DescribeConnectionsOnInterconnect
    - DescribeDirectConnectGatewayAssociations
    - DescribeDirectConnectGatewayAttachments
    - DescribeDirectConnectGateways
    - DescribeHostedConnections
    - DescribeInterconnects
- サービス API オペレーションを使用して、ルートテーブルが使用している仮想プライベートゲートウェイを収集します。
    - DescribeVpnGateways
    - DescribeRouteTables
- サービス API オペレーションを使用してルートテーブルが使用している AWS Marketplace アプリケーションを収集します。
    - DescribeRouteTables

リソース

関連するドキュメント:

- APN パートナー: ネットワークの計画を支援できるパートナー
- AWS Direct Connect の回復性に関する推奨事項
- ネットワークインフラストラクチャ向け AWS Marketplace
- Amazon Virtual Private Cloud 接続オプションホワイトペーパー
- Multiple data center HA network connectivity
- 冗長な Site-to-Site VPN 接続を使用してフェイルオーバーを提供する
- Direct Connect Resiliency Toolkit を使用して開始する
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- 「Amazon VPC とは?」
- Transit Gateway とは?
- AWS Site-to-Site VPN とは?
- Direct Connect ゲートウェイの操作

関連動画:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)

REL02-BP03 拡張性と可用性を考慮した IP サブネットの割り当てを確実に行う

Amazon VPC IP アドレスの範囲は、将来の拡張や アベイラビリティゾーンをまたがるサブネットへの IP アドレスの割り当てを考慮し て、ワークロードの要件に対応するのに十分な大きさでなければなりません。これには、ロードバランサー、EC2 インスタンス、コンテナーベースのアプリケーションが含まれます。

ネットワークトポロジの計画は、IP アドレス空間の定義から始めます。プライベート IP アドレス範囲 (RFC 1918 ガイドラインに準拠) は、VPC ごとに割り当てる必要があります。このプロセスの一環として、次の要件を満たすようにします。

- リージョンごとに複数の VPC 用の IP アドレス空間を割り当てる。
- VPC 内で、複数のアベイラビリティーゾーンにまたがる複数のサブネット用の空間を割り当てます。
- 将来の拡張のために、常に未使用の CIDR ブロック空間を VPC 内に残しておきます。
- 機械学習用のスポットフリート、Amazon EMR クラスター、Amazon Redshift クラスターなど、使用する可能性のある EC2 インスタンスの一時的なフリートのニーズを満たす IP アドレス空間があることを確認します。
- 各サブネット CIDR ブロックの最初の 4 つの IP アドレスと最後の IP アドレスはリザーブドのため、お客様はご使用いただけません。
- 大きな VPC CIDR ブロックのデプロイを計画する必要があります。VPC に割り当てられた最初の VPC CIDR ブロックは変更または削除することはできませんが、重複していない CIDR ブロックを VPC に追加することはできます。サブネット IPv4 CIDR は変更できませんが、IPv6 CIDR は変更できます。最大規模の VPC (/16) をデプロイする場合、65,000 を超える IP アドレスが割り当てられることになります。ベース 10.x.x.x IP アドレス空間だけで、そのような VPC を 255 個プロビジョニングできます。したがって、VPC の管理を容易にするためには、小さすぎて失敗するよりも、大きすぎて失敗するほうが良いでしょう。

一般的なアンチパターン:

- 小さな VPC を作成する。
- 小さなサブネットを作成するため、増大に伴ってサブネットを設定に追加する必要がある。
- Elastic Load Balancing が使用できる IP アドレスの数を不正確に見積もる。

- 多数の高トラフィックロードバランサーを同じサブネットにデプロイする。

このベストプラクティスを確立するメリット: これにより、ワークロードの増大に対応し、スケールアップ時に可用性を引き続き提供できます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 拡張、コンプライアンス、他のネットワークとの統合に対応できるようにネットワークを計画します。適切に計画しないと、拡張の見積もりが甘くなったり、規制コンプライアンスが変わったり、取得やプライベートネットワーク接続の設定が難しくなったりする場合があります。
  - サービス要件、レイテンシー、規制、およびディザスタリカバリ (DR) 要件に基づいて、関連する AWS アカウント とリージョンを選択します。
  - リージョン別 VPC デプロイのニーズを明確にします。
  - VPC のサイズを明確にします。
    - マルチ VPC 接続をデプロイするかどうかを判断します。
      - Transit Gateway とは?
      - 単一リージョンの複数 VPC 接続
    - 規制要件のためネットワークの分離が必要かどうかを判断します。
    - VPC を可能な限り大きくします。VPC に割り当てられた最初の VPC CIDR ブロックは変更または削除することはできませんが、重複していない CIDR ブロックを VPC に追加することはできます。ただし、この場合、アドレス範囲が断片化される可能性があります。
    - VPC を可能な限り大きくします。VPC に割り当てられた最初の VPC CIDR ブロックは変更または削除することはできませんが、重複していない CIDR ブロックを VPC に追加することはできます。ただし、この場合、アドレス範囲が断片化される可能性があります。

リソース

関連するドキュメント:

- APN パートナー: ネットワークの計画を支援できるパートナー
- ネットワークインフラストラクチャ向け AWS Marketplace
- Amazon Virtual Private Cloud の接続オプションホワイトペーパー
- Multiple data center HA network connectivity

- 単一リージョンの複数 VPC 接続
- 「Amazon VPC とは?」

関連動画:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303) (高度な VPC 設計と Amazon VPC の新機能)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1) (多くの VPC に対応した AWS Transit Gateway のリファレンスアーキテクチャ)

REL02-BP04 多対多メッシュよりもハブアンドスポークトポロジを優先する

3 つ以上のネットワークアドレス空間 (VPC およびオンプレミスネットワークなど) が VPC ピア接続、AWS Direct Connect、または VPN 経由で接続されている場合は、AWS Transit Gateway が提供するようなハブアンドスポークモデルを使用します。

そのようなネットワークが 2 つしかない場合は、それらを互いに接続するだけで済みますが、ネットワークの数が増えるにつれて、そのようなメッシュ接続の複雑さは維持できないものになります。AWS Transit Gateway は、維持しやすいハブアンドスポークモデルを提供し、複数のネットワーク間でトラフィックをルーティングできるようにします。

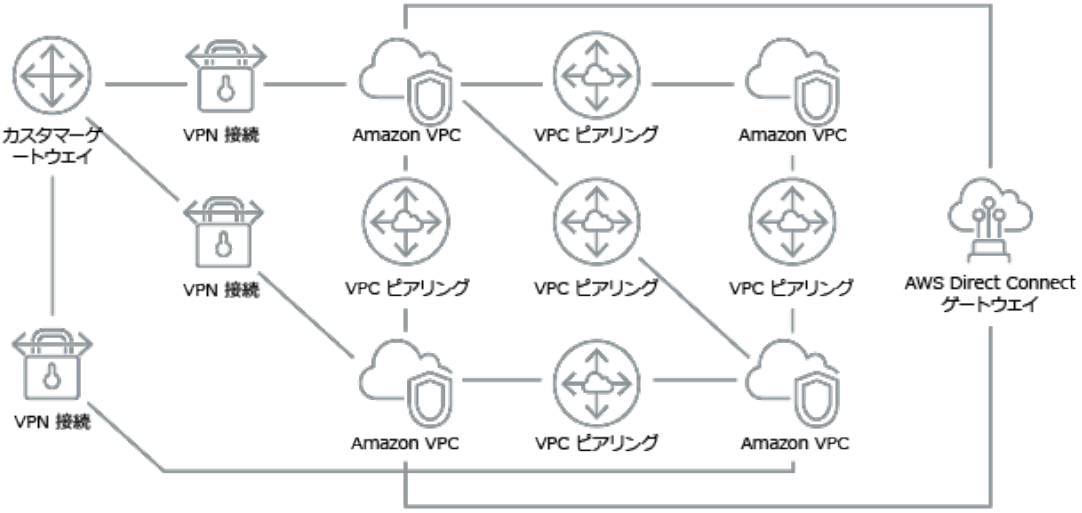


図 1: AWS Transit Gateway なしの場合: VPN 接続を使用して、各 Amazon VPC を相互にピア接続し、オンサイトの各ロケーションをピア接続する必要があります。これは、拡張するにつれ複雑さが増す可能性があります。

図 2: AWS Transit Gateway ありの場合: 各 Amazon VPC または VPN を AWS Transit Gateway に接続するだけで、各 VPC または VPN との間でトラフィックをルーティングできます。

一般的なアンチパターン:

- VPC ピア接続を使用して 3 つ以上の VPC を接続する。
- 各 VPC に対して複数の BGP セッションを確立し、複数の AWS リージョン における仮想プライベートクラウド (VPC) にまたがる接続を確立する。

このベストプラクティスを確立するメリット: ネットワークの数が増えるにつれて、このようなメッシュ接続の複雑さは受容できないものになります。AWS Transit Gateway は、維持しやすいハブアンドスポークモデルを提供し、複数のネットワーク間でトラフィックをルーティングできるようにします。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 多対多メッシュよりもハブアンドスポークトポロジを優先します。3 つ以上のネットワークアドレス空間 (VPC およびオンプレミスネットワークなど) が VPC ピア接続、AWS Direct Connect、または VPN 経由で接続されている場合は、AWS Transit Gateway が提供するようなハブアンドスポークモデルを使用します。
  - そのような 2 つのネットワークのみについては、それらを互いに接続するだけで済みますが、ネットワークの数が増えるにつれて、そのようなメッシュ接続の複雑さは受容できないものにな

ります。AWS Transit Gateway は、維持しやすいハブアンドスポークモデルを提供し、複数のネットワーク間でトラフィックをルーティングできるようにします。

- Transit Gateway とは?

リソース

関連するドキュメント:

- APN パートナー: ネットワークの計画を支援できるパートナー
- ネットワークインフラストラクチャ向け AWS Marketplace
- Multiple data center HA network connectivity
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- 「Amazon VPC とは?」
- Transit Gateway とは?

関連動画:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303) (高度な VPC 設計と Amazon VPC の新機能)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1) (多くの VPC に対応した AWS Transit Gateway のリファレンスアーキテクチャ)

REL02-BP05 接続されているすべてのプライベートアドレス空間において、重複しないプライベート IP アドレス範囲を指定する

各 VPC の IP アドレス範囲が、ピア接続時または VPN 経由での接続時に重複しないようにする必要があります。同様に、VPC とオンプレミス環境の間、または使用する他のクラウドプロバイダーとの IP アドレスの競合を回避する必要があります。また、必要に応じてプライベート IP アドレス範囲を割り当てる方法を用意する必要もあります。

これには、IP アドレス管理 (IPAM) システムが役立ちます。複数の IPAM は AWS Marketplace から入手できます。

一般的なアンチパターン:

- オンプレミスまたは社内ネットワークにあるのと同じ IP 範囲を VPC で使用する。

- ワークロードのデプロイに使用されている VPC の IP 範囲を追跡しない。

このベストプラクティスを確立するメリット: ネットワークを積極的に計画することで、相互接続されたネットワークで同じ IP アドレスが複数出現しないようにできます。これにより、異なるアプリケーションを使用しているワークロードの一部でルーティングの問題が発生するのを防ぐことができます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- CIDR の使用をモニタリングし、管理します。AWS での予想される使用量を評価して、CIDR の範囲を既存の VPC に追加し、計画的な使用量の増加を可能にする VPC を作成します。
  - 現在の CIDR 消費量 (VPC、サブネットなど) を把握します。
    - サービス API オペレーションを使用して、現在の CIDR 消費量を収集します。
  - 現在のサブネットの使用量を把握します。
    - サービス API オペレーションを使用して、各リージョンの VPC ごとにサブネットを収集します。
      - DescribeSubnets
    - 現在の使用量を記録します。
    - 重複している IP 範囲を作成したかどうか確認します。
    - 予備容量を計算します。
    - 重複している IP 範囲を特定します。重複する範囲に接続する必要がある場合は、新しいアドレス範囲に移行するか、AWS Marketplace の Network and Port Translation (NAT) アプライアンスを使用できます。

リソース

関連するドキュメント:

- APN パートナー: ネットワークの計画を支援できるパートナー
- ネットワークインフラストラクチャ向け AWS Marketplace
- Amazon Virtual Private Cloud の接続オプションホワイトペーパー
- Multiple data center HA network connectivity
- 「Amazon VPC とは?」

- IPAM とは

関連動画:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303) (高度な VPC 設計と Amazon VPC の新機能)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1) (多くの VPC に対応した AWS Transit Gateway のリファレンスアーキテクチャ)

ワークロードアーキテクチャ

質問

- REL 3 ワークロードサービスアーキテクチャをどのように設計すればよいですか?
- REL 4 障害を防ぐために、分散システムでの操作をどのように設計すればよいですか?
- REL 5 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?

REL 3 ワークロードサービスアーキテクチャをどのように設計すればよいですか?

サービス指向アーキテクチャ (SOA) またはマイクロサービスアーキテクチャを使用して、拡張性と信頼性の高いワークロードを構築します。サービス指向アーキテクチャ (SOA) は、サービスインターフェイスを介してソフトウェアコンポーネントを再利用できるようにする方法です。マイクロサービスアーキテクチャは、その一歩先を行き、コンポーネントをさらに小さくシンプルにしています。

ベストプラクティス

- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL03-BP02 特定のビジネスドメインと機能に重点を置いたサービスを構築する
- REL03-BP03 API ごとにサービス契約を提供する

REL03-BP01 ワークロードをセグメント化する方法を選択する

アプリケーションの回復力要件を決定する際に、ワークロードのセグメント化は重要です。モノリシックアーキテクチャはできるだけ避ける必要があります。代わりに、どのアプリケーションコンポーネントをマイクロサービスに分けられるかを注意深く考慮します。アプリケーションの要件によっては、最終的にサービス指向アーキテクチャ (SOA) とマイクロサービスの組み合わせになるこ

ともあります。ステートレス化が可能なワークロードは、マイクロサービスとしてデプロイすることができます。

期待される成果: ワークロードは、サポート可能で、スケーラブルであり、可能な限り疎結合であるべきです。

ワークロードのセグメント化について選択を行う場合は、複雑さに対してどれだけメリットがあるかを考えてください。新製品のローンチ時に適しているものは、初期からスケーリングのことを考えて構築したワークロードとは異なります。既存のモノリスをリファクタリングする場合、アプリケーションがステートレスへの分解をどの程度サポートできるかを検討する必要があります。サービスを小さく分割することで、小規模で明確なチームが開発、管理することができます。しかし、サービスの規模が小さくなると、レイテンシーの増加、デバッグの複雑化、運用負荷の増大など、複雑な問題が発生する可能性があります。

一般的なアンチパターン:

- 「マイクロサービス Death Star」とは、アトミックコンポーネントが強く依存しあっているために、1 つの失敗がより大きな失敗となり、コンポーネントがモノリスのように柔軟性が低く、壊れやすくなっている状態のことです。

このプラクティスを活用するメリット:

- より特化したセグメントは、高い俊敏性、組織の柔軟性、およびスケーラビリティにつながる。
- サービス中断の影響が小さくなる。
- アプリケーションコンポーネントには異なる可用性要件があり、より特化したセグメント化によってサポートされることがある。
- ワークロードをサポートするチームの責任が明確に定義される。

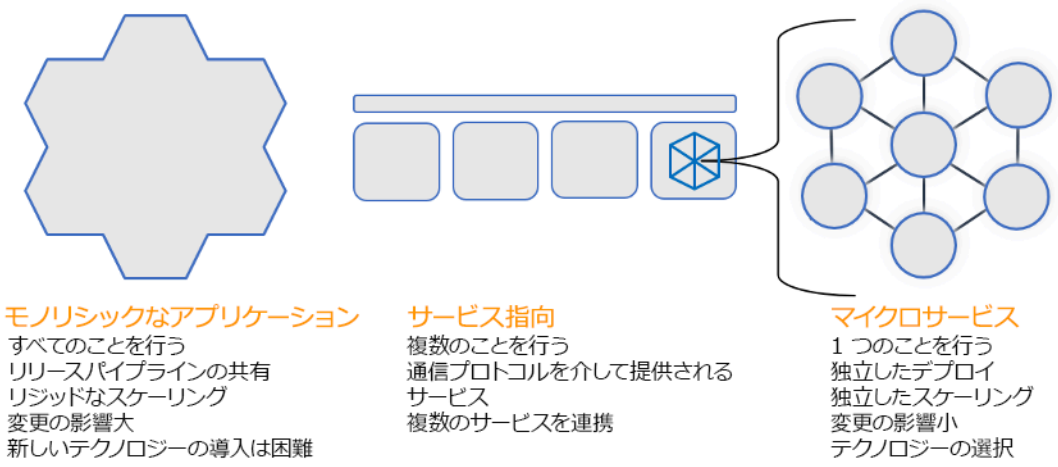
このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードをセグメント化する方法に基づいてアーキテクチャタイプを選択します。SOA またはマイクロサービスアーキテクチャ (場合によってはモノリシックアーキテクチャ) を選択します。モノリスアーキテクチャから開始する場合でも、それがモジュラー型で、ユーザーの導入に合わせて製品がスケールされるにつれて最終的に SOA またはマイクロサービスに進化できることを確認する必要があります。SOA とマイクロサービスは、それぞれより小さなセグメントを提供し、最新のス

ケーラブルで信頼性の高いアーキテクチャとして好まれていますが、特にマイクロサービスアーキテクチャを展開する際には、トレードオフを考慮しなければなりません。

主なトレードオフとしては、分散コンピューティングアーキテクチャを採用することになり、ユーザーのレイテンシー要件を達成するのが難しくなることと、ユーザーインタラクションのデバッグとトレースにさらなる複雑さが生じることが挙げられます。AWS X-Ray を使ってこの問題の解決に役立てることができます。また、管理するアプリケーションの数が増え、複数の独立したコンポーネントを配置する必要があるため、運用が複雑になることも考慮しなければなりません。



モノリシック、サービス指向、マイクロサービスアーキテクチャ

実装手順

- アプリケーションのリファクタリングやビルドに適したアーキテクチャを決定します。SOA とマイクロサービスは、それぞれより小さなセグメンテーションを提供し、最新のスケーラブルで信頼性の高いアーキテクチャとして好まれます。SOA は、マイクロサービスの複雑さを回避しながら、より小さなセグメント化を達成するための優れた折衷案となり得ます。詳細については、マイクロサービスのトレードオフ.
- ワークロードが適していて、組織がサポートできる場合は、最高の俊敏性と信頼性を実現するために、マイクロサービスアーキテクチャを使用すべきです。詳細については、AWS でのマイクロサービスの実装
- モノリスを Strangler Fig パターン に従って、より小さいコンポーネントにリファクタリングします。これには、特定のアプリケーションコンポーネントを新しいアプリケーションとサービスに徐々に置き換えることが含まれます。AWS Migration Hub Refactor Spaces は、増分リファクタリングの開始点として機能します。詳細については、「オンプレミスのレガシーワークロードをシームレスに移行する」を参照してください。

- マイクロサービスを実装する場合、これらの分散したサービスが互いに通信できるようにするためのサービス検出メカニズムが必要になる場合があります。AWS App Mesh をサービス指向アーキテクチャとともに使用することで、高い信頼性をもってサービスを検出し、サービスにアクセスできます。 AWS Cloud Map は、動的 DNS ベースのサービス検出にも使用できます。
- モノリスから SOA へ移行する場合、Amazon MQ は、レガシーアプリケーションをクラウドで再設計する際に、サービスバスとしてギャップを埋めるのに役立ちます。
- 単一の共有されたデータベースがある既存のモノリスには、データを再編成して小さなセグメントにする方法を選択します。これは、ビジネスユニット、アクセスパターン、またはデータ構造によって行うことができます。リファクタリングプロセスのこの時点では、リレーショナルまたは非リレーショナル (NoSQL) タイプのデータベースを選択して進めていく必要があります。詳細については、「 SQL から NoSQL へ」を参照してください。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- REL03-BP02 特定のビジネスドメインと機能に重点を置いたサービスを構築する

関連するドキュメント:

- Amazon API Gateway: OpenAPI を使用した REST API の設定
- サービス指向アーキテクチャとは
- 境界付けられたコンテキスト (ドメイン駆動設計の中心的なパターン)
- AWS でのマイクロサービスの実装
- マイクロサービスのトレードオフ
- Microservices - a definition of this new architectural term
- AWS でのマイクロサービス
- AWS App Mesh とは

関連する例:

- Iterative App Modernization Workshop

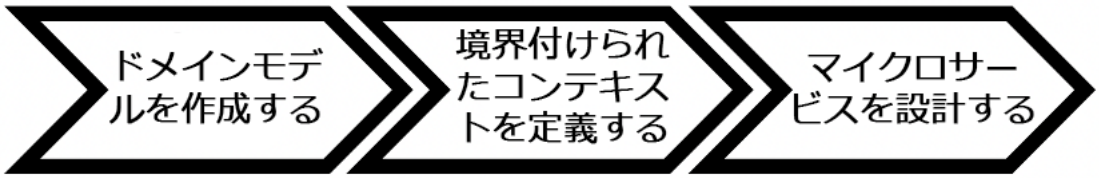
関連動画:

- Delivering Excellence with Microservices on AWS

REL03-BP02 特定のビジネスドメインと機能に重点を置いたサービスを構築する

サービス指向アーキテクチャ (SOA) は、ビジネスニーズに合わせて明確に定義された機能を備えたサービスを構築します。マイクロサービスはドメインモデルと境界付けられたコンテキストを使用してこれをさらに制限し、各サービスが 1 つのことだけを実行するようにします。特定の機能に焦点を当てることで、さまざまなサービスの信頼性要件を差別化し、より具体的に的を絞って投資することができます。簡潔なビジネス上の問題と各サービスに関連付けられた小さなチームにより、組織のスケーリングも容易になります。

マイクロサービスアーキテクチャを設計する際は、ドメイン駆動設計 (DDD) を使用して、エンティティでビジネス上の問題をモデル化すると便利です。例えば、Amazon.com ウェブサイトの場合、エンティティには、パッケージ、配送、スケジュール、料金、割引、通貨などがあります。その後、モデルは 境界コンテキストを使用してさらに小さなモデルに分割され、そこで類似した機能と属性を共有するエンティティがグループ化されます。したがって、Amazon.com の例で言うと、パッケージ、配送、スケジュールは発送コンテキストの一部となり、料金、割引、通貨は料金コンテキストの一部となります。モデルをコンテキストに分割したら、マイクロサービスを境界で区切る方法のテンプレートが現れます。



このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ビジネスドメインとそれぞれの機能に基づいてワークロードを設計します。特定の機能に焦点を当てることで、さまざまなサービスの信頼性要件を差別化し、より具体的に的を絞って投資することができます。簡潔なビジネス上の問題と各サービスに関連付けられた小さなチームにより、組織のスケーリングも容易になります。
  - ドメイン分析を実行して、ワークロードのドメイン駆動型設計 (DDD) をマッピングします。次に、ワークロードのニーズを満たすアーキテクチャタイプを選択できます。
    - モノリスをマイクロサービスに分割する方法

    - レガシーシステムに囲まれているときの DDD の使用開始
    - Eric Evans “Domain-Driven Design: Tackling Complexity in the Heart of Software”
    - AWS でのマイクロサービスの実装
- サービスをできるだけ小さなコンポーネントに分解します。マイクロサービスアーキテクチャを使用すると、最小限の機能でワークロードをコンポーネントに分割し、全社的なスケーリングと俊敏性を実現できます。
    - ワークロードの API とその設計目標、制約、使用に関するその他の検討事項を定義します。
        - API を定義します。
            - 拡張と追加パラメータが実現可能になるように API を定義します。
        - 設計時の可用性を定義します。
            - さまざまな機能に関して API の設計目標を複数立てることができます。
        - 制約を決める
            - テスティングを利用して、ワークロードの能力の上限を定義します。

リソース

関連するドキュメント:

- Amazon API Gateway: OpenAPI を使用した REST API の設定
- 境界付けられたコンテキスト (ドメイン駆動設計の中心的なパターン)
- Eric Evans “Domain-Driven Design: Tackling Complexity in the Heart of Software”
- レガシーシステムに囲まれているときの DDD の使用開始
- モノリスをマイクロサービスに分割する方法
- AWS でのマイクロサービスの実装
- マイクロサービスのトレードオフ
- Microservices - a definition of this new architectural term
- AWS でのマイクロサービス

REL03-BP03 API ごとにサービス契約を提供する

サービス契約は、サービス統合に関するチーム間で文書化した合意で、機械で読み取ることができる API 定義、レート制限、パフォーマンスの期待値が含まれます。バージョニング戦略により、クライアントは既存の API を引き続き使用し、準備ができたらアプリケーションを新しい API に移行でき

ます。契約に違反しない限り、デプロイはいつでも行うことができます。サービスプロバイダーチームは、選択した技術スタックを使用して、API 契約の条件を満たすことができます。同様に、サービスコンシューマーは独自のテクノロジーを使用できます。

マイクロサービスはサービス指向アーキテクチャ (SOA) という概念の進化形であり、マイクロサービスでは最小限の機能を備えたサービスを構築します。各サービスでは、サービスを使用するための API、設計目標、制限、その他の考慮事項が公開されています。これにより、アプリケーションの呼び出しを備えた 契約 が確立されます。これには次のような 3 つのメリットがあります。

- このサービスでは、対応すべきビジネスの課題が簡潔で、それを共有するチームの規模は小さくなります。これにより組織の拡大が可能となります。
- API の要件やその他の契約条件を満たしている限り、チームはいつでもデプロイできます。
- API の要件やその他の契約条件を満たしている限り、チームはあらゆる技術スタックを使用することができます。

Amazon API Gateway は、デベロッパーがあらゆる規模の API の作成、公開、保守、モニタリング、保護を簡単に行えるようにするためのフルマネージド型サービスです。Amazon API Gateway は、トラフィック管理、認可とアクセスコントロール、モニタリング、API バージョン管理など、最大数十万個の同時 API 呼び出しの受け入れと処理に伴うすべてのタスクを取り扱います。以前は Swagger 仕様として知られていた OpenAPI 仕様 (OAS) を使用して、API 契約を定義し、API Gateway にインポートできます。API Gateway を使用すると、API をバージョン管理してデプロイできます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- API ごとにサービス契約を提供するサービス契約は、サービス統合に関するチーム間の文書化された合意であり、機械で読み取ることができる API 定義、レート制限、パフォーマンスの期待値が含まれます。
  - Amazon API Gateway: OpenAPI を使用した REST API の設定
    - バージョニング戦略により、クライアントは既存の API を引き続き使用し、準備ができたらアプリケーションを新しい API に移行できます。
    - Amazon API Gateway は、開発者が規模を問わず簡単に API を作成できる完全マネージド型サービスです。以前は Swagger 仕様として知られていた OpenAPI 仕様 (OAS) を使用して、API 契約を定義し、API Gateway にインポートできます。API Gateway を使用すると、API をバージョン管理してデプロイできます。

リソース

関連するドキュメント:

- Amazon API Gateway: OpenAPI を使用した REST API の設定
- 境界付けられたコンテキスト (ドメイン駆動設計の中心的なパターン)
- AWS でのマイクロサービスの実装
- マイクロサービスのトレードオフ
- Microservices - a definition of this new architectural term
- AWS でのマイクロサービス

REL 4 障害を防ぐために、分散システムでの操作をどのように設計すればよいですか?

分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。これらのベストプラクティスは、故障を防ぎ、平均故障間隔 (MTBF) を改善します。

ベストプラクティス

- REL04-BP01 必要な分散システムの種類を特定する
- REL04-BP02 疎結合の依存関係を実装する
- REL04-BP03 継続動作を行う
- REL04-BP04 すべての応答に冪等性を持たせる

REL04-BP01 必要な分散システムの種類を特定する

ハードなリアルタイム分散システムでは、応答を同期的かつ迅速に行えるようにする必要がありますが、ソフトなリアルタイムシステムでは、応答に数分以上の余裕をもった時間枠があります。オフラインシステムは、バッチ処理または非同期処理を通じて応答を処理します。ハードなリアルタイム分散システムは、最も厳格な信頼性要件を持っています。

最も難しい 分散型システムの課題 は、リクエスト / 応答サービスとも呼ばれるハードなリアルタイム分散システムです。それを困難にしているのは、リクエストが前触れもなく送信され、直ちに応答

しなくてはならないという点です (お客様がレスポンスを待っているなど)。この例には、フロントエンドウェブサーバー、オーダーパイプライン、クレジットカードトランザクション、すべての AWS API、テレフォニーなどがあります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 必要な分散システムの種類を特定します。分散型システムの課題としては、レイテンシー、スケーリング、ネットワーキング API の理解、データのマーシャリングとアンマーシャリング、および Paxos などのアルゴリズムの複雑性に関するものがありました。システムが大きくなり、分散化が進むにつれて、理論上のエッジケースが日常的に発生するようになります。
  - The Amazon Builders' Library: 分散システムの課題
    - ハードなリアルタイム分散システムでは、応答を同期的かつ迅速に与える必要があります。
    - ソフトなリアルタイムシステムでは、応答に数分以上の余裕をもった時間枠があります。
    - オフラインシステムは、バッチ処理または非同期処理を通じて応答を処理します。
    - ハードなリアルタイム分散システムは、最も厳格な信頼性要件を持っています。

リソース

関連するドキュメント:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: 分散システムの課題
- The Amazon Builders' Library: 信頼性、動作の継続、一杯のコーヒー
- 「Amazon EventBridge とは?」
- 「Amazon Simple Queue Service とは何ですか?」

関連動画:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (疎結合、継続動作、静的安定性を含む)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP02 疎結合の依存関係を実装する

キューイングシステム、ストリーミングシステム、ワークフロー、ロードバランサーなどの依存関係は、緩やかに結合しています。疎結合は、コンポーネントの動作をそれに依存する他のコンポーネントから分離するのに役立ち、弾力性と俊敏性を高めます。

1 つのコンポーネントを変更すると、それに依存する他のコンポーネントも強制的に変更される場合、それらは 緊密に 結合されています。 疎 結合はこの依存関係を壊すため、依存コンポーネントが知る必要があるのは、バージョン管理されて公開されたインターフェイスのみです。依存関係があるコンポーネント間に疎結合を実装すると、あるコンポーネントの障害が別のコンポーネントに影響を及ぼさないようにすることができます。

疎結合により、そのコンポーネントに依存する他のコンポーネントのリスクを最小限に抑えながら、コンポーネントにコードまたは機能を自由に追加できます。また、スケールアウトしたり、依存関係の基盤となる実装を変更したりできるため、スケーラビリティが向上します。

疎結合によって弾力性をさらに向上させるには、可能な場合はコンポーネント間のやりとりを非同期にします。このモデルは、即時応答を必要とせず、リクエストが登録されていることの確認で十分な状況では、どのような対話にも最適です。イベントを生成するコンポーネントと、イベントを消費するコンポーネントがあります。2 つのコンポーネントは、直接的なポイントツーポイントのやりとりではなく、通常、SQS キューのような中間的な耐久性の高いストレージレイヤーや Amazon Kinesis のようなストリーミングデータプラットフォーム、または AWS Step Functions を介して統合されます。

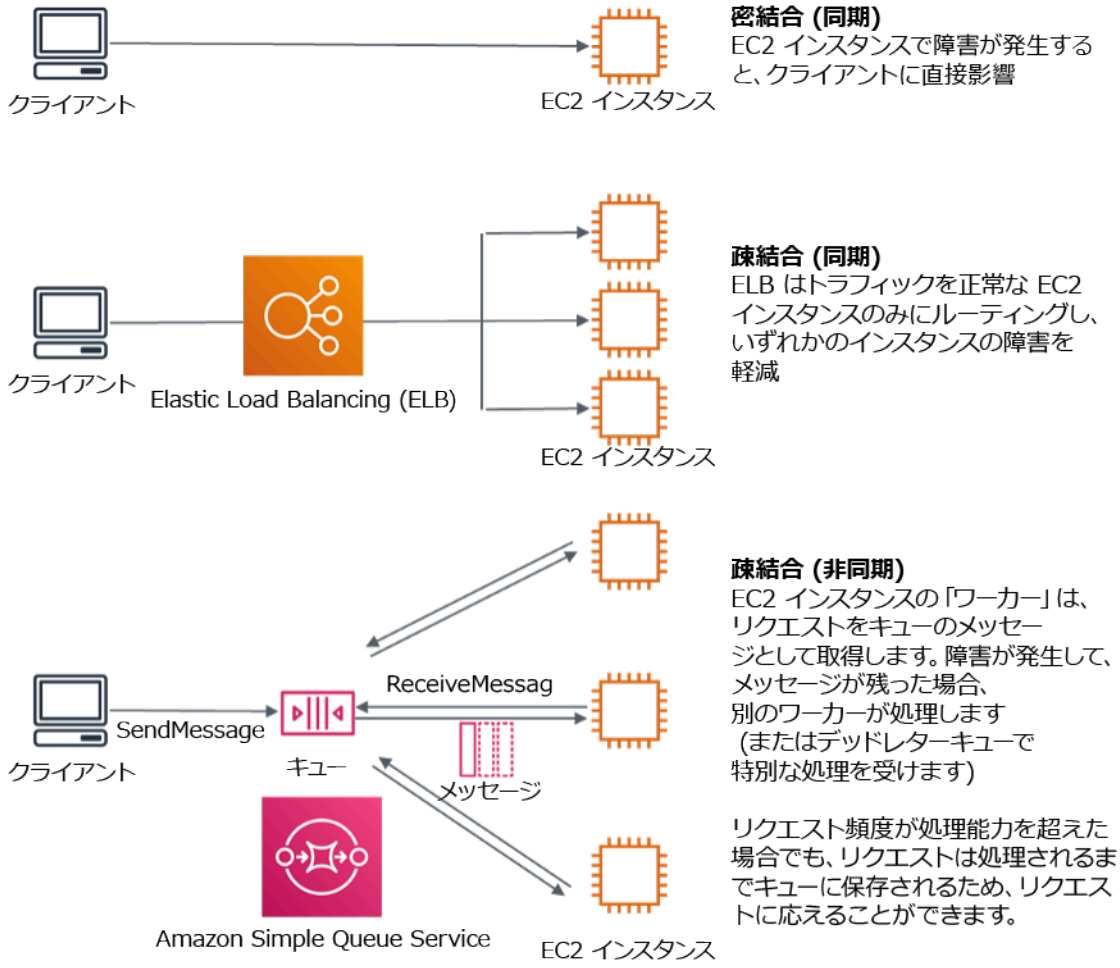


図 4: キューイングシステムやロードバランサーなどの依存関係が疎結合されています。

Amazon SQS キューと Elastic Load Balancing は、疎結合の中間レイヤーを追加する方法のうちの 2 つにすぎません。イベント駆動型アーキテクチャは、Amazon EventBridge を使用して AWS クラウド に構築することもできます。これにより、依存しているサービス (イベントコンシューマー) からクライアント (イベントプロデューサー) を抽出することができます。Amazon Simple Notification Service (Amazon SNS) は、高スループット、プッシュベースの多対多メッセージングが必要な場合に効果的なソリューションです。Amazon SNS トピックを使用すると、パブリッシャーシステムは、メッセージを多数のサブスクライバーエンドポイントにファンアウトして、並列処理できます。

キューにはいくつかの利点がありますが、ほとんどのハードリアルタイムシステムでは、しきい値の時間 (多くの場合、数秒) よりも長時間かかっているリクエストは古くなっていると見なされ (クライアントが停止し、応答を待機しなくなる)、処理されません。このように、古くなったリクエストの代わりに、より新しい (そしておそらくまだ有効な) リクエストを処理することができます。

一般的なアンチパターン:

- ワークロードの一部としてシングルトンをデプロイする。
- リクエストのフェイルオーバーや非同期処理を行うことはできない状態で、ワークロード層間で直接 API を呼び出す。

このベストプラクティスを活用するメリット: 疎結合は、コンポーネントの動作をそれに依存する他のコンポーネントから分離するのに役立ち、弾力性と俊敏性を高めます。1 つのコンポーネントの障害は他のコンポーネントから分離されます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 疎結合の依存関係を実装します。キューイングシステム、ストリーミングシステム、ワークフロー、ロードバランサーなどの依存関係は、緩やかに結合しています。疎結合は、コンポーネントの動作をそれに依存する他のコンポーネントから分離するのに役立ち、弾力性と俊敏性を高めます。
  - AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
  - 「Amazon EventBridge とは?」
  - 「Amazon Simple Queue Service とは何ですか?」
    - Amazon EventBridge では、疎結合で分散されたイベント駆動型アーキテクチャを構築できます
      - AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
    - 1 つのコンポーネントを変更すると、それに依存する他のコンポーネントも強制的に変更される場合、それらは密結合されています。疎結合はこの依存関係を壊すため、依存関係コンポーネントが知る必要があるのは、バージョン付きで公開されたインターフェイスのみです。
    - 可能な限り、コンポーネントのインタラクションを非同期にします。このモデルは、即時応答を必要とせず、要求が登録されていることの確認が十分である状況では、どのような対話にも最適です。
      - AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)

リソース

関連するドキュメント:

- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- Amazon EC2: 冪等性の確保
- The Amazon Builders' Library: 分散システムの課題
- The Amazon Builders' Library: 信頼性、動作の継続、一杯のコーヒー
- 「Amazon EventBridge とは?」
- 「Amazon Simple Queue Service とは何ですか?」

関連動画:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (疎結合、継続動作、静的安定性を含む)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda (API304)

REL04-BP03 継続動作を行う

負荷が急激に大きく変化すると、システム障害が発生することがあります。例えば、ワークロードで何千台ものサーバーのヘルスをモニタリングするヘルスチェックを実行する場合、毎回同じサイズのペイロード (現在の状態の完全なスナップショット) を送信しています。障害が発生しているサーバーがなくても、またはそのすべてに障害が発生していても、ヘルスチェックシステムは、大規模で急激な変更なしに常に作業を行っています。

たとえば、ヘルスチェックシステムが 100,000 台のサーバーをモニタリングしている場合、通常のサーバー障害率が軽いときは、その負荷はわずかです。ただし、重大なイベントによってこれらのサーバーの半分が異常な状態になると、ヘルスチェックシステムは、通知システムを更新し、クライアントに状態を通知しようとして過負荷になるでしょう。したがって、ヘルスチェックシステムは現在の状態の完全なスナップショットを毎回送信する必要があります。サーバー 100,000 台 のヘルス状態 (それぞれ 1 ビットで表される) のペイロードは、12.5 KB にすぎません。サーバーに障害が発生していないか、またはすべてに発生しているかにかかわらず、ヘルスチェックシステムは定期的に作業を行っているため、大規模の急激な変化はシステムの安定性を脅かすものではありません。これは実際に Amazon Route 53 がエンドポイントのヘルスチェック (IP アドレスなど) によってエンドユーザーがどのようにルーティングされているかを調べる際の方法です。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 負荷が急激に変化してシステム障害が発生しないように、継続動作を行います。
- 疎結合の依存関係を実装します。キューイングシステム、ストリーミングシステム、ワークフロー、ロードバランサーなどの依存関係は、緩やかに結合しています。疎結合は、コンポーネントの動作をそれに依存する他のコンポーネントから分離するのに役立ち、弾力性と俊敏性を高めます。
  - The Amazon Builders' Library: 信頼性、動作の継続、一杯のコーヒー
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (ループを閉じ、発想を開く: 大小さまざまなシステムをコントロールする方法) ARC337 (継続動作を含む)
    - 100,000 台のサーバーをモニタリングする健康診断システムの例の場合、成功または失敗の数に関係なく、ペイロードサイズが一定になるように、ワークロードを設計します。

リソース

関連するドキュメント:

- Amazon EC2: べき等性の確保
- The Amazon Builders' Library: 分散システムの課題
- The Amazon Builders' Library: 信頼性、動作の継続、一杯のコーヒー

関連動画:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205) (イベント駆動型アーキテクチャーと Amazon EventBridge 入門)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (ループを閉じ、発想を開く: 大小さまざまなシステムをコントロールする方法) ARC337 (継続動作を含む)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (ループを閉じ、発想を開く: 大小さまざまなシステムをコントロールする方法) ARC337 (疎結合、継続動作、静的安定性を含む)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308) (イベント駆動型アーキテクチャへの移行)

REL04-BP04 すべての応答に冪等性を持たせる

べき等のサービスは、各リクエストが 1 回だけで完了することを約束します。そのため、同一のリクエストを複数回行っても、リクエストを 1 回行ったのと同じ効果しかありません。べき等サービスを使用すると、リクエストが誤って複数回処理されることを恐れる必要がなくなるため、クライアントが再試行を行いやすくなります。このために、クライアントはべき等性トークンを使用して API リクエストを発行できます。リクエストが繰り返される場合は常に同じトークンが使われます。べき等サービス API はトークンを使用して、リクエストが最初に完了したときに返された応答と同じ応答を返します。

分散システムでは、アクションを最大で 1 回 (クライアントがリクエストを 1 回だけ行う)、または少なくとも 1 回 (クライアントが成功を確認するまでリクエストを続ける) 実行するのは簡単です。ただし、アクションがべき等であることを保証することは困難です。つまり、 正確に 1 回だけ実行し、同一のリクエストを複数回行っても、リクエストを 1 回行うのと同じ効果を得るようにするということです。API でべき等性トークンを使用すると、サービスは、重複レコードや副作用を生むことなく、変更リクエストを 1 回または複数回受け取ることができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- すべての応答に冪等性を持たせます。べき等のサービスは、各リクエストが 1 回だけで完了することを約束します。そのため、同一のリクエストを複数回行っても、リクエストを 1 回行ったのと同じ効果しかありません。
  - クライアントはべき等性トークンを使用して API リクエストを発行できます。リクエストが繰り返される場合は常に同じトークンが使われます。べき等サービス API はトークンを使用して、リクエストが最初に完了したときに返された応答と同じ応答を返します。
    - Amazon EC2: 冪等性の確保

リソース

関連するドキュメント:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: 分散システムの課題
- The Amazon Builders' Library: 信頼性、動作の継続、一杯のコーヒー

関連動画:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (疎結合、継続動作、静的安定性を含む)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL 5 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?

分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。これらのベストプラクティスに従うことで、ワークロードはストレスや障害に耐え、より迅速に復旧し、そのような障害の影響を軽減できます。その結果、平均復旧時間 (MTTR) が向上します。

ベストプラクティス

- REL05-BP01 該当するハードな依存関係をソフトな依存関係に変換するため、グレイスフルデグラデーションを実装する
- REL05-BP02 リクエストのスロットル
- REL05-BP03 再試行呼び出しを制御および制限する
- REL05-BP04 フェイルファストとキューの制限
- REL05-BP05 クライアントタイムアウトを設定する
- REL05-BP06 可能な限りサービスをステートレスにする
- REL05-BP07 緊急レバーを実装する

REL05-BP01 該当するハードな依存関係をソフトな依存関係に変換するため、グレイスフルデグラデーションを実装する

コンポーネントの依存関係が異常な場合でも、コンポーネント自体は機能しますが、パフォーマンスが低下します。たとえば、依存関係の呼び出しが失敗した場合、事前に定義された静的レスポンスにフェイルオーバーします。

サービス A によって呼び出されたサービス B が、次にサービス C を呼び出すとします。

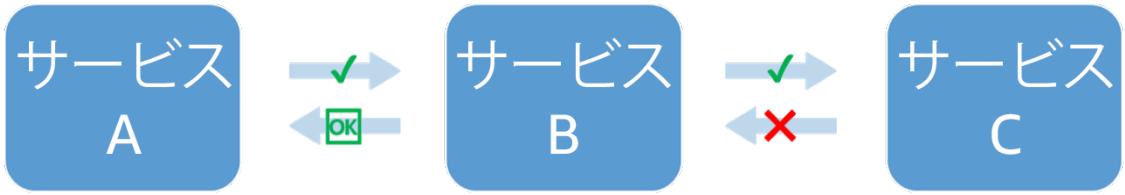


図 5: サービス C は、サービス B から呼び出されると失敗します。サービス B は、低下した応答をサービス A に返します。

サービス B がサービス C を呼び出すと、エラーまたはタイムアウトを受け取ります。サービス B は、サービス C (およびそれに含まれるデータ) からの応答がないため、返せるものを返します。これは、最後にキャッシュされた適切な値であるかもしれません。または、サービス B は、サービス C から受け取るはずだったものを所定の静的応答に置き換える可能性もあります。次に、低下した応答を呼び出し元のサービス A に返すことでしょう。この静的応答がない場合、サービス C で障害が発生すると、サービス B を介してサービス A にカスケードされ、可用性が失われます。

強い依存関係の可用性方程式の乗数的因子により (「強い依存関係を持つ可用性を計算する」を参照)、C の可用性が低下すると、B の有効な可用性に重大な影響を与えます。静的応答を返すことによりサービス B は、C での障害を軽減し、パフォーマンスが低下しますが、サービス C が 100% の可用性を実現しているように見せます (エラー条件下で確実に静的応答を返すと仮定します)。静的応答は単純にエラーを返す代わりに行う手段で、別の手段を使って応答を再計算する試みではないことに注意してください。まったく異なるメカニズムで同じ結果を達成しようとする試みは、フォールバック動作と呼ばれ、回避すべきアンチパターンです。

グレースフルデグラデーションのもう 1 つの例は サーキットブレーカーパターン.障害が一時的な場合は、再試行戦略を用いるのがよいでしょう。障害が一時的ではなく、操作が失敗する可能性が高い場合、サーキットブレーカーパターンは、失敗する可能性が高いリクエストをクライアントが実行できないようにします。リクエストが正常に処理されると、サーキットブレーカーは閉じられ、リクエストは正常に流されます。リモートシステムがエラーを返し始めるか、レイテンシーが高くなると、サーキットブレーカーが開かれ、依存関係が無視されるか、結果的に返される応答は単純に取得されたが包括的ではない応答 (単なる応答キャッシュである場合もあります) に置き換えられます。システムは、依存性が回復したかどうかを判断するために、依存関係を定期的に呼び出そうとします。依存関係が確認できたら、サーキットブレーカーは閉じられます。

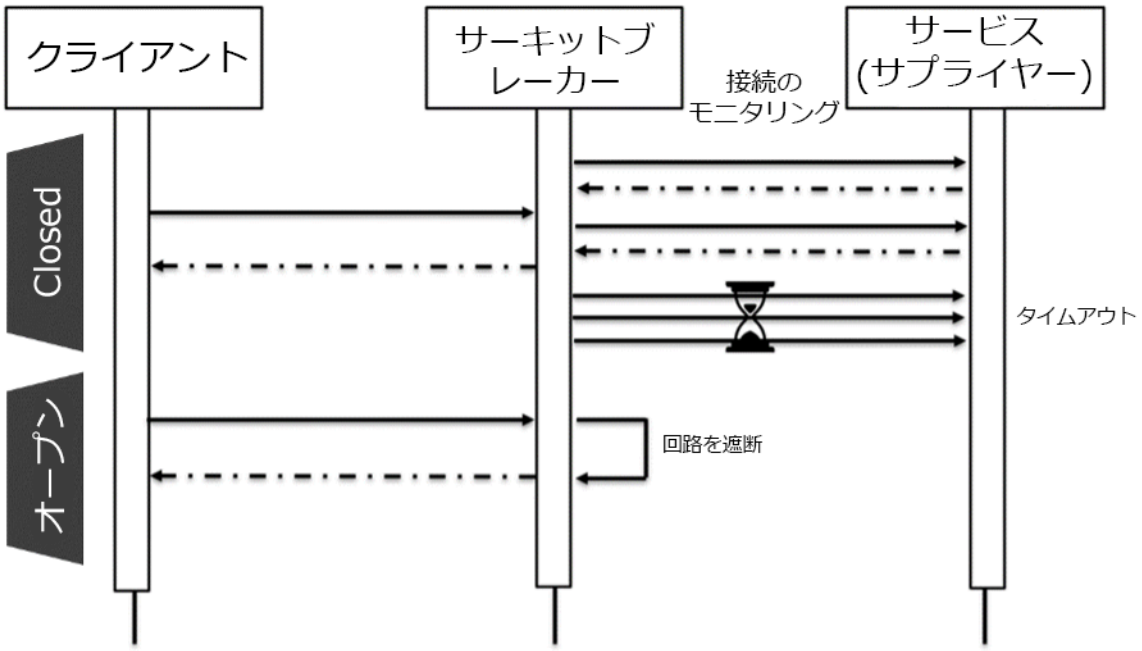


図 6: サーキットブレーカーが閉じた状態と開いた状態を示した図。

図に示されている閉じた状態と開いた状態に加えて、開いた状態で設定可能な期間が経過すると、サーキットブレーカーは半分開いた状態に移行することもあります。この状態では、通常よりはるかに低いレートで定期的にサービスを呼び出そうとします。このプローブは、サービスの状態を確認するために使用します。半開状態で何度か成功すると、サーキットブレーカーは閉じた状態に移行し、通常のリクエストフローが再開されます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 該当するハードな依存関係をソフトな依存関係に変換するため、グレイスフルデグラデーションを実装します。コンポーネントの依存関係が異常な場合でも、コンポーネント自体は機能しますが、パフォーマンスが低下します。たとえば、依存関係の呼び出しが失敗した場合、事前に定義された静的レスポンスにフェイルオーバーします。
  - 静的応答を返すことで、ワークロードは依存関係で発生する障害を軽減します。
    - Well-Architected ラボ: レベル 300: ヘルスチェックを実装し依存関係を管理して、信頼性を向上する
  - 再試行オペレーションが失敗する可能性がある場合を検出し、クライアントがサーキットブレーカーパターンで失敗した呼び出しを行わないようにします。
    - CircuitBreaker

リソース

関連するドキュメント:

- Amazon API Gateway: スループット向上に向けた API リクエストのスロットリング
- CircuitBreaker (「Release It!」よりサーキットブレーカーをまとめたもの)
- AWS でのエラーの再試行とエクスポネンシャルバックオフ
- Michael Nygard「Release It! Design and Deploy Production-Ready Software」
- The Amazon Builders' Library: 分散システムでのフォールバックの回避
- The Amazon Builders' Library: 克服できないキューバックログの回避
- The Amazon Builders' Library: キャッシュの課題と戦略
- The Amazon Builders' Library: タイムアウト、再試行、ジッターによるバックオフ

関連動画:

- 再試行、バックオフ、ジッター: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

関連する例:

- Well-Architected ラボ: レベル 300: ヘルスチェックを実装し依存関係を管理して、信頼性を向上する

REL05-BP02 リクエストのスロットル

リクエストのスロットリングは、予想外の需要の増加に対応するための軽減パターンです。一部のリクエストは受け入れられますが、定義された制限を超えるリクエストは拒否され、スロットルされたことを示すメッセージが返されます。クライアントの期待は、リクエストが戻されて放棄されるか、遅い速度で再試行することです。

サービスは、各ノードまたはセルが処理できる既知のリクエスト容量に合わせて設計する必要があります。この容量は、負荷テストによって設定できます。リクエストの到着率をトラッキングし、到着率が一時的に制限を超えると、リクエストが適切にスロットリングされたことを示す応答があります。これにより、ユーザーは、利用可能なキャパシティを持つ可能性のある別のノードまたはセルに再試行することができます。Amazon API Gateway には、リクエストをスロットリングするためのメソッドが用意されています。Amazon SQS と Amazon Kinesis はリクエストをバッファリング

し、リクエスト率を平準化し、非同期で対応できるリクエストのスロットリングの必要性を軽減することができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- リクエストをスロットリングします。これは、予想外の需要の増加に対応するための軽減パターンです。一部のリクエストは受け入れられますが、定義された制限を超えるリクエストは拒否され、スロットルされたことを示すメッセージが返されます。クライアントの期待は、リクエストが戻されて放棄されるか、遅い速度で再試行することです。
  - Amazon API Gateway を使用します。
    - スループット向上に向けた API リクエストのスロットリング

リソース

関連するドキュメント:

- Amazon API Gateway: スループット向上に向けた API リクエストのスロットリング
- AWS でのエラー再試行とエクスポネンシャルバックオフ
- The Amazon Builders' Library: 分散システムでのフォールバックの回避
- The Amazon Builders' Library: 克服できないキューバックログの回避
- The Amazon Builders' Library: タイムアウト、再試行、ジッターによるバックオフ
- スループット向上に向けた API リクエストのスロットリング

関連動画:

- Retry, backoff, and jitter (再試行、バックオフ、ジッター): AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328) (The Amazon Builders' Library のご紹介)

REL05-BP03 再試行呼び出しを制御および制限する

エクスポネンシャルバックオフを使用して、徐々に長い間隔で再試行します。これらの再試行間隔をランダム化するジッターを導入し、再試行の最大数を制限します。

分散ソフトウェアシステムの一般的なコンポーネントには、サーバー、ロードバランサー、データベース、DNS サーバーが含まれます。操作中に障害が発生すると、これらのコンポーネントのいず

れかにエラーが発生し始める可能性があります。エラーを処理するデフォルトの手法は、クライアント側で再試行を行うことです。この手法により、アプリケーションの信頼性と可用性が向上します。ただし、再試行が大規模に行われた場合 (またエラーが発生してからすぐにクライアントが失敗した操作を再試行しようとすると)、ネットワークは、新しいリクエストと再試行されたリクエストですぐに飽和状態になり、それぞれがネットワーク帯域幅を奪い合うことになる可能性があります。これにより 再試行の大混乱が生じて、 サービスの可用性が低下します。このパターンは、システムが完全にダウンするまで続くかもしれません。

このようなシナリオを回避するには、一般的なエクスポネンシャルバックオフなどの バックオフアルゴリズム を使用する必要があります。エクスポネンシャルバックオフアルゴリズムは、再試行が行われる速度を徐々に下げて、ネットワークの輻輳を回避します。

多くの SDK およびソフトウェアライブラリ (AWS のものを含む) は、これらのアルゴリズムのバージョンを実装しています。ただし、 バックオフアルゴリズムが存在することは想定しないでください。必ずこれをテストして検証してください。

分散システムでは、すべてのクライアントが同時にバックオフし、再試行呼び出しのクラスターが発生する可能性があるため、単にバックオフするだけでは不十分です。Marc Brooker 氏のブログ記事「エクスポネンシャルバックオフとジッター」は、エクスポネンシャルバックオフで wait() 関数を変更して、再試行呼び出しのクラスターを防ぐ方法を説明しています。解決策は、 ジッター を wait() 関数に追加することです。時間がかかり過ぎる再試行を行わないようにするには、実装ではバックオフを最大値に制限する必要があります。

最後に、 再試行の最大数 または最大経過時間を設定し、これを超えると再試行が失敗するようにすることが重要です。AWS SDK はデフォルトでこれを実装しており、設定を変更することもできます。スタックの下位にあるサービスの場合、再試行の上限を 0 または 1 にするとリスクが緩和されますが、スタックの上位にあるサービスに再試行が委任されるため、効果的な方法です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 再試行呼び出しを制御および制限します。エクスポネンシャルバックオフを使用して、徐々に長い間隔で再試行します。これらの再試行間隔をランダム化するジッターを導入し、再試行の最大数を制限します。
  - AWS でのエラーの再試行とエクスポネンシャルバックオフ
    - Amazon SDK は、デフォルトで再試行とエクスポネンシャルバックオフを実装しています。お客様独自の依存サービスを呼び出す場合は、同類の口ジックを依存関係レイヤーに実装します。タイムアウトの時間と、再試行をいつ停止するのかをユースケースに基づいて決めます。

リソース

関連するドキュメント:

- Amazon API Gateway: スループット向上に向けた API リクエストのスロットリング
- AWS でのエラーの再試行とエクスポネンシャルバックオフ
- The Amazon Builders' Library: 分散システムでのフォールバックの回避
- The Amazon Builders' Library: 克服できないキューバックログの回避
- The Amazon Builders' Library: キャッシュの課題と戦略
- The Amazon Builders' Library: タイムアウト、再試行、ジッターによるバックオフ

関連動画:

- 再試行、バックオフ、ジッター: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL05-BP04 フェイルファストとキューの制限

ワークロードがリクエストに正常に応答できない場合は、すぐに失敗するようにします。これにより、リクエストに関連付けられたリソースを解放でき、リソースが不足した場合にサービスを復旧できます。ワークロードは正常に応答できるが、リクエスト頻度が高すぎる場合は、代わりにキューを使用してリクエストをバッファします。ただし、長いキューは許可しないでください。クライアントがすでに処理を停止している古いリクエストを処理する原因となる可能性があるためです。

このベストプラクティスは、サーバー側、つまりリクエストのレシーバーに当てはまります。

キューはシステムの複数のレベルで作成される可能性があるため、応答が必要な新しいリクエストの前に (もはや応答を必要としない) 古い応答が処理されると、迅速に復旧する能力が著しく阻害される可能性があることに注意してください。キューがどこに存在するかに注意を払ってください。ワークフローや、データベースに記録された作業の中に隠れていることもよくあります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- すぐに失敗し、キューを制限します。ワークロードがリクエストに正常に応答できない場合は、すぐに失敗するようにします。これにより、リクエストに関連付けられたリソースを解放でき、リソースが不足した場合にサービスを復旧できます。ワークロードは正常に応答できるが、リクエス

ト頻度が高すぎる場合は、代わりにキューを使用してリクエストをバッファします。ただし、長いキューは許可しないでください。クライアントがすでに処理を停止している古いリクエストを処理する原因となる可能性があるためです。

- サービスへの負荷が過剰になったときのフェイルファストを実装します。
  - 速やかな失敗
- キューベースのシステムでは、処理が停止してもメッセージが到着し続けると、メッセージの負債が蓄積されて、大きなバックログになり、処理時間が長くなることがあります。作業の完了が遅すぎて、結果が役に立たなくなることがあります。これにより、基本的にキューイングが防御することを意図していた、可用性への悪影響が発生します。
  - The Amazon Builders' Library: 克服できないキューバックログの回避

リソース

関連するドキュメント:

- AWS でのエラーの再試行とエクスポネンシャルバックオフ
- 速やかな失敗
- The Amazon Builders' Library: 分散システムでのフォールバックの回避
- The Amazon Builders' Library: 克服できないキューバックログの回避
- The Amazon Builders' Library: キャッシュの課題と戦略
- The Amazon Builders' Library: タイムアウト、再試行、ジッターによるバックオフ

関連動画:

- 再試行、バックオフ、ジッター: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL05-BP05 クライアントタイムアウトを設定する

タイムアウトを適切に設定し、体系的に検証します。デフォルト値は通常高すぎるため、デフォルト値のままにしないでください。

このベストプラクティスは、クライアント側、つまりリクエストの送信者に当てはまります。

リモート呼び出しに接続タイムアウトとリクエストタイムアウトの両方を設定します。またこの設定は、プロセス全体のすべての呼び出しに一般的に行います。多くのフレームワークには組み込みの

タイムアウト機能がありますが、その多くのデフォルト値は無限または高すぎるため、注意が必要です。値が高すぎると、クライアントがタイムアウトの発生を待機している間もリソースが消費され続けるため、タイムアウトの有用性が低下します。値が小さすぎると、再試行されるリクエストが多くなりすぎるため、バックエンドのトラフィックが増加し、レイテンシーが高くなってしまいます。場合によっては、すべてのリクエストが再試行されることになるため、完全な機能停止につながる恐れもあります。

Amazon がタイムアウト、再試行、およびジッターによるバックオフを使用する方法について詳しくは Amazon Builders' Library の「ジッターを伴うタイムアウト、再試行、およびバックオフ」.

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- リモート呼び出しに接続タイムアウトとリクエストタイムアウトの両方を設定します。またこの設定は、プロセス全体のすべての呼び出しに一般的に行います。多くのフレームワークには組み込みのタイムアウト機能がありますが、その多くのデフォルト値は無限または高すぎるため、注意が必要です。値が高すぎると、クライアントがタイムアウトの発生を待機している間もリソースが消費され続けるため、タイムアウトの有用性が低下します。値が小さすぎると、再試行されるリクエストが多くなりすぎるため、バックエンドのトラフィックが増加し、レイテンシーが高くなってしまいます。場合によっては、すべてのリクエストが再試行されることになるため、完全な機能停止につながる恐れもあります。
  - AWS SDK: 再試行とタイムアウト

リソース

関連するドキュメント:

- AWS SDK: 再試行とタイムアウト
- Amazon API Gateway: スループット向上に向けた API リクエストのスロットリング
- AWS でのエラーの再試行とエクスポネンシャルバックオフ
- The Amazon Builders' Library: タイムアウト、再試行、ジッターによるバックオフ

関連動画:

- 再試行、バックオフ、ジッター: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL05-BP06 可能な限りサービスをステートレスにする

サービスは、ステートを必要としないか、またはステートをオフロードして、異なるクライアントのリクエスト間で、ディスクやメモリのローカルに保存されたデータに依存しないようにする必要があります。これにより、可用性に影響を与えずにサーバーをいつでも交換できます。Amazon ElastiCache または Amazon DynamoDB は、オフロード状態の送信先として適しています。

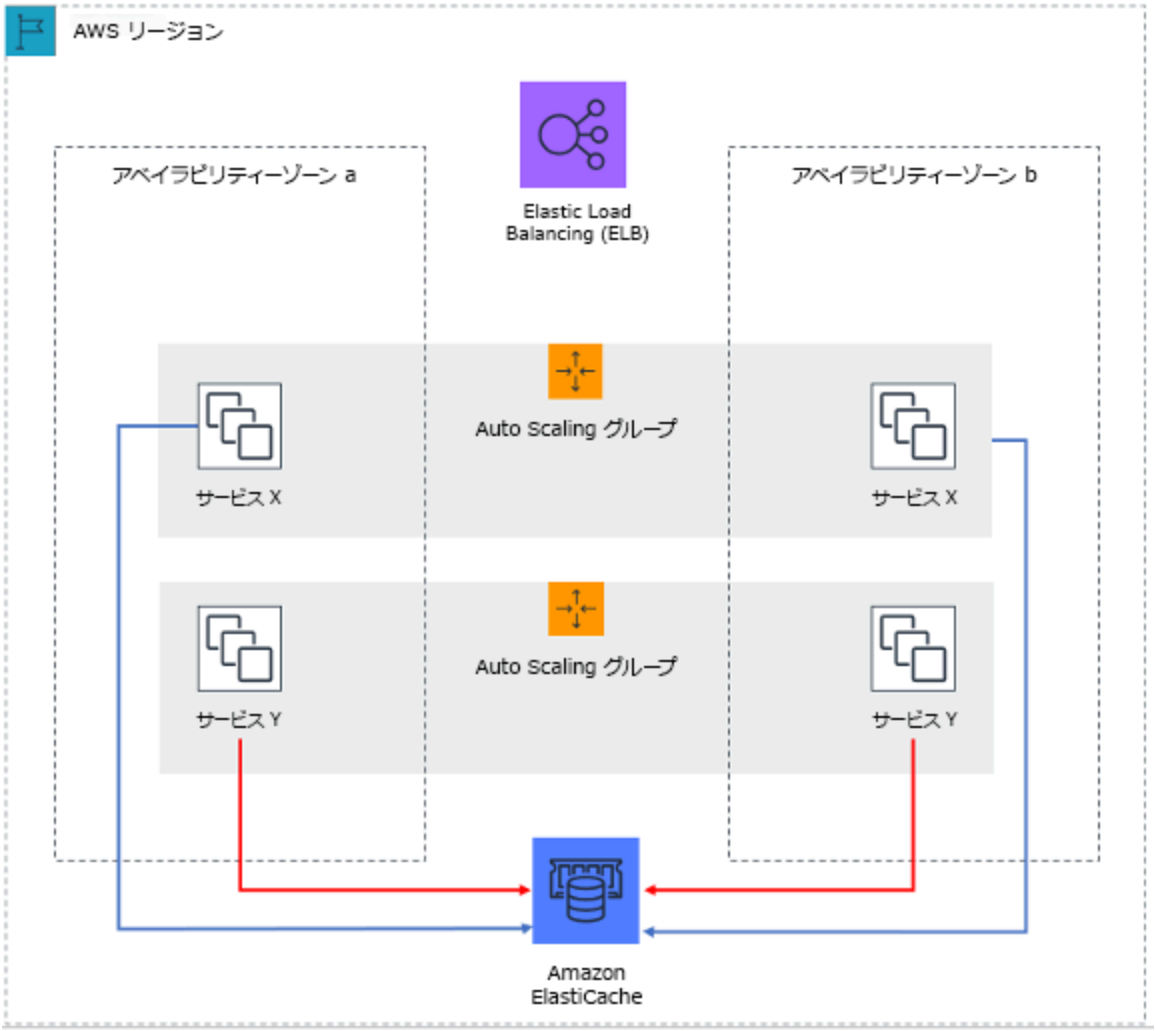


図 7: このステートレスウェブアプリケーションでは、セッション状態は Amazon ElastiCache にオフロードされます。

ユーザーまたはサービスがアプリケーションと対話するとき、セッションを形成する一連のやりとりを頻繁に実行します。セッションは、ユーザーがアプリケーションを使用している間、リクエスト間で持続するユーザー固有のデータです。ステートレスアプリケーションは、以前のやりとりの知識を必要とせず、セッション情報を保存しません。

ステートレスな設計にすれば、あとは AWS Lambda や AWS Fargate などのサーバーレスコンピューティングサービスを利用できます。

サーバーの置き換えに加えて、ステートレスアプリケーションのもう 1 つの利点は、利用可能なコンピューティングリソース (EC2 インスタンスや AWS Lambda 関数など) がどのようなリクエストにも対応できるため、水平方向にスケーリングできることです。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- アプリケーションをステートレスにします。ステートレスアプリケーションは、水平方向へのスケーリングが可能であり、個別ノードのエラーに耐性があります。
  - リクエストパラメータに実際に格納できるステートを削除する
  - ステートが必要かどうかを調べてから、あらゆるステート追跡を Amazon ElastiCache、Amazon RDS、Amazon DynamoDB などの回復力のあるマルチゾーンキャッシュやデータストア、またはサードパーティの分散データソリューションに移動します。移動できないステートをエラーに強いデータストアに格納します。
    - 一部のデータ (Cookie など) は、ヘッダーまたはクエリパラメータで渡すことができます。
    - リクエストですばやく渡すことができるステートを削除するためにリファクタリングします。
    - 実際には毎回のリクエストで必要のないデータはオンデマンドで取得できます。
    - 非同期で取得できるデータを削除します。
    - 必要なステートの条件を満たしているデータストアを決めます。
    - リレーショナル型ではないデータには NoSQL データベースを検討します。

リソース

関連するドキュメント:

- The Amazon Builders' Library: 分散システムでのフォールバックの回避
- The Amazon Builders' Library: 克服できないキューバックログの回避
- The Amazon Builders' Library: キャッシュの課題と戦略

REL05-BP07 緊急レバーを実装する

緊急レバーは、ワークロードの可用性に対する影響を軽減できる迅速なプロセスです。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 緊急レバーを実装します。これは、ワークロードの可用性に対する影響を軽減できる可能性がある迅速なプロセスです。根本原因がなくても操作できます。緊急レバーは、完全に決定的なアクティブ化と非アクティブ化の基準を提供することにより、リゾルバーの認知負荷をゼロに減らせるものが理想的です。緊急レバーは多くの場合手動ですが、自動化することもできます
  - 例えば、次のようなレバーがあります。
    - すべてのロボットトラフィックをブロックする
    - 動的ページの代わりに静的ページを表示する
    - 依存関係への呼び出しの頻度を減らす
    - 依存関係からの呼び出しをスロットリングする
  - 緊急レバーを実装して使用するためのヒント
    - 緊急レバーがアクティブになったら、実行数を増やすのではなく、減らす
    - シンプルに保ち、バイモーダルな行動は避ける
    - 緊急レバーを定期的にテストする
  - これらは、緊急レバーではないアクションの例です
    - キャパシティーを追加する
    - サービスに依存するクライアントのサービス所有者を呼び出して、呼び出しを減らすよう依頼する
    - コードを変更してリリースする

変更管理

質問

- REL 6 ワークロードリソースをモニタリングするにはどうすればよいですか?
- REL 7 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?
- REL 8 変更はどのように実装するのですか?

REL 6 ワークロードリソースをモニタリングするにはどうすればよいですか?

ログとメトリクスは、ワークロードの状態についての洞察を得るための強力なツールです。ワークロードは、しきい値を超えたり重大なイベントが発生したりしたときに、ログとメトリクスがモニ

タリングされて通知が送信されるように構成できます。モニタリングにより、ワークロードは、低パフォーマンスのしきい値を超えたときや障害が発生したときにそれを認識できるため、それに応じて自動的に復旧できます。

ベストプラクティス

- REL06-BP01 ワークロードのすべてのコンポーネントをモニタリングする (生成)
- REL06-BP02 メトリクスを定義および計算する (集計)
- REL06-BP03 通知を送信する (リアルタイム処理とアラーム)
- REL06-BP04 レスポンスを自動化する (リアルタイム処理とアラーム)
- REL06-BP05 分析
- REL06-BP06 定期的にレビューを実施する
- REL06-BP07 システムを通じたリクエストのエンドツーエンドのトレースをモニタリングする

REL06-BP01 ワークロードのすべてのコンポーネントをモニタリングする (生成)

ワークロードのコンポーネントは、Amazon CloudWatch またはサードパーティー製ツールを使用してモニタリングします。AWS サービスを AWS Health ダッシュボードでモニタリングします。

フロントエンド、ビジネスロジック、ストレージ層など、ワークロードのすべてのコンポーネントをモニタリングする必要があります。主要なメトリクスと、必要に応じてそれをログから抽出する方法を定義し、対応するアラームイベントを起動させるためのしきい値を設定します。メトリクスがワークロードの重要業績評価指標 (KPI) に関連していることを確認し、メトリクスとログを使用して、サービス低下の早期警告サインを識別します。例えば、1 分間に正常に処理されたオーダー数など、ビジネス成果に関するメトリクスは、CPU 使用率などの技術的メトリクスより早く、ワークロード問題を示すことができます。AWS Health ダッシュボードは、AWS リソースの基盤となる AWS のサービスのパフォーマンスと可用性をパーソナライズして表示するために使用します。

クラウドでのモニタリングは新しい機会をもたらします。ほとんどのクラウドプロバイダーは、カスタマイズ可能なフックを開発して、ワークロードの複数のレイヤーをモニタリングする際に役立つインサイトを提供しています。Amazon CloudWatch などの AWS サービスは、統計的な機械学習アルゴリズムを応用して、システムとアプリケーションのメトリクスを継続的に分析し、正常なベースラインを決定し、最小限のユーザー介入で異常を表面化します。異常検出アルゴリズムでは、メトリクスの季節変動とトレンドの変化が考慮されます。

AWS では、豊富なモニタリングおよびログ情報を公開しており、これらを使用して、ワークロード固有のメトリクスと需要変化プロセスを定義し、機械学習の知識に関わらず、機械学習技法を適応させることができます。

さらに、すべての外部エンドポイントをモニタリングし、それらがベースとなる実装から独立していることを確認します。このアクティブモニタリングは、合成トランザクション (ユーザーカナリアと呼ばれることもありますが、canary デプロイと混同しないでください) で行うことができます。これは、ワークロードのクライアントによって実行されるアクションに相当する多数の一般的タスクを定期的に実行するというものです。これらのタスクは、短期間に保ち、テスト中にワークロードに負荷をかけすぎないようにしてください。Amazon CloudWatch Synthetics を使用すると、 Synthetics Canary を作成して、 エンドポイントと API をモニタリングできます。合成 Canary クライアントノードと AWS X-Ray コンソールを組み合わせて、選択した期間中にエラー、障害、スロットリング率で問題が発生している合成 Canary を特定することもできます。

期待される成果:

ワークロードのすべてのコンポーネントから重要なメトリクスを収集して使用し、ワークロードの信頼性と最適なユーザーエクスペリエンスを確保します。ワークロードがビジネス成果を達成していないことを検出した場合は、障害を迅速に宣言して、インシデントから復旧できます。

一般的なアンチパターン:

- ワークロードへの外部インターフェイスのみをモニタリングする。
- ワークロード固有のメトリクスを生成せず、ワークロードが使用している AWS から提供されるメトリクスにのみ依存する。
- ワークロードの技術的メトリクスを使用するだけで、ワークロードが貢献する非技術的な KPI に関するメトリクスをモニタリングしない。
- 本番トラフィックとシンプルなヘルスチェックに依存して、ワークロード状態をモニタリングし、評価する。

このベストプラクティスを確立するメリット: ワークロードのすべての階層でモニタリングすることで、ワークロードを構成するコンポーネントの問題をより迅速に予測し、解決できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

1. 可能な限りログを有効にします。 ワークロードのすべてのコンポーネントからモニタリングデータを取得する必要があります。S3 Access Logs など、追加のロギングをオンにして、ワークロードがワークロード固有のデータをログに記録できるようにします。Amazon ECS、Amazon EKS、Amazon EC2、Elastic Load Balancing、AWS Auto Scaling、Amazon EMR などのサービスから、CPU、ネットワーク I/O、およびディスク I/O の平均に関するメトリクスを収集します。把

握 CloudWatch メトリクスを発行する AWS のサービス で、CloudWatch にメトリクスを発行する AWS のサービスのリストを確認できます。

2. デフォルトのメトリクスをすべてレビューし、データ収集にギャップがないか確認します。 すべてのサービスはデフォルトのメトリクスを生成します。デフォルトのメトリクスを収集することで、ワークロードのコンポーネント間の依存関係と、コンポーネントの信頼性とパフォーマンスがワークロードに及ぼす影響をより深く理解できます。また、独自のメトリクスを 作成して、AWS CLI または API を使用して CloudWatch に発行することもできます。 *** please leave this segment blank since the source does not make any sense in context ***
3. すべてのメトリクスを評価して、ワークロード内の各 AWS サービスに対してどのメトリクスでアラートを出すかを決定します。 ワークロードの信頼性に大きな影響を持つメトリクスのサブセットを選択することもできます。重要なメトリクスとしきい値に焦点を当てることで、 アラート の数を絞り込み、偽陽性を最小化できます。
4. アラートを定義し、アラートが起動された後のワークロードの復旧プロセスを定義します。 アラートを定義することで、通知とエスカレーションを迅速に行い、インシデントからの復旧に必要なステップに従い、所定の目標復旧時間 (RTO) を満たすことができます。専用のインフラストラクチャで Amazon CloudWatch アラーム を使用すると、定義されたしきい値に基づいて自動ワークフローを起動し、回復手順を開始することができます。
5. 合成トランザクションを使用して、ワークロードの状態に関する関連データを収集することを検討しましょう。 合成モニタリングは、顧客と同じルートに従って同じアクションを実行するため、ワークロードに顧客のトラフィックがない場合でも、継続的にカスタマーエクスペリエンスを検証することが可能になります。また、 合成トランザクションを使用することで、顧客より先に問題を発見できます。

リソース

関連するベストプラクティス:

- REL11-BP03 すべてのレイヤーの修復を自動化する

関連するドキュメント:

- Getting started with your AWS Health Dashboard – Your account health (AWS Health ダッシュボードの使用開始 - アカウントのヘルス)
- CloudWatch メトリクスを発行する AWS のサービス
- Network Load Balancer のアクセスログ
- Application Load Balancer のアクセスログ

- Accessing Amazon CloudWatch Logs for AWS Lambda
- Amazon S3 サーバーアクセスのログ記録
- Classic Load Balancer のアクセスログの有効化
- Amazon S3 へのログデータのエクスポート
- CloudWatch エージェントを Amazon EC2 インスタンスにインストールする
- カスタムメトリクスの発行
- Amazon CloudWatch ダッシュボードの使用
- Using Amazon CloudWatch Metrics
- Canary の使用 (Amazon CloudWatch Synthetics)
- Amazon CloudWatch Logs とは

ユーザーガイド:

- 追跡の作成
- Amazon EC2 Linux インスタンスのメモリとディスクのメトリクスのモニタリング
- コンテナインスタンスでの CloudWatch Logs の使用
- VPC フローログ
- Amazon DevOps Guru とは
- 「AWS X-Ray とは何ですか。」

関連ブログ:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

関連する例とワークショップ:

- AWS Well-Architected ラボ: 運用上の優秀性 - 依存関係のモニタリング
- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める
- Observability workshop (可観測性ワークショップ)

REL06-BP02 メトリクスを定義および計算する (集計)

ログデータを保存し、必要に応じてフィルターを適用します。これにより、特定のログイベントのカウントや、ログイベントのタイムスタンプから計算されたレイテンシーなどのメトリクスを計算できます。

Amazon CloudWatch と Amazon S3 は、主要な集約レイヤーおよびストレージレイヤーとして機能します。AWS Auto Scaling や Elastic Load Balancing などの一部のサービスでは、クラスターまたはインスタンス全体の CPU 負荷または平均的なリクエストのレイテンシーについて、デフォルトのメトリクスが提供されます。VPC フローログや AWS CloudTrail などのストリーミングサービスの場合、イベントデータは CloudWatch Logs に転送されるため、メトリクスフィルターを定義して適用し、イベントデータからメトリクスを抽出する必要があります。これにより、時系列データが提供されます。これは、アラートをトリガーするために定義した CloudWatch アラームへの入力データとして機能します。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- メトリクスを定義および計算します (集計)。特定のログイベントのカウントや、ログイベントのタイムスタンプから計算されたレイテンシーなどのメトリクスを計算するため、ログデータを保存し、必要に応じてフィルターを適用する
  - メトリクスフィルターは、CloudWatch Logs に送信されるログデータから検索する条件とパターンを定義します。CloudWatch Logs は、これらのメトリクスフィルターを使用して、ログデータを数値の CloudWatch メトリクスに変換し、グラフ化やアラームの設定を可能にします。
    - ログデータの検索およびフィルタリング
  - 信頼できるサードパーティーを使用してログを集計します。
    - サードパーティーの指示に従います。ほとんどのサードパーティー製品は、CloudWatch および Amazon S3 と統合されています。
  - 一部の AWS のサービスでは、ログを直接 Amazon S3 に発行できます。ログを Amazon S3 に保存することが主な要件であれば、追加のインフラを設定することなく、ログを生成するサービスに直接 Amazon S3 に送信させることが簡単にできます。
    - Sending Logs Directly to Amazon S3 (Amazon S3 へのログの直接送信)

リソース

関連するドキュメント:

- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- 1 つの可観測性ワークショップ
- ログデータの検索およびフィルタリング

- Sending Logs Directly to Amazon S3 (Amazon S3 への口グの直接送信)
- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める

REL06-BP03 通知を送信する (リアルタイム処理とアラーム)

重要なイベントが発生すると、把握する必要のある組織に通知が送信されます。

Amazon Simple Notification Service (Amazon SNS) トピックにアラートを送信し、任意の数の登録者にプッシュすることができます。例えば Amazon SNS では、E メールエイリアスにアラートを転送して、技術スタッフが対応できるようにしています。

一般的なアンチパターン:

- 低すぎるしきい値でアラームを設定することで、多すぎる通知が送信される。
- 今後の調査のためにアラームをアーカイブしない。

このベストプラクティスを確立するメリット: イベントの通知 (対応し、自動的に解決できるものであっても) により、イベントの記録を保持し、将来的に別の方法で対処できる場合があります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- リアルタイムの処理とアラームを実行します。重要なイベントが発生すると、把握しておく必要のある組織が通知を受信します
  - Amazon CloudWatch ダッシュボードは、CloudWatch コンソール内のカスタマイズ可能なホームページであり、これを使用すると、異なるリージョンにまたがっているリソースであっても、単一のビューでリソースをモニタリングできます。
    - Amazon CloudWatch ダッシュボードの使用
  - メトリクスが制限を超える場合にアラームを作成します。
    - Amazon CloudWatch アラームの使用

リソース

関連するドキュメント:

- 1 つの可観測性ワークショップ

- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める
- Amazon CloudWatch アラームの使用
- Amazon CloudWatch ダッシュボードの使用
- Using Amazon CloudWatch Metrics

REL06-BP04 レスポンスを自動化する (リアルタイム処理とアラーム)

自動化を使用して、イベントが検出されたときにアクションを実行します (例えば、障害が発生したコンポーネントを交換します)。

アラートは、クラスターが需要の変化に対応できるように、AWS Auto Scaling イベントをトリガーします。アラートは、サードパーティチケットシステムの統合ポイントとして機能する Amazon Simple Queue Service (Amazon SQS) に送信できます。AWS Lambda は、アラートをサブスクライブして、変更に対して動的に対応する非同期サーバーレスモデルをユーザーに提供することもできます。AWS Config は AWS リソースの構成を継続的にモニタリングして記録し、 AWS Systems Manager Automation をトリガーして 問題を修復できます。

Amazon DevOps Guru は、異常な動作についてアプリケーションリソースを自動的にモニタリングし、的を絞ったレコメンデーションを提供することにより、問題の識別を速めて修復時間を短縮します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- Amazon DevOps Guru を使用して、自動化アクションを実行します。Amazon DevOps Guru は、異常な動作についてアプリケーションリソースを自動的にモニタリングし、的を絞ったレコメンデーションを提供することにより、問題の識別を速めて修復時間を短縮します。
  - 「Amazon DevOps Guru とは何ですか?」
- AWS Systems Manager を使用して、自動化アクションを実行します。AWS Config は AWS リソースの設定を継続的にモニタリングおよび記録し、AWS Systems Manager Automation をトリガーして問題を修復できます。
  - AWS Systems Manager Automation をトリガーして
    - Systems Manager Automation ドキュメントを作成して使用します。これらは、オートメーションプロセスの実行時に Systems Manager がマネージドインスタンスおよび他の AWS リソースに対して実行するアクションを定義します。
    - オートメーションドキュメント (プレイブック) の使用

- Amazon CloudWatch は、状態変更イベントを Amazon EventBridge に警告します。EventBridge ルールを作成して、レスポンスを自動化します。
  - AWS リソースからのイベントでトリガーする EventBridge ルールの作成
- 応答を自動化する計画を作成して実行します。
  - すべてのアラート応答手順をインベントリします。タスクをランク付けする前に、アラートレスポンスを計画する必要があります。
  - 実行する必要がある特定のアクションを含むすべてのタスクをインベントリします。これらのアクションのほとんどは、ランブックに記載されています。また、予期しないイベントのアラートに対するプレイブックも必要です。
  - すべての自動化可能なアクションについて、ランブックとプレイブックを調べます。一般に、アクションを定義できる場合は、ほとんどの場合、自動化できます。
  - エラーが発生しやすいアクティビティや時間のかかるアクティビティを上位にランク付けます。エラーの原因を取り除き、解決までの時間を短縮することが最も有益です。
  - オートメーションを完了する計画を立てます。自動化と、自動化を更新するためのアクティブな計画を維持します。
  - オートメーションの機会に関する手動要件を調べます。手動プロセスの自動化機会に挑戦します。

リソース

関連するドキュメント:

- AWS Systems Manager Automation をトリガーして
- AWS リソースからのイベントでトリガーする EventBridge ルールの作成
- 1 つの可観測性ワークショップ
- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める
- 「Amazon DevOps Guru とは何ですか?」
- オートメーションドキュメント (プレイブック) の使用

REL06-BP05 分析

ログファイルとメトリクスの履歴を収集し、これらを分析して、幅広いトレンドとワークロードの洞察が得られます。

Amazon CloudWatch Logs Insights は、 シンプルかつ強力なクエリ言語をサポートし、 ログデータの分析に使用できます。Amazon CloudWatch Logs ではさらに、シームレスにデータを Amazon S3 に送ってデータを使用したり、または Amazon Athena に送ってデータをクエリしたりできるサブスクリプションもサポートしています。豊富な種類のフォーマットのクエリがサポートされています。把握 サポートされる SerDes とデータ形式 詳細については、Amazon Athena ユーザーガイドを参照してください。巨大なログファイルセットの分析では、Amazon EMR クラスターを実行してペタバイト規模の分析を実行できます。

集計、処理、保存、分析を実行できる多数のツールが AWS パートナーや
サードパーティによって提供されています。このようなツールには、New
Relic、Splunk、Loggly、Logstash、CloudHealth、Nagios などがあります。ただし、システムやアプリケーションログの外で行うデータ生成は各クラウドプロバイダーに固有であり、また多くの場合サービスごとに固有です。

モニタリングプロセスで見落とされがちな点は、データ管理です。モニタリングのためのデータ保存要件を決定し、それに応じたライフサイクルポリシーを適用する必要があります。Amazon S3 は S3 バケットレベルのライフサイクル管理をサポートしています。このライフサイクル管理には、バケット内のパスごとに異なる管理方法を適用できます。ライフサイクルの最終段階では、データを Amazon S3 Glacier に移行して長期保存し、保存期間の終了後には期限切れにすることができます。S3 Intelligent-Tiering ストレージクラスは、パフォーマンスへの影響や運用のオーバーヘッドなしに、データを最も費用対効果の高いアクセス階層に自動的に移動することにより、コストを最適化できるように設計されています。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- CloudWatch Logs Insights を使用すると、Amazon CloudWatch Logs でログデータをインタラクティブに検索して分析できます。
  - CloudWatch Logs Insights を使用したログデータの分析
  - Amazon CloudWatch Logs Insights Sample Queries
- 使用できる場合は、Amazon CloudWatch Logs を使用してログを Amazon S3 に送信するか、Amazon Athena を使用してデータをクエリします。
  - Athena を使用して Amazon S3 サーバーのアクセスログを分析するにはどうすればよいですか?
    - サーバーアクセスログバケットの S3 ライフサイクルポリシーを作成します。ライフサイクルポリシーを設定して、定期的にログファイルを削除します。そうすることで、Athena が各クエリについて分析するデータ量が削減されます。

- S3 バケットのライフサイクルポリシーを作成する方法

リソース

関連するドキュメント:

- Amazon CloudWatch Logs Insights Sample Queries
- CloudWatch Logs Insights を使用したログデータの分析
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- S3 バケットのライフサイクルポリシーを作成する方法
- Athena を使用して Amazon S3 サーバーのアクセスログを分析するにはどうすればよいですか?
- 1 つの可観測性ワークショップ
- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める

REL06-BP06 定期的にレビューを実施する

ワークロードモニタリングがどのように実装されているかを頻繁に確認し、重要なイベントや変更に基づいて更新します。

効果的なモニタリングは、主要なビジネスメトリクスが原動力になります。ビジネスの優先順位が変化したときに、メトリクスがワークロードに確実に対応できるようにします。

モニタリングを監査することで、アプリケーションがどのタイミングで可用性の目標を満たしているかを確実に把握できます。根本原因の分析には、障害発生時に何が起こったかを発見する機能が必要です。AWS は、インシデント時にサービスの状態を追跡できるサービスを提供しています。

- Amazon CloudWatch Logs: このサービスにログを保存してその内容を調査できます。
- Amazon CloudWatch Logs Insights: 数秒で大量のログを分析できるフルマネージドサービスです。高速でインタラクティブなクエリと視覚化が行えます。
- AWS Config: さまざまな時点でどの AWS インフラストラクチャが使用されているかを確認できます。
- AWS CloudTrail: どの AWS API が、いつどのプリンシパルに呼び出されたかを確認できます。

AWS では、週に一度のミーティングを実施して、 運用パフォーマンスをレビューし、 学んだ教訓をチーム間で共有しています。AWS には多数のチームが存在するため、 私たちは The Wheel を

作成し、 ワークロードをランダムに選んで確認できるようにしました。運用パフォーマンスのレビューと知識の共有を定期的に行うことで、運用チームのパフォーマンスを向上させることができます。

一般的なアンチパターン:

- デフォルトのメトリクスのみを収集する。
- モニタリング戦略を設定し、見直さない。
- 主要な変更がデプロイされる際に、モニタリングについて話し合わない。

このベストプラクティスを活用するメリット: モニタリングを定期的にレビューすることで、予期される問題が実際に発生したときに通知に反応する代わりに、潜在的な問題を予測できるようになります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ワークロード用に複数のダッシュボードを作成します。主要なビジネスメトリクスと、使用状況の変化に応じて予測されるワークロードの状態に最も関連性があるものとして特定した技術メトリクスを含む最上位のダッシュボードが必要です。また、検査が可能なさまざまなアプリケーション層や依存関係のダッシュボードも必要があります。
  - Amazon CloudWatch ダッシュボードの使用
- ワークロードダッシュボードの定期的なレビューをスケジュールし、実施します。ダッシュボードの定期的な検査を行います。検査する深度に応じて異なる頻度にすることができます。
  - メトリクスの傾向を検査します。メトリクス値と履歴値を比較して、調査が必要なものを示唆している可能性がある傾向があるかどうかを確認します。これには、レイテンシーの増加、主要なビジネス機能の減少、失敗レスポンスの増加などがあります。
  - メトリクスの外れ値/異常を検査します。平均値または中央値は、外れ値と異常値を覆い隠すことがあります。期間中の最大値と最低値を調べ、極端なスコアの原因を調査します。これらの原因の排除を続行しながら、極値の定義を低くしていくことで、ワークロードパフォーマンスの一貫性を継続して向上させることができます。
  - 行動の急変を探します。メトリクスの数量または方向性の突然の変化は、アプリケーションに変更があったこと、または追跡するためにさらなるメトリクスを追加する必要がある外部要因があることを示唆している可能性があります。

リソース

関連するドキュメント:

- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- 1 つの可観測性ワークショップ
- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める
- Amazon CloudWatch ダッシュボードの使用

REL06-BP07 システムを通じたリクエストのエンドツーエンドのトレースをモニタリングする

AWS X-Ray またはサードパーティ製のツールを使用することで、デベロッパーは分散システムの分析とデバッグをより簡単に行い、アプリケーションとその基盤となるサービスのパフォーマンスを把握できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- システムを通じたリクエストのエンドツーエンドのトレースをモニタリングします。AWS X-Ray は、アプリケーションが処理するリクエストに関するデータを収集するサービスであり、データの表示、フィルタリング、インサイトの取得によって、問題や最適化の機会を特定するためのツールを提供します。アプリケーションへのトレースされたリクエストについて、リクエストとレスポンスだけでなく、アプリケーションがダウンストリーム AWS リソース、マイクロサービス、データベース、ウェブ API に対して行う呼び出しに関する詳細情報も確認できます。
  - 「AWS X-Ray とは何ですか。」
  - Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

リソース

関連するドキュメント:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- 1 つの可観測性ワークショップ
- The Amazon Builders' Library: 分散システムを装備して、運用の可視性を高める
- 模擬モニタリングの使用 (Amazon CloudWatch Synthetics)

- 「AWS X-Ray とは何ですか。」

REL 7 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?

スケーラブルなワークロードには、リソースを自動で追加または削除する伸縮性があるので、リソースは常に、現行の需要に厳密に適合します。

ベストプラクティス

- REL07-BP01 リソースの取得またはスケーリング時に自動化を使用する
- REL07-BP02 ワークロードの障害を検出したときにリソースを取得する
- REL07-BP03 ワークロードにより多くのリソースが必要であることを検出した時点でリソースを取得する
- REL07-BP04 ワークロードの負荷テストを実施する

REL07-BP01 リソースの取得またはスケーリング時に自動化を使用する

障害のあるリソースを交換したり、ワークロードをスケーリングしたりする場合は、Amazon S3 や AWS Auto Scaling などのマネージド型の AWS のサービスを使用してプロセスを自動化します。サードパーティのツールや AWS SDK を使用して、スケーリングを自動化することもできます。

マネージド型の AWS のサービスとしては、Amazon S3、Amazon CloudFront、AWS Auto Scaling、AWS Lambda、Amazon DynamoDB、AWS Fargate、および Amazon Route 53 があります。

AWS Auto Scaling では、障害のあるインスタンスを検出して置き換えることができます。また、以下を含むリソースのスケーリングプランを構築することもできます。 Amazon EC2 インスタンスとスポットフリート、 Amazon ECS タスク、 Amazon DynamoDB テーブルとインデックス、および Amazon Aurora レプリカ。

EC2 インスタンスをスケーリングする場合は、複数のアベイラビリティゾーン (できれば少なくとも 3 つ) を使用し、容量を追加または削除して、これらのアベイラビリティゾーン間のバランスを維持します。ECS タスクまたは Kubernetes ポッド (Amazon Elastic Kubernetes Service を使用しているとき) も複数のアベイラビリティゾーンに分散してください。

AWS Lambda を使用しているときには、インスタンスは自動的にスケーリングされます。AWS Lambda は、関数のイベント通知を受信するたびに、コンピューティングフリート内の空き容量を

すばやく見つけ、割り当てられた同時実行数までコードを実行します。特定の Lambda とService Quotas で、必要な同時実行数が確実に設定されているようにしてください。

Amazon S3 は、高いリクエスト頻度を処理できるように自動的にスケーリングします。たとえば、アプリケーションはバケット内のプレフィックスごとに 1 秒あたり 3,500 件以上の PUT/COPY/POST/DELETE リクエストまたは 5,500 件以上の GET/HEAD リクエストを送信できます。バケット内のプレフィックス数に制限はありません。読み取りを並列化することで、読み取りまたは書き込みのパフォーマンスを向上させることができます。例えば、Amazon S3 バケットに 10 個のプレフィックスを作成して読み取りを並列化する場合、読み取りパフォーマンスを 1 秒あたり 55,000 件の読み取りリクエストにスケーリングできます。

Amazon CloudFront または信頼できるコンテンツ配信ネットワーク (CDN) を設定して使用します。CDN は、より迅速なエンドユーザーレスポンスタイムを提供でき、コンテンツのリクエストをキャッシュから処理できるため、ワークロードをスケーリングする必要性が少なくなります。

一般的なアンチパターン:

- 自動ヒーリングのために Auto Scaling グループを実装しますが、伸縮性は実装しません。
- トラフィックの大幅な増加に対応するために自動スケーリングを使用する。
- ステートフル性が高いアプリケーションをデプロイし、伸縮性を排除する。

このベストプラクティスを活用するメリット: 自動化により、リソースのデプロイと廃棄で手動エラーが発生する可能性がなくなります。自動化は、デプロイや廃棄のニーズへの応答が遅いことによるコストの超過やサービス拒否のリスクを排除します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- AWS Auto Scaling を設定して使用します。これにより、アプリケーションをモニタリングし、安定した予測可能なパフォーマンスを可能な限り低いコストで維持するためのキャパシティーを自動的に調整します。AWS Auto Scaling を使用すると、複数のサービスにまたがる複数のリソースに対してアプリケーションのスケーリングをセットアップできます。
  - 「AWS Auto Scaling とは何ですか?」
    - Amazon EC2 インスタンスとスポットフリート、Amazon ECS タスク、Amazon DynamoDB のテーブルとインデックス、Amazon Aurora のレプリカ、および AWS Marketplace アプライアンスなど、該当する者に対して Auto Scaling を設定します。
      - DynamoDB Auto Scaling によるスループットキャパシティの自動管理

        - サービス API を操作して、アラーム、スケーリングポリシー、ウォームアップ時間、およびクールダウン時間を指定します。
- Elastic Load Balancing を使用します。ロードバランサーは、パスまたはネットワーク接続ごとに負荷を分散することができます。
    - 「Elastic Load Balancing とは何ですか?」
        - Application Load Balancers は、負荷をパスごとに分散できます。
            - Application Load Balancer とは?
                - Application Load Balancer を設定して、ドメイン名の下のパスに基づいてトラフィックをさまざまなワークロードに分散します。
                - Application Load Balancers を使用すると、AWS Auto Scaling と統合して需要を管理するという方法で負荷を分散できます。
                    - Auto Scaling グループでロードバランサーを使用する
        - Network Load Balancers は、接続ごとに負荷を分散することができます。
            - Network Load Balancer とは?
                - Network Load Balancer は、TCP を使用してトラフィックをさまざまなワークロードに分散するか、ワークロードの IP アドレスの一定のセットが含まれるように設定します。
                - Network Load Balancer を使用すると、AWS Auto Scaling と統合して需要を管理するという方法で負荷を分散できます。
- 可用性の高い DNS プロバイダーを使用します。DNS 名により、ユーザーは、IP アドレスの代わりに DNS 名を入力してワークロードにアクセスでき、この情報を、定義されたスコープ (通常はワークロードのユーザーに対してグローバルに定義されたスコープ) に分散できます。
    - Amazon Route 53 または信頼できる DNS プロバイダーを使用します。
        - 「Amazon Route 53 とは何ですか?」
    - Route 53 を使用して、CloudFront ディストリビューションとロードバランサーを管理します。
        - 管理する予定のドメインとサブドメインを決定します。
        - ALIAS レコードまたは CNAME レコードを使用して適切なレコードセットを作成します。
            - レコードの操作
- AWS グローバルネットワークを使用して、ユーザーからアプリケーションへのパスを最適化します。AWS Global Accelerator は、アプリケーションエンドポイントの状態を継続的にモニタリングし、トラフィックを 30 秒未満で正常なエンドポイントにリダイレクトします。
    - AWS Global Accelerator は、ローカルまたはグローバルユーザーが使用するアプリケーションの可用性とパフォーマンスを向上させるサービスです。Application Load Balancers、Network

Load Balancer、Amazon EC2 インスタンスなど、単一または複数の AWS リージョンのアプリケーションエンドポイントへの固定エントリポイントとして機能する静的 IP アドレスが提供されます。

  - AWS Global Accelerator とは?

- Amazon CloudFront または信頼できるコンテンツ配信ネットワーク (CDN) を設定して使用します。コンテンツ配信ネットワークは、エンドユーザーの応答時間を短縮し、ワークロードの不要なスケーリングを引き起こす原因となるコンテンツのリクエストを処理できます。
  - 「Amazon CloudFront とは」
    - ワークロード用の Amazon CloudFront ディストリビューションを設定するか、サードパーティの CDN を使用します。
      - エンドポイントセキュリティグループまたはアクセスポリシーで CloudFront の IP 範囲を使用することで、ワークロードへのアクセスを CloudFront からのみに制限できます。

リソース

関連するドキュメント:

- APN Partner: partners that can help you create automated compute solutions
- AWS Auto Scaling: スケーリングプランの仕組み
- AWS Marketplace: Auto Scaling で使用できる製品
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Auto Scaling グループでロードバランサーを使用する
- AWS Global Accelerator とは?
- 「What Is Amazon EC2 Auto Scaling?」
- 「AWS Auto Scaling とは何ですか?」
- 「Amazon CloudFront とは」
- 「Amazon Route 53 とは何ですか?」
- 「Elastic Load Balancing とは何ですか?」
- Network Load Balancer とは?
- Application Load Balancer とは?
- レコードの操作

REL07-BP02 ワークロードの障害を検出したときにリソースを取得する

可用性が影響を受ける場合、必要に応じてリソースをリアクティブにスケールし、ワークロードの可用性を復元します。

まず、ヘルスチェックとこのチェックの基準を設定して、リソースの不足が可用性に影響を与えるタイミングを示す必要があります。次に、適切な担当者に通知してリソースを手動でスケーリングするか、自動操作をトリガーしてリソースを自動的にスケーリングします。

スケーリングはワークロードに合わせて手動で調整できます。例えば、Auto Scaling グループの EC2 インスタンスの数の変更や、DynamoDB テーブルのスループットの変更は、AWS Management Console または AWS CLI で行うことができます。ただし、可能な限りオートメーションを使用する必要があります (詳細は リソースの取得またはスケーリング時に自動化を使用する) を指定する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ワークロードの障害を検出したときにリソースを取得します。可用性が影響を受ける場合、必要に応じてリソースをリアクティブにスケールし、ワークロードの可用性を復元します。
  - AWS Auto Scaling のコアコンポーネントであるスケーリングプランを使用して、リソースをスケーリングするための一連の指示を設定します。AWS CloudFormation を操作する場合や、タグを AWS リソースに追加する場合、アプリケーションごとに、さまざまなリソースセットのスケーリングプランをセットアップできます。AWS Auto Scaling は各リソースに応じてカスタマイズされたスケーリング戦略についてレコメンデーションを提供します。スケーリングプランを作成すると、AWS Auto Scaling は、動的スケーリングと予測スケーリング方法を組み合わせて、スケーリング戦略をサポートします。
    - AWS Auto Scaling: スケーリングプランの仕組み
  - Amazon EC2 Auto Scaling を使用すると、アプリケーションの負荷を処理するための正しい数の Amazon EC2 インスタンスを利用できます。Auto Scaling グループと呼ばれる EC2 インスタンスのコレクションを作成します。各 Auto Scaling グループのインスタンスの最小数を指定でき、Amazon EC2 Auto Scaling では、グループがこのサイズを下回ることはありません。各 Auto Scaling グループのインスタンスの最小数を指定でき、Amazon EC2 Auto Scaling では、グループがこのサイズを上回ることはありません。
    - 「What Is Amazon EC2 Auto Scaling?」
  - Amazon DynamoDB Auto Scaling は、AWS Application Auto Scaling サービスを使用して、実際のトラフィックパターンに応じて、お客様に代わってプロビジョニングされたスループットキャ

パシティを動的に調整します。これにより、テーブルまたはグローバルセカンダリインデックスは、プロビジョニングされた読み込みおよび書き込みキャパシティーを増やすことができ、スロットリングなしでトラフィックの急激な増加を処理できます。

- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling

リソース

関連するドキュメント:

- APN Partner: partners that can help you create automated compute solutions
- AWS Auto Scaling: スケーリングプランの仕組み
- AWS Marketplace: Auto Scaling で使用できる製品
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- 「What Is Amazon EC2 Auto Scaling?」

REL07-BP03 ワークロードにより多くのリソースが必要であることを検出した時点でリソースを取得する

需要に合わせてリソースをプロアクティブにスケールし、可用性への影響を回避します。

多くの AWS サービスは、需要に合わせて自動的にスケールします。Amazon EC2 インスタンスまたは Amazon ECS クラスターを使用している場合、ワークロードの需要に対応する使用状況のメトリクスに基づいて Auto Scaling を実行するように設定できます。Amazon EC2 では、平均 CPU 使用率、ロードバランサーリクエスト数、またはネットワーク帯域幅を使用して、EC2 インスタンスをスケールアウト (またはスケールイン) できます。Amazon ECS では、平均 CPU 使用率、ロードバランサーリクエスト数、およびメモリ使用率を使用して、ECS タスクをスケールアウト (またはスケールイン) できます。AWS で Target Auto Scaling を使用すると、オートスケーラーは家庭用サーモスタットのように機能し、指定したターゲット値 (例えば、CPU 使用率 70%) を維持するためにリソースを追加または削除します。

AWS Auto Scaling はまた、 Predictive Auto Scaling も実行できます。これは、機械学習を使用して各リソースの過去のワークロードを分析し、次の 2 日間の負荷を定期的に予測します。

リトルの法則は、必要なコンピューティングインスタンス (EC2 インスタンス、同時実行の Lambda 関数など) 数を計算するのに役立ちます。

$L = \lambda W$

L = インスタンス数 (またはシステムの平均同時実行数)

λ = リクエストが到着する平均レート (リクエスト/秒)

W = 各リクエストがシステムで費やす平均時間 (秒)

例えば、100 rps では、各リクエストの処理に 0.5 秒かかる場合、需要に対応するには 50 インスタンスが必要です。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ワークロードにより多くのリソースが必要であることを検出した時点でリソースを取得します。需要に合わせてリソースをプロアクティブにスケールし、可用性への影響を回避します。
  - 特定のリクエストレートを処理するために必要なコンピューティングリソースの数 (コンピューティングの同時実行) を計算します。
    - リトルの法則について語る
  - 使用状況の履歴パターンがあるときには、Amazon EC2 Auto Scaling のスケジュールされたスケーリングをセットアップします。
    - Amazon EC2 Auto Scaling のスケジュールされたスケーリング
  - AWS 予測スケーリングを使用します。
    - 機械学習を利用した EC2 の予測スケーリング

リソース

関連するドキュメント:

- AWS Auto Scaling: スケーリングプランの仕組み
- AWS Marketplace: Auto Scaling で使用できる製品
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- 機械学習を利用した EC2 の予測スケーリング
- Amazon EC2 Auto Scaling のスケジュールされたスケーリング
- リトルの法則について語る
- 「What Is Amazon EC2 Auto Scaling?」

REL07-BP04 ワークロードの負荷テストを実施する

負荷テスト手法を採用して、スケーリングアクティビティがワークロード要件を満たすかどうかを測定します。

持続的な負荷テストを実行することが重要です。負荷テストによって、ブレークポイントを発見し、ワークロードのパフォーマンスをテストします。AWS は、生産ワークロードのスケールをモデル化する一次的なテスト環境のセットアップを容易にします。クラウド上では、本稼働スケールのテスト環境をオンデマンドで作成し、テスト完了後にリソースを解放できます。テスト環境の#払いは実#時にのみ発#するため、オンプレミスでテストを実施する場合と比べてわずかなコストで、本番環境をシミュレートできます。

本番環境での負荷テストは、ゲームデーの一部として考える必要もあります。その中で、顧客の使用率が低い時間帯に本番システムに負荷をかけ、担当者全員がテスト結果を解釈して発生した問題に対処できるようにします。

一般的なアンチパターン:

- 本稼働環境と同じ設定ではないデプロイで負荷テストを実行する。
- ワークロード全体ではなく、ワークロードの個々の部分に対してのみ負荷テストを実行する。
- 実際のリクエストの代表的なセットではなく、リクエストのサブセットを使用して負荷テストを実行する。
- 予想される負荷を超える小さな安全率に対して負荷テストを実行する。

このベストプラクティスを活用するメリット: 負荷がかかっているときにアーキテクチャのどのコンポーネントが失敗するかを把握し、問題に対処すべく、その負荷が近づいていることを示すために監視するメトリクスを特定し、その障害の影響を防ぐことができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 負荷テストを実行して、キャパシティを追加または削除する必要があるワークロードの側面を特定します。負荷テストには、本稼働環境で受け取るものと同様の代表的なトラフィックを用いる必要があります。設定したメトリクスを監視しながら負荷を増やし、リソースを追加または削除する必要があるタイミングをどのメトリクスが示唆しているのかを判断します。
  - AWS での分散負荷テスト: 接続された数千のユーザーをシミュレートする

- リクエストの組み合わせを特定します。なされるリクエストの組み合わせはさまざまであるため、トラフィックの混在を組み合わせを特定する際は、さまざまな時間枠を確認する必要があります。
- ロードドライバーを実装します。カスタムコード、オープンソース、または商用ソフトウェアを使用して、ロードドライバーを実装できます。
- 最初は小さなキャパシティを使用して負荷テストを実施します。1 つのインスタンスまたはコンテナと同じくらいの少なさのキャパシティーに負荷をかけることで、すぐに効果が現れます。
- 大きなキャパシティに対して負荷テストを実施します。この効果は分散された負荷によって異なるため、できるだけ製品環境に近いに対してテストする必要があります。

リソース

関連するドキュメント:

- AWS での分散負荷テスト: 接続された数千のユーザーをシミュレートする

REL 8 変更はどのように実装するのですか?

変更制御は、新しい機能をデプロイしたり、アプリケーションと運用環境で既知のソフトウェアが実行されており、予測できる方法でパッチを適用または置換できることを確認したりするために必要です。変更が制御されていないと、変更の影響を予測したり、変更によって発生した問題に対処したりすることが困難になります。

ベストプラクティス

- REL08-BP01 デプロイなどの標準的なアクティビティにランブックを使用する
- REL08-BP02 デプロイの一部として機能テストを統合する
- REL08-BP03 デプロイの一部として回復力テストを統合する
- REL08-BP04 イミュータブルなインフラストラクチャを使用してデプロイする
- REL08-BP05 オートメーションを使用して変更をデプロイする

REL08-BP01 デプロイなどの標準的なアクティビティにランブックを使用する

ランブックは、特定の成果を達成するための事前定義された手順です。手動または自動のどちらでも、標準的なアクティビティを実行するにはランブックを使用します。例えば、ワークロードのデプロイ、ワークロードへのパッチの適用、DNS の変更などがあります。

例えば、デプロイ中のロールバックの安全性を確保するためのプロセスを導入します。顧客側の中断なしでデプロイをロールバックできるようにすることは、サービスの信頼性を高める上で重要です。

ランブックの手順については、有効で効果的な手動プロセスから開始し、それをコードで実装して、適切な場合は自動実行をトリガーします。

高度に自動化された高機能のワークロードの場合でも、ランブックは本番環境で実行したり、厳格なレポートや監査の要件を満たしたりするのに役立ちます。

プレイブックは特定のインシデントに対応するために用いられ、ランブックは特定の結果を達成するために使用されます。多くの場合、ランブックは日常的なアクティビティ用で、プレイブックは非日常的なイベントに応えるために使用します。

一般的なアンチパターン:

- 本稼働環境の設定に対して、計画されていない変更を実行する。
- デプロイを高速化するために計画の手順をスキップすることで、デプロイを失敗させる。
- 変更を戻すことができるかどうかをテストせずに変更を加える。

このベストプラクティスを確立するメリット: 効果的な変更計画は、影響を受けるすべてのシステムを考慮しているため、変更を正常に実行する能力を強化します。テスト環境で変更を検証すると、信頼が強化されます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- ランブックに手順を文書化することにより、一貫性を保ち、汎用イベントにすみやかに対応できるようになります。
  - AWS Well-Architected Framework: Concepts: Runbook (AWS Well-Architected フレームワーク: 概念: ランブック)
- Infrastructure as Code の原則を適用して、インフラストラクチャを定義します。AWS CloudFormation (または信頼できるサードパーティ) を使用してインフラストラクチャを定義することにより、バージョン管理ソフトウェアを使用して、バージョン管理および変更の追跡を行うことができます。
  - AWS CloudFormation (または信頼できるサードパーティプロバイダー) を使用して、インフラストラクチャを定義します。

    - AWS CloudFormation とは
- 優れたソフトウェア設計の原則を使用して、単一または分離されたテンプレートを作成します。
    - 実装にあたって、必要な権限、テンプレート、責任者を決定します。
        - AWS Identity and Access Management によるアクセスの制御
    - バージョン管理には、AWS CodeCommit や信頼できるサードパーティ製ツールのようなバージョン管理用ソースコントロールを使用します。
        - AWS CodeCommit とは

リソース

関連するドキュメント:

- APN パートナー: 自動化されたデプロイソリューションの作成を支援できるパートナー
- AWS Marketplace: products that can be used to automate your deployments
- AWS Well-Architected Framework: Concepts: Runbook (AWS Well-Architected フレームワーク: 概念: ランブック)
- AWS CloudFormation とは
- AWS CodeCommit とは

関連する例:

- Automating operations with Playbooks and Runbooks (プレイブックとランブックによるオペレーションの自動化)

REL08-BP02 デプロイの一部として機能テストを統合する

機能テストは、自動デプロイの一部として実行されます。成功条件を満たさない場合、パイプラインは停止またはロールバックされます。

このようなテストは、パイプラインの本番稼働前にステージングされた本番稼働前環境で実行されます。これは、デプロイパイプラインの一部として行うのが理想的です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- デプロイの一部として機能テストを統合します。機能テストは、自動デプロイの一部として実行されます。成功条件を満たさない場合、パイプラインは停止またはロールバックされます。

- AWS CodePipeline でモデル化されたソフトウェアリリースパイプラインの「テストアクション」中に AWS CodeBuild を呼び出します。この機能により、ユニットテスト、静的コード分析、統合テストなど、コードに対してさまざまなテストを簡単に実行できます。
  - AWS CodePipeline が、AWS CodeBuild を使用したユニットテストおよびカスタム統合テストのサポートを追加
- AWS Marketplace ソリューションを使用して、ソフトウェア配信パイプラインの一部として自動テストを実行します。
  - ソフトウェアテストのオートメーション

リソース

関連するドキュメント:

- AWS CodePipeline が、AWS CodeBuild を使用したユニットテストおよびカスタム統合テストのサポートを追加
- ソフトウェアテストのオートメーション
- 「AWS CodePipeline とは何ですか?」

REL08-BP03 デプロイの一部として回復力テストを統合する

回復力テスト (カオスエンジニアリングの原則を使用した) は、本番稼働前環境で自動デプロイパイプラインの一部として実行されます。

このようなテストはステージングされ、本番稼働前にパイプラインで実行されます。ゲームデーの一部としても実行してください。 本番環境で実行する.

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- デプロイの一部として回復力テストを統合します。ワークロードの実験規律であるカオスエンジニアリングを使用して、本番環境での絶えず変化する条件に耐えるワークロードの能力に信頼を築きます。
  - 回復力テストでは、障害やリソースの低下を挿入して、設計した回復力をもってワークロードが応答することを評価します。
    - Well-Architected ラボ: レベル 300: EC2 RDS と S3 の回復力をテストする
  - これらのテストは、自動デプロイパイプラインの本番稼働前の環境で定期的に実行できます。

- また、スケジュールされたゲームデーの一環として、本稼働環境で実行する必要があります。
- カオスエンジニアリングの原則を使用して、さまざまな障害下でワークロードがどのように実行されるかについての仮説を提案し、回復力テストを使用して仮説をテストします。
  - カオスエンジニアリングの原則

リソース

関連するドキュメント:

- カオスエンジニアリングの原則
- AWS Fault Injection Simulator とは?

関連する例:

- Well-Architected ラボ: レベル 300: EC2 RDS と S3 の回復力をテストする

REL08-BP04 イミュータブルなインフラストラクチャを使用してデプロイする

イミュータブルなインフラストラクチャは、本番ワークロードで更新、セキュリティパッチ、または設定変更がインプレースで行われないように義務付けるモデルです。変更が必要な場合、アーキテクチャは新しいインフラストラクチャに構築され、本番環境にデプロイされます。

イミュータブルインフラストラクチャパラダイムを実装したものとして最も一般的なのが、 イミュータブルサーバーです。つまり、サーバーに更新または修正が必要な場合、既存のサーバーを更新するのではなく、新しいサーバーをデプロイします。そのため、アプリケーションのすべての変更は、SSH 経由でサーバーにログインしてソフトウェアバージョンを更新するのではなく、コードリポジトリにソフトウェアをプッシュすることから始まります (たとえば git push)。イミュータブルインフラストラクチャでは変更が許可されていないため、デプロイされたシステムの状態をしっかりと把握します。イミュータブルインフラストラクチャは、本質的に一貫性があり、信頼性が高く、予測可能であり、ソフトウェアの開発と運用の多くの側面を簡素化します。

イミュータブルインフラストラクチャにアプリケーションをデプロイする場合は、Canary デプロイまたはブルー/グリーンデプロイを使用します。

カナリアデプロイ は、通常は単一のサービスインスタンス (Canary) で実行される新しいバージョンに、少数の顧客を誘導する方法です。次に、生じた動作の変更やエラーを詳細に調べます。重大な問題が発生した場合、Canary からトラフィックを削除して、ユーザーを以前のバージョンに戻すことができます。デプロイが成功したら、変更やエラーをモニタリングしながら、希望の速度で完全にデ

プロイされるまでデプロイを続行できます。AWS CodeDeploy では、デプロイ設定で Canary デプロイを有効にすることでことができます。

ブルーグリーンデプロイ は Canary デプロイに似ていますが、アプリケーション全体が並行してデプロイされる点が異なります。2 つのスタック (青と緑) 間でデプロイを交互に行います。この場合も、トラフィックを新しいバージョンに送信したときにデプロイに問題が発生した場合は、古いバージョンにフォールバックできます。通常、すべてのトラフィックが一度に切り替えられますが、各バージョンへのトラフィックの一部を使用し、Amazon Route 53 の加重 DNS ルーティング機能を使用して、新しいバージョンの採用をダイヤルアップすることもできます。AWS CodeDeploy と AWS Elastic Beanstalk は、ブルー/グリーンデプロイを有効にするデプロイ構成で設定できます。

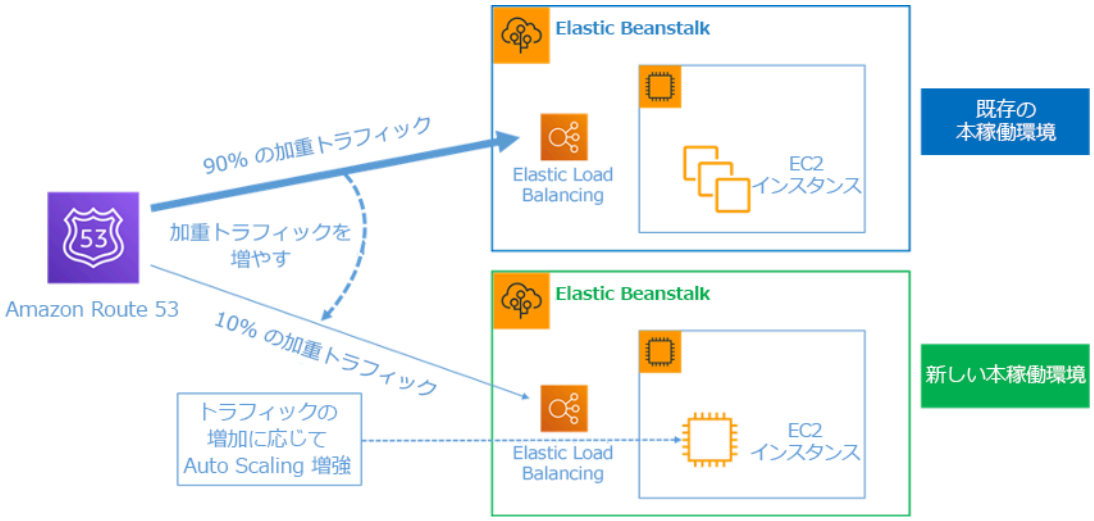


図 8: AWS Elastic Beanstalk と Amazon Route 53 によるブルー/グリーンデプロイ

イミュータブルインフラストラクチャの利点:

- 設定ドリフトの削減: サーバーをバージョン管理された既知の基本設定から頻繁に交換することにより、インフラストラクチャは既知の状態に リセットされ、 設定ドリフトを回避できます。
- 簡単なデプロイ: アップグレードをサポートする必要がないため、デプロイが簡素化されます。単に新たにデプロイすることが、アップグレードになります。
- 信頼性の高いアトミックデプロイ: デプロイは正常に完了するか、何も変更されません。デプロイプロセスの信頼性が高まります。
- 迅速なロールバックと復旧プロセスによる安全なデプロイ: 以前の作業バージョンは変更されないため、より安全にデプロイできます。エラーが検出された場合は、ロールバックできます。
- 一貫したテスト環境とデバッグ環境: すべてのサーバーが同じイメージを使用するため、環境間による違いはありません。1 つのビルドが複数の環境にデプロイされます。また、環境の整合性が失われるのを防ぎ、テストとデバッグが簡素化されます。

- スケーラビリティの向上: サーバーはベースイメージを使用し、一貫性があり、再現性があるため、とても簡単に自動スケーリングできます。
- 簡素化されたツールチェーン: 本番用ソフトウェアのアップグレードを管理する設定管理ツールが不要になるため、ツールチェーンが簡素化されます。サーバーにツールやエージェントが追加でインストールされることはありません。変更はベースイメージに加えられ、テストされてロールアウトされます。
- セキュリティの向上: サーバーへのすべての変更を拒否することで、インスタンスの SSH 接続を無効にし、キーを削除できます。これにより攻撃経路が減少し、組織のセキュリティ体制が向上します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- イミュータブルなインフラストラクチャを使用してデプロイします。イミュータブルインフラストラクチャは、更新、セキュリティパッチ、または設定の変更が本番システムで インプレースで 行われないように義務付けるモデルです。変更が必要な場合、新しいバージョンのアーキテクチャが構築され、本番環境にデプロイされます。
  - ブルー/グリーンデプロイの概要
  - サーバーレスアプリケーションの段階的なデプロイ
  - イミュータブルなインフラストラクチャ: 不変性を通じた信頼性、一貫性、確信
  - CanaryRelease

リソース

関連するドキュメント:

- CanaryRelease
- サーバーレスアプリケーションの段階的なデプロイ
- イミュータブルなインフラストラクチャ: 不変性を通じた信頼性、一貫性、確信
- ブルー/グリーンデプロイの概要
- The Amazon Builders' Library: デプロイ時におけるロールバックの安全性の確保

REL08-BP05 オートメーションを使用して変更をデプロイする

デプロイとパッチ適用は自動化されて、悪影響を排除します。

本番システムに変更を加えることは、多くの企業にとって最大級のリスクの 1 つです。当社は、ソフトウェアで解決するビジネス課題と同じくらい、デプロイを最優先課題としてとらえています。今日においてデプロイとは、変更のテストと導入、容量の追加と削除、データの移行など、実運用のあらゆる場所における自動化の導入を意味します。AWS CodePipeline により、ワークロードを解放するために必要なステップを管理できます。これには、AWS CodeDeploy を使用してデプロイされた状態が含まれ、Amazon EC2 インスタンス、オンプレミスインスタンス、サーバーレス Lambda 関数、または Amazon ECS サービスへのアプリケーションコードのデプロイを自動化します。

> 推奨
>
> 運用上の最も難しい手順に人を関与させることが一般通念で推奨されていますが、最も難しい手順については、まさにこの理由から自動化を推奨します。

一般的なアンチパターン:

- 手動で変更を実行する。
- 緊急ワークフローを通じて自動化の手順をスキップする。
- 計画に従わない。

このベストプラクティスを確立するメリット: 自動化を使用してすべての変更をデプロイすると、人為的ミスの発生の可能性がなくなり、本番環境を変更する前にテストして、計画が完了したことを確認できます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- デプロイパイプラインを自動化します。デプロイパイプラインを使用すると、自動テストおよび異常の検出を呼び出せるようになります。また、本番環境へのデプロイを行う前の特定のステップでパイプラインを休止したり、変更を自動的にロールバックしたりできます。
  - The Amazon Builders' Library: デプロイ時におけるロールバックの安全性の確保
  - The Amazon Builders' Library: 継続的デリバリーによる高速化
    - AWS CodePipeline (または信頼できるサードパーティ製品) を使用して、パイプラインを定義し実行します。
      - 変更がコードリポジトリにコミットされた時点で処理を開始するようにパイプラインを設定します。

    - AWS CodePipeline とは?
- Amazon Simple Notification Service (Amazon SNS) と Amazon Simple Email Service (Amazon SES) を使用して、パイプラインの問題に関する通知を送信するか、Amazon Chime などのチームチャットツールに統合します。
    - Amazon Simple Notification Service とは
    - Amazon SES とは?
    - Amazon Chime とは?
    - Webhook を使用してチャットメッセージを自動化する

リソース

関連するドキュメント:

- APN パートナー: 自動化されたデプロイソリューションの作成を支援できるパートナー
- AWS Marketplace: products that can be used to automate your deployments
- Webhook を使用してチャットメッセージを自動化する
- The Amazon Builders' Library: デプロイ時におけるロールバックの安全性の確保
- The Amazon Builders' Library: 継続的デリバリーによる高速化
- 「AWS CodePipeline とは何ですか?」
- 「What Is CodeDeploy?」
- AWS Systems Manager パッチマネージャーを使用して
- Amazon SES とは?
- Amazon Simple Notification Service とは

関連動画:

- AWS Summit 2019: CI/CD on AWS (AWS における CI/CD)

障害管理

質問

- REL 9 データはどのようにバックアップするのですか?
- REL 10 ワークロードを保護するために、障害分離をどのように使用すればよいですか?

- REL 11 コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?
- REL 12 信頼性はどのようにテストすればよいですか?
- REL 13 災害対策 (DR) はどのように計画するのですか?

REL 9 データはどのようにバックアップするのですか?

目標復旧時間 (RTO) と目標復旧時点 (RPO) の要件を満たすように、データ、アプリケーション、設定をバックアップします。

ベストプラクティス

- REL09-BP01 バックアップが必要なすべてのデータを特定し、バックアップする、またはソースからデータを再現する
- REL09-BP02 バックアップを保護し、暗号化する
- REL09-BP03 データバックアップを自動的に実行する
- REL09-BP04 データの定期的な復旧を行ってバックアップの完全性とプロセスを確認する

REL09-BP01 バックアップが必要なすべてのデータを特定し、バックアップする、またはソースからデータを再現する

すべての AWS データストアは、バックアップ機能を備えています。Amazon RDS や Amazon DynamoDB などのサービスは、ポイントインタイムリカバリ (PITR) を有効にする自動バックアップを追加でサポートします。これにより、現在時刻の 5 分前までの任意の時刻にバックアップを復元することができます。多くの AWS サービスは、バックアップを別の AWS リージョン にコピーする機能を備えています。AWS Backup は、複数の AWS のサービスにまたがってデータ保護を一元化し、自動化できるツールです。

Amazon S3 をセルフマネージドおよび AWS マネージドデータソースのバックアップ先として使用できます。Amazon EBS、Amazon RDS、Amazon DynamoDB などの AWS サービスには、バックアップを作成する機能が組み込まれています。サードパーティー製のバックアップソフトウェアも使用できます。

オンプレミスのデータを AWS クラウド にバックアップするには、 AWS Storage Gateway または AWS DataSyncを使用します。Amazon S3 バケットは、このデータを AWS に保存するために使用できます。Amazon S3 は、 Amazon S3 Glacier または S3 Glacier Deep Archive など複数のストレージ階層を提供して、データストレージのコストを削減します。

他のソースからデータを再生成することによって、データリカバリのニーズを満たすこともできます。例えば、 Amazon ElastiCache レプリカノード または RDS リードレプリカ を使用して、プライマリが失われた場合にデータを再生成できます。このようなソースを使用して、 目標復旧時点 (RPO) と目標復旧時間 (RTO)を満たすことができる場合、バックアップは不要です。別の例として、Amazon EMR を操作している場合、 S3 から EMR にデータを再生成できる限り、HDFS データストアをバックアップする必要はありません.

バックアップ戦略を選択するときには、データの復旧にかかる時間を考慮してください。データの復旧に必要な時間は、バックアップのタイプ (バックアップ戦略の場合) やデータ再生成メカニズムの複雑性に依存します。この時間は、ワークロードの RTO 以内でなければなりません。

期待される成果:

データソースが識別され、重要性に基づいて分類されている。次に、RPO に基づいてデータ復旧戦略を確立します。この戦略には、これらのデータソースをバックアップするか、他のソースからデータを再生成する能力を持つことが含まれます。データ損失の場合、実装された戦略によって、定義された RPO および RTO 内でのデータの復旧または再生成が可能になります。

クラウド成熟度フェーズ: Fondational

一般的なアンチパターン:

- ワークロードのすべてのデータソースとそれらの重要性を認識していない。
- 重要なデータソースのバックアップを取っていない。
- 重要性を評価基準として使用せずに、一部のデータソースのみのバックアップを取る。
- RPO が定義されていないか、バックアップの頻度が RPO を満たしていない。
- バックアップが必要かどうか、またはデータを他のソースから再生成できるかどうかを評価していない。

このベストプラクティスを確立するメリット: バックアップが必要な場所を特定し、バックアップを作成するメカニズムを実装するか、外部ソースからデータを再生成できるようにすることで、停止時にデータを復元し、復旧する能力が高まります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードが使用する AWS のサービスとリソースのバックアップ機能を理解し、使用します。ほとんどの AWS のサービスは、ワークロードデータをバックアップする機能を提供します。

実装手順:

1. ワークロードのすべてのデータソースを特定します.データは、データベース、 からの脱却io1 ボリュームio1 ファイルシステムio1 ロギングシステム、および オブジェクトストレージなど、多くのリソースに保存できます.ウェブアプリケーションのバックエンドに関する推奨事項については、 リソース セクションで、 関連するドキュメント から、データが保存されるさまざまな AWS のサービスと、これらのサービスが提供するバックアップ機能に関するものを参照してください。
2. 重要性に基づいてデータソースを分類します.データセットごとにワークロードにとっての重要度が異なるため、回復力の要件も異なります。例えば、一部のデータは重要であり、ゼロに近い RPO を必要とするかもしれませんが、その他のデータは重要度が低く、より高い RPO や部分的なデータ損失に耐えられるかもしれません。同様に、データセットごとに RTO 要件も異なります。
3. AWS またはサードパーティのサービスを使用して、データのバックアップを作成します. AWS Backup は、AWS にさまざまなデータソースのバックアップを作成できるマネージドサービスです。また、これらのサービスのほとんどは、バックアップを作成する機能をネイティブで備えています。AWS Marketplace には、これらの機能を提供する多数のソリューションも用意されています。ウェブアプリケーションのバックエンドに関する推奨事項については、 リソース 以下に記載されているリソースの中で、さまざまな AWS のサービスからデータのバックアップを作成する方法に関する情報を参照してください。
4. バックアップしないデータについては、データ再生成メカニズムを確立します.さまざまな理由から、他のソースから再生成できるデータはバックアップしないという選択をすることもあるでしょう。バックアップの保管にコストがかかるため、バックアップを作成するよりも、必要なときにソースからデータを再現したほうが安いという状況もあるかもしれません。別の例としては、バックアップからの復元にかかる時間が、ソースからデータを再生成するよりも長く、結果として RTO に反する場合です。このような状況では、トレードオフを考慮して、データ復旧が必要なときに、これらのソースからデータを再生成する方法について、十分に定義されたプロセスを確立してください。例えば、データの分析を行うために、Amazon S3 からデータウェアハウス (Amazon Redshift など) 、または MapReduce クラスター (Amazon EMR など) にデータをロードしてある場合、これは他のソースから再生成できるデータの例になるかもしれません。これらの分析の結果がどこかに保存されているか再現可能である限り、データウェアハウスまたは MapReduce クラスターで発生した障害でデータが失われることはありません。ソースから再現できる例には他にも、キャッシュ (Amazon ElastiCache など) や RDS リードレプリカなどがあります。
5. データをバックアップするサイクルを確立します.データソースのバックアップの作成は定期的なプロセスであり、頻度は RPO に依存します。

実装計画の工数レベル: 中

リソース

関連するベストプラクティス:

REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する

REL13-BP02 復旧目標を満たすため、定義された復旧戦略を使用する

関連するドキュメント:

- 「AWS Backup とは。」
- AWS DataSync とは?
- ボリュームゲートウェイとは?
- APN パートナー: バックアップを支援できるパートナー
- AWS Marketplace: バックアップに活用できる製品
- Amazon EBS スナップショット
- Amazon EFS のバックアップ
- Amazon FSx for Windows File Server のバックアップ
- ElastiCache for Redis のバックアップと復元
- Creating a DB Cluster Snapshot in Neptune (Neptune での DB クラスタースナップショットの作成)
- DB スナップショットの作成
- スケジュールに従って実行する EventBridge ルールの作成
- クロスリージョンレプリケーション と Amazon S3
- EFS-to-EFS AWS Backup
- Amazon S3 へのログデータのエクスポート
- オブジェクトのライフサイクル管理
- On-Demand Backup and Restore for DynamoDB (DynamoDB のオンデマンドバックアップと復元)
- DynamoDB のポイントインタイムリカバリ
- Working with Amazon OpenSearch Service Index Snapshots (Amazon OpenSearch Service インデックススナップショットの操作)

関連動画:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS (AWS によるバックアップ、ディザスタリカバリ、ランサムウェア保護)
- AWS Backup Demo: Cross-Account and Cross-Region Backup (デモ: クロスアカウントおよびクロスリージョンバックアップ)
- AWS re:Invent 2019: Deep dive on AWS Backup (AWS Backup の深掘り), ft.Rackspace (STG341)

関連する例:

- Well-Architected ラボ: Amazon S3 の双方向クロスリージョンレプリケーション (CRR) の実装
- Well-Architected ラボ: データのバックアップと復元のテスト
- Well-Architected ラボ: 分析ワークロードのフェイルバックによるバックアップと復元
- Well-Architected ラボ: ディザスタリカバリ - バックアップと復元

REL09-BP02 バックアップを保護し、暗号化する

AWS IAM などの認証と承認を使用して、バックアップへのアクセスを制御し、検出します。暗号化によりバックアップのデータ保全性が損なわれることを防止、検出します。

Amazon S3 は、保管時のデータを暗号化するための方法をいくつかサポートしています。Amazon S3 はサーバー側の暗号化を使用して、オブジェクトを暗号化されていないデータとして受け入れてから、保存時に暗号化します。クライアント側の暗号化を使用すると、ワークロードアプリケーションはデータを Amazon S3 に送信する前に暗号化することに対して責任を負います。どちらの方法でも、AWS Key Management Service (AWS KMS) を使ってデータキーを作成して保存することもできますし、自分でキーを用意し、そのキーに責任を持つこともできます。AWS KMS を使用すると、IAM を使用してポリシーを設定し、データキーと復号化されたデータにアクセスできるユーザーとアクセスできないユーザーにわけることができます。

Amazon RDS では、データベースの暗号化を選択すると、バックアップも暗号化されます。DynamoDB のバックアップは常に暗号化されます。

一般的なアンチパターン:

- データに対するのと同一の、バックアップおよび復元オートメーションへのアクセスを設定する。
- バックアップを暗号化しない。

このベストプラクティスを確立するメリット: バックアップを保護することで、データの改ざんを防止し、データの暗号化により、誤って公開されたデータへのアクセスが防止されます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- 各データストアで暗号化を使用します。ソースデータが暗号化されている場合、バックアップも暗号化されます。
  - RDS での暗号化を有効にします。RDS インスタンスの作成時に、AWS Key Management Service を使用して、保管時の暗号化を設定できます。
    - Amazon RDS リソースの暗号化
  - EBS ボリュームの暗号化を有効にします。デフォルトの暗号化を設定するか、ボリュームの作成時に一意のキーを指定できます。
    - Amazon EBS 暗号化
  - 必要な Amazon DynamoDB 暗号化を使用します。DynamoDB は保管中のすべてのデータを暗号化します。AWS 所有の AWS KMS キーを使用するか、AWS マネージド KMS キーを使用して、アカウントに保存されるキーを指定できます。
    - 保管時の DynamoDB 暗号化
    - 暗号化されたテーブルの管理
  - Amazon EFS に保存されているデータを暗号化します。ファイルシステムを作成するときに暗号化を設定します。
    - EFS でのデータとメタデータの暗号化
  - 送信元と送信先のリージョンで暗号化を設定します。KMS に保存されているキーを使用して Amazon S3 で保管時の暗号化を設定できますが、キーはリージョン固有です。レプリケーションを設定するときに、送信先キーを指定できます。
    - CRR 追加設定: AWS KMS に保存された暗号化キーを使用したサーバー側の暗号化 (SSE) で作成されたオブジェクトをレプリケートする
- バックアップにアクセスするための最小特権のアクセス許可を実装します。セキュリティのベストプラクティスに従って、バックアップ、スナップショット、およびレプリカへのアクセスを制限します。
  - セキュリティの柱: AWS Well-Architected

リソース

関連するドキュメント:

- AWS Marketplace: バックアップに活用できる製品

- Amazon EBS 暗号化
- Amazon S3: 暗号化を使用しデータを保護する
- CRR 追加設定: AWS KMS に保存された暗号化キーを使用したサーバー側の暗号化 (SSE) で作成されたオブジェクトをレプリケートする
- 保管時の DynamoDB 暗号化
- Amazon RDS リソースの暗号化
- EFS でのデータとメタデータの暗号化
- AWS でのバックアップの暗号化
- 暗号化されたテーブルの管理
- セキュリティの柱: AWS Well-Architected

関連する例:

- Well-Architected ラボ: Amazon S3 の双方向クロスリージョンレプリケーション (CRR) の実装

REL09-BP03 データバックアップを自動的に実行する

目標復旧時点 (RPO) によって、またはデータセット内の変更によって通知される定期的なスケジュールに基づいて、バックアップが自動的に行われるように設定します。データ損失の少ない重要なデータセットは頻繁に自動バックアップする必要がありますが、多少の損失は許容できる重要度の低いデータは、バックアップの頻度を少なくすることができます。

AWS Backup を使用して、さまざまな AWS データソースの自動化されたデータバックアップを作成できます。Amazon RDS インスタンスは 5 分ごとにほぼ連続的にバックアップでき、Amazon S3 オブジェクトは 15 分ごとにほぼ連続的にバックアップできます。これにより、バックアップ履歴内の特定の時点へのポイントインタイムリカバリ (PITR) が可能になります。Amazon EBS ボリューム、Amazon DynamoDB テーブル、Amazon FSx ファイルシステムなど、その他の AWS データソースについては、AWS Backup は 1 時間ごとの頻度で自動バックアップを実行できます。これらのサービスでは、バックアップ機能もネイティブに提供されています。ポイントインタイムリカバリを備えた自動バックアップを提供する AWS サービスとしては、 Amazon DynamoDBio1 Amazon RDS、および Amazon Keyspaces (Apache Cassandra 向け) があります。これらはバックアップ履歴内の特定の時点への復元が可能です。その他の AWS データストレージサービスのほとんどが、1 時間ごとの定期バックアップをスケジュールする機能を提供しています。

Amazon RDS と Amazon DynamoDB は、ポイントインタイムリカバリを使用した継続的なバックアップを提供しています。Amazon S3 バージョニングは、一度有効にすると、自動で実行されま

す。Amazon Data Lifecycle Manager を使用して、Amazon EBS スナップショットの作成、コピー、および削除を自動化できます。また、Amazon EBS-backed Amazon マシンイメージ (AMI) とその基盤となる Amazon EBS スナップショットの作成、コピー、廃止、および登録解除も自動化できます。

バックアップの自動化と履歴を一元的に確認できるようにするために、AWS Backup は完全マネージド型の、ポリシーベースのバックアップソリューションを提供します。AWS Storage Gateway を使用して、クラウド内およびオンプレミスの複数の AWS のサービスにわたってデータのバックアップを一元化および自動化します。

バージョン管理に加えて、Amazon S3 はレプリケーション機能も備えています。S3 バケット全体を同じまたは異なる AWS リージョン にある別のバケットに自動的にレプリケートできます。

期待される成果:

一定の周期でデータソースのバックアップを作成する自動化されたプロセス。

一般的なアンチパターン:

- バックアップを手動で実行する。
- バックアップ機能があるが、自動化にバックアップが含まれていないリソースを使用する。

このベストプラクティスを確立するメリット: バックアップを自動化することで、バックアップが RPO に基づいて定期的に作成され、作成されない場合はアラートが送信されます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

1. 現在、手動でバックアップされている データソースを特定します。AWS に ANF サーバーを構築するための詳細については、 REL09-BP01 バックアップが必要なすべてのデータを特定し、バックアップする、またはソースからデータを再現する これに関するガイダンスを参照してください。
2. ワークロードの RPO を 決定します。AWS に ANF サーバーを構築するための詳細については、REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する これに関するガイダンスを参照してください。
3. 自動バックアップソリューションまたはマネージドサービスを使用します。AWS Backup はフルマネージドサービスであり、 AWS のサービス、クラウド、およびオンプレミスでのデータ保護の一元化と自動化を容易にします.バックアッププランは AWS Backup の機能であり、バックアッ

プするリソースと、これらのバックアップを作成する頻度を定義するルールの作成を可能にします。この頻度は、ステップ 2 で確立した RPO によって通知される必要があります。 この WA ラボは、 AWS Backup を使用して自動バックアップを作成する方法に関するハンズオンガイダンスを提供します。データを保存するほとんどの AWS サービスでは、バックアップ機能がネイティブで提供されています。例えば、RDS は、ポイントインタイムリカバリ (PITR) 付きの自動バックアップに利用できます。

4. オンプレミスデータソースやメッセージキューなど、自動バックアップソリューションまたはマネージドサービスによってサポートされないデータソースについては、 信頼できるサードパーティソリューションを使用して自動バックアップを作成することを検討してください。または、AWS CLI または SDK を使用してこれを行うオートメーションを作成することができます。AWS Lambda 関数または AWS Step Functions を使用して、データバックアップの作成にかかわるロジックを定義し、Amazon EventBridge を使用して RPO (ステップ 2 で確立) に基づく頻度で実行することができます。

実装計画の工数レベル: 低

リソース

関連するドキュメント:

- APN パートナー: バックアップを支援できるパートナー
- AWS Marketplace: バックアップに活用できる製品
- スケジュールに従って実行する EventBridge ルールの作成
- 「AWS Backup とは。」
- 「AWS Step Functions とは何ですか?」

関連動画:

- AWS re:Invent 2019: Deep dive on AWS Backup (AWS Backup の深掘り), ft.Rackspace (STG341)

関連する例:

- Well-Architected ラボ: データのバックアップと復元のテスト

REL09-BP04 データの定期的な復旧を行ってバックアップの完全性とプロセスを確認する

復旧テストを実行して、バックアッププロセスの実装が目標復旧時間 (RTO) と目標復旧時点 (RPO) を満たしていることを検証します。

AWS を使用して、テスト環境を立ち上げ、そこにバックアップを復元して RTO および RPO が機能するかを評価し、データコンテンツと完全性のテストを実行できます。

さらに、Amazon RDS および Amazon DynamoDB はポイントインタイムリカバリ (PITR) を許可します。継続的バックアップを使用すると、データセットを指定された日時の状態に復元できます。

期待される成果: バックアップからのデータを、十分に定義されたメカニズムを使用して定期的に復旧することで、ワークロードについて確立された目標復旧時間 (RTO) 内での復旧が可能であることを確認できます。バックアップからの復元により、オリジナルデータを含むリソースになり、破損したりアクセス不能になっていたりするデータがなく、目標復旧時点 (RPO) 内のデータ損失であることを確認します。

一般的なアンチパターン:

- バックアップを復元しますが、復元が使用可能であることを確認するためのデータのクエリや取得は行いません。
- バックアップが存在することを前提とする。
- システムのバックアップが完全に動作可能であり、そこからデータを復旧できることを前提とする。
- バックアップからデータを復元または復旧する時間がワークロードの RTO の範囲内であることを前提とする。
- バックアップに含まれるデータがワークロードの RPO の範囲内であることを前提とする。
- ランブックを使用せずに、または確立された自動手順の外部で、アドホックに復元する。

このベストプラクティスを活用するメリット: バックアップの復旧をテストすることで、データの紛失や破損を心配せずに、必要なときにデータを復元できること、ワークロードの RTO 内で復元と復旧が可能であること、ならびにデータ損失がワークロードの RPO の範囲内であることを確認できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

バックアップおよび復元機能をテストすることで、停止時にこれらのアクションを実行できるという安心感が得られます。定期的にバックアップを新しい場所に復元して、テストを実行し、データの完全性を確認します。実行すべき一般的なテストは、

すべてのデータが使用可能であり、破損しておらず、アクセス可能であり、データ損失がワークロードの RPO の範囲内であることを確認します。そのようなテストは、復旧メカニズムが十分に高速であり、ワークロードの RTO に対応できることを確認するのにも役立ちます。

1. 現在、手動でバックアップされている データソースと、これらのバックアップの保存場所を特定します。AWS に ANF サーバーを構築するための詳細については、 REL09-BP01 バックアップが必要なすべてのデータを特定し、バックアップする、またはソースからデータを再現する この実装方法に関するガイダンスを参照してください。
2. 各データソースの データ検証基準を確立します。データのタイプが異なると、プロパティも異なり、異なる検証メカニズムが必要になることがあります。本番環境での使用を決定する前に、このデータを検証する方法を考慮してください。データを検証するための一般的な方法としては、データタイプ、フォーマット、チェックサム、サイズ、またはこれらの組み合わせなど、データとバックアッププロパティをカスタム検証ロジックで使用することです。例えば、復元されたリソースと、バックアップが作成された時点でのデータソースの間でチェックサム値を比較します。
3. データの重要度に基づいて、データ復元の RTO と RPO を確立します。AWS に ANF サーバーを構築するための詳細については、 REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する この実装方法に関するガイダンスを参照してください。
4. データ復旧機能を評価します.バックアップおよび復元戦略をレビューして、RTO および RPO を満たせるかどうかを理解し、必要に応じて戦略を調整します。分析のために、 AWS レジリエンスハブを使用して、ワークロードのアセスメントを実行できます。アセスメントは、回復力ポリシーに対してアプリケーション設定を評価し、RTO および RPO 目標を満たすことができるかどうかを報告します。
5. 本番環境で現在使われている確立されたデータ復元プロセスを使用して、 テスト復元を行います。これらのプロセスは、オリジナルデータソースのバックアップ方法、バックアップそのもののフォーマットとストレージ場所、またはデータが他のソースから再生成されるかどうかによって異なります。例えば、 AWS Backup などのマネージドサービスを使用している場合、バックアップを新しいリソースに容易に復元できます.AWS Elastic Disaster Recovery を使用した場合、リカバリドリルを開始できます.
6. (前のステップから) 復元されたリソースからのデータリカバリを ステップ 2 でデータ検証のために確立した基準に基づいて検証します。復元され、復旧されたデータには、バックアップ時点で

最新のレコード/アイテムが含まれていますか? このデータはワークロードの RPO の範囲内ですか?

7. 復元と復旧に必要な時間を測定して、 ステップ 3 で確立された RTO と比較します。このプロセスは、ワークロードの RTO の範囲内ですか? 例えば、復元プロセスが開始されたときのタイムスタンプと復旧検証が完了したときのタイムスタンプを比較して、このプロセスの所要時間を計算します。すべての AWS API 呼び出しにはタイムスタンプが付けられるため、この情報を使用できます AWS CloudTrail.この情報から復元プロセスが開始したときの詳細がわかりますが、検証が完了したときの終了タイムスタンプが検証ロジックによって記録される必要があります。自動化プロセスを使用している場合、 Amazon DynamoDB などのサービスを使用して、この情報を保存できます。さらに、多くの AWS のサービスは、特定のアクションが発生したときのタイムスタンプ付きの情報を提供するイベント履歴を備えています。AWS Backup 内では、バックアップおよび復元アクションは ジョブと呼ばれ、これらのジョブにはメタデータの一部としてタイムスタンプ情報が含まれ、復元と復旧の所要時間の測定に使用できます。
8. データ検証に失敗した場合や、 復元と復旧に必要な時間がワークロードについて確立された RTO を超えている場合は、関係者に通知します。これを行うオートメーションを実装するときには、 このラボのように、Amazon Simple Notification Service (Amazon SNS) などのサービスを使用して、メールや SMS などで関係者にプッシュ通知を送信できます。 これらのメッセージは、Amazon Chime、Slack、Microsoft Teams などのメッセージングアプリケーションに発行したり、 または AWS Systems Manager OpsCenter を使用して OpsItems としてタスクを作成するために使用したりすることもできます.
9. このプロセスを定期的に実行するように自動化します.例えば、AWS Lambda や AWS Step Functions の状態マシンなどのサービスを使用して、復元および復旧プロセスを自動化でき、Amazon EventBridge を使用して、下のアーキテクチャ図に示されているように、このオートメーションワークフローを定期的にトリガーすることができます。データ復旧検証を AWS Backup で自動化する方法について確認してください.さらに、 この Well-Architected ラボは、 いくつかのステップを自動化するための 1 つの方法について、ハンズオンエクスペリエンスを提供します。

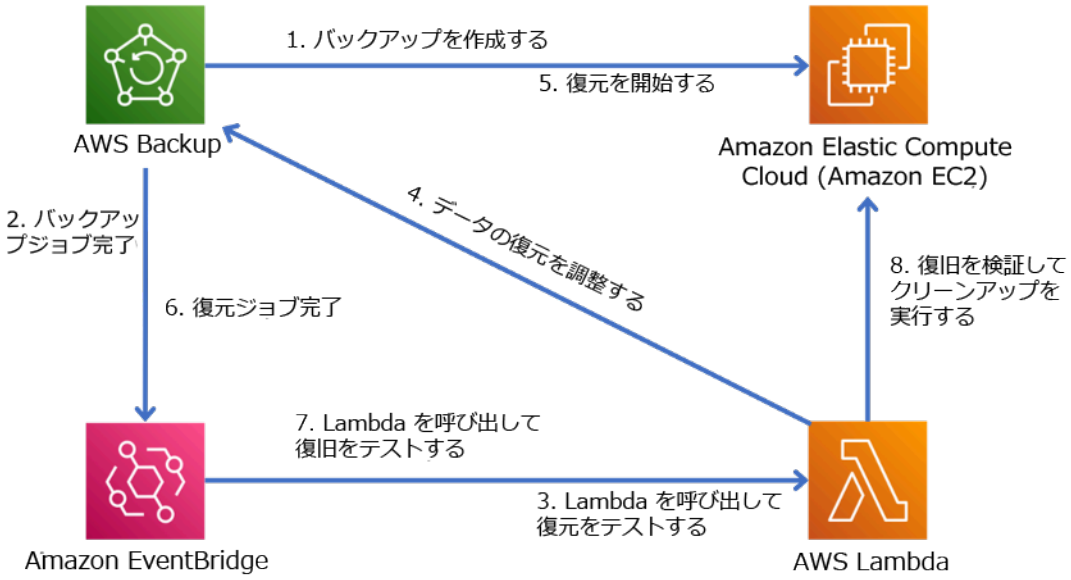


図 9.自動化されたバックアップおよび復元プロセス

実装計画の工数レベル: 検証基準の複雑性に応じて、中 ~ 高。

リソース

関連するドキュメント:

- AWS Backup でデータ復旧検証を自動化する
- APN パートナー: バックアップを支援できるパートナー
- AWS Marketplace: バックアップに活用できる製品
- スケジュールに従ってトリガーする EventBridge ルールの作成
- DynamoDB のオンデマンドバックアップと復元
- 「AWS Backup とは。」
- 「AWS Step Functions とは何ですか?」
- AWS Elastic Disaster Recovery とは
- AWS Elastic Disaster Recovery

関連する例:

- Well-Architected ラボ: データのバックアップと復元のテスト

REL 10 ワークロードを保護するために、障害分離をどのように使用すればよいですか?

障害部分を分離した境界は、ワークロード内の障害の影響を限られた数のコンポーネントに限定します。境界外のコンポーネントは、障害の影響を受けません。障害部分を切り離した境界を複数使用すると、ワークロードへの影響を制限できます。

ベストプラクティス

- REL10-BP01 複数の場所にワークロードをデプロイする
- REL10-BP02 マルチロケーションデプロイ用の適切な場所の選択
- REL10-BP03 単一のロケーションに制約されるコンポーネントのリカバリを自動化する
- REL10-BP04 バルクヘッドアーキテクチャを使用して影響範囲を制限する

REL10-BP01 複数の場所にワークロードをデプロイする

ワークロードのデータとリソースを複数のアベイラビリティゾーンに分散するか、必要に応じて複数の AWS リージョン にまたがって分散します。これらのロケーションは、必要に応じて多様化できます。

AWS のサービス設計の基本原則の 1 つは、基盤となる物理インフラストラクチャの単一障害点を回避することです。これによって、複数のアベイラビリティーゾーンを使用して単一ゾーンで起こる障害に耐えられるソフトウェアおよびシステムを構築することができます。これと同様に、システムは単一のコンピューティングノード、単一のストレージボリューム、単一のデータベースインスタンスの障害に対する弾力性を持つように設計されています。冗長コンポーネントに依存するシステムを構築するときには、それぞれのコンポーネントが独立して動作すること、また、AWS リージョン の場合は、それぞれのリージョンが自律して動作することが重要です。冗長化されたコンポーネントを使用した理論的な可用性の計算から得られるメリットは、これが当てはまる場合にのみ有効です。

アベイラビリティゾーン (AZ)

AWS リージョン は、互いに独立するように設計された 2 つ以上のアベイラビリティゾーンで構成されます。各アベイラビリティゾーンは、火災、洪水、竜巻などの自然災害による障害の影響を回避するため、ほかのゾーンから物理的に大きな距離を隔てています。各アベイラビリティゾーンは、商用電源への専用接続、スタンドアロンのバックアップ電源、独立したメカニカルサービス、アベイラビリティゾーン内外の独立したネットワーク接続など、独立した物理インフラストラクチャも備えています。この設計により、これらのシステムのいずれかに障害が発生した場合、影響を受ける AZ は 1 だけで済みます。地理的には分離されていても、アベイラビリティゾーンは、同じリージョ

ンのエリアに配置されているため、高スループットで低レイテンシーなネットワーク接続が可能です。AWS リージョン 全体 (すべてのアベイラビリティゾーンにまたがり、複数の物理的に独立したデータセンターで構成される) をワークロードの単一の論理的なデプロイ先として扱うことができ、同期したデータレプリケーション (例えば、データベース間で) が可能です。このようにして、アクティブ/アクティブまたはアクティブ/スタンバイの設定でアベイラビリティゾーンを使用することができます。

アベイラビリティゾーンは独立しているため、ワークロードが複数のゾーンを使用するように設定された場合、ワークロードの可用性が向上します。一部の AWS サービス (Amazon EC2 インスタンスデータプレーンを含む) は、厳密なゾーンサービスとしてデプロイされるため、属するアベイラビリティゾーンに左右されます。ただし、他の AZ 内の Amazon EC2 インスタンスは影響を受けず、機能し続けます。同様に、アベイラビリティゾーンの障害によって Amazon Aurora データベースに障害が発生した場合、影響を受けない AZ のリードレプリカ Aurora インスタンスは自動的にプライマリに昇格できます。一方、Amazon DynamoDB などのリージョンにおける AWS のサービスは、サービスの可用性の設計目標を達成するために、内部ではアクティブ/アクティブ設定で複数のアベイラビリティゾーンを使用するため、AZ 配置を設定する必要はありません。

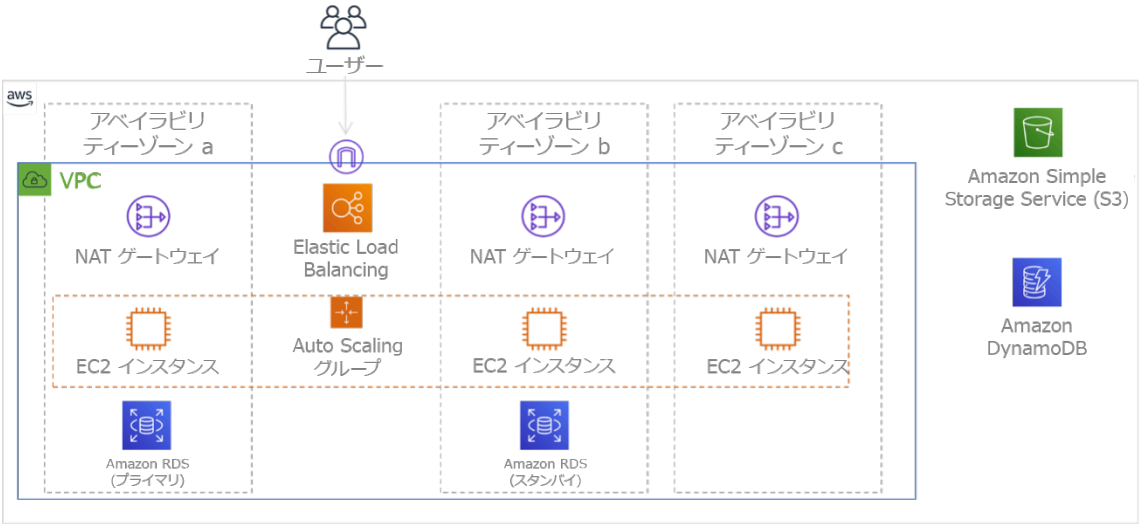


図 9: 3 つのアベイラビリティーゾーンにまたがってデプロイされる多階層アーキテクチャ。Amazon S3 と Amazon DynamoDB は常に自動的にマルチ AZ であることに注意してください。ELB も 3 つのゾーンすべてにデプロイされます。

AWS コントロールプレーンは、通常、リージョン全体 (複数のアベイラビリティゾーン) 内のリソースを管理する機能を提供しますが、特定のコントロールプレーン (Amazon EC2 および Amazon EBS を含む) は、結果を単一のアベイラビリティゾーンにフィルタリングする機能を備えています。これを実行すると、指定されたアベイラビリティーゾーン内でのみリクエストが処理されるため、その他のアベイラビリティーゾーンで起こる障害からの影響を軽減できます。この AWS CLI の例は、us-

east-2c アベイラビリティゾーンからのみ Amazon EC2 インスタンス情報を取得する例を示しています。

```
AWS ec2 describe-instances --filters Name=availability-zone,Values=us-east-2c
```

AWS ローカルゾーン

AWS ローカルゾーンは、サブネットや EC2 インスタンスなど、ゾーン内の AWS リソースの配置場所として選択できるという点で、それぞれの AWS リージョン リージョン内でアベイラビリティゾーンと同じように機能します。それらが特別なのは、関連する AWS リージョン リージョンにあるのではなく、現在 AWS リージョン が存在しない大規模な人口、産業、IT センターの近くにあることです。それでも、ローカルゾーンのローカルワークロードと AWS リージョン で実行されているワークロードの間で、高帯域幅で安全な接続を維持します。ワークロードをユーザーに近い場所にデプロイし、低レイテンシー要件を満たすには、AWS ローカルゾーンを使用する必要があります。

Amazon Global Edge Network

Amazon Global Edge Network は、世界中の都市のエッジロケーションで構成されます。Amazon CloudFront は、このネットワークを使用して、コンテンツをエンドユーザーに低レイテンシーで配信します。AWS Global Accelerator を使用すると、これらのエッジロケーションにワークロードエンドポイントを作成して、ユーザーに近い AWS グローバルネットワークへのオンボーディングを提供できます。Amazon API Gateway は、CloudFront ディストリビューションを使用してエッジ最適化された API エンドポイントを有効にし、最も近いエッジロケーションを経由してクライアントアクセスを容易にします。

AWS リージョン

AWS リージョン は自律するように設計されているため、マルチリージョンアプローチを使用するには、各リージョンにサービスの専用コピーをデプロイします。

マルチリージョンアプローチは、1 回限りの大規模なイベントが発生したときに復旧目標を満たすための ディザスタリカバリ 戦略に一般的です。把握 災害対策 (DR) を計画する これらの戦略の詳細について確認してください。ただし、ここでは、 可用性に焦点を当てて、平均アップタイム目標の実現を目指します。高可用性目標については、マルチリージョンアーキテクチャは、一般に、アクティブ/アクティブとして設計され、(それぞれのリージョンの) 各サービスコピーはアクティブです (リクエストを処理します)。

推奨

ほとんどのワークロードの可用性目標は、単一の AWS リージョン 内でマルチ AZ 戦略を使用して満たすことができます。マルチリージョンアーキテクチャは、ワークロードの可用性要件が極端に高いか、その他のビジネス目標のために、マルチリージョンアーキテクチャが必要とされる場合のみ検討してください。

AWS は、お客様がリージョンをまたいでサービスを運用できるようにしています。例えば、AWS は、Amazon Simple Storage Service (Amazon S3) レプリケーション、Amazon RDS リードレプリカ (Aurora リードレプリカを含む)、Amazon DynamoDB グローバルテーブルを使用して、連続的な非同期データレプリケーションを提供します。連続レプリケーションでは、アクティブリージョンのそれぞれでデータのバージョンをほとんどすぐに使用できます。

AWS CloudFormation を使用して、インフラストラクチャを定義し、複数の AWS アカウント と複数の AWS リージョン にまたがって一貫してデプロイできます。また、AWS CloudFormation StackSets は、この機能を拡張し、複数のアカウントとリージョンにまたがって単一のオペレーションで AWS CloudFormation スタックの作成、更新、または削除が可能です。Amazon EC2 インスタンスのデプロイの場合、AMI (Amazon マシンイメージ) は、ハードウェア設定やインストールされたソフトウェアなどの情報を提供するために使用されます。Amazon EC2 Image Builder パイプラインを実装して、必要な AMI を作成し、これらをアクティブリージョンにコピーできます。これにより、これらの ゴールデン AMI は、新しい各リージョンにワークロードをデプロイし、スケールアウトするために必要なすべてを備えることになります。

トラフィックをルーティングするために、Amazon Route 53 と AWS Global Accelerator の両方により、どのユーザーをどのアクティブリージョンエンドポイントに向かわせるかを決定するポリシーを定義できます。Global Accelerator では、トラフィックダイヤルを設定して、各アプリケーションエンドポイントに向かうトラフィックのパーセンテージを制御します。Route 53 は、このパーセンテージアプローチをサポートするほか、地理的近接性やレイテンシーに基づくものなど、他の複数の可用性ポリシーもサポートします。Global Accelerator は、AWS エッジサーバーの拡張ネットワークを自動的に利用して、AWS ネットワークバックボーンへのトラフィックを可能な限りすぐにオンボードするため、リクエストのレイテンシーが低下します。

これらの機能はすべて、各リージョンの自律性を保つように動作します。このアプローチには、ごくわずかな例外があり、AWS Identity and Access Management (IAM) サービスのコントロールプレーンと共に、グローバルエッジデリバリーを提供するサービス (Amazon CloudFront や Amazon Route 53 など) が含まれます。大部分のサービスが、完全に単一リージョン内で運用されています。

オンプレミスのデータセンター

オンプレミスのデータセンターで実行されるワークロードに関しては、可能な場合はハイブリッドエクスペリエンスを設計します。AWS Direct Connect は、オンプレミスから AWS への専用ネットワーク接続を提供し、両方での実行が可能になります。

もう 1 つのオプションは、AWS Outposts を使用して、AWS インフラストラクチャとサービスをオンプレミスで実行することです。AWS Outposts は、AWS インフラストラクチャ、AWS のサービス、API、ツールをデータセンターに拡張する完全マネージド型サービスです。AWS クラウドで使用されているのと同じハードウェアインフラストラクチャがデータセンターにインストールされます。その後、AWS Outposts は最も近い AWS リージョン に接続されます。その結果、AWS Outposts を使用して、低レイテンシーまたはローカルデータ処理を要求されるワークロードをサポートできます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 複数のアベイラビリティゾーンと AWS リージョン を使用します。ワークロードのデータとリソースを複数のアベイラビリティゾーンに分散するか、必要に応じて複数の AWS リージョン にまたがって分散します。これらの口ケーションは、必要に応じて多様化できます。
  - リージョンサービスは初めからアベイラビリティーゾーンにデプロイされます。
    - これには、Amazon S3、Amazon DynamoDB、AWS Lambda (VPC に接続されていない場合) が含まれます。
  - コンテナ、インスタンス、機能ベースのワークロードを複数のアベイラビリティーゾーンにデプロイします。キャッシュを含め、マルチゾーンデータストアを使用します。EC2 Auto Scaling、ECS タスク配置、AWS Lambda 関数の設定 (VPC で実行するとき)、および ElastiCache クラスターの機能を使用します。
    - Auto Scaling グループをデプロイするときには、アベイラビリティーゾーンの異なるサブネットを使用します。
      - 例: 複数のアベイラビリティーゾーンにインスタンスを分散する
      - Amazon ECS タスク配置戦略
      - Amazon VPC 内のリソースにアクセスできるように AWS Lambda 関数を設定する
      - リージョンとアベイラビリティーゾーンを選択する
    - Auto Scaling グループをデプロイするときには、アベイラビリティーゾーンの異なるサブネットを使用します。
      - 例: 複数のアベイラビリティーゾーンにインスタンスを分散する

    - タスク配置パラメータを使用して、DB サブネットグループを指定します。
        - Amazon ECS タスク配置戦略
    - VPC で実行する関数を設定するには、複数のアベイラビリティーゾーンでサブネットを使用します。
        - Amazon VPC 内のリソースにアクセスできるように AWS Lambda 関数を設定する
    - ElastiCache クラスターには複数のアベイラビリティーゾーンを使用します。
        - リージョンとアベイラビリティーゾーンを選択する
- ワークロードを複数のリージョンにデプロイする必要がある場合は、マルチリージョン戦略を選択します。信頼性に関するほとんどのニーズは、マルチアベイラビリティゾーン戦略を使用して単一の AWS リージョン 内で満たすことができます。必要に応じて、ビジネスニーズを満たすためにマルチリージョン戦略を使用します。
    - AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
        - 別の AWS リージョン にバックアップすると、必要なときにデータが利用可能になるという保証のレイヤーを追加できます。
        - ワークロードによっては、マルチリージョン戦略の使用を必要とする規制要件があります。
- ワークロードの AWS Outposts を評価します。ワークロードでオンプレミスのデータセンターへの低レイテンシーが必要な場合、またはローカルのデータ処理要件がある場合があります。その場合、AWS Outposts を使用して、オンプレミスで AWS インフラストラクチャとサービスを実行します。
    - 「What is AWS Outposts?」
- AWS Local Zones がユーザーにサービスを提供するのに役立つかどうかを判断します。低レイテンシー要件がある場合は、AWS Local Zones がユーザーの近くにあるかどうかを確認してください。近くにある場合は、それらのユーザーの近くにワークロードをデプロイするのに使用します。
    - AWS Local Zones のよくある質問

リソース

関連するドキュメント:

- AWS グローバルインフラストラクチャ
- AWS Local Zones のよくある質問
- Amazon ECS タスク配置戦略
- リージョンとアベイラビリティーゾーンを選択する

- 例: 複数のアベイラビリティーゾーンにインスタンスを分散する
- グローバルテーブル: DynamoDB を使用したマルチリージョンレプリケーション
- Amazon Aurora グローバルデータベースの使用
- AWS のサービスによるマルチリージョンアプリケーションの作成 (ブログシリーズ)
- 「What is AWS Outposts?」

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure (NET339)

REL10-BP02 マルチロケーションデプロイ用の適切な場所の選択

期待される成果

高可用性のためには、(可能なときには) 常に、図 10 に示されているように、ワークロードコンポーネントを複数のアベイラビリティゾーン (AZ) にデプロイします。回復力要件が極端に高いワークロードについては、マルチリージョンアーキテクチャのオプションを慎重に評価してください。

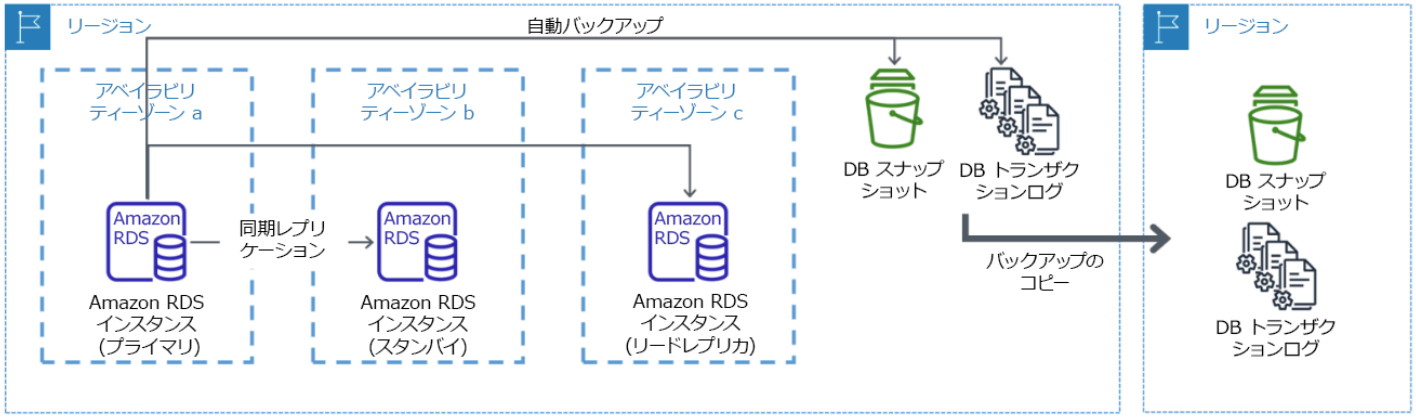


図 10: 別の AWS リージョンへのバックアップを備えた回復力の高いマルチ AZ データベースデプロイ

一般的なアンチパターン:

- マルチ AZ アーキテクチャで要件を満たせるときに、マルチリージョンアーキテクチャの設計を選択する。

- 回復力要件とマルチロケーション要件がアプリケーションコンポーネント間で異なる場合に、コンポーネント間の依存関係を考慮しない。

このベストプラクティスを活用するメリット:

回復力のためには、複数の防御層を構築するアプローチを使用する必要があります。1 つの層では、複数の AZ を使用して高可用性アーキテクチャを構築することによって、小規模で一般的な混乱に対して保護します。もう 1 つの防御層では、広範囲の自然災害やリージョンレベルの混乱など、まれな出来事に対して保護します。この 2 番目の層には、複数の AWS リージョン にまたがるアプリケーションの設計が必要です。

- 99.5% の可用性と 99.9% の可用性の違いは、1 か月あたり 3.5 時間に相当します。複数の AZ で期待されるワークロードの可用性を達成するには、99.99% が必要です。
- 複数の AZ でワークロードを実行することにより、電力、冷却、ネットワークの障害、および火災や洪水などの自然災害のほとんどを分離できます。
- ワークロードのためのマルチリージョン戦略の実装は、国の広い範囲に影響を与える自然災害や、リージョン全体に及ぶ技術的障害に対する保護に役立ちます。マルチリージョンアーキテクチャの実装はかなり複雑になることがあり、通常、ほとんどのワークロードには不要である点に注意してください。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

1 つのアベイラビリティゾーンの混乱または部分的損失に基づく災害イベントについては、単一の AWS リージョン 内の複数のアベイラビリティゾーンで高可用性ワークロードを実装すると、自然災害または技術的な災害の軽減に役立ちます。各 AWS リージョン は複数のアベイラビリティゾーンで構成され、各ゾーンは他のゾーンの障害から隔離され、かなりの距離によって分離されます。ただし、相互にかなりの距離を隔てた複数のアベラビリティゾーンコンポーネントの損失リスクがある災害については、リージョン規模の障害を緩和するディザスタリカバリオプションを実装する必要があります。極端に高い回復力を必要とするワークロードについては (重要なインフラストラクチャ、医療関連のアプリケーション、財務システムインフラストラクチャなど)、マルチリージョン戦略が必要なことがあります。

実装手順

1. ワークロードを評価して、回復力ニーズをマルチ AZ アプローチ (単一の AWS リージョン) で満たせるか、それともマルチリージョンアプローチが必要かを判断します。これらの要件を満たす

ためにマルチリージョンアーキテクチャを実装すると、複雑性が増すため、ユースケースと要件を慎重に考慮してください。回復力の要件は、ほぼ常に、単一の AWS リージョン を使用して満たすことができます。複数のリージョンを使用する必要があるかどうかを判断するときには、次のような要件を考慮してください。

a. ディザスタリカバリ (DR): 1 つのアベイラビリティゾーンの混乱または部分的損失に基づく災害イベントについては、単一の AWS リージョン 内の複数のアベイラビリティゾーンで高可用性ワークロードを実装すると、自然災害または技術的な災害の軽減に役立ちます。相互にかなりの距離を隔てた複数のアベイラビリティゾーンコンポーネントの損失リスクがある災害については、リージョン規模の自然災害や技術的障害を緩和するために、複数のリージョンにまたがるディザスタリカバリを実装する必要があります。

b. 高可用性 (HA): マルチリージョンアーキテクチャ (各リージョンで複数の AZ を使用) を使用して、99.99% 以上の可用性を達成できます。

c. スタックローカリゼーション: 世界中のオーディエンスにワークロードをデプロイするときには、異なる AWS リージョン にローカライズしたスタックをデプロイして、それらのリージョンのオーディエンスに対応できます。ローカリゼーションには、言語、通貨、および保存されるデータのタイプが含まれます。

d. ユーザーへの近接性: 世界中のオーディエンスにワークロードをデプロイするときには、エンドユーザーに近い AWS リージョン にスタックをデプロイすることによって、レイテンシーを軽減できます。

e. データレジデンシー: 一部のワークロードはデータレジデンシー要件の対象であり、特定のユーザーからのデータは特定の国境内にとどめる必要があります。該当する規制に基づいて、それらの国境内の AWS リージョン にスタック全体をデプロイするか、データだけをデプロイするかを選ぶことができます。

2. AWS のサービスによって提供されるマルチ AZ 機能の例をいくつか示します。

a. EC2 または ECS を使用するワークロードを保護するには、コンピューティングリソースの前に Elastic Load Balancer をデプロイします。Elastic Load Balancing は、異常のあるゾーンのインスタンスを検出して、正常なゾーンにトラフィックをルーティングするソリューションを提供します。

   i. Application Load Balancers の開始方法

   ii. Network Load Balancer の開始方法

b. ロードバランシングをサポートしない市販のソフトウェアを実行する EC2 インスタンスの場合、マルチ AZ ディザスタリカバリ方法論を実装することによって、一種の耐障害性を達成できます。

   i. the section called “REL13-BP02 復旧目標を満たすため、定義された復旧戦略を使用する”

c. Amazon ECS タスクの場合は、3 つの AZ に均等にサービスをデプロイして、可用性とコストのバランスを達成します。

   i. Amazon ECS の可用性のベストプラクティス | コンテナ

d. 非 Aurora Amazon RDS の場合、マルチ AZ を設定オプションとして選ぶことができます。プライマリデータベースインスタンスに障害が発生した場合、Amazon RDS はスタンバイデータベースを自動的に昇格させて、別のアベイラビリティゾーンのトラフィックを受け取ります。マルチリージョンリードレプリカを作成して、回復力を高めることもできます。

   i. Amazon RDS マルチ AZ デプロイ

   ii. 別の AWS リージョン でのリードレプリカの作成

3. AWS のサービスによって提供されるマルチリージョン機能の例をいくつか示します。

a. マルチ AZ の可用性がサービスによって自動的に提供される Amazon S3 ワークロードの場合、マルチリージョンデプロイが必要な場合は、マルチリージョンアクセスポイントを検討してください。

   i. Amazon S3 のマルチリージョンアクセスポイント

b. マルチ AZ の可用性がサービスによって自動的に提供される DynamoDB テーブルの場合、既存のテーブルをグローバルテーブルに容易に変換して、複数リージョンの利点を活かすことができます。

   i. 単一リージョン Amazon DynamoDB テーブルをグローバルテーブルに変換する

c. ワークロードの前に Application Load Balancers または Network Load Balancer がある場合には、AWS Global Accelerator を使用し、正常なエンドポイントを含んでいる複数のリージョンにトラフィックを向けることで、アプリケーションの可用性を高めます。

   i. AWS Global Accelerator にける標準アクセラレーター用エンドポイント - AWS Global Accelerator (amazon.com)

d. AWS EventBridge を利用するアプリケーションの場合、クロスリージョンバスによってイベントを、選択したほかのリージョンに転送することを検討してください。

   i. Amazon EventBridge イベントの AWS リージョン 間での送受信

e. Amazon Aurora データベースの場合、複数の AWS リージョンにまたがる Aurora グローバルデータベースを検討してください。既存のクラスターを変更して、新しいリージョンも追加できます。

   i. Amazon Aurora グローバルデータベースの開始方法

f. ワークロードに AWS Key Management Service (AWS KMS) 暗号化キーが含まれる場合は、マルチリージョンキーがアプリケーションに適切かどうかを考慮してください。

   i. AWS KMS のマルチリージョンキー

g. その他の AWS サービスの機能については、このブログシリーズの AWS のサービスによるマルチリージョンアプリケーションの作成シリーズを参照してください。

実装計画の工数レベル: 中 ~ 高

リソース

関連するドキュメント:

- AWS のサービスによるマルチリージョンアプリケーションの作成シリーズを参照してください。
- AWS のディザスタリカバリ (DR) アーキテクチャ、パート IV: マルチサイトアクティブ/アクティブ
- AWS グローバルインフラストラクチャ
- AWS Local Zones のよくある質問
- AWS のディザスタリカバリ (DR) アーキテクチャ、パート I: クラウドでのリカバリ戦略
- クラウド上の災害対策はさまざまある
- グローバルテーブル: DynamoDB を使用したマルチリージョンレプリケーション

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Auth0: 自動フェイルオーバーにより、月あたり 15 億回以上のログインにスケールするマルチリージョンの高可用性アーキテクチャ

関連する例:

- AWS のディザスタリカバリ (DR) アーキテクチャ、パート I: クラウドでのリカバリ戦略
- DTCC はオンプレミスで実行できることをはるかに超えた回復力を達成
- Expedia Group は専有 DNS サービスでマルチリージョン、マルチアベイラビリティゾーンを使用して、アプリケーションに回復力を追加
- Uber: マルチリージョン Kafka の災害対策
- Netflix: マルチリージョンの回復力のためのアクティブ/アクティブ
- Atlassian Cloud のデータ回復力を構築する方法
- Intuit TurboTax は 2 つのリージョンで実行

REL10-BP03 単一の口ケーションに制約されるコンポーネントのリカバリを自動化する

ワークロードのコンポーネントが 1 つのアベイラビリティゾーンまたはオンプレミスのデータセンターでのみ実行できる場合は、定義された復旧目標内でワークロードを全面的に再構築する機能を実装する必要があります。

技術的な制約のためにワークロードを複数の口ケーションにデプロイするベストプラクティスが不可能な場合は、弾力性を確保するための代替パスを採り入れる必要があります。このような場合、必要なインフラストラクチャを再作成し、アプリケーションを再デプロイし、必要なデータを再作成する機能を自動化する必要があります。

例えば、Amazon EMR は同じアベイラビリティーゾーンで特定のクラスターのすべてのノードを起動します。これは、同じゾーンでクラスターを実行すると、データアクセス率が高くなり、ジョブフローのパフォーマンスが向上するためです。このコンポーネントがワークロードの回復力のために必要な場合は、クラスターとそのデータを再デプロイする方法が必要です。また、Amazon EMR では、マルチ AZ を使用する以外の方法で冗長性をプロビジョニングする必要があります。複数のマスターノードを プロビジョニングできます.EMR ファイルシステム (EMRFS) を使用すると、EMR のデータを Amazon S3 に保存でき、次にそのデータを複数のアベイラビリティゾーンまたは AWS リージョン にわたってレプリケートできます。

Amazon Redshift も同様に、デフォルトでは、選択した AWS リージョン 内のランダムに選択されたアベイラビリティゾーンにクラスターをプロビジョニングします。すべてのクラスターノードが同じゾーンにプロビジョニングされます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 自己修復を実装します。可能であれば自動スケーリングを利用して、インスタンスとコンテナをデプロイします。自動スケーリングを利用できない場合は、EC2 インスタンスの自動復旧機能を利用するか、Amazon EC2 または ECS のコンテナのライフサイクルイベントに基づいた自己修復自動化を実装します。
  - 単一インスタンス IP アドレスや、プライベート IP アドレス、Elastic IP アドレス、インスタンスメタデータを必要としないインスタンスとコンテナのワークロードには、Auto Scaling グループを使用します。
    - EC2 Auto Scaling とは?
    - サービスのオートスケーリング
      - 起動設定ユーザーデータを使用して、ほとんどのワークロードを自己修復できるオートメーションを実装できます。

- 単一インスタンス IP アドレスや、プライベート IP アドレス、elastic IP アドレス、インスタンスメタデータを必要とするワークロードには、EC2 インスタンスの自動復旧機能を使用します。
  - インスタンスの復旧
    - 自動復旧は、インスタンスの障害が検出されると、復旧ステータスアラートを SNS トピックに送信します。
- オートスケーリングや EC2 の復旧機能を利用できない場合は、EC2 インスタンスのライフサイクルイベントや ECS イベントを利用して、自己修復を自動化します。
  - EC2 Auto Scaling ライフサイクルフック
  - Amazon ECS イベント
    - 必要なプロセスロジックに従ってコンポーネントを修復するオートメーションを呼び出すには、イベントを利用します。

リソース

関連するドキュメント:

- Amazon ECS イベント
- EC2 Auto Scaling ライフサイクルフック
- インスタンスの復旧
- サービスのオートスケーリング
- EC2 Auto Scaling とは?

REL10-BP04 バルクヘッドアーキテクチャを使用して影響範囲を制限する

このパターンは、船の隔壁のように、障害がリクエストまたはユーザーの小さなサブセットに確実にとどまるようにすることで、障害のあるリクエストの数を制限し、ほとんどが エラーなしで続行できるようにします。多くの場合、データの隔壁はパーティションと呼ばれ、サービスの隔壁はセルと呼ばれます。

セルベースのアーキテクチャ では、各セルは独立した完全なサービスのインスタンスで、最大サイズは固定されています。負荷が増加すると、セルが追加されることでワークロードが増大します。着信トラフィックでパーティションキーを使用して、リクエストを処理するセルを決定します。障害は発生した単一のセルに限定され、他のセルが エラーなしで継続するため、障害のあるリクエストの数が制限されます。セル間の相互作用を最小限に抑え、各リクエストに複雑なマッピングサービスを含

める必要がないように、適切なパーティションキーを特定することが重要です。複雑なマッピングを必要とするサービスは、問題をマッピングサービスにシフトするだけですが、セル間の相互作用を必要とするサービスは、セルの独立性が低下します (そのため、これにより想定される可用性の改善が低下)。

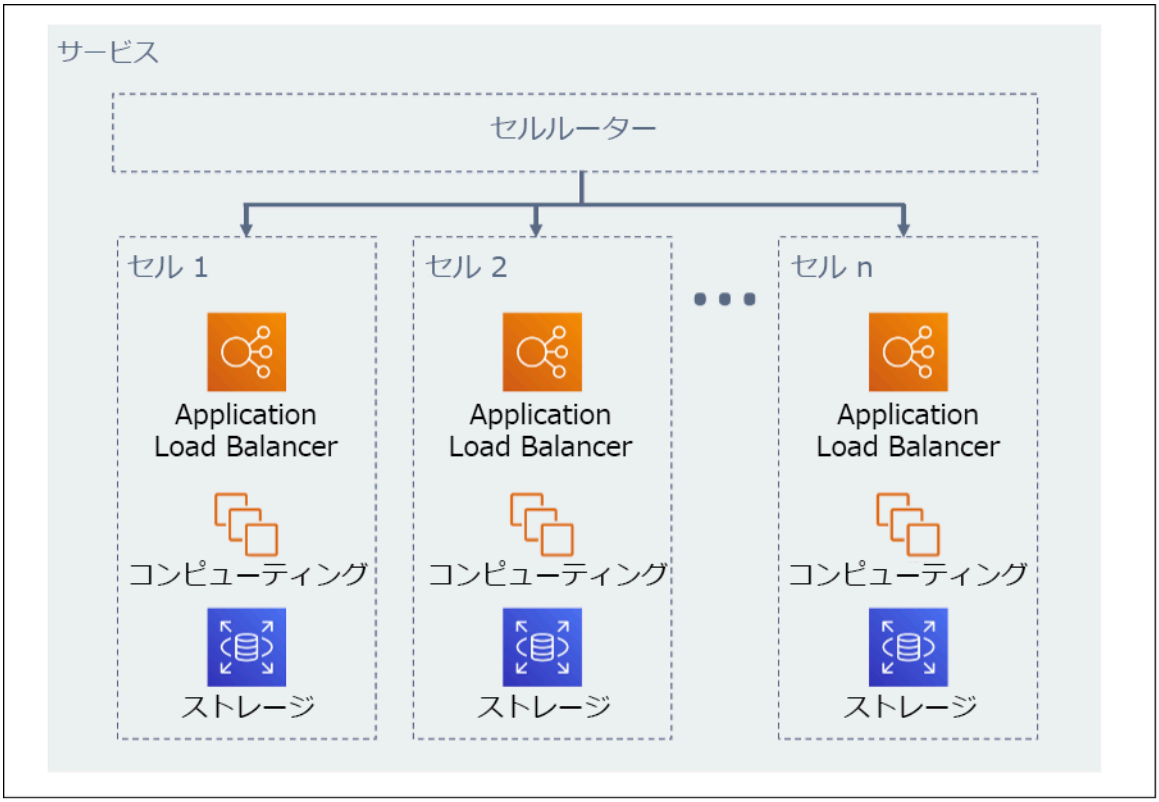


図 11: セルベースアーキテクチャ

AWS ブログ投稿で、Colm MacCarthaigh 氏は Amazon Route 53 が シャッフルシャーディング の概念を使用して、顧客リクエストをシャードに分離する方法を説明します。この場合、シャードは 2 つ以上のセルで構成されます。パーティションキーに基づいて、顧客 (またはリソース、または分離対象) からのトラフィックは、割り当てられたシャードにルーティングされます。シャードごとに 2 つのセルを持つ 8 つのセルがあり、顧客が 4 つのシャードに分割された場合、問題が発生した場合に影響を受ける顧客は全体の 25% です。

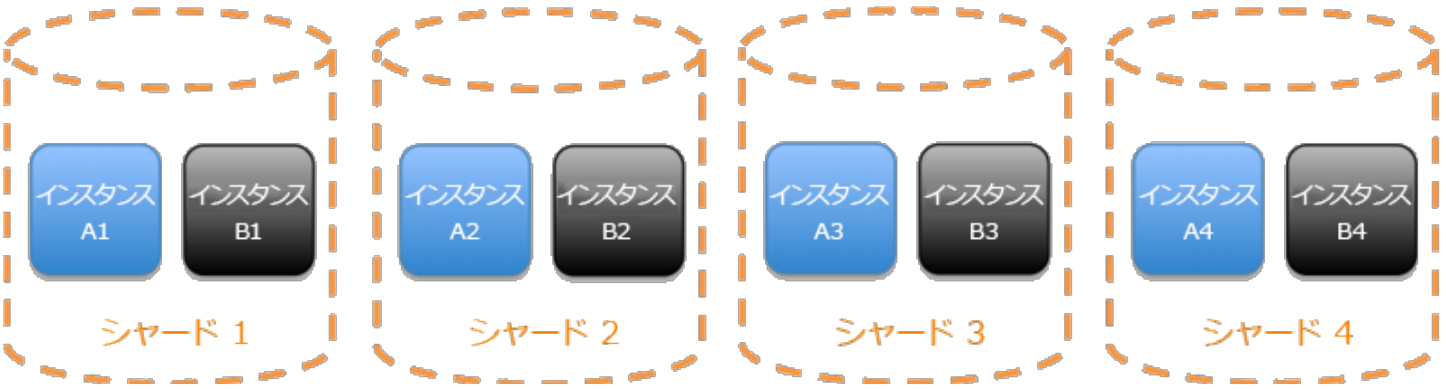


図 12: それぞれ 2 つのセルを持つ 4 つの従来のシャードに分割されたサービス

シャッフルシャーディングでは、それぞれ 2 つのセルを持つ仮想シャードを作成し、顧客をそれらの仮想シャードの 1 つに割り当てます。問題が発生した場合でも、サービス全体の 4 分の 1 が失われる可能性がありますが、顧客またはリソースが割り当てられる方法から、シャッフルシャーディングでは影響を受ける範囲が 25% を大幅に下回ることになります。8 つのセル中の 2 つのセルには 28 のユニークな組み合わせがあります。つまり、シャッフルシャード (仮想シャード) が 28 通りあります。数百または数千の顧客がいて、各顧客を 1 つのシャッフルシャードに割り当てた場合、問題が発生したことで影響を受ける範囲はわずか 1/28 です。これは通常のシャーディングより 7 倍優れています。

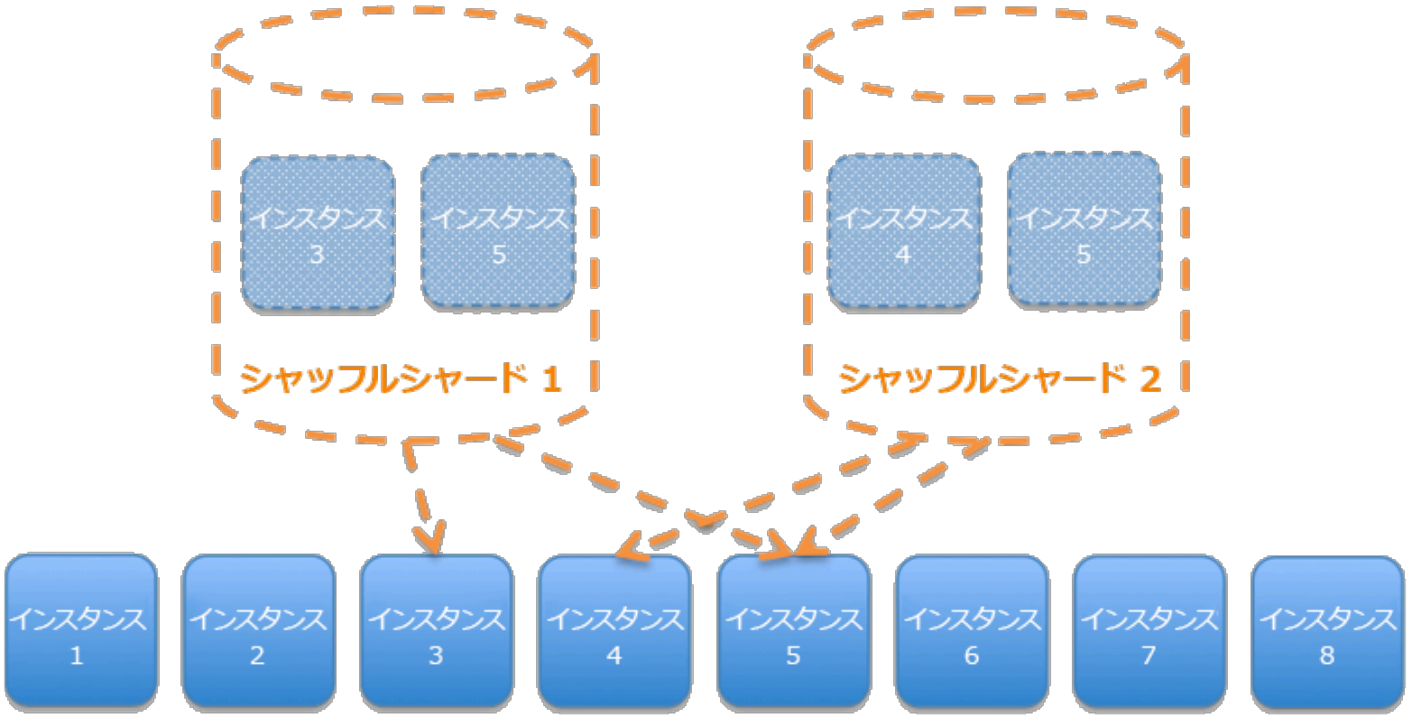


図 13: それぞれ 2 つのセルを持つ 28 のシャッフルシャード (仮想シャード) に分割されたサービス (28 通りのうち 2 つのシャッフルシャードのみを表示)

シャードは、セルだけでなく、サーバー、キュー、またはその他のリソースにも使用できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- バルクヘッドアーキテクチャを使用します。このパターンは、船の隔壁のように、障害がリクエストまたはユーザーの小さなサブセットに確実にとどまるようにすることで、障害のあるリクエストの数を制限し、ほとんどがエラーなしで続行できるようにします。多くの場合、データの隔壁はパーティションと呼ばれ、サービスの隔壁はセルと呼ばれます。
  - Well-Architected ラボ: シャッフルシャーディングを使用した障害分離
  - Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
  - AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- ワークロードのセルベースのアーキテクチャを評価します。セルベースのアーキテクチャでは、各セルは独立した完全なサービスのインスタンスで、最大サイズは固定されています。負荷が増加すると、セルが追加されることでワークロードが増大します。着信トラフィックでパーティションキーを使用して、リクエストを処理するセルを決定します。障害は発生した単一のセルに限定され、他のセルがエラーなしで継続するため、障害のあるリクエストの数が制限されます。セル間の相互作用を最小限に抑え、各リクエストに複雑なマッピングサービスを含める必要がないように、適切なパーティションキーを特定することが重要です。複雑なマッピングを必要とするサービスは、問題をマッピングサービスにシフトするだけですが、セル間の相互作用を必要とするサービスは、セルの自律性が低下します (そのため、予想された可用性の向上も低下します)。
  - Colm MacCarthaigh は、 AWS ブログの記事で、Amazon Route 53 がシャッフルシャーディングの概念を用いて顧客のリクエストをシャードに分離する方法を説明しています。
    - シャッフルシャーディング: 大量およびマジカルな障害切り離し

リソース

関連するドキュメント:

- シャッフルシャーディング: 大量およびマジカルな障害切り離し
- The Amazon Builders' Library: シャッフルシャーディングを使ったワークロードの分離

関連動画:

- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

関連する例:

- Well-Architected ラボ: シャッフルシャーディングを使用した障害分離

REL 11 コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?

高い可用性と低い平均復旧時間 (MTTR) の要件を持つワークロードは、回復力を考慮した設計をする必要があります。

ベストプラクティス

- REL11-BP01 ワークロードのすべてのコンポーネントをモニタリングして障害を検知する
- REL11-BP02 正常なリソースにフェイルオーバーする
- REL11-BP03 すべてのレイヤーの修復を自動化する
- REL11-BP04 復旧中はコントロールプレーンではなくデータプレーンを利用する
- REL11-BP05 静的安定性を使用してバイモーダル動作を防止する
- REL11-BP06 イベントが可用性に影響する場合に通知を送信する

REL11-BP01 ワークロードのすべてのコンポーネントをモニタリングして障害を検知する

ワークロードの状態を継続的にモニタリングすることで、お客様と自動化システムがパフォーマンスの低下や完全な障害の発生を速やかに把握できるようにします。ビジネス価値に基づいて重要業績評価指標 (KPI) をモニタリングします。

復旧および修復メカニズムはすべて、問題を迅速に検出する機能から開始する必要があります。技術的な障害を最初に検出して、解決できるようにするのです。ただし、可用性はワークロードがビジネス価値を提供する能力に基づいているため、これを測定する重要業績評価指標 (KPI) が検出および修正戦略の一部である必要があります。

一般的なアンチパターン:

- アラームが設定されていないため、停止は通知なしで発生します。
- アラームは存在しますが、そのしきい値では対応するために十分な時間がありません。
- メトリクスは、目標復旧時間 (RTO) を満たすのに十分な頻度で収集されません。
- ワークロードの顧客向け階層のみがアクティブにモニタリングされます。
- 技術的なメトリクスのみ収集し、ビジネス関数のメトリクスは収集しません。

- ワークロードのユーザーエクスペリエンスを測定するメトリクスはありません。

このベストプラクティスを活用するメリット: すべてのレイヤーで適切なモニタリングを行うことで、検出までの時間を短縮して、復旧時間を短縮できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 復旧の目標に基づいて、コンポーネントの収集間隔を決定します。
  - モニタリング間隔は、どの程度迅速に復旧する必要があるかによって異なります。復旧時間は復旧にかかる時間によって決まるため、この時間と目標復旧時間 (RTO) を考慮して、収集の頻度を決定する必要があります。
- コンポーネントの詳細モニタリングを設定します。
  - EC2 インスタンスと Auto Scaling の詳細モニタリングが必要かどうかを判断します。詳細モニタリングは 1 分間隔メトリクスを提供し、デフォルトのモニタリングは 5 分間隔メトリクスを提供します。
    - インスタンスの詳細モニタリングの有効化/無効化
    - Amazon CloudWatch を使用した Auto Scaling グループおよびインスタンスのモニタリング
  - RDS の拡張モニタリングが必要かどうかを判断します。拡張モニタリングでは、RDS インスタンスのエージェントを使用して、RDS インスタンスのさまざまなプロセスまたはスレッドに関する有益な情報を取得します。
    - 拡張モニタリング
- ビジネスの重要業績評価指標 (KPI) を測定するカスタムメトリクスを作成します。ワークロードは主要なビジネス機能を実装します。これらの関数は、いつ間接的な問題が発生したのかを特定するのに役立つ KPI として使用される必要があります。
  - カスタムメトリクスの発行
- 模擬モニタリングを使用して、障害に対するユーザーエクスペリエンスをモニタリングします。顧客の行動を実行およびシミュレートできる合成トランザクションテスト (「カナリアテスト」とも呼ばれますが、カナリアデプロイと混同しないでください) は、最も重要なテストプロセスの 1 つです。さまざまなリモートロケーションからワークロードエンドポイントに対してこれらのテストを常に実行します。
  - Amazon CloudWatch Synthetics により模擬モニタリングを作成可能
- ユーザーのエクスペリエンスを追跡するカスタムメトリクスを作成します。顧客のエクスペリエンスを測定できる場合は、コンシューマーエクスペリエンスが低下するタイミングを判断できます。

  - カスタムメトリクスの発行
- ワークロードの一部が正常に動作していないことを検出し、リソースを Auto Scaling するタイミングを示すアラームを設定します。アラームはダッシュボードに視覚的に表示したり、Amazon SNS または E メールでアラートを送信したり、Auto Scaling と連携してワークロードのリソースをスケールアップまたはスケールダウンしたりできます。
  - Amazon CloudWatch アラームの使用
- ダッシュボードを作成してメトリクスを視覚化します。ダッシュボードは、傾向や異常値などの潜在的な問題の指標を視覚的に確認したり、調査対象となり得る問題の存在の示唆を与えたりするために使用できます。
  - CloudWatch ダッシュボードの使用

リソース

関連するドキュメント:

- Amazon CloudWatch Synthetics により模擬モニタリングを作成可能
- インスタンスの詳細モニタリングの有効化/無効化
- 拡張モニタリング
- Amazon CloudWatch を使用した Auto Scaling グループおよびインスタンスのモニタリング
- カスタムメトリクスの発行
- Amazon CloudWatch アラームの使用
- CloudWatch ダッシュボードの使用

関連する例:

- Well-Architected ラボ: レベル 300: ヘルスチェックを実装し依存関係を管理して、信頼性を向上する

REL11-BP02 正常なリソースにフェイルオーバーする

リソース障害の発生時に、正常なリソースが引き続きリクエストに対応できるようにします。ロケーション障害 (アベイラビリティゾーンや AWS リージョン など) に対しては、障害のないロケーションの正常なリソースにフェイルオーバーするシステムを用意します。

Elastic Load Balancing や AWS Auto Scaling などの AWS のサービスは、複数のリソースおよびアベイラビリティゾーンへの負荷分散に役立ちます。そのため、個々のリソース (EC2 インスタンスなど) の障害や、アベイラビリティゾーンの障害を、残りの正常なリソースにトラフィックをシフトすることによって緩和できます。マルチリージョンのワークロードの場合、状況はさらに複雑です。例えば、クロスリージョンリードレプリカを使用すると、データを複数の AWS リージョン にデプロイできますが、障害が発生した場合は、リードレプリカをプライマリに昇格させ、そこにトラフィックを向かわせる必要があります。Amazon Route 53 と AWS Global Accelerator は、AWS リージョン 間のトラフィックのルーティングを容易にします。

ワークロードが Amazon S3 や Amazon DynamoDB などの AWS のサービスを使用している場合、自動的に複数のアベイラビリティゾーンにデプロイされます。障害が発生した場合、AWS コントロールプレーンはトラフィックを正常なロケーションに自動的にルーティングします。データは複数のアベイラビリティゾーンに冗長的に保存され、使用可能なままとなります。Amazon RDS の場合、設定オプションとしてマルチ AZ を選択する必要があります。その場合、障害が発生すると、AWS はトラフィックを正常なインスタンスに自動的にルーティングします。Amazon EC2 インスタンス、Amazon ECS タスク、または Amazon EKS ポッドの場合、デプロイ先のアベイラビリティゾーンを選択します。Elastic Load Balancing は、異常なゾーンのインスタンスを検出し、正常なゾーンにトラフィックをルーティングするソリューションを提供します。Elastic Load Balancing は、オンプレミスのデータセンターのコンポーネントにトラフィックをルーティングすることもできます。

マルチリージョンのアプローチ (オンプレミスのデータセンターも含まれる場合があります) の場合、Amazon Route 53 はインターネットドメインを定義し、ヘルスチェックを含むルーティングポリシーを割り当てて、トラフィックが正常なリージョンにルーティングされるようにします。または、AWS Global Accelerator は、アプリケーションへの固定エントリポイントとして機能する静的 IP アドレスを提供します。そして、インターネットの代わりに AWS グローバルネットワークを使用して、選択した AWS リージョン のエンドポイントにルーティングして、パフォーマンスと信頼性を向上させます。

AWS は、障害復旧を念頭に置いてサービスの設計に取り組んでいます。当社は、障害からの復旧時間とデータへの影響を最小限に抑えるサービスを設計しています。当社のサービスは主にデータストアを使用しており、リクエストが認識されるのは、リージョン内の複数のレプリカにわたりデータが永続的に保存された後です。これらのサービスとリソースには Amazon Aurora、Amazon Relational Database Service (Amazon RDS) マルチ AZ DB インスタンス、Amazon S3、Amazon DynamoDB、Amazon Simple Queue Service、Amazon SQS、および Amazon Elastic File System (Amazon EFS) が含まれます。これらのサービスは、セル単位の分離とアベイラビリティーゾーンにより提供される障害切り分けを活用するように構成されています。当社は、運用上の手順の多くで自

動化を幅広く使用しています。また、中断から迅速に復旧するために、置換と再起動の機能を最適化しています。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 正常なリソースにフェイルオーバーします。リソース障害の発生時に、正常なリソースが引き続きリクエストに対応できるようにします。ロケーション障害 (アベイラビリティゾーンや AWS リージョン など) に対しては、障害のないロケーションの正常なリソースにフェイルオーバーするシステムを用意します。
  - ワークロードが Amazon S3 や Amazon DynamoDB などの AWS のサービスを使用している場合、自動的に複数のアベイラビリティゾーンにデプロイされます。障害が発生した場合、AWS コントロールプレーンはトラフィックを正常なロケーションに自動的にルーティングします。
  - Amazon RDS の場合、設定オプションとしてマルチ AZ を選択する必要があります。その場合、障害が発生すると、AWS はトラフィックを正常なインスタンスに自動的にルーティングします。
    - Amazon RDS の高可用性 (マルチ AZ)
  - Amazon EC2 インスタンスまたは Amazon ECS タスクの場合、デプロイ先のアベイラビリティーゾーンを選択します。Elastic Load Balancing は、異常なゾーンのインスタンスを検出し、正常なゾーンにトラフィックをルーティングするソリューションを提供します。Elastic Load Balancing は、オンプレミスのデータセンターのコンポーネントにトラフィックをルーティングすることもできます。
  - マルチリージョンのアプローチ (オンプレミスのデータセンターが含まれる場合もあります) の場合は、正常なロケーションのデータとリソースが、引き続きリクエストに対応できることを確認します。
    - 例えば、クロスリージョンリードレプリカを使用すると、データを複数の AWS リージョンにデプロイできますが、プライマリロケーションに障害が発生した場合は、リードレプリカをマスターに昇格させ、そこにトラフィックを向かわせる必要があります。
      - Amazon RDS リードレプリカの概要
    - Amazon Route 53 は、インターネットドメインを定義し、ヘルスチェックを含むルーティングポリシーを割り当てて、トラフィックが正常なリージョンに確実にルーティングされるようにする方法を提供します。または、AWS Global Accelerator は、アプリケーションへの固定エントリポイントとして機能する静的 IP アドレスを提供します。そして、インターネットの代わりに AWS グローバルネットワークを使用して、選択した AWS リージョン のエンドポイントにルーティングして、パフォーマンスと信頼性を向上させます。

- Amazon Route 53: ルーティングポリシーを選択する
- 「AWS Global Accelerator とは何ですか?」

リソース

関連するドキュメント:

- APN パートナー: 耐障害性のオートメーションを支援できるパートナー
- AWS Marketplace: 耐障害性に関して活用できる商品
- AWS OpsWorks: 自動ヒーリングを使用して障害のあるインスタンスを置き換える
- Amazon Route 53: ルーティングポリシーを選択する
- Amazon RDS の高可用性 (マルチ AZ)
- Amazon RDS リードレプリカの概要
- Amazon ECS タスク配置戦略
- 複数のアベイラビリティゾーンの Kubernetes Auto Scaling グループの作成
- 「AWS Global Accelerator とは何ですか?」

関連する例:

- Well-Architected ラボ: レベル 300: ヘルスチェックを実装し依存関係を管理して、信頼性を向上する

REL11-BP03 すべてのレイヤーの修復を自動化する

障害を検出したら、自動化機能を使用して修復するアクションを実行します。

再起動する機能 は、障害を修復するための重要なツールです。分散システムについて前述したように、ベストプラクティスは、可能な場合はサービスをステートレスにすることです。これにより、再起動時のデータまたは可用性が失われるのを防ぎます。クラウドでは、再起動の一環として、リソース全体 (EC2 インスタンス、Lambda 関数など) を置き換えることができます (通常はそうする必要があります)。再起動自体は、障害から復旧するための簡単で信頼できる方法です。ワークロードでは、さまざまなタイプの障害が発生します。障害は、ハードウェア、ソフトウェア、通信、オペレーションなどさまざまな部分で発生する可能性があります。さまざまなタイプの障害をそれぞれ捕捉、特定、修正するための新しいメカニズムを構築するのではなく、さまざまなカテゴリの障害を同じ復旧戦略にマッピングします。ハードウェアの障害、オペレーティングシステムのバグ、メモ

リリーク、その他の原因で、インスタンスが機能しなくなることがあります。状況ごとにカスタム修復を構築するのではなく、そのいずれかをインスタンスの障害として扱います。インスタンスを終了し、AWS Auto Scaling がそのインスタンスを置き換えることを許可します。その後、障害が発生したリソースの分析を帯域外で実行します。

もう 1 つの例は、ネットワークリクエストを再起動する機能です。依存関係にあるシステムからエラーが返された場合、ネットワークのタイムアウトの場合と依存関係にあるシステムの障害の両方に同じ復旧アプローチを適用します。どちらのイベントもシステムに類似の影響を与えるため、どちらかのイベントを「特例」とするのではなく、エクスポネンシャルバックオフとジッターで限定的に再試行するという類似の戦略を適用します。

再起動する機能 は、復旧指向コンピューティング (ROC) と高可用性クラスターアーキテクチャを特徴とする復旧メカニズムです。

Amazon EventBridge を使用して、CloudWatch アラームなどのイベントや他の AWS のサービスの状態の変化をモニタリングおよびフィルタリングできます。イベント情報に基づいて、AWS Lambda、AWS Systems Manager Automation、または他のターゲットをトリガーして、ワークロードに対してカスタム修正ロジックを実行できます。

Amazon EC2 Auto Scaling は、EC2 インスタンスの状態をチェックするように設定できます。インスタンスが実行中以外の状態にある場合、またはシステムステータスが損なわれている場合、Amazon EC2 Auto Scaling はインスタンスが異常であると見なし、代替インスタンスを起動します。AWS OpsWorks を使用している場合は、OpsWorks レイヤーレベルで EC2 インスタンスの自動ヒーリングを設定できます。

大規模な置き換え (アベイラビリティーゾーン全体の喪失など) の場合、複数の新しいリソースを一度に取得するのではなく、静的安定性が高可用性のために優先されます。

一般的なアンチパターン:

- インスタンスまたはコンテナにアプリケーションを個別にデプロイします。
- 自動復旧を使用せずに、複数のロケーションにデプロイできないアプリケーションをデプロイします。
- 自動スケーリングと自動復旧が修復に失敗するアプリケーションを手動で修復します。

このベストプラクティスを活用するメリット: ワークロードが一度に 1 つのロケーションにしかデプロイできない場合でも、自動ヒーリングによって、復旧までの平均時間が短縮され、ワークロードの可用性を確保できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- Auto Scaling グループを使用して、ワークロードに階層をデプロイします。オートスケーリングは、ステートレスなアプリケーションで自己修復を実行し、キャパシティーを追加および削除できます。
  - AWS Auto Scaling の仕組み
- 複数のロケーションにデプロイできないアプリケーションがデプロイされている EC2 インスタンスに自動復旧を実装し、障害時の再起動を許容できます。自動復旧は、アプリケーションが複数のロケーションにデプロイできない場合に、障害が発生したハードウェアを交換してインスタンスを再起動するために使用できます。インスタンスメタデータおよび関連する IP アドレスは保持されます。また、Amazon EBS ボリュームと Elastic File Systems または File Systems for Lustre and Windows へのマウントポイントも保持されます。
  - Amazon EC2 自動復旧
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon Elastic File System (Amazon EFS)
  - Amazon FSx for Lustre とは?
  - Amazon FSx for Windows File Server とは?
    - AWS OpsWorks を使用している場合は、レイヤーレベルで EC2 インスタンスの自動ヒーリングを設定できます。
      - AWS OpsWorks: 自動ヒーリングを使用して、障害のあるインスタンスを置き換える
- 自動スケーリングまたは自動復旧を使用できない場合、または自動復旧が失敗した場合は、AWS Step Functions と AWS Lambda を使用して自動復旧を実装します。自動スケーリングを使用できず、さらに、自動復旧が使用できないか、自動復旧が失敗した場合は、AWS Step Functions と AWS Lambda を使用して修復を自動化できます。
  - 「AWS Step Functions とは何ですか?」
  - 「AWS Lambda とは何ですか?」
    - Amazon EventBridge を使用して、CloudWatch アラームなどのイベントや他の AWS のサービスの状態の変化をモニタリングおよびフィルタリングできます。イベント情報に基づいて、AWS Lambda (または他のターゲット) をトリガーし、ワークロードに対してカスタム修正ロジックを実行できます。
      - 「Amazon EventBridge とは?」
      - Amazon CloudWatch アラームの使用

リソース

関連するドキュメント:

- APN パートナー: 耐障害性のオートメーションを支援できるパートナー
- AWS Marketplace: 耐障害性に関して活用できる商品
- AWS OpsWorks: 自動ヒーリングを使用して、障害のあるインスタンスを置き換える
- Amazon EC2 自動復旧
- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- AWS Auto Scaling の仕組み
- Amazon CloudWatch アラームの使用
- 「Amazon EventBridge とは?」
- 「AWS Lambda とは何ですか?」
- AWS Systems Manager Automation をトリガーして
- 「AWS Step Functions とは何ですか?」
- Amazon FSx for Lustre とは?
- Amazon FSx for Windows File Server とは?

関連動画:

- AWS の静的安定性: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

関連する例:

- Well-Architected ラボ: レベル 300: ヘルスチェックを実装し依存関係を管理して、信頼性を向上する

REL11-BP04 復旧中はコントロールプレーンではなくデータプレーンを利用する

コントロールプレーンはリソースの設定に使用し、データプレーンはサービスを提供します。通常、データプレーンの可用性設計目標はコントロールプレーンよりも高く、複雑さは低くなっています。回復力に影響する可能性があるイベントに対して復旧または軽減対策を実装するときは、コントロールプレーンを使用すると、アーキテクチャの全体的な回復力が下がる可能性があります。例えば、Amazon Route 53 データプレーンを利用して、ヘルスチェックに基づいて DNS クエリを確実に

ルーティングできますが、Route 53 ルーティングポリシーの更新にはコントロールプレーンを使用するため、これを復旧には利用しないでください。

Route 53 データプレーンは、DNS クエリに応答し、ヘルスチェックを実行し、評価します。グローバルに分散され、100% の可用性サービスレベルアグリーメント (SLA) として設計されています。 Route 53 のリソースを作成、更新、削除する Route 53 管理 API およびコンソールは、コントロールプレーンで実行します。コントロールプレーンは、DNS の管理に必要な強力な一貫性と耐久性を重視するように設計されています。これを達成するために、コントロールプレーンは単一のリージョン、US East (N. Virginia) に配置されています。どちらのシステムも非常に高い信頼性で構築されていますが、コントロールプレーンは SLA には含まれません。まれに、データプレーンの回復力設計によって可用性を維持できるときでも、コントロールプレーンでは維持でない場合があります。ディザスタリカバリおよびフェイルオーバーメカニズムについては、データプレーンの機能を使用して、可能な限り最善の信頼性を提供してください。

データプレーン、コントロールプレーン、および AWS が高可用性目標を満たすためにサービスを構築する方法の詳細については、 アベイラビリティーゾーンを使用した静的安定性 ペーパーと Amazon Builders' Library を参照してください。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ディザスタリカバリに Amazon Route 53 を使用するときには、コントロールプレーンではなくデータプレーンを利用します。Route 53 Application Recovery Controller は、準備状況のチェックとルーティングコントロールを使用して、フェイルオーバーの管理と調整を支援します。これらの機能により、障害から回復するアプリケーションの能力を継続的にモニタリングし、複数の AWS リージョン、アベイラビリティゾーン、およびオンプレミスにまたがってアプリケーションの回復を管理できます。
  - Route 53 Application Recovery Controller とは
  - Amazon Route 53 を使用したディザスタリカバリメカニズムの作成
  - Amazon Route 53 を使用した回復力の高いアプリケーションの構築、パート 1: シングルリージョンスタック
  - Amazon Route 53 を使用した回復力の高いアプリケーションの構築、パート 1: シングルリージョンスタック
- データプレーンでの運用と、コントロールプレーンでの運用を理解します。
  - The Amazon Builders' Library: より小規模なサービスを管理して、分散システムの過負荷を回避する

- Amazon DynamoDB API (コントロールプレーンとデータプレーン)
- AWS Lambda の実行 (コントロールプレーンとデータプレーンに分割されています)
- AWS Lambda の実行 (コントロールプレーンとデータプレーンに分割されています)

リソース

関連するドキュメント:

- APN パートナー: 耐障害性のオートメーションを支援できるパートナー
- AWS Marketplace: 耐障害性に関して活用できる商品
- The Amazon Builders' Library: より小規模なサービスを管理して、分散システムの過負荷を回避する
- Amazon DynamoDB API (コントロールプレーンとデータプレーン)
- AWS Lambda の実行 (コントロールプレーンとデータプレーンに分割されています)
- AWS Elemental MediaStore のデータプレーン
- Amazon Route 53 を使用した回復力の高いアプリケーションの構築、パート 1: シングルリージョンスタック
- Amazon Route 53 を使用した回復力の高いアプリケーションの構築、パート 1: シングルリージョンスタック
- Amazon Route 53 を使用したディザスタリカバリメカニズムの作成
- Route 53 Application Recovery Controller とは

関連する例:

- Amazon Route 53 Application Recovery Controller の概要

REL11-BP05 静的安定性を使用してバイモーダル動作を防止する

バイモーダル動作とは、たとえばアベイラビリティーゾーンに障害が発生した場合に新しいインスタンスの起動に依存するなど、通常モードと障害モードでワークロードが異なる動作を示す場合をいいます。バイモーダル動作を防止するために、静的に安定し、1 つのモードでのみ動作するワークロードを構築する必要があります。この場合、1 つの AZ が削除された場合にワークロードの負荷を処理するのに十分な数のインスタンスを各アベイラビリティーゾーンにプロビジョニングしてから、Elastic Load Balancing または Amazon Route 53 ヘルスチェックを使用して、障害のあるインスタンスから負荷を分散します。

EC2 インスタンスやコンテナなどのコンピューティングデプロイの静的安定性があると、信頼性が最も高くなります。これは、コストがかかる懸念と比較検討する必要があります。プロビジョニングするコンピューティングキャパシティーを減らし、障害が発生した場合は新しいインスタンスを起動する方が、コストは低くなります。ただし、大規模な障害 (アベイラビリティーゾーンの障害など) が発生した場合には、効果が低くなります。このアプローチは障害が発生する前に準備するのではなく、障害が発生したときに事後的に対処することになるためです。ソリューションを考える際は、信頼性とワークロードのコストのニーズを比較検討する必要があります。より多くのアベイラビリティーゾーンを使用することで、静的安定性に必要なコンピューティングキャパシティーが減少します。

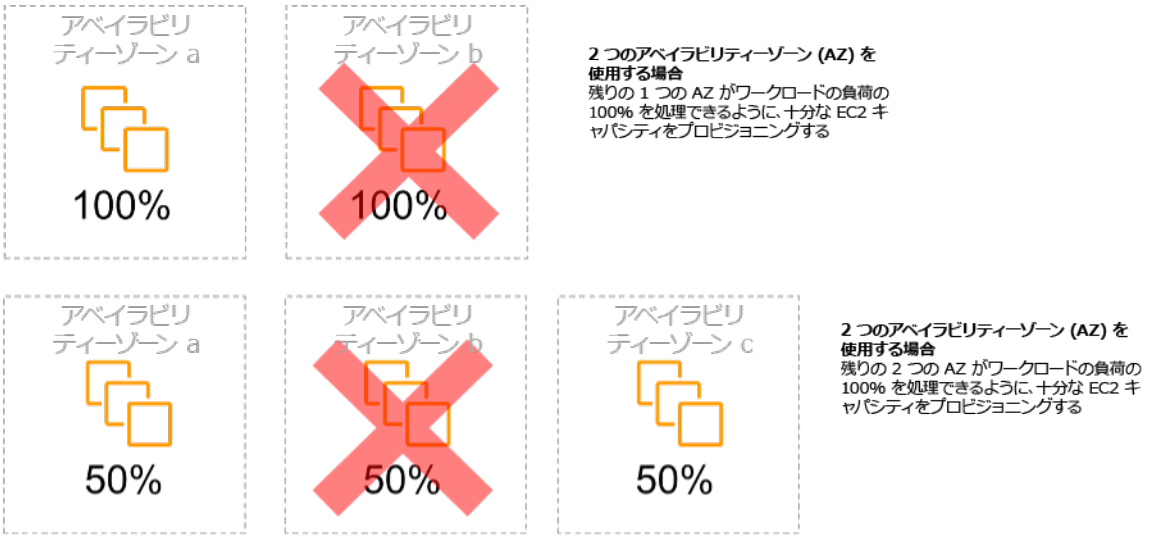


図 14: 複数のアベイラビリティゾーンにわたる EC2 インスタンスの静的安定性

トラフィックがシフトした後、AWS Auto Scaling を使用して、障害が発生したゾーンのインスタンスを非同期で置き換え、正常なゾーンで起動します。

バイモーダル動作のもう 1 つの例に、ネットワークのタイムアウトにより、システム全体の設定状態の再読み込みが始まることがあります。これにより想定外の負荷が別のコンポーネントに加わり、そのコンポーネントで障害が発生し、想定外の結果につながる可能性があります。この負のフィードバックループは、ワークロードの可用性に影響を与えます。そこで、静的に安定し、1 つのモードでのみ動作するシステムを構築する必要があります。静的に安定した設計は、一定の作業を行い、常に一定の周期で設定状態を更新することになるでしょう。呼び出しに失敗すると、ワークロードは以前にキャッシュされた値を使用し、アラームをトリガーします。

バイモーダル動作のもう 1 つの例は、障害発生時にクライアントがワークロードキャッシュをバイパスできるようにすることです。これは、クライアントのニーズに対応するソリューションのように思われるかもしれませんが、ワークロードのリクエストを大幅に変更し、障害が発生する可能性が高いため、許可すべきではありません。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 静的安定性を使用してバイモーダル動作を防止します。バイモーダル動作とは、たとえばアベイラビリティーゾーンに障害が発生した場合に新しいインスタンスの起動に依存するなど、通常モードと障害モードでワークロードが異なる動作を示す場合をいいます。
  - 災害対策プランでの依存関係の最小化
  - The Amazon Builders' Library: アベイラビリティーゾーンを使用した静的安定性
  - AWS の静的安定性: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
    - バイモーダル動作を防止するために、静的に安定し、1 つのモードでのみ動作するシステムを構築する必要があります。この場合、1 つの AZ が削除された場合にワークロードの負荷を処理するのに十分な数のインスタンスを各ゾーンにプロビジョニングしてから、Elastic Load Balancing または Amazon Route 53 ヘルスチェックを使用して、障害のあるインスタンスから負荷をシフトします。
    - バイモーダル動作のもう 1 つの例は、障害発生時にクライアントがワークロードキャッシュをバイパスできるようにすることです。これは、クライアントのニーズに対応するソリューションのように思われるかもしれませんが、ワークロードのリクエストを大幅に変更し、障害が発生する可能性が高いため、許可すべきではありません。

リソース

関連するドキュメント:

- 災害対策プランでの依存関係の最小化
- The Amazon Builders' Library: アベイラビリティーゾーンを使用した静的安定性

関連動画:

- AWS の静的安定性: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL11-BP06 イベントが可用性に影響する場合に通知を送信する

重大なイベントが検出されると、イベントによって引き起こされた問題が自動的に解決された場合でも、通知が送信されます。

自動ヒーリング機能により、ワークロードの信頼性を高めることができます。ただし、対処する必要のある根本的な問題もあいまいになる可能性があります。根本原因の問題を解決できるように、自動ヒーリングによって対処されたものを含む問題のパターンを検出できるように、適切なモニタリングとイベントを実装します。Amazon CloudWatch アラームは、発生した障害に基づいてトリガーできます。また、実行された自動ヒーリングアクションに基づいてトリガーすることもできます。CloudWatch アラームは、Amazon SNS 統合を使用して、E メールを送信するか、サードパーティのインシデント追跡システムにインシデントを記録するように設定できます。

一般的なアンチパターン:

- 誰もアクションを実行しないアラームを送信する。
- オートヒーリングのオートメーションを実行したが、ヒーリングが必要とされたことは通知しない。

このベストプラクティスを確立するメリット: 復旧イベントの通知により、まれに発生する問題を無視することがなくなります。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- ビジネスの重要業績評価指標が低しきい値を超えたときに警告します。ビジネス KPI に低しきい値を設定すると、ワークロードが利用不可または機能していない場合にそれを認識できます。
  - 静的しきい値に基づいて CloudWatch アラームを作成する
- ヒーリングオートメーションを呼び出すイベントについて警告します。SNS API を直接呼び出して、作成したオートメーションで通知を送信できます。
  - Amazon Simple Notification Service とは

リソース

関連するドキュメント:

- 静的しきい値に基づいて CloudWatch アラームを作成する
- 「Amazon EventBridge とは?」
- Amazon Simple Notification Service とは

REL 12 信頼性はどのようにテストすればよいですか?

本番環境のストレスに耐えられるようにワークロードを設計した後、ワークロードが意図したとおりに動作し、期待する弾力性を実現することを確認する唯一の方法が、テストを行うことです。

ベストプラクティス

- REL12-BP01 プレイブックを使用して障害を調査する
- REL12-BP02 インシデント後の分析を実行する
- REL12-BP03 機能要件をテストする
- REL12-BP04 スケーリングおよびパフォーマンス要件をテストする
- REL12-BP05 カオスエンジニアリングを使用して回復力をテストする
- REL12-BP06 定期的にゲームデーを実施する

REL12-BP01 プレイブックを使用して障害を調査する

調査プロセスをプレイブックに文書化することで、よく理解されていない障害シナリオに対する一貫性のある迅速な対応が可能になります。プレイブックは、障害シナリオの原因となる要因を特定するために実行される事前定義されたステップです。プロセスステップの結果は、問題が特定されるか、エスカレーションされるまで、次のステップを決定するために使用されます。

プレイブックは、対応措置を効果的に実行できるようにするために立てる必要があるプロアクティブな計画です。本番環境でプレイブックに含まれていない障害シナリオが発生した場合は、まず問題に対処します (火を消します)。その後、振り返って問題に対処するために実行した手順を見て、これらの手順を用いてプレイブックに新しいエントリを追加します。

プレイブックは特定のインシデントに対応するために用いられる一方、ランブックは特定の結果を達成するために使用されます。多くの場合、ランブックは日常的なアクティビティに用いられる一方、プレイブックは非日常的なイベントに応えるために使用します。

一般的なアンチパターン:

- 問題の診断やインシデントへの対応を行うためのプロセスを知ることなくワークロードのデプロイを計画する。
- イベントを調査するときに、ログとメトリクスを収集するシステムに関する計画外の決定。
- データを取得するためにメトリクスとイベントを十分な期間保持していない。

このベストプラクティスを活用するメリット: プレイブックをキャプチャすることで、プロセスへの一貫した遵守が実現できます。プレイブックを成文化することによって、手動のアクティビティによるエラーの発生が抑制されます。プレイブックのオートメーションは、チームメンバーの介入の必要性をなくし、または介入の開始時に追加情報を提供することによって、イベントへの対応時間を短縮します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- プレイブックを使用して問題を特定します。プレイブックは、問題を調査するための文書化されたプロセスです。プロセスをプレイブックに文書化することで、障害シナリオに対する一貫性のある迅速な対応が可能になります。プレイブックには、十分なスキルを持った人物が該当する情報の収集、障害の潜在的原因の特定、障害の切り分け、寄与する要因の特定 (インシデント後の分析の実行) を行うために必要な情報とガイダンスが含まれている必要があります。
  - プレイブックをコードとして実装します。プレイブックをスクリプト化することにより、運用をコードとして実行し、一貫性を保ち、手動プロセスによって発生するエラーを抑制または低減します。プレイブックは、問題に寄与する要因を特定するために必要となり得るさまざまなステップを表す複数のスクリプトで構成できます。ランブックのアクティビティは、プレイブックのアクティビティの一部としてトリガーまたは実行するか、特定されたイベントへの応答としてプレイブックの実行を引き起こす場合があります。
    - AWSSystems Manager を使用して運用上のプレイブックを自動化する
    - AWS Systems Manager Run コマンド
    - AWS Systems Manager Automation をトリガーして
    - 「AWS Lambda とは何ですか?」
    - 「Amazon EventBridge とは?」
    - Amazon CloudWatch アラームの使用

リソース

関連するドキュメント:

- AWS Systems Manager Automation をトリガーして
- AWS Systems Manager Run コマンド
- AWSSystems Manager を使用して運用上のプレイブックを自動化する
- Amazon CloudWatch アラームの使用

- カナリアの使用 (Amazon CloudWatch Synthetics)
- 「Amazon EventBridge とは?」
- 「AWS Lambda とは何ですか?」

関連する例:

- プレイブックとランブックによるオペレーションの自動化

REL12-BP02 インシデント後の分析を実行する

顧客に影響を与えるイベントを確認し、寄与する要因と予防措置を特定します。この情報を使用して、再発を制限または回避するための緩和策を開発します。迅速で効果的な対応のための手順を開発します。対象者に合わせて調整された、寄与する要因と是正措置を必要に応じて伝えます。必要に応じて根本原因を他の人に伝える方法を確立します。

既存のテストで問題が見つからなかった理由を評価します。テストがまだ存在しない場合は、このケースのテストを追加します。

一般的なアンチパターン:

- 寄与因子を見つけるが、他の潜在的な問題やリスクの軽減策についてさらに詳しく調べない。
- 人的エラーの原因を特定するだけで、人的ミスを防止し得るトレーニングやオートメーションを実施しない。

このベストプラクティスを活用するメリット: インシデント後の分析を実施し、結果を共有することで、他のワークロードが同じ寄与因子を実装した場合のリスクを軽減し、インシデントが発生する前に軽減策または自動復旧を実装できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- インシデント後の分析の標準を確立します。優れたインシデント後の分析は、システムの別の場所で使用されているアーキテクチャパターンの問題に対して、共通のソリューションを提案する機会になります。
  - 寄与する要因の記述が正直であり、非難の対象にならないようにします。
  - 問題を文書化しないと、問題を修正できません。

    - 提案された是正措置を冷静に検討し、アプリケーションチームにおける誠実な自己評価とコラボレーションを促進できるようにするため、インシデント後の分析が非難の場にならないようにします。
- プロセスを使用して、寄与した要因を判断します。イベントに寄与した要因を特定してドキュメント化するプロセスを用意しておき、再発を抑制または防止する緩和策と、迅速で効果的な対応手順を展開できるようにしておきます。対象者に合わせて調整された、寄与因子を必要に応じて伝えます。
    - ログ分析とは?

リソース

関連するドキュメント:

- ログ分析とは?
- エラーの修正 (COE) を開発すべき理由

REL12-BP03 機能要件をテストする

必要な機能を検証する単体テストや統合テストなどの技法を使用します。

これらのテストがビルドおよびデプロイアクションの一部として自動的に実行されると、最良の結果が得られます。例えば、デベロッパーは AWS CodePipeline を使用して、CodePipeline が変更を自動的に検出するソースリポジトリに変更をコミットします。このような変更が構築されたら、テストが実行されます。テストが完了すると、ビルドされたコードがテストのためステージングサーバーにデプロイされます。CodePipeline はステージングサーバーから統合テストや負荷テストなど、より多くのテストを実行します。これらのテストが正常に完了すると、CodePipeline はテストおよび承認されたコードを本番稼働インスタンスにデプロイします。

また、経験上、合成トランザクションテスト (カナリアテストとも呼ばれますが、カナリアデプロイと混同しないでください) は、顧客の行動を実行およびシミュレートでき、最も重要なテストプロセスの 1 つです。さまざまなリモートロケーションからワークロードエンドポイントに対してこれらのテストを常に実行します。Amazon CloudWatch Synthetics を使用すると、 Canary を作成して エンドポイントと API をモニタリングできます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- 機能要件をテストします。これには、必要な機能を検証する単体テストと統合テストが含まれます。
  - AWS CodeBuild で CodePipeline を使用してコードをテストし、ビルドを実行する
  - AWS CodePipeline が、AWS CodeBuild を使用した単体テストとカスタム統合テストのサポートを追加
  - 継続的デリバリーと継続的インテグレーション
  - カナリアの使用 (Amazon CloudWatch Synthetics)
  - ソフトウェアテストのオートメーション

リソース

関連するドキュメント:

- APN パートナー: 継続的インテグレーションパイプラインの実装を支援できるパートナー
- AWS CodePipeline が、AWS CodeBuild を使用した単体テストとカスタム統合テストのサポートを追加
- AWS Marketplace: 継続的インテグレーションに利用できる製品
- 継続的デリバリーと継続的インテグレーション
- ソフトウェアテストのオートメーション
- AWS CodeBuild で CodePipeline を使用してコードをテストし、ビルドを実行する
- カナリアの使用 (Amazon CloudWatch Synthetics)

REL12-BP04 スケーリングおよびパフォーマンス要件をテストする

負荷テストなどの技法を使用して、ワークロードがスケーリングおよびパフォーマンス要件を満たしていることを検証します。

クラウドでは、ワークロードに合わせて、本番稼働規模のテスト環境を作成できます。スケールダウンしたインフラストラクチャでこれらのテストを実行する場合、観測された結果を、本番環境で予想される事態にスケーリングする必要があります。実際のユーザーに影響を与えないように注意する場合は、本番環境でも負荷テストとパフォーマンステストを行います。その際、実際のユーザーデータと混合したり、使用統計や本番レポートを破損しないないようにテストデータにタグを付けます。

テストでは、ベースリソース、スケーリング設定、サービスクォータ、および弾力性設計が負荷がかかる時に想定どおりに動作することを確認します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- スケーリングおよびパフォーマンス要件をテストします。ワークロードがスケーリングおよびパフォーマンスの要件を満たしていることを検証するための負荷テストを実行します。
  - AWS での分散負荷テスト: 接続された数千のユーザーをシミュレートする
  - Apache JMeter
    - 本番環境と同じ環境にアプリケーションをデプロイして、負荷テストを実施します。
      - コードとしてのインフラストラクチャの概念を使用して、できるだけ本番環境と類似した環境を作成する

リソース

関連するドキュメント:

- AWS での分散負荷テスト: 接続された数千のユーザーをシミュレートする
- Apache JMeter

REL12-BP05 カオスエンジニアリングを使用して回復力をテストする

不利な条件下でシステムがどのように反応するかを理解するために、本番環境またはできるだけそれに近い環境で定期的にカオス実験を行います。

期待される成果:

イベント発生時のワークロードの既知の期待動作を検証する回復力テストに加え、フォールトインジェクション実験や想定外の負荷の注入という形でカオスエンジニアリングを適用し、ワークロードの回復力を定期的に検証します。カオスエンジニアリングと回復力テストの両方を組み合わせることで、ワークロードがコンポーネントの故障に耐え、予期せぬ中断から影響を最小限に抑えて回復できることへの信頼を得ることができます。

一般的なアンチパターン:

- 耐障害性を考慮した設計でありながら、障害発生時にワークロードが全体としてどのように機能するかを検証していない。

- 実際の条件および予期された負荷の下で実験を一切行わない。
- 実験をコードとして処理しないか、開発サイクルを通して維持しない。
- CI/CD パイプラインの一部、またはデプロイの外部のどちらとしても、カオス実験を実行しない。
- どの障害で実験するかを考慮する際に、過去のインシデント後の分析を軽視する。

このベストプラクティスを活用するメリット: ワークロードの回復力を検証するために障害を発生させることにより、耐障害性設計の回復手順が実際の障害発生時にも機能するという信頼性を得られます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

カオスエンジニアリングは、サービスプロバイダ、インフラストラクチャ、ワークロード、コンポーネントレベルにおいて、現実世界の障害 (シミュレーション) を継続的に発生させる機能を提供し、顧客には最小限の影響しか与えません。これにより、チームは障害から学び、ワークロードの回復力を観察、測定、改善することができます。また、イベント発生時にアラートが発せられ、チームに通知されることを確認することもできます。

カオスエンジニアリングを継続的に実施することで、ワークロードの欠陥が浮き彫りになり、そのままにしておくと可用性や オペレーションに悪影響が及ぶ可能性があります。

> **Note**
>
> カオスエンジニアリングとは、あるシステムで実験を行い、実稼働時の混乱状態に耐えることができるかどうかの確信を得るための手法です。 カオスエンジニアリングの原則

もし、システムがこれらの混乱に耐えられるなら、カオス実験は自動化された回帰テストとして維持されるはずです。このように、カオス実験はシステム開発ライフサイクル (SDLC) の一部および CI/CD の一部として実行される必要があります。

ワークロードがコンポーネントの障害に耐えられることを確認するために、実際のイベントを実験の一部として挿入します。たとえば、Amazon EC2 インスタンスの喪失やプライマリ Amazon RDS データベースインスタンスのフェイルオーバーを実験し、ワークロードに影響がないこと (または最小限の影響にとどまること) を確認します。コンポーネントの障害の組み合わせを使用して、アベイラビリティーゾーンで中断によって発生する可能性のあるイベントをシミュレートします。

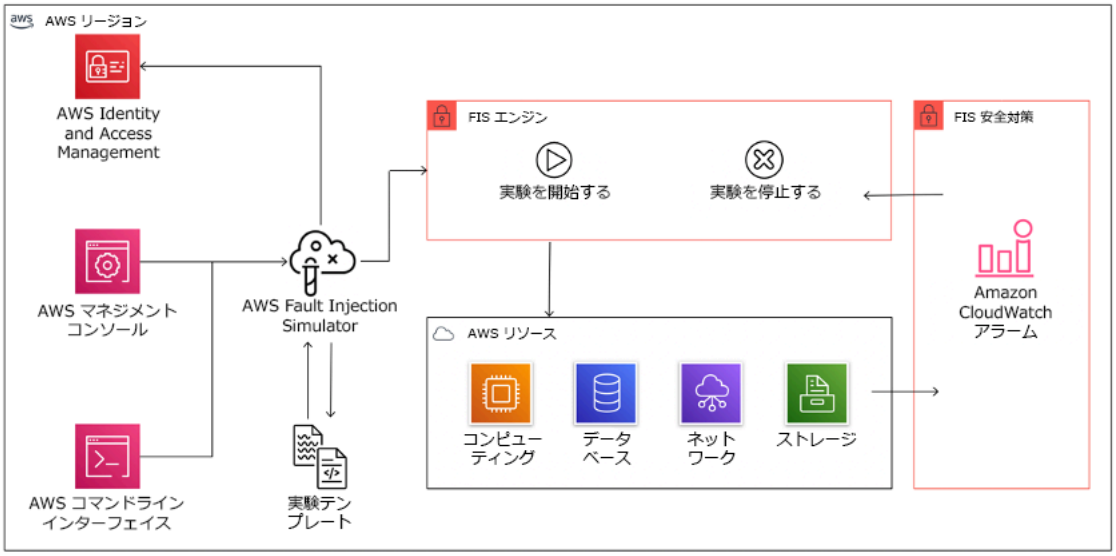
アプリケーションレベルの障害 (クラッシュなど) では、メモリや CPU の枯渇などのストレス要因から始めます。

断続的なネットワークの中断による外部依存のための フォールバックまたはフェイルオーバーメカニズム を検証するために、コンポーネントは、数秒から数時間の指定された期間、サードパーティプロバイダへのアクセスをブロックすることにより、そのようなイベントをシミュレートする必要があります。

その他の劣化モードでは、機能の低下や応答の遅れが発生し、サービスの中断につながることがよくあります。このパフォーマンス低下の一般的な原因は、主要サービスのレイテンシー増加と、信頼性の低いネットワーク通信 (パケットのドロップ) です。レイテンシー、メッセージのドロップ、DNS 障害などのネットワーク効果を含むこれらの障害の実験には、名前を解決できない、DNS サービスへリーチできない、または依存サービスへの接続を確立できないなどの可能性があります。

カオスエンジニアリングのツール:

AWS Fault Injection Service (AWS FIS) は、フォールトインジェクション実験を実行する完全マネージド型サービスであり、CD パイプラインの一部として、またはパイプラインの外で使用することができます。AWS FIS は、カオスエンジニアリングゲームデー中に使用するのに適しています。Amazon EC2、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、および Amazon RDS などを含む、異なるタイプのリソースに同時に障害を導入することをサポートします。これらの障害には、リソースの終了、強制フェイルオーバー、CPU にストレスをかける、スロットリング、またはメモリー、レイテンシー、およびパケットの損失が含まれます。Amazon CloudWatch アラームと連携しているため、ガードレールとして停止条件を設定し、予期せぬ影響を与えた場合に実験をロールバックすることができます。



ワークロードのフォールトインジェクション実験を実行するため、AWS リソースと統合された AWS Fault Injection Service。

フォールトインジェクション実験のためのサードパーティオプションもいくつかあります。これには、次のようなオープンソースのツールが含まれます。 Chaos Toolkit、Chaos Mesh、および Litmus Chaos、Gremlin などの商用オプションです。AWS に挿入できる障害のスコープを拡張するために、AWS FIS は Chaos Mesh および Litmus Chaos と統合して、複数のツール間でフォールトインジェクションワークフローの調整を可能にします。たとえば、AWS FIS 障害アクションを使用して、ランダムに選択した割合のクラスターノードを終了する間に、Chaos Mesh または Litmus 障害を使用してポッドの CPU のストレステストを実行することができます。

実装手順

- どの障害を実験に使用するかを決定する。

  回復力に関してワークロードの設計を評価します。そのような設計 (Well-Architected フレームワークのベストプラクティスを使用して作成) では、重要な依存関係、過去のイベント、既知の問題、およびコンプライアンス要件に基づくリスクが考慮されています。回復力を維持するために意図された設計の各要素と、それを軽減するために設計された障害をリストアップします。そのようなリストの作成に関する詳細については、 運用準備状況の確認に関するホワイトペーパー を確認し、過去のインシデントの再発を防ぐためのプロセスを作成する方法を学びます。障害モードと影響の分析 (FMEA) プロセスにより、障害とそれがワークロードに与える影響をコンポーネントレベルで解析するためのフレームワークが提供されます。FMEA については Adrian Cockcroft 氏による「 Failure Modes and Continuous Resilience」内に詳しく記載されています。
- それぞれの障害に優先度を割り当てる。

  「高」「中」「低」などの大まかな分類から始めます。優先度を評価するためには、障害の発生頻度と障害によるワークロード全体への影響度を考慮します。

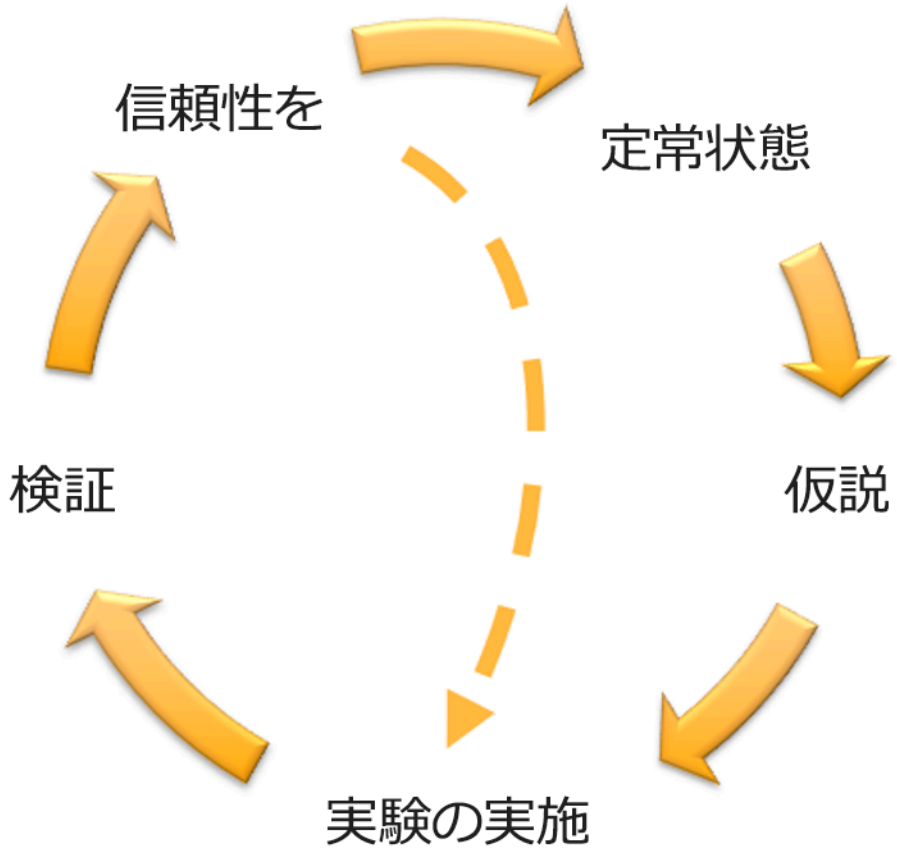
  ある障害の発生頻度を考慮する場合、利用可能であれば、そのワークロードの過去のデータを分析します。利用できない場合は、類似の環境において実行されている他のワークロードのデータを使用します。

  ある障害の影響を考慮する場合、一般に、障害の範囲が大きければ大きいほど、影響も大きくなります。また、ワークロードの設計と目的も考慮します。たとえば、ソースデータストアにアクセスする機能は、データ変換や分析を行うワークロードにとって重要です。この場合、アクセス障害、スロットルアクセス、レイテンシー挿入の実験を優先させることになります。

障害発生後の分析は、障害モードの頻度と影響の両方を理解するための良いデータソースとなります。

割り当てた優先度を使用して、どの障害を最初に実験するか、および新しいフォールトインジェクション実験を開発する順序を決定します。

- 実行するそれぞれの実験に対して、カオスエンジニアリングと継続的な回復力のフライホイールに従います。



Adrian Hornsby 氏による、科学的手法を用いたカオスエンジニアリングと継続的な回復力のフライホイール。

- 定常状態とは、正常な動作を示すワークロードの測定可能な出力であると定義する。

  ワークロードは、信頼性が高く、期待通りに動作していれば、定常状態を示しています。したがって、定常状態を定義する前に、ワークロードが正常であることを検証します。障害が発生した場合、一定の割合で許容範囲内である可能性があるため、定常状態は、必ずしもワークロード

に影響を与えないことを意味するものではありません。定常状態は、実験中に観察する基準値であり、次のステップで定義した仮説が予想通りにならない場合に、異常が浮き彫りになります。

たとえば、決済システムの定常状態は、成功率 99%、往復時間 500ms で 300TPS を処理することと定義することができます。

- ワークロードがどのように障害に対応するかについての仮説を立てる。

  良い仮説とは、定常状態を維持するために、ワークロードがどのように障害を軽減すると予想されるかに基づいています。仮説は、特定のタイプの障害が発生した場合、システムまたはワークロードが定常状態を維持することを示しています。なぜなら、ワークロードは特定の緩和策を講じて設計されているからです。特定の種類の障害および緩和策は、仮説の中で特定さ れる必要があります。

  次のテンプレートが仮説に使用できます (ただし、他の表現も許容されます)。

  > **Note**
  >
  > もし ##### が発生した場合、 ####### ワークロードは ####### して ########## ##############。

  例:

  - Amazon EKS ノードグループの 20% のノードが停止しても、[Transaction Create API] は 100 ミリ秒未満で 99% のリクエストに対応し続けます (定常状態)。Amazon EKS ノードは 5 分以内に回復し、ポッドは実験開始後 8 分以内にスケジューリングされてトラフィックを処理するようになります。3 分以内にアラートが発せられます。
  - Amazon EC2 インスタンスの単一障害が発生した場合、注文システムの Elastic Load Balancing ヘルスチェックにより、Amazon EC2 Auto Scaling が障害インスタンスを置き換える間、Elastic Load Balancing は残りの健全なインスタンスにのみリクエストを送信し、サーバーサイド (5xx) エラーの増加を 0.01% 未満に維持します (定常状態)。
  - プライマリ Amazon RDS データベースインスタンスに障害が発生した場合、サプライチェーンデータ収集ワークロードはフェイルオーバーし、スタンバイ Amazon RDS データベースインスタンスに接続して、データベースの読み取りまたは書き込みエラーを 1 分未満に維持します (定常状態)。

- 障害を挿入して実験を実行する。

実験はデフォルトでフェイルセーフであり、ワークロードが耐えることができる必要があります。ワークロードが失敗することが分かっている場合は、実験を実行しないでください。カオスエンジニアリングは、既知の未知、または未知なる未知を見つけるために使用されるべきです。既知の未知 とは、認識はしているが完全に理解していないことであり、 未知なる未知 は、認識も完全な理解もしていないことです。壊れていると分かっているワークロードに対して実験を行っても、新しいインサイトを得ることはできません。実験は慎重に計画し、影響の範囲を明確にし、予期せぬ乱れが発生した場合に適用できるロールバックメカニズムを提供する必要があります。デューディリジェンスにより、ワークロードが実験に耐えられることが分かったら、実験を進めてください。障害を挿入するには、いくつかの方法があります。AWS 上のワークロードに対して、AWS FIS は多くの事前定義された障害シミュレーションを挿入します。これは、アクションと呼ばれます。また、AWS FIS で実行するカスタムアクションも定義します。これには AWS Systems Manager ドキュメントを使用します。

カオス実験にカスタムスクリプトを使用することは、スクリプトがワークロードの現在の状態を理解する機能を持ち、ログを出力でき、可能であればロールバックと停止条件のメカニズムを提供しない限り、推奨されません。

カオスエンジニアリングをサポートする効果的なフレームワークやツールセットは、実験の現在の状態を追跡し、ログを出力し、実験の制御された実行をサポートするためのロールバックメカニズムを提供する必要があります。AWS FIS のように、実験範囲を明確に定義し、実験によって予期せぬ乱れが生じた場合に実験をロールバックする安全なメカニズムを備えた実験を行うことができる、確立されたサービスから始めてみてください。AWS FIS を使用した、より幅広い実験に関する詳細については、「 カオスエンジニアリングラボでの回復力と Well-Architected アプリ」も参照してください。また、 AWS Resilience Hub はワークロードを分析し、AWS FIS で実装、実行することを選択できるような実験を作成します。

> **Note**
>
> すべての実験について、その範囲と影響を明確に理解します。本番環境で実行する前に、まず非本番環境で障害をシミュレートすることをお勧めします。

実験は、可能な限り、対照系と実験系の両方をスピンアップする canary デプロイ を使用して、実世界の負荷で実稼働させる必要があります。オフピークの時間帯に実験を行うことは、本番で初めて実験を行う際に潜在的な影響を軽減するための良い習慣です。また、実際の顧客トラフィックを使用するとリスクが高すぎる場合は、本番インフラストラクチャの制御環境と実験環

境に対して合成トラフィックを使用して実験を実行することができます。本番環境での実験が不可能な場合は、できるだけ本番環境に近いプレ本番環境で実験を行ってください。

実験が本番トラフィックや他のシステムに許容範囲を超えて影響を与えないように、ガードレールを確立してモニタリングする必要があります。停止条件を設定し、定義したガードレールのメトリクスでしきい値に達した場合に実験を停止するようにします。これには、ワークロードの定常状態のメトリクスと、障害を注入するコンポーネントに対するメトリク スを含める必要があります。ユーザー canary とも呼ばれる 合成モニタリング は、通常、ユーザープロキシとして含む必要がある 1 つのメトリクスです。AWS FIS への停止条件 は、実験テンプレートの一部としてサポートされており、1 テンプレートあたり最大 5 つの停止条件を設定することが可能です。

カオスの原則の 1 つは、実験の範囲とその影響を最小化することです。

短期的な悪影響は許容されなければなりませんが、実験の影響を最小化し、抑制することはカオスエンジニアの責任であり義務です。

範囲や潜在的な影響を検証する方法としては、本番環境で直接実験を行うのではなく、まず非本番環境で実験を行い、実験中に停止条件のしきい値が想定通りに作動するか、例外をキャッチするための観測性があるかどうかを確認することが挙げられます。

フォールトインジェクション実験を実施する場合、すべての責任者に情報が十分に提供されるようにします。オペレーションチーム、サービス信頼性チーム、カスタマーサポートなどの適切なチームとコミュニケーションをとり、実験がいつ実行され、何を期待されるかを伝えます。これらのチームには、何か悪影響が見られた場合に、実験を行っているチームに知らせるためのコミュニケーションツールを与えます。

ワークロードとその基盤となるシステムを元の既知の良好な状態に復元する必要があります。多くの場合、ワークロードの回復力のある設計が自己回復を行います。しかし、一部の障害設計や実験の失敗により、ワークロードが予期せぬ障害状態に陥ることがあります。実験が終了するまでに、このことを認識し、ワークロードとシステムを復旧させなければなりません。AWS FIS では、アクションのパラメータ内にロールバック設定 (ポストアクションとも呼ばれる) を設定することができます。ポストアクションは、アクションが実行される前にある状態にターゲットを返します。自動化 (AWS FIS の使用など) であれ手動であれ、これらのポストアクションは、障害を検出し処理する方法を説明するプレイブックの一部であるべきです。

- 仮説を検証する。

  カオスエンジニアリングの原則 は、ワークロードの定常状態を検証する方法について、このようなガイダンスを示しています。

システムの内部属性ではなく、測定可能な出力に焦点を当てます。その出力を短期間で測定することによって、システムの安定状態のプロキシが構成されます。システム全体のスループット、エラーレート、レイテンシーのパーセンタイルはすべて、定常状態の動作を表す重要なメトリクスになり得ます。実験中にシステム的な動作パターンに注目することで、カオスエンジニアリングは、システムがどのように動作するかを検証するのではなく、システムが動作していることを検証します。

先の 2 つの例では、サーバーサイド (5xx) エラーの増加率が 0.01% 未満、データベースの読み取りと書き込みのエラーが 1 分未満という定常状態の測定基準が含まれています。

5xx エラーは、ワークロードのクライアントが直接経験する障害モードの結果であるため、良いメトリクスです。データベースエラーの測定は、障害の直接的な結果として適切ですが、顧客からのリクエストの失敗や、顧客に表面化したエラーなど、顧客への影響も測定して補足する必要があります。さらに、ワークロードのクライアントが直接アクセスする API や URI に、合成モニタリング (ユーザー canary としても知られる) を含めるようにしましょう。

- 回復力を高めるためのワークロード設計を改善する。

  定常状態が維持されなかった場合、障害を軽減するためにワークロード設計をどのように改善できるかを調査し、 AWS Well-Architected 信頼性の柱のベストプラクティスを適用します。その他のガイダンスとリソースは AWS Builder's Libraryにあり、ここでは、他の記事と共に ヘルスチェックを見直す 方法、または アプリケーションコード内のバックオフを使用した再試行の採用に関する記事が掲載されています。

  これらの変更を実施した後、再度実験を行い (カオスエンジニアリングフライホイールの点線で表示)、その効果を判断します。検証の結果、仮説が正しいことがわかれば、ワークロードは定常状態になり、このサイクルが続きます。

- 実験を定期的に実施する。

  カオス実験はサイクルであり、実験はカオスエンジニアリングの一環として定期的に実施される必要があります。ワークロードが実験の仮説を満たした後、CI/CD パイプラインの回帰部分として継続的に実行されるように実験を自動化する必要があります。この方法については、このブログの「 how to run AWS FIS experiments using AWS CodePipeline」を参照してください。このラボでは、 CI/CD パイプラインで AWS FIS 実験 を繰り返し行うことで、実践的に作業することができます。

  フォールトインジェクション実験は、ゲームデーの一部でもあります (参照: REL12-BP06 定期的にゲームデーを実施する)。ゲームデーでは、障害やイベントをシミュレートし、システム、プロ

セス、チームの対応を検証します。その目的は、例外的な出来事が発生した場合にチームが実行することになっているアクションを実際に実行することです。

- 実験結果をキャプチャし、保存する。

  フォールトインジェクション実験の結果は、キャプチャおよび保持される必要があります。実験結果や傾向を後で分析できるように、必要なデータ (時間、ワークロード、条件など) をすべて含めておきましょう。結果の例には、ダッシュボードのスクリーンショット、メトリクスのデータベースからの CSV ダンプ、実験中のイベントや観察結果を手書きで記録したものなどがあります。AWS FIS での実験記録 もデータキャプチャの一部となり得ます。

リソース

関連するベストプラクティス:

- REL08-BP03 デプロイの一部として回復力テストを統合する
- REL13-BP03 ディザスタリカバリの実装をテストし、実装を検証する

関連するドキュメント:

- AWS Fault Injection Service とは
- AWS Resilience Hub とは
- カオスエンジニアリングの原則
- カオスエンジニアリング: 最初の実験を計画する
- 回復力エンジニアリング: 失敗から学ぶ
- カオスエンジニアリングのストーリー
- 分散システムでのフォールバックの回避
- カオス実験の canary デプロイ

関連動画:

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

関連する例:

- Well-Architected ラボ: レベル 300: Amazon EC2 Amazon RDS と Amazon S3 の回復力をテストする
- AWS ラボでのカオスエンジニアリング
- カオスエンジニアリングラボでの回復力と Well-Architected アプリ
- サーバーレスカオスラボ
- アプリケーションの回復力を AWS Resilience Hub ラボを使用して測定し、向上させる

関連ツール:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh
- Litmus

REL12-BP06 定期的にゲームデーを実施する

ゲームデーを使用して、実際の障害シナリオに関わる人々と、可能な限り本番環境に近い環境 (本番環境を含む) でのイベントと障害の対処手順を定期的に実行します。ゲームデーでは、本番環境のイベントがユーザーに影響を与えないようにするための対策を講じます。

ゲームデーでは、障害やイベントをシミュレーションして、システム、プロセス、チームの対応をテストします。その目的は、例外的な出来事が発生した場合にチームが実行することになっているアクションを実際に実行することです。これは、改善できる箇所を把握し、組織がイベントに対応することを経験するのに役#ちます。こうしたゲームデーを定期的に実施することで、チームは対応方法に関する 「基礎体力」 をつけることができます。

弾力性を考慮した設計が整い、本番環境以外の環境でテストした後、本番環境ですべてが計画どおりに機能することを確認するのがゲームデーです。ゲームデー、特に初日は、「全員が総力を挙げた」取り組みです。いつ起こるか、そして何が起こるかについてエンジニアと運用担当者に通知します。ランブックを用意します。障害イベントも含めて、シミュレートされたイベントが本番稼働システムで所定の方法で実行され、影響が評価されます。すべてのシステムが設計どおりに動作すると、検出と自己修復が行われ、影響はほとんどありません。ただし、負の影響が観察された場合、テストはロールバックされ、ワークロードの問題が必要に応じて (ランブックを参照して) 手動で修正されます。ゲームデーは本番環境で行われることが多いため、顧客の可用性に影響を与えないように、あらゆる予防策を講じる必要があります。

一般的なアンチパターン:

- 手順は文書化するが、実行しない。
- テスト演習にビジネス上の意思決定者を含めない。

このベストプラクティスを活用するメリット: ゲームデーを定期的に実施することで、実際のインシデントが発生したときにすべてのスタッフがポリシーと手順に従っていることを確認し、それらのポリシーと手順が適切であることを検証できます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- ゲームデーをスケジュールして定期的にランブックおよびプレイブックを使ってみるゲームデーには、事業主、開発スタッフ、運用スタッフ、インシデント対応チームといった、本番環境でのイベントに関与すると思われるすべての人員が参加する必要があります。
  - 負荷テストやパフォーマンステストを実施した後、障害注入を実施します。
  - ランブックのおかしな点やプレイブックを使う機会を探します。
    - ランブックから逸脱したら、対応マニュアルを改善するか行動を修正します。プレイブックを使用したら、使用すべきだったランブックを特定するか新しいランブックを作成します。

リソース

関連するドキュメント:

- AWS GameDay とは?

関連動画:

- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)

関連する例:

- AWS Well-Architected ラボ - Testing Resiliency

REL 13 災害対策 (DR) はどのように計画するのですか?

バックアップと冗長ワークロードコンポーネントを配置することは、DR 戦略の出発点です。RTO と RPO は ワークロードの復旧目標です。これは、ビジネスニーズに基づいて設定します。ワークロードのリソースとデータのロケーションと機能を考慮して、目標を達成するための戦略を実装します。ワークロードの災害対策を提供することのビジネス価値を伝えるには、中断の可能性と復旧コストも重要な要素となります。

ベストプラクティス

- REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する
- REL13-BP02 復旧目標を満たすため、定義された復旧戦略を使用する
- REL13-BP03 ディザスタリカバリの実装をテストし、実装を検証する
- REL13-BP04 DR サイトまたはリージョンでの設定ドリフトを管理する
- REL13-BP05 復旧を自動化する

REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する

ワークロードには、目標復旧時間 (RTO) と目標復旧時点 (RTO) が定義されます。

目標復旧時間 (RTO) RTO は、サービスの中断からサービスの復元までの最大許容遅延です。これにより、サービスが利用できないときに許容可能と見なされる時間枠が決まります。

目標復旧時点 (RPO) RPO は、最後のデータ復旧ポイントからの最大許容時間です。これにより、最後の復旧ポイントからサービスの中断までの間に許容可能と見なされるデータ損失が決まります。

RTO 値と RPO 値は、ワークロードに適したディザスタリカバリ (DR) 戦略を選択する際の重要な考慮事項です。これらの目標は企業によって決定され、技術チームによって DR 戦略の選択と実装のために使用されます。

期待される成果:

すべてのワークロードに、ビジネスへの影響に基づいて定義された RTO と RPO が割り当てられます。ワークロードが事前に定義された改装に割り当てられ、該当する RTO および RPO とともに、サービスの可用性と許容可能なデータ損失を定義します。そのような階層化ができない場合には、後で階層を作成する目的で、ワークロードごとに別注を割り当てることもできます。RTO と RPO は、ワークロードのディザスタリカバリ戦略実装を選択する際の主要な考慮事項の 1 つとして使用されます。DR 戦略を選択する際のその他の考慮事項としては、コストの制約、ワークロードの依存関係、運用要件があります。

RTO については、停止時間に基づく影響を理解してください。線形か、それとも非線形の意味合いがあるか (例えば、4 時間後に、次のシフトの開始まで製造ラインをシャットダウンしておく) 。

次のようなディザスタリカバリマトリックスは、ワークロードが復旧目標にどの程度関係しているかを理解するのに役立ちます。(X 軸と Y 軸の実際の値は、組織のニーズに合わせてカスタマイズしてください)。

ディザスタリカバリマトリックス						
		目標復旧時点				
		1 分未満	1 時間未満	6 時間未満	1 日未満	1 日以上
目標復旧時間	10 分未満	重要	重要	高	中	中
	2 時間未満	重要	高	中	中	低
	8 時間未満	高	中	中	低	低
	24 時間未満	中	中	低	低	低
	24 時間以上	中	低	低	低	低

図16: ディザスタリカバリマトリックス

一般的なアンチパターン:

- 復旧目標を定義していない。
- 任意の復旧目標を選択する。
- 過度に寛大で、ビジネス目標を満たさない復旧目標を選択する。
- ダウンタイムとデータ損失の影響を理解していない。
- 復旧時間ゼロやデータ損失ゼロなど、ワークロード設定では達成できない恐れのある非現実的な復旧目標を選択する。
- 実際のビジネス目標よりも厳格な復旧目標を選択する。これにより、ワークロードが必要とするよりもコストが高く、複雑な DR 実装を強いられます。
- 依存するワークロードの復旧目標とは互換性のない復旧目標を選択する。
- 復旧目標で規制コンプライアンス要件が考慮されていない。
- ワークロードの RTO と RPO は定義されたが、テストされていない。

このベストプラクティスを活用するメリット: 時間とデータ損失の復旧目標は、DR 実装の指針とするために必要です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

特定のワークロードについて、ダウンタイムとデータ損失がビジネスに与える影響を理解する必要があります。一般に、ダウンタイムが長いほど、またはデータ損失が大きいほど、影響は増加しますが、この増加の形状はワークロードのタイプによって異なります。例えば、1 時間までのダウンタイムなら耐えられ、影響もほとんどないかもしれませんが、その後は影響が急増するかもしれません。ビジネスへの影響は、金銭的コスト (減益など)、顧客の信頼 (と評判への影響)、運用上の問題 (給与未払いや生産性の低下など)、規制リスクなど、多くの形態をとります。以下のステップを使用して、これらの影響を理解し、ワークロードの RTO と RPO を設定してください。

実装手順

1. このワークロードのビジネスステークホルダーを決め、これらのステップを実装するように促します。ワークロードの復旧目標は、ビジネス上の決定です。技術チームはビジネスステークホルダーと協力して、これらの目標に基づいて DR 戦略を選択します。

> **Note**
>
> ステップ 2 と 3 については、以下を使用してください。 the section called “実装ワークシート”.

2. 以下の質問に答えることによって、決定を下すために必要な情報を集めます。
3. ワークロードが組織に与える影響について、重要度のカテゴリまたは階層がありますか?
   a. ある場合、このワークロードをカテゴリに割り当てます。
   b. ない場合は、これらのカテゴリを確立します。5 つ以下のカテゴリを作成し、それぞれの目標復旧時間の範囲を絞り込みます。カテゴリの例としては、重要、高、中、低などがあります。ワークロードがどのようにカテゴリにマッピングされるかを理解するには、ワークロードがミッションクリティカルであるか、ビジネスにとって重要であるか、それともビジネスを駆動するものではないかを考慮します。
   c. カテゴリに基づいて、ワークロードの RTO と RPO を設定します。このステップに入るときに計算した元の値より厳しいカテゴリ (低い RTO および RPO) を選ぶようにします。この結果、値の変化が不適切に大きくなる場合には、新しいカテゴリの作成を検討します。
4. これらの回答に基づいて、RTO 値と RPO 値をワークロードに割り当てます。これは直接行うか、ワークロードを事前定義のサービス階層に割り当てることで行うことができます。

5. このワークロードのディザスタリカバリプラン (DRP) を文書化し、組織の ビジネス継続性計画 (BCP)の一部とし、ワークロードチームとステークホルダーがアクセスできる場所に保管します。
   a. RTO および RPO と、これらの値を決めるために使用した情報を記録します。ワークロードがビジネスに与える影響を評価するために使用した戦略も含めます。
   b. RTO と RPO のほかに、ディザスタリカバリ目標のために追跡しているか、追跡する予定のその他のメトリクスも記録します。
   c. DR 戦略とランブックを作成したときには、これらの詳細をこのプランに追加します。
6. 図 15 のようなマトリックスでワークロードの重要性を調べることで、組織で定義される事前定義のサービス階層の確立を開始できます。
7. に従って DR 戦略 (または DR 戦略の概念実証) を実装した後the section called “REL13-BP02 復旧目標を満たすため、定義された復旧戦略を使用する”、この戦略をテストして、ワークロードの実際の RTC (復旧時間機能) と RPC (復旧時点機能) を判断します。これらがターゲットの復旧目標を満たさない場合は、ビジネスステークホルダーと協力して目標を調整するか、DR 戦略に変更を加えて、ターゲット目標を満たします。

主な質問

1. ワークロードがダウンしてからビジネスに重大な影響が出るまでの最大時間はどのくらいですか。
   a. ワークロードが中断した場合にビジネスに及ぼす 1 分間あたりの金銭的コスト (直接的な経済的影響) を判断します。
   b. 影響が常に線形とは限らないことを考慮します。影響は最初は限定的でも、臨界時点を超えると急増することがあります。
2. ビジネスに重大な影響が出るデータ損失の最大量はどのくらいですか。
   a. 最も重要なデータストアについて、この値を考慮します。その他のデータストアのそれぞれの重要度を特定します。
   b. ワークロードデータが失われた場合、再作成できますか? これがバックアップと復元よりも運用上容易な場合は、ワークロードデータの再作成に使用されるソースデータの重要度に基づいて RPO を選びます。
3. このワークロードに依存するワークロード (ダウンストリーム) またはこのワークロードが依存するワークロード (アップストリーム) の復旧目標と可用性期待は何ですか?
   a. このワークロードがアップストリームの依存関係の要件を満たすことができる復旧目標を選びます。

b. ダウンストリームの依存関係の復旧機能を前提として達成可能な復旧目標を選びます。重要でないダウンストリームの依存関係 (「対処」できるもの) は除外できます。または、必要な場合は、ダウンストリームの重要な依存関係と協力して、復旧能力を高めます。

その他の質問

以下の質問と、これらがこのワークロードにどのように適用されるか考慮してください。

4. 停止のタイプ (リージョン対AZ など) に応じた異なる RTO および RPO がありますか?
5. RTO/RPO が変更される特定の時期 (季節性、販売イベント、製品の発売) がありますか? その場合、異なる測定と時間境界は何ですか?
6. ワークロードが中断した場合、何人の顧客が影響を受けますか?
7. ワークロードが中断した場合、評判への影響は何ですか?
8. ワークロードが中断した場合に発生する可能性のある、その他の運用上の影響は何ですか? 例えば、E メールシステムが使用できない場合や、給与システムがトランザクションを送信できない場合の従業員の生産性への影響などです。
9. ワークロードの RTO および RPO は基幹業務および組織の DR 戦略とどのように連携しますか?
10. サービスの提供に関する内部契約上の義務がありますか? 満たすことができなかった場合の罰則はありますか?
11. データに関する規制またはコンプライアンス制約は何ですか?

実装ワークシート

このワークシートは、実装ステップ 2 および 3 に使用できます。質問を追加するなど、特定のニーズに応じてこのワークシートを調整することができます。

ステップ 2: 主な質問	ワークロードに適用されますか？	ワークロード RTO	ワークロード RPO	RTO 調整	RPO 調整	手順
[1] ワークロードの最大ダウン時間						サービス停止の発生から復旧までの時間で測定
[2] 失われるデータの最大量						復元可能な最後の既知のデータセットからの時間で測定
[3a] アップストリームの依存関係						最も厳しいアップストリームリカバリ目標を入力します
[3b] ダウンストリームの依存関係						最も緩いダウンストリームリカバリ目標を入力します
[3a] 調整されたアップストリームの依存関係						アップストリームの値が現在の値よりも小さく、ダウンストリームの値が大きい場合は、依存関係を調整し、調整した値をここに入力します
[3b] 調整されたダウンストリームの依存関係						
[3] 依存関係						アップストリームの依存関係を満たすために値を下げるか、ダウンストリームの依存関係の機能に基づいて値を上げます
ステップ 2: その他の質問						質問に該当する場合は、ご記入ください。該当しない場合は、飛ばしてください
ベースの RTO/RPO						上記の RTO と RPO の値をここに入れます
[4] サービス停止の種類	[ ]はい / [ ]いいえ					イベントタイプについて要件が最も厳しい復旧目標を入力します
[5] 特定の時間ベースの目標	[ ]はい / [ ]いいえ					時間について要件が最も厳しい復旧目標を入力します
[6] 顧客の混乱	[ ]はい / [ ]いいえ					ダウンタイムまたはデータ損失の関数として影響を受ける顧客をグラフ化します。これをもとに、顧客への影響を考慮して許容される最大 RTO と RPO を入力します
[7] 評判への影響	[ ]はい / [ ]いいえ					ビジネスと連携し、評判への影響を考慮した最大の RTO と RPO を決定します
[8] 運用上の影響	[ ]はい / [ ]いいえ					運用上の影響に基づいて、最大 RTO と RPO を入力します
[9] 組織の調整	[ ]はい / [ ]いいえ					LOB および組織の要件に従って、このタイプのワークロードの最大 RTO と RPO を入力します
[10] 契約上の義務	[ ]はい / [ ]いいえ					契約上の義務に基づいて、最大 RTO と RPO を入力します
[11] 規制コンプライアンス	[ ]はい / [ ]いいえ					適用される規制コンプライアンスに基づいて、最大 RTO と RPO を入力します
その他の質問に基づくターゲット						Q の 4 ～ 11 から最小値 (より厳しい値) をここに入力します
調整済みターゲット						上記の目標に対応できない場合は、ステークホルダーと協力して制約を緩め、ここに新しい最小値を入力します
調整済み RTO/RPO						ベースの RPO/RTO 値、または調整済みターゲットのいずれか低い方を入力します
ステップ 3						
事前定義されたカテゴリまたは層にマッピング						両方の値を下方 (より厳密) に調整して、最も近い定義された層に合わせます

ワークシート

実装計画の工数レベル: 低

リソース

関連するベストプラクティス:

- the section called “REL09-BP04 データの定期的な復旧を行ってバックアップの完全性とプロセスを確認する”
- the section called “REL13-BP02 復旧目標を満たすため、定義された復旧戦略を使用する”
- the section called “REL13-BP03 ディザスタリカバリの実装をテストし、実装を検証する”

関連するドキュメント:

- AWS アーキテクチャブログ: Disaster Recovery Series
- AWS でのワークロードのディザスタリカバリ: クラウドでの復旧 (AWS ホワイトペーパー)
- AWS レジリエンスハブによる回復力ポリシーの管理
- APN パートナー: 災害対策を支援できるパートナー

- AWS Marketplace: 災害対策に活用できる商品

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Disaster Recovery of Workloads on AWS

REL13-BP02 復旧目標を満たすため、定義された復旧戦略を使用する

ワークロードの復旧目標を満たすディザスタリカバリ (DR) 戦略を定義します。バックアップと復元、スタンバイ (アクティブ/パッシブ)、アクティブ/アクティブなどの戦略を選択します。

DR 戦略は、プライマリロケーションでワークロードを実行できなくなった場合に復旧サイトでワークロードに耐えられる能力に依存します。最も一般的な復旧目標は、RTO と RPO です REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する.

単一の AWS リージョン 内の複数のアベイラビリティゾーン (AZ) にまたがる DR 戦略は、火災、洪水、大規模な停電などの災害イベントに対して影響を緩和できます。ワークロードを特定の AWS リージョン で実行できなくなるような、可能性の低いイベントに対する保護を実装する必要がある場合には、複数のリージョンを使用する DR 戦略を使用できます。

複数のリージョンにまたがる DR 戦略を設計するときには、以下のいずれかの戦略を選んでください。戦略は、コストと複雑さの昇順、および RTO と RPO の降順でリストされています。 復旧リージョン とは、ワークロードで使用されるプライマリ以外の AWS リージョン を指します。

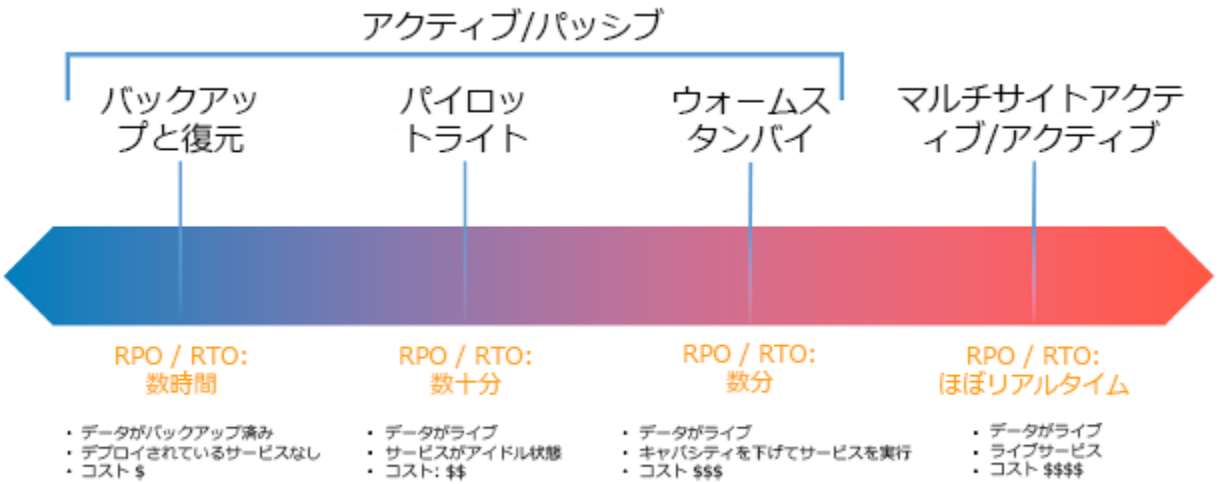


図 17: ディザスタリカバリ (DR) 戦略

- バックアップと復元 (数時間での RPO、24 時間以下での RTO): データとアプリケーションを復旧リージョンにバックアップします。自動化されたバックアップまたは連続バックアップを使用すると、ポイントインタイムリカバリが可能であり、場合によっては RPO を 5 分間に短縮できます。災害の際には、インフラストラクチャをデプロイし (インフラストラクチャをコードとして使用して RTO を削減)、コードをデプロイし、バックアップされたデータを復元して、復旧リージョンで災害から復旧します。
- パイロットライト （ 数分間の RPO、数十分間の RTO): コアワークロードインフラストラクチャのコピーを復旧リージョンにプロビジョニングします。データを復旧リージョンにレプリケートして、そこでバックアップを作成します。データベースやオブジェクトストレージなど、データのレプリケーションとバックアップのサポートに必要なリソースは、常にオンです。アプリケーションサーバーやサーバーレスコンピューティングなど、その他の要素はデプロイされませんが、必要なときには、必須の設定とアプリケーションコードで作成できます。
- ウォームスタンバイ (数秒間の RPO、数分間の RTO): 完全に機能する縮小バージョンのワークロードを復旧リージョンで常に実行している状態に保ちます。ビジネスクリティカルなシステムは完全に複製され、常に稼働していますが、フリートは縮小されています。データは復旧リージョンでレプリケートされ、使用可能です。復旧時には、システムをすばやくスケールアップして本番環境の負荷を処理できるようにします。ウォームスタンバイの規模が大きいほど、RTO とコントロールプレーンへの依存は低くなります。これを完全にスケールアップしたものは ホットスタンバイと呼ばれます.
- マルチリージョン (マルチサイト) アクティブアクティブ (ゼロに近い RPO、ほぼゼロの RTO): ワークロードは複数の AWS リージョン にデプロイされ、そこからトラフィックにアクティブに対応します。この戦略では、複数のリージョン間でデータを同期する必要があります。2 つの異なるリージョンレプリカ内の同じレコードへの書き込みによって生じる矛盾を回避または処理する必要があり、これは複雑になることがあります。データレプリケーションは、データの同期に便利であり、特定のタイプの災害から保護しますが、ソリューションがポイントインタイムリカバリのためのオプションを含んでいない限り、データの破損や破壊からは保護しません。

> **Note**
>
> パイロットライトとウォームスタンバイの違いは、理解しにくいかもしれません。どちらも、プライマリリージョンアセットのコピーがある復旧リージョン内の環境を含みます。違いは、パイロットライトが最初に追加アクションを取らなければリクエストを処理できないのに対して、ウォームスタンバイはトラフィックを直ちに (削減された能力レベルで) 処理できることです。パイロットライトでは、サーバーをオンにして、おそらく追加の (非コア) イ

ンフラストラクチャをデプロイし、スケールアップする必要があるのに対して、ウォームスタンバイでは、スケールアップするだけです (すべてがすでにデプロイされ、実行しています)。RTO と RPO のニーズに基づいて、両者の中から選択してください。

期待される成果:

各ワークロードについて、定義され、実装された DR 戦略があり、ワークロードは DR 目標を達成できます。ワークロード間の DR 戦略では、再利用可能なパターンを利用します (以前に記載された戦略など)。

一般的なアンチパターン:

- 同じような DR 目標を持つワークロードについて、一貫性のない復旧手順を実装する。
- DR 戦略は、災害が発生したときにアドホックに実装すればよいとする。
- DR のプランがない。
- 復旧時にコントロールプレーンのオペレーションに依存する。

このベストプラクティスを活用するメリット:

- 定義された復旧戦略を使用すると、一般的なツールとテスト手順を使用できます。
- 定義された復旧戦略を使用することで、チーム間での知識の共有が効率的になり、それぞれのチームが所有するワークロードの DR の実装が容易になります。

このベストプラクティスを活用しない場合のリスクレベル: 高

- 計画され、実装され、テストされた DR 戦略がなければ、災害発生時に復旧目標を達成できない可能性があります。

実装のガイダンス

これらのステップのそれぞれについて、以下の詳細を参照してください。

1. このワークロードの復旧要件を満たす DR 戦略を決定します。
2. 選択した DR 戦略の実装パターンをレビューします。
3. ワークロードのリソースと、それらの設定がフェイルオーバー前 (正常なオペレーション時) に復旧リージョンでどうなるかを評価します。

4. 必要なとき (災害発生時) に復旧リージョンをフェイルオーバーに備える方法を決定し、実装します。
5. 必要なとき (災害発生時) にフェイルオーバーするトラフィックを再ルーティングする方法を決定し、実装します。
6. ワークロードをフェイルバックする方法のプランを設計します。

実装ステップ

1. このワークロードの復旧要件を満たす DR 戦略を決定します。

DR 戦略を選ぶということは、ダウンタイムとデータ損失の削減 (RTO と RPO) と、戦略を実装するコストと複雑さのトレードオフです。必要以上に厳格な戦略の実装は、不要なコストにつながるため避けてください。

例えば、次の図では、許容可能な最大 RTO と、サービス復元戦略に費やすことができるコストの限界を決定しています。特定のビジネス目標の場合、パイロットライトまたはウォームスタンバイの DR 戦略は、RTO とコスト基準の両方を満たすことができます。

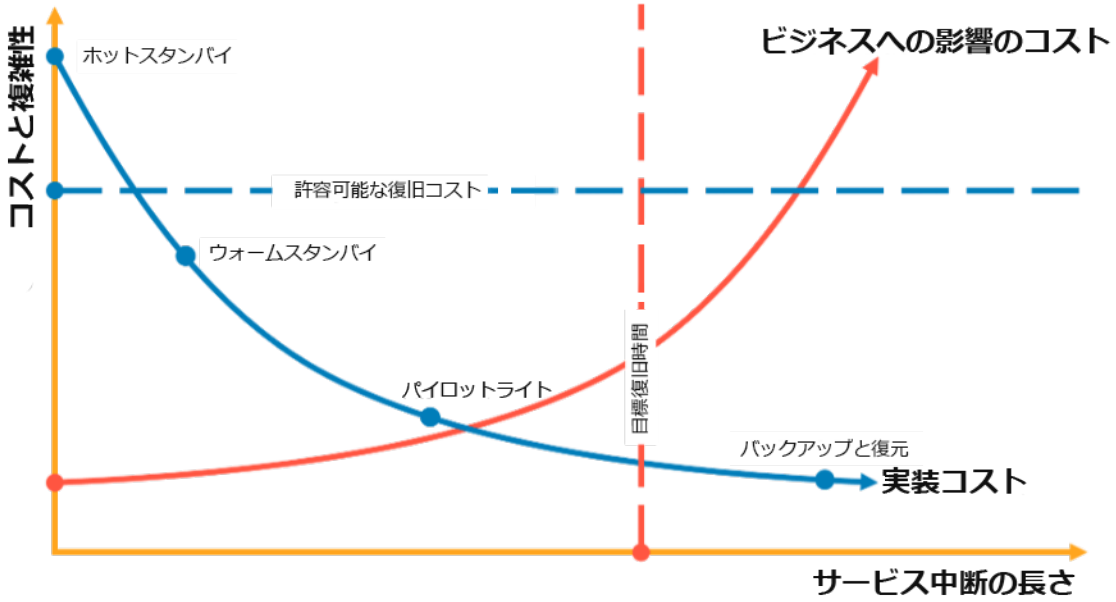


図 18: RTO とコストに基づく DR 戦略の選択を示すグラフ

詳細については、 ビジネス継続性計画 (BCP) を参照してください.

2. 選択した DR 戦略の実装パターンをレビューします。

このステップでは、選択した戦略の実装方法を理解します。戦略は、プライマリサイトと復旧サイトとしての AWS リージョン を使用して説明されています。ただし、単一リージョン内のアベイラビリティゾーンを DR 戦略として使用することもでき、その場合は、これら複数の戦略の要素を利用します。

このステップの後のステップでは、戦略を特定のワークロードに適用します。

バックアップと復元

バックアップと復元 は、実装の複雑さが最も少ない戦略ですが、ワークロードの復元に必要な時間と労力が多く、より高い RTO と RPO につながります。常にデータのバックアップを取り、これらを別のサイト (別の AWS リージョン など) にコピーすることをお勧めします。

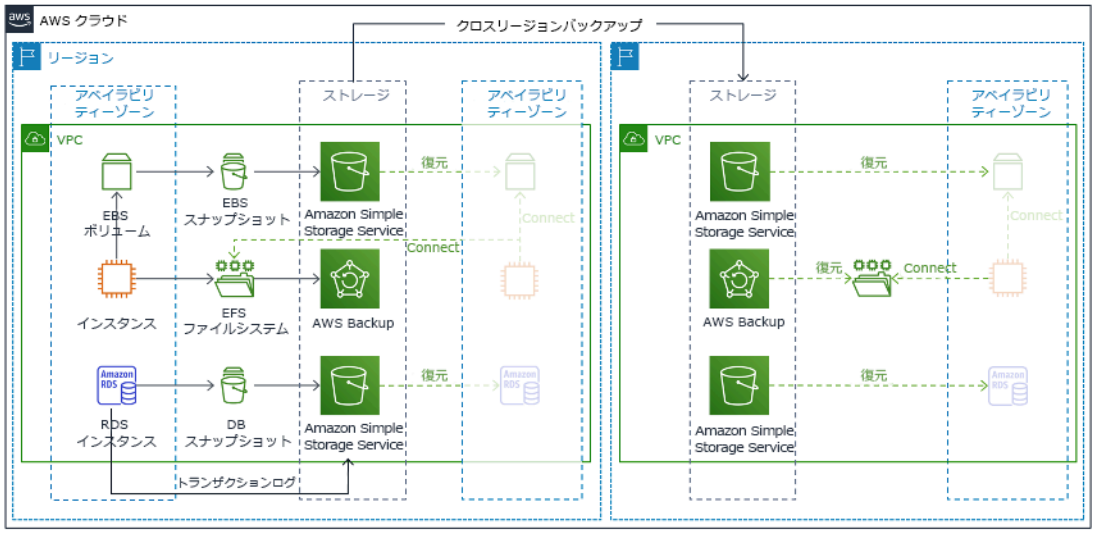


図 19: バックアップと復元アーキテクチャ

この戦略の詳細については、下記を参照してください。 AWS でのディザスタリカバリ (DR) アーキテクチャ、パート II: 迅速な復旧によるバックアップと復元.

パイロットライト

ように、 パイロットライト アプローチでは、プライマリリージョンから復旧リージョンにデータをレプリケートします。ワークロードインフラストラクチャに使用されるコアリソースは復旧リージョンにデプロイされますが、これを機能するスタックにするには、やはり追加のリソースと依存関係が必要です。例えば、図 20 では、コンピューティングインスタンスはデプロイされていません。

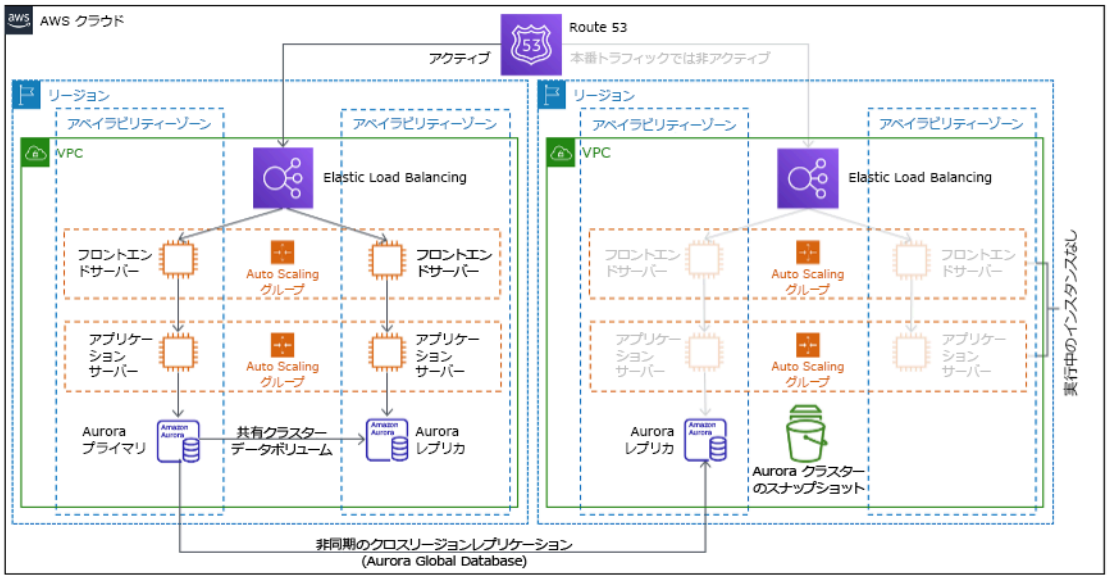


図 20: パイロットライトアーキテクチャ

この戦略の詳細については、下記を参照してください。 AWS でのディザスタリカバリ (DR) アーキテクチャ、パート III: パイロットライトとウォームスタンバイ.

ウォームスタンバイ

それらの インフラストラクチャ アプローチでは、本番稼働環境の完全に機能する縮小コピーを別のリージョンに用意する必要があります。このアプローチは、パイロットライトの概念を拡張して、ワークロードが別のリージョンに常駐するため、復旧時間が短縮されます。復旧リージョンが完全なキャパシティでデプロイされた場合は、 ホットスタンバイと呼ばれます.

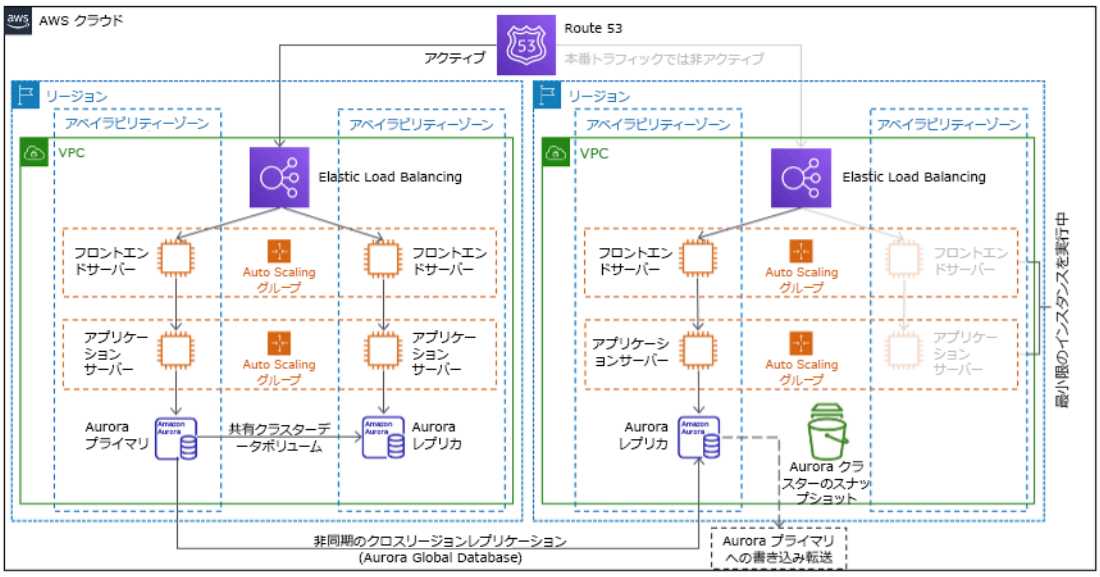


図 21: ウォームスタンバイアーキテクチャ

ウォームスタンバイまたはパイロットライトを使用するには、復旧リージョンのリソースをスケールアップする必要があります。必要なときにキャパシティが使用可能であるためには、EC2 インスタンスの キャパシティ予約 の使用を検討してください。AWS Lambda を使用する場合、 プロビジョニングされた同時実行数 により、機能の呼び出しにすぐに応答できる実行環境を確保できます。

この戦略の詳細については、下記を参照してください。 AWS でのディザスタリカバリ (DR) アーキテクチャ、パート III: パイロットライトとウォームスタンバイ.

マルチサイトアクティブ/アクティブ

マルチサイトアクティブ/アクティブ戦略の一部として、 複数のリージョンで同時に ワークロードを実行できます。マルチサイトアクティブ/アクティブは、デプロイされたすべてのリージョンからのトラフィックを処理します。顧客は、DR 以外の理由でこの戦略を選択することもあります。可用性を高めるためや、グローバルオーディエンスにワークロードをデプロイするときに (エンドポイントをエンドユーザーに近づけるためや、そのリージョン内のオーディエンスに対してローカライズされたスタックをデプロイするため) 使用できます。DR 戦略としては、ワークロードがデプロイされている AWS リージョン の 1 つでワークロードをサポートできない場合、そのリージョンは隔離され、残りのリージョンを使用して可用性を維持します。マルチサイトアクティブ/アクティブは、運用が最も複雑な DR 戦略であり、ビジネス要件上、必須の場合のみ選択してください。

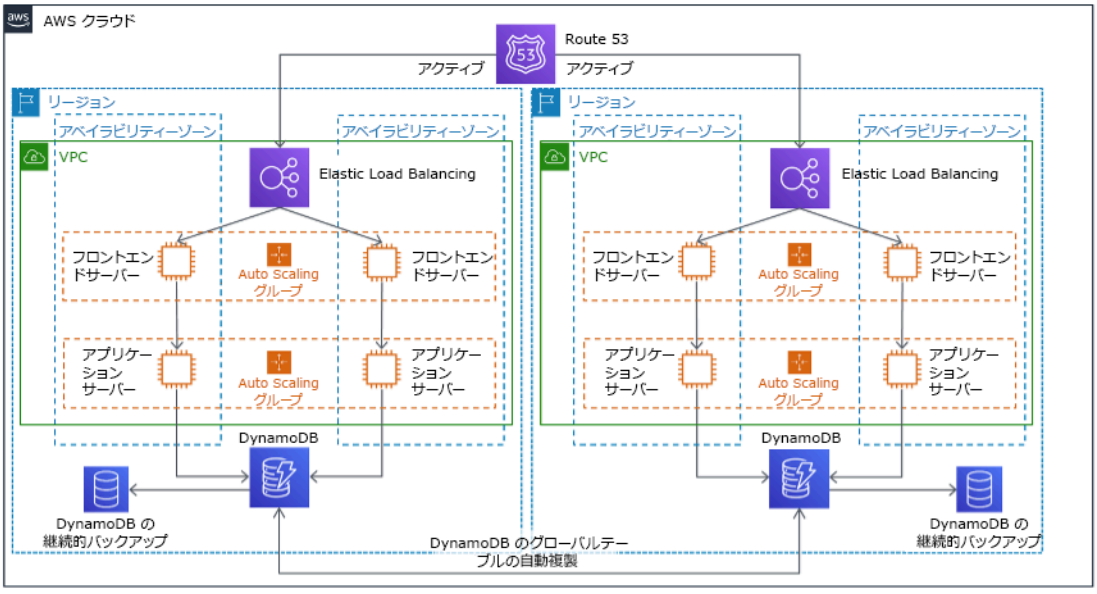


図 22: マルチサイトアクティブ/アクティブアーキテクチャ

この戦略の詳細については、下記を参照してください。 AWS のディザスタリカバリ (DR) アーキテクチャ、パート IV: マルチサイトアクティブ/アクティブ.

データを保護するためのその他のプラクティス

どの戦略でも、データ災害に対する緩和も必要です。連続的なデータレプリケーションは、特定のタイプの災害から保護しますが、戦略に、保存データのバージョニングまたはポイントインタイムリカバリのためのオプションが含まれていない限り、データの破損や破壊からは保護しません。復旧サイトにレプリケートしたデータもバックアップして、レプリカに加えて、ポイントインタイムバックアップを作成する必要があります。

単一の AWS リージョン 内の複数のアベイラビリティゾーン (AZ) の使用

単一のリージョン内の複数の AZ を使用する場合、DR 実装は上記の戦略の複数の要素を使用します。まず、図 23 に示されている複数の AZ を使用して、高可用性 (HA) アーキテクチャを作成する必要があります。このアーキテクチャは、マルチサイトアクティブ/アクティブアプローチを Amazon EC2 インスタンスとして利用し、 および Elastic Load Balancing はリソースを複数の AZ にデプロイして、リクエストをアクティブに処理します。このアーキテクチャは、ホットスタンバイのデモンストレーションにもなります。ホットスタンバイでは、プライマリ Amazon RDS インスタンスに障害が発生した場合 (または AZ そのものに障害が発生した場合)、スタンバイインスタンスがプライマリに昇格します。

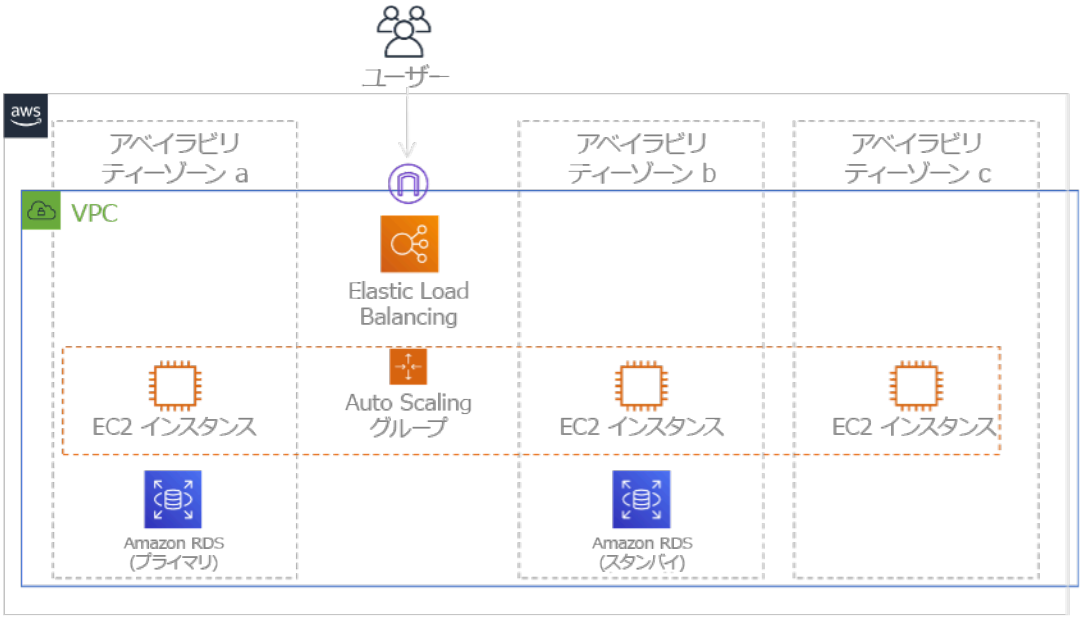


図 23: マルチ AZ アーキテクチャ

この HA アーキテクチャに加えて、ワークロードの実行に必要なすべてのデータのバックアップを追加する必要があります。これは、単一のゾーンに制約されるデータの場合に特に重要です。例えば、 Amazon EBS ボリューム または Amazon Redshift クラスターなどです.AZ に障害が発生した場合、このデータを別の AZ に復元する必要があります。可能な場合には、追加の保護層として、データバックアップも別の AWS リージョン にコピーしてください。

単一リージョン、マルチ AZ DR に対する、あまり一般的でない代替アプローチが下記のブログ投稿で説明されています。 Amazon Route 53 Application Recovery Controller を使用した回復力の高いアプリケーションの構築、パート 1: シングルリージョンスタック.ここでは、戦略は、AZ 間の分離をできるだけ高く維持して、リージョンのように動作させることです。この代替戦略を使用すると、アクティブ/アクティブまたはアクティブ/パッシブアプローチを選ぶことができます。

注: 一部のワークロードには、規制によるデータレジデンシー要件があります。現在 AWS リージョン が 1 つだけの地域のワークロードにこれが該当する場合、マルチリージョンではビジネスニーズに適しません。マルチ AZ 戦略は、ほとんどの災害に対して良好な保護を提供します。

3. ワークロードのリソースと、それらの設定がフェイルオーバー前 (正常なオペレーション時) に復旧リージョンでどうなるかを評価します。

インフラストラクチャと AWS リソースについては、 AWS CloudFormation などのコードとしてのインフラストラクチャや、Hashicorp Terraform などのサードパーティ製ツールを使用してください。複数のアカウントとリージョンに単一の操作でデプロイするには、 AWS CloudFormation StackSets を使用できます.マルチサイトアクティブ/アクティブとホットスタンバイ戦略の場合、復旧リージョンにデプロイされるインフラストラクチャはプライマリリージョンと同じリソースを持ちます。パイロットライトとウォームスタンバイ戦略の場合、デプロイされたインフラストラクチャを本番稼働で使用するには追加のアクションが必要です。CloudFormation の チューニング および 条件付きロジックを使用すると、デプロイされるスタックがアクティブかスタンバイかを単一のテンプレートで制御できます。このような CloudFormation テンプレートの例が、 このブログ投稿に含まれています.

すべての DR 戦略では、データソースが AWS リージョン の範囲内にバックアップされ、その後、それらのバックアップが復旧リージョンにコピーされる必要があります。AWS Backup は、これらのリソースのバックアップの設定、スケジュール、モニタリングができる一元的なビューを提供します。パイロットライト、ウォームスタンバイ、およびマルチサイトアクティブ/アクティブについては、プライマリリージョンのデータを復旧リージョンのデータリソース、例えば、 Amazon Relational Database Service (Amazon RDS) DB インスタンスや Amazon DynamoDB テーブルなどにレプリケートする必要もあります。したがって、これらのデータリソースはライブであり、復旧リージョンのリクエストに対応できます。

複数のリージョンにまたがる AWS サービスの動作の詳細については、火気に関するこのブログシリーズを参照してください。 AWS サービスによるマルチリージョンアプリケーションの作成.

4. 必要なとき (災害発生時) に復旧リージョンをフェイルオーバーに備える方法を決定し、実装します。

マルチサイトアクティブ/アクティブの場合、フェイルオーバーとは、リージョンを隔離して、残りのアクティブリージョンに頼ることを意味します。一般に、これらのリージョンはトラフィックを受け入れる準備ができています。パイロットライトとウォームスタンバイ戦略の場合、復旧アクションとして、図 20 の EC2 インスタンスなど、不足しているリソースやその他の不足リソースをデプロイする必要があります。

上記の戦略のすべてで、データベースの読み取り専用インスタンスを昇格して、プライマリの読み書きインスタンスにしなければならない場合があります。

バックアップと復元の場合、バックアップからのデータの復元によって、EBS ボリューム、RDS DB インスタンス、DynamoDB テーブルなど、そのデータのリソースを作成します。インフラストラクチャを復元し、コードをデプロイする必要もあります。AWS Backup を使用して、データを復旧リージョンに復元できます。把握 REL09-BP01 バックアップが必要なすべてのデータを特定し、バックアップする、またはソースからデータを再現する をご覧ください。インフラストラクチャの再構築には、 Amazon Virtual Private Cloud (Amazon VPC) は、、サブネット、必要なセキュリティグループに加えて、EC2 インスタンスなどのリソースの作成も含まれます。復元プロセスの大部分を自動化できます。方法については、 このブログ投稿を参照してください.

5. 必要なとき (災害発生時) にフェイルオーバーするトラフィックを再ルーティングする方法を決定し、実装します。

このフェイルオーバー操作は、自動または手動で開始できます。ヘルスチェックまたはアラームに基づくフェイルオーバーの自動開始を使用するときには、不要なフェイルオーバー (誤ったアラーム) によって、使用できないデータやデータ損失などのコストが発生するため、注意が必要です。そのため、多くの場合、手動によるフェイルオーバーの開始が使用されます。この場合でも、フェイルオーバーのステップを自動化できるため、手動開始はボタンを押すようなものです。

AWS サービスを使用するときに検討すべき、いくつかのトラフィック管理オプションがあります。1 つのオプションは、火気を使用することです Amazon Route 53.Amazon Route 53 を使用すると、1 つ以上の AWS リージョン の複数の IP エンドポイントを Route 53 ドメイン名に関連付けることができます。手動開始フェイルオーバーを実装するには、 Amazon Route 53 Application Recovery Controllerを使用できます。これは、高可用性データプレーン API を提供して、トラフィックを復旧リージョンに再ルーティングします。フェイルオーバーを実装するときには、火気で説明されているように、データプレーン操作を使用し、コントロールプレーンを避けてください REL11-BP04 復旧中はコントロールプレーンではなくデータプレーンを利用する.

このオプションやその他のオプションの詳細については、 ディザスタリカバリホワイトペーパーのこのセクションを参照してください.

6. ワークロードをフェイルバックする方法のプランを設計します。

フェイルバックとは、災害イベントの終息後、ワークロード操作をプライマリリージョンに戻すことを言います。インフラストラクチャとコードをプライマリリージョンにプロビジョニングするときには、一般に、最初に使用したのと同じステップに従い、コードとしてのインフラストラクチャとコードデプロイパイプラインに依存します。フェイルバックでの課題は、データストアを復元し、動作中の復旧リージョンとの一貫性を確認することです。

フェイルオーバー状態では、復旧リージョンのデータベースはライブであり、最新データを保持しています。目的は、復旧リージョンからプライマリリージョンへ再同期して、最新であることを確認することです。

いくつかの AWS のサービスは、これを自動的に行います。例えば、 Amazon DynamoDB グローバルテーブルを使用している場合、プライマリリージョンのテーブルが使用できなくなった場合でも、オンラインに復帰すると、DynamoDB が保留中の書き込みの伝播を再開します。例えば、 Amazon Aurora グローバルデータベース を使用し、 マネージドプランドフェイルオーバーを使用している場合、Aurora グローバルデータベースの既存のレプリケーショントポロジが維持されます。そのため、プライマリリージョンの以前の読み書きインスタンスがレプリカになり、復旧リージョンから更新を受け取ります。

これが自動でない場合、プライマリリージョンで復旧リージョンのデータベースのレプリカとしてデータベースを再確立する必要があります。多くの場合、これには、古いプライマリデータベースを削除して、新しいレプリカを作成する必要があります。例えば、Amazon Aurora グローバルデータベースでこれを行う方法の説明については、 計画外の フェイルオーバーを前提として、次のラボを参照してください。 グローバルデータベースのフェイルバック.

フェイルオーバー後、復旧リージョンでの実行を続行できる場合は、これを新しいプライマリリージョンにすることを検討してください。その場合でも、上記のすべてのステップを実行して、前のプライマリリージョンを復旧リージョンにします。一部の組織は、計画的ローテーションを実行して、プライマリリージョンと復旧リージョンを定期的に (3 か月ごとなど) 交換しています。

フェイルオーバーとフェイルバックに必要なすべてのステップをプレイブックに記載して、チームのメンバー全員が使用できるようにし、定期的にレビューする必要があります。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- the section called “REL09-BP01 バックアップが必要なすべてのデータを特定し、バックアップする、またはソースからデータを再現する”
- the section called “REL11-BP04 復旧中はコントロールプレーンではなくデータプレーンを利用する”
- the section called “REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する”

関連するドキュメント:

- AWS アーキテクチャブログ: ディザスタリカバリシリーズ
- AWS でのワークロードのディザスタリカバリ: クラウドでの復旧 (AWS ホワイトペーパー)
- クラウド内の災害対策オプション
- サーバーレス、マルチリージョン、アクティブ/アクティブのバックエンドソリューションを 1 時間で構築する
- マルチリージョンのサーバーレスバックエンド – 再ロード
- RDS: リージョン間でのリードレプリカのレプリケーション方法
- Route 53: DNS フェイルオーバーの設定
- S3: クロスリージョンレプリケーション
- 「AWS Backup とは。」
- Route 53 Application Recovery Controller とは?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: 開始方法 - AWS
- APN パートナー: 災害対策を支援できるパートナー
- AWS Marketplace: 災害対策に活用できる商品

関連動画:

- AWS でのワークロードのディザスタリカバリ
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

関連する例:

- AWS Well-Architected ラボ - ディザスタリカバリ - DR 戦略を説明するワークショップシリーズ

REL13-BP03 ディザスタリカバリの実装をテストし、実装を検証する

復旧サイトへの定期的なテストフェイルオーバーにより、適切な操作と、RTO および RPO が満たされることを確認します。

回避すべきパターンは、まれにしか実行されない復旧経路を作ることです。たとえば、読み取り専用のクエリに使用されるセカンダリデータストアがあるとします。データストアの書き込み時にプライマリデータストアで障害が発生した場合、セカンダリデータストアにフェイルオーバーします。もしこのフェイルオーバーを頻繁にテストしない場合、セカンダリデータストアの機能に関する前提が正しくない可能性があります。セカンダリデータストアの容量は、最後にテストしたときには十分だったかもしれませんが、このシナリオでは負荷に耐えられなくなる可能性があります。エラー復旧がうまくいくのは頻繁にテストする経路のみであることは、これまでの経験からも明らかです。少数の復旧経路を用意することがベストであるのはそのためです。復旧パターンを確立して定期的にテストできます。復旧経路が複雑な場合や重大な場合に復旧経路が正常に機能するという確信を持つには、本番環境でその障害を定期的に実行する必要があります。前述の例では、その必要性に関係なく、スタンバイへのフェイルオーバーを定期的に行う必要があります。

一般的なアンチパターン:

- 本番環境ではフェイルオーバーを実行しない。

このベストプラクティスを活用するメリット: 災害対策プランを定期的にテストすることで、必要なときに機能することや、チームが戦略の実行方法を把握していることを確認できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

- ワークロードを復旧用にエンジニアリングします。復旧経路を定期的にテストする Recovery Oriented Computing (ROC) は、復旧を強化するシステムの特性を特定します。以下がその特性です。隔離と冗長性、システム全体の変更のロールバック機能、正常性を監視し判断する機能、診断する機能、自動的な復旧、モジュラー設計、そして再起動する機能。復旧経路を訓練して、指定された時間内に指定された状態に復旧できるようにします。この復旧中にランブックを使用して問題を文書化し、次のテストの前に解決策を見つけます。
  - バークレー/スタンフォード大学の復旧指向コンピューティングプロジェクト
- CloudEndure Disaster Recovery を使用して DR 戦略を実装し、テストします。

- CloudEndure を使用した災害対策ソリューションのテスト
- CloudEndure Disaster Recovery
- AWS への CloudEndure Disaster Recovery

リソース

関連するドキュメント:

- APN パートナー: 災害対策を支援できるパートナー
- AWS アーキテクチャブログ: Disaster Recovery Series
- AWS Marketplace: 災害対策に活用できる商品
- CloudEndure Disaster Recovery
- AWS でのワークロードのディザスタリカバリ: クラウドでの復旧 (AWS ホワイトペーパー)
- CloudEndure を使用した災害対策ソリューションのテスト
- バークレー/スタンフォード大学の復旧指向コンピューティングプロジェクト
- AWS Fault Injection Simulator とは?

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS (STG208)

関連する例:

- AWS Well-Architected Labs - Testing for Resiliency

REL13-BP04 DR サイトまたはリージョンでの設定ドリフトを管理する

インフラストラクチャ、データ、設定が DR サイトまたはリージョンで必要とされるとおりであることを確認します。たとえば、AMI と Service Quotas が最新であることを確認します。

AWS Config は AWS リソース設定を継続的にモニタリングおよび記録します。これにより AWS Systems Manager Automation のドリフトを検出、トリガーでき、 修正してアラームを発生させます。AWS CloudFormation は、さらにデプロイしたスタックのドリフトを検出できます。

一般的なアンチパターン:

- プライマリロケーションで設定またはインフラストラクチャに変更を加えたときに、復旧ロケーションの更新を行わない。
- プライマリロケーションと復旧ロケーションの潜在的な制限 (サービスの違いなど) を考慮しない。

このベストプラクティスを確立するメリット: DR 環境が既存の環境と一致していることを確認することで、完全な復旧が保証されます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- デリバリーパイプラインがプライマリサイトとバックアップサイトの両方に配信しているようにします。アプリケーションを本番環境にデプロイするための配信パイプラインは、開発環境やテスト環境など、指定されたすべての災害対策戦略のロケーションに分散する必要があります。
- AWS Config を有効にして、潜在的なドリフトロケーションを追跡します。AWS Config ルールを使用して、ディザスタリカバリ戦略を実施するシステムを構築し、ドリフトを検出したときにアラートを生成します。
  - AWS Config ルール による非準拠 AWS リソースの修復
  - AWS Systems Manager Automation をトリガーして
- AWS CloudFormation を使用して、インフラストラクチャをデプロイします。AWS CloudFormation は、CloudFormation テンプレートが指定するものと実際にデプロイされているものとの間のドリフトを検出できます。
  - AWS CloudFormation: CloudFormation スタック全体のドリフトを検出する

リソース

関連するドキュメント:

- APN パートナー: 災害対策を支援できるパートナー
- AWS アーキテクチャブログ: ディザスタリカバリシリーズ
- AWS CloudFormation: CloudFormation スタック全体のドリフトを検出する
- AWS Marketplace: 災害対策に活用できる商品
- AWS Systems Manager Automation をトリガーして

- AWS でのワークロードの災害対策: クラウド内での復旧 (AWS ホワイトペーパー)
- How do I implement an Infrastructure Configuration Management solution on AWS? (AWS でインフラストラクチャ設定管理ソリューションを実装するにはどうすればよいですか?)
- AWS Config ルール による非準拠 AWS リソースの修復

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2) (マルチリージョンアクティブ/アクティブアプリケーションのアーキテクチャパターン)

REL13-BP05 復旧を自動化する

AWS またはサードパーティ製のツールを使用して、システムの復旧を自動化し、トラフィックを DR サイトまたはリージョンにルーティングします。

設定されたヘルスチェックに基づいて、Elastic Load Balancing や AWS Auto Scaling などの AWS サービスは、正常なアベイラビリティゾーンに負荷を分散できますが、Amazon Route 53、や AWS Global Accelerator などのサービスは、正常な AWS リージョン に負荷をルーティングできます。Route 53 Application Recovery Controller は、準備状況のチェックとルーティングコントロール機能を使用して、フェイルオーバーの管理と調整を支援します。これらの機能は、障害から回復するアプリケーションの能力を継続的にモニタリングするため、複数の AWS リージョン、アベイラビリティゾーン、およびオンプレミスにまたがってアプリケーションの回復を管理できます。

既存の物理または仮想データセンターまたはプライベートクラウド上のワークロードについては、 AWS Elastic Disaster Recovery(AWS Marketplace から入手可能) により、組織は自動ディザスタリカバリ戦略を AWS にセットアップできます。CloudEndure は、AWS のクロスリージョン/クロス AZ ディザスタリカバリもサポートしています。

一般的なアンチパターン:

- 同一の自動フェイルオーバーとフェイルバックを実装すると、障害が発生したときにフラッピングが発生する可能性があります。

このベストプラクティスを活用するメリット: 自動復旧により、手動エラーの可能性が排除され、復旧時間が短縮されます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 復旧経路を自動化します。復旧時間が短い場合に人が判断して対処する方法は、高い可用性シナリオには利用できません。システムはあらゆる状況下で自動的に復旧する必要があります。
  - CloudEndure Disaster Recover を自動化したフェイルオーバーとフェイルバックに使用する CloudEndure Disaster Recovery は、マシン (オペレーティングシステム、システム状態設定、データベース、アプリケーション、ファイルなど) をターゲット AWS アカウント および希望するリージョンの低コストのステージングエリアに継続的にレプリケートします。災害が発生した場合、CloudEndure Disaster Recovery に指示して、数千台のマシンを数分で完全にプロビジョニングされた状態で自動的に起動できます。
    - 災害対策フェイルオーバーとフェイルバックの実行
    - CloudEndure Disaster Recovery

リソース

関連するドキュメント:

- APN パートナー: 災害対策を支援できるパートナー
- AWS アーキテクチャブログ: ディザスタリカバリシリーズ
- AWS Marketplace: 災害対策に活用できる商品
- AWS Systems Manager Automation をトリガーして
- AWS への CloudEndure Disaster Recovery
- AWS でのワークロードのディザスタリカバリ: クラウドでの復旧 (AWS ホワイトペーパー)

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

パフォーマンス効率

トピック

- 選択
- レビュー
- モニタリング

- トレードオフ

選択

質問

- PERF 1 どのように最良パフォーマンスのアーキテクチャを選択するのですか?
- PERF 2 コンピューティングソリューションはどのように選択すればよいですか?
- PERF 3 ストレージソリューションはどのように選択すればよいですか?
- PERF 4 データベースソリューションはどのように選択すればよいですか?
- PERF 5 ネットワーキングソリューションはどのように選択すればよいですか?

PERF 1 どのように最良パフォーマンスのアーキテクチャを選択するのですか?

多くの場合、ワークロード全体での最適なパフォーマンスのためには、複数のアプローチが必要になります。Well-Architected なシステムでは、パフォーマンスを向上させるために複数のソリューションと機能が使用されています。

ベストプラクティス

- PERF01-BP01 利用可能なサービスやリソースを理解する
- PERF01-BP02 アーキテクチャにかかわる選択プロセスを決める
- PERF01-BP03 意思決定においてコスト要件を考慮する
- PERF01-BP04 ポリシーやリファレンスアーキテクチャを使用する
- PERF01-BP05 クラウドプロバイダー、または適切なパートナーからのガイダンスを利用する
- PERF01-BP06 既存のワークロードのベンチマークを実施する
- PERF01-BP07 ワークロードの負荷テスト

PERF01-BP01 利用可能なサービスやリソースを理解する

クラウドで利用できる幅広いサービスやリソースに関する情報を取得し、その内容を理解します。お客様のワークロードに関連するサービスや設定の選択肢を認識した上で、最適なパフォーマンスを実現する方法を理解してください。

既存のワークロードを評価している場合は、それによって消費されるさまざまなサービスとリソースのインベントリを作成する必要があります。この一覧は、どのコンポーネントをマネージドサービス、および新しいテクノロジーに置き換えることができるかを評価するために役立ちます。

一般的なアンチパターン:

- クラウドをコロケーションされたデータセンターとして使用する。
- 永続的なストレージを必要とするすべてに共有ストレージを使用する。
- 自動スケーリングを使用しない。
- 現在の基準に最も近いインスタンスタイプを使用するが、必要に応じてより大きいインスタンスタイプを使用しない。
- マネージドサービスとして使用できるテクノロジーをデプロイおよび管理する。

このベストプラクティスを活用するメリット: 使い慣れていないサービスを検討することで、インフラストラクチャのコストと、サービスの維持に必要な労力を大幅に削減できる可能性があります。新しいサービスや機能をデプロイすることで、市場投入までの時間を短縮できる場合があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

関連サービスのワークロードソフトウェアとアーキテクチャの一覧を作成する: ワークロードのインベントリを収集し、詳細を確認する製品のカテゴリを決定します。パフォーマンスを向上させ、運用の複雑さを軽減するために、マネージドサービスに置き換えることができるワークロードコンポーネントを特定します。

リソース

関連ドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

関連サンプル:

- AWS Samples
- AWS SDK サンプル

PERF01-BP02 アーキテクチャにかかわる選択プロセスを決める

クラウドに関する社内の経験と知識、公開されたユースケース、関連ドキュメント、ホワイトペーパーなどの外部リソースを利用して、リソースとサービスを選択するプロセスを決定します。お客様のワークロードで利用できるサービスについて、実験とベンチーマークを促すプロセスを定義するようにしてください。

アーキテクチャに重要なユーザーストーリーを記述するときは、それぞれの重要なストーリーをどの程度迅速に実行する必要があるかを明記するなど、パフォーマンス要件を含めるようにしてください。これらの重要なストーリーには、要件に対してこれらのストーリーがどのように実行されるかについての可視性を確保するために、スクリプト化されたユーザージャーニーを追加で実装するようにしてください。

一般的なアンチパターン:

- あなたは、現在のアーキテクチャが今後は静的なものとなり、しばらく更新されないと考えています。
- あなたは、理由なしで、時間の経過とともにアーキテクチャの変更を導入します。

このベストプラクティスを活用するメリット: アーキテクチャの変更を行うためのプロセスを定義することで、収集されたデータを使用して、時間の経過とともにワークロード設計を適宜変更します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

アーキテクチャのアプローチを選択する: パフォーマンス要件を満たすアーキテクチャの種類を特定します。配信用のメディア (デスクトップ、ウェブ、モバイル、IoT)、従来の要件、統合などの制約を特定します。リファクタリングなどの再利用の機会を把握します。他のチーム、アーキテクチャ図、および AWS ソリューションアーキテクト、AWS リファレンスアーキテクチャ、AWS パートナーなどのリソースを参考にし、アーキテクチャ選びに役立てます。

カスタマーエクスペリエンスを使用して、最も重要なメトリクスを特定します。メトリクスごとに、ターゲット、測定アプローチ、および優先順位を特定します。カスタマーエクスペリエンスを定義します。顧客に必要なパフォーマンスのエクスペリエンスと、顧客がワークロードのパフォーマンスをどのように評価するかを文書化します。重要なユーザーストーリーのエクスペリエンスの懸念事項に

ついて優先順位を付けます。要件に対してこのストーリーがどのように実行されるかを知ることができるように、パフォーマンス要件を含め、スクリプト化されたユーザージャーニーを実装します。

リソース

関連ドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

関連サンプル:

- AWS Samples
- AWS SDK サンプル

PERF01-BP03 意思決定においてコスト要件を考慮する

ワークロードには、多くの場合、運用のコスト要件があります。社内のコスト管理を使用し、予測されたリソースニーズに基づいて、リソースのタイプとサイズを選択してください。

ワークロードの各要素がマネージドデータベース、インメモリキャッシュ、ETL サービスなどのフルマネージド型サービスに置き換えることができるかを判断します。自社で運用すべきワークロードを削減することによって、リソースをビジネス成果に集中させることが可能になります。

コスト要件のベストプラクティスについては、 費用対効果の高いリソース セクションにある コスト最適化の柱に関するホワイトペーパーを参照してください.

一般的なアンチパターン:

- インスタンスの 1 つのファミリーのみを使用する。
- ライセンスソリューションとオープンソースソリューションを比較しない。

- ブロックストレージのみを使用する。
- 一般的なソフトウェアを EC2 インスタンスと、マネージドサービスとして使用できる Amazon EBS ボリュームまたはエフェメラルボリュームにデプロイする。

このベストプラクティスを活用するメリット: 選択を行う際にコストを考慮することで、他の投資が可能となります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ワークロードコンポーネントを最適化し、伸縮性を実現することで、コストを削減し、コンポーネントの効率を最大化します。マネージドデータベース、インメモリキャッシュ、リバースプロキシなど、必要に応じてマネージドサービスに置き換えることができるワークロードコンポーネントを判断します。

リソース

関連ドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター
- AWS Compute Optimizer

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture
- Optimize performance and cost for your AWS compute (CMP323-R1)

関連サンプル:

- AWS Samples
- AWS SDK サンプル
- Rightsizing with Compute Optimizer and Memory utilization enabled

- AWS Compute Optimizer デモコード

PERF01-BP04 ポリシーやリファレンスアーキテクチャを使用する

内部ポリシーと既存のリファレンスアーキテクチャを評価し、独自の分析を使用してワークロードのサービスと設定を選択することによって、パフォーマンスと効率性を最大化します。

一般的なアンチパターン:

- あなたは、会社の管理オーバーヘッドに影響する可能性のあるテクノロジーの選択を幅広く使用することを許可します。

このベストプラクティスを確立するメリット: アーキテクチャ、テクノロジー、ベンダーの選択に関するポリシーを確立することで、迅速な意思決定が可能になります。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

既存のポリシーまたはリファレンスアーキテクチャを使用してワークロードをデプロイする: サービスをクラウドデプロイに統合し、パフォーマンステストを使用して、パフォーマンス要件を継続的に満たすことができることを確認します。

リソース

関連ドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

関連サンプル:

- AWS Samples
- AWS SDK Examples (AWS SDK サンプル)

PERF01-BP05 クラウドプロバイダー、または適切なパートナーからのガイダンスを利用する

判断の指針とするために、ソリューションアーキテクト、プロフェッショナルサービス、または適切なパートナーなどのクラウド企業のリソースを利用します。それらのリソースはお客様のアーキテクチャにおける最適なパフォーマンスを実現するためのレビューと改善に役立ちます。

追加のガイダンス、または製品情報が必要な場合は、AWS までお問い合わせください。AWS ソリューションアーキテクトおよび AWS プロフェッショナルサービス は、ソリューションの実装に関するガイダンスを提供します。 AWS パートナー は、ビジネスの俊敏性とイノベーションを引き出すために AWS の専門知識を提供します。

一般的なアンチパターン:

- 一般的なデータセンタープロバイダーとして AWS を利用する。
- 意図されていない方法で AWS のサービスを利用する。

このベストプラクティスを確立するメリット: プロバイダーやパートナーに相談することで、意思決定に対する自信を高めることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

AWS リソースにサポートを依頼する: AWS ソリューションアーキテクトとプロフェッショナルサービスは、ソリューションの実装におけるガイダンスを提供します。APN パートナーは、ビジネスの俊敏性とイノベーションを引き出すために AWS の専門知識を提供します。

リソース

関連ドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

関連サンプル:

- AWS Samples
- AWS SDK Examples (AWS SDK サンプル)

PERF01-BP06 既存のワークロードのベンチマークを実施する

既存のワークロードのパフォーマンスにベンチマーク結果を参考に、クラウドでの実行状況を把握します。ベンチマークから収集されたデータを使用して、アーキテクチャ面での判断を導き出します。

合成テストと実際のユーザーのモニタリングによるベンチマークを使用して、ワークロードの各コンポーネントがどのように機能するかに関するデータを生成します。ベンチマークは概して負荷テストよりも迅速にセットアップでき、特定のコンポーネントに対するテクノロジーを評価するために使用されます。ベンチマークは、まだ負荷テストができるほどソリューションが完成していないプロジェクトの初期段階によく使用されます。

独自のカスタムベンチマークテストを構築するか、TPC-DS などの業界標準テストを 使用できます (データウェアハウジングのワークロードをベンチマークする場合)。業界ベンチマークは、環境を比較する場合に有用です。カスタムベンチマークは、アーキテクチャで採用する予定の特殊なオペレーションを対象にするときに役立ちます。

ベンチマークを実施するときは、有効な結果が得られることを確実にするためにテスト環境の事前暖気を行うことが重要です。同じベンチマークを複数回実行して、時系列での変動をとらえておくようにしてください。

ベンチマークは概して負荷テストよりも速く実行されるため、デプロイパイプラインの早い時期に使用でき、パフォーマンスの逸脱に関するフィードバックもより迅速に提供されます。ベンチマークは、コンポーネント、またはサービスにおける大幅な変更を評価する場合に、その変更を行う労力を正当化できるかどうかを見極める近道となり得ます。負荷テストでは、ワークロードが本番環境でどのように機能するかに関する情報が得られることから、ベンチマークは負荷テストと併せて使用することが重要です。

一般的なアンチパターン:

- あなたは、ワークロードの特性を示唆しない一般的なベンチマークに依存しています。
- 顧客からのフィードバックと認識を唯一のベンチマークとして使用している。

このベストプラクティスを活用するメリット: 現在の実装をベンチマークすることで、パフォーマンスの向上を測定できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

開発中にパフォーマンスをモニタリングする: ワークロードの進化に合わせて、パフォーマンスを目で見て確認できるプロセスを実装します。

配信パイプラインに統合する: 配信パイプラインで負荷テストを自動的に実行します。テスト結果を事前定義された主要業績評価指標 (KPI) やしきい値と比較して、引き続きパフォーマンス要件を満たせるようにします。

ユーザージャーニーをテストする: 負荷テストには、本番データの合成またはサニタイズされたバージョン (機密情報や個人が特定できる情報は削除する) を使用します。アプリケーション全体で再生またはプログラミング済みのユーザージャーニーを大規模に使用して、アーキテクチャ全体を練習として動かします。

実際のユーザーのモニタリング: CloudWatch RUM を使用して、アプリケーションのパフォーマンスに関するクライアント側のデータを収集、表示できます。このデータを使用して、実際のユーザーのパフォーマンスベンチマークを確立します。

リソース

関連ドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター
- Amazon CloudWatch RUM を使用する
- Amazon CloudWatch Synthetics とは

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics のデモ

関連サンプル:

- AWS Samples
- AWS SDK サンプル
- 分散負荷テスト
- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP07 ワークロードの負荷テスト

異なるリソースタイプとサイズを使用して、最新のワークロードアーキテクチャをクラウドにデプロイします。デプロイメントをモニタリングして、ボトルネック、または過剰なキャパシティーを特定するパフォーマンスメトリクスを取得してください。このパフォーマンス情報を使用して、お客様のアーキテクチャとリソースの選択を改善します。

負荷テストでは 実際の ワークロードを使用するため、ソリューションが本番環境でどのように機能するかを確認できます。ロードテストは、本番データの合成バージョンまたはサニタイズバージョン (機密情報または識別情報を削除) を使用して実行する必要があります。大規模にアーキテクチャ全体に対して負荷をかけるため、ユーザーのサービス利用方法をリプレイする、もしくはプログラムによって再現できるようにします。デリバリーパイプラインの一環として負荷テストを自動的に実行し、その結果を事前定義された KPI およびしきい値と比較します。こうすることで、必要とされるパフォーマンスの継続的な達成が保証されます。

一般的なアンチパターン:

- あなたは、ワークロード全体ではなく、ワークロードの個々の部分について負荷テストを行います。
- あなたは、本番環境とは異なるインフラストラクチャで負荷テストを行います。
- あなたは、今後問題が発生する可能性を予測するのに役立てるため、予想される負荷に対してのみ、負荷テストを実施し、それを超える負荷に対しては負荷テストを実施しません。

- AWS Support に通知することなく負荷テストを実行し、あたかもサービス拒否のようなイベントが発生することで、テストが打ち切られてしまいます。

このベストプラクティスを確立するメリット: 負荷テストでパフォーマンスを測定すると、負荷の増加に伴って影響を受ける場所が判明します。これにより、必要な変更がワークロードに影響を与える前に予測できるようになります。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

負荷テストによってアプローチを検証する: パフォーマンス要件を満たすかどうかを確認するために、概念実証のための負荷テストを行います。AWS のサービスを使用して、アーキテクチャをテストするための本番規模の環境を実行することができます。料金はテスト環境が必要となる場合にのみ発生することから、オンプレミス環境を使用する場合に比べてわずかなコストで本格的なテストを実施できます。

メトリクスをモニタリングする: Amazon CloudWatch では、アーキテクチャ内のリソース全体のメトリクスを収集できます。また、カスタムメトリクスを収集および発行して、ビジネスメトリクスまたは導出メトリクスを表面化することも可能です。CloudWatch またはサードパーティーのソリューションを使用して、しきい値を超過したことを示すアラームを設定します。

大規模にテストする: 負荷テストは実際のワークロードを使用するため、ソリューションが本番環境でどのように機能するかを確認できます。AWS のサービスを使用して、アーキテクチャをテストするための本番規模の環境を実行することができます。テスト環境に対する支払いはテスト環境が必要なときにのみ発生するため、オンプレミスの環境を使用する場合より低いコストで大規模なテストを実行できます。AWS クラウドを活用してワークロードをテストし、どこでスケールしないのか、あるいは非線形にスケールしているのかを発見してください例えば、低コストで負荷を生成し、本番前にボトルネックを発見するには、スポットインスタンスを使用します。

リソース

関連ドキュメント:

- AWS CloudFormation
- Building AWS CloudFormation Templates using CloudFormer (CloudFormer を使った AWS CloudFormation テンプレートの構築)
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics

- AWS での分散負荷テスト

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- Optimize applications through Amazon CloudWatch RUM (Amazon CloudWatch RUM によるアプリケーションの最適化)
- Amazon CloudWatch Synthetics のデモ

関連サンプル:

- AWS での分散負荷テスト

PERF 2 コンピューティングソリューションはどのように選択すればよいですか?

ワークロードにとって最適なコンピューティングソリューションは、アプリケーションの設計、使用パターン、設定に応じて異なります。各アーキテクチャでは、コンポーネントごとに異なるコンピューティングソリューションが使用される可能性があるため、パフォーマンスを向上させるための機能も異なります。アーキテクチャに不適切なコンピューティングソリューションを選択すると、パフォーマンス効率が低下する可能性があります。

ベストプラクティス

- PERF02-BP01 使用可能なコンピューティングオプションを評価する
- PERF02-BP02 利用可能なコンピューティング設定オプションについて理解する
- PERF02-BP03 コンピューティング関連のメトリクスを収集する
- PERF02-BP04 適切なサイジングによって必要な設定を決定する
- PERF02-BP05 利用可能な伸縮性のあるリソースを使用する
- PERF02-BP06 メトリクスに基づいてコンピューティングニーズを再評価する

PERF02-BP01 使用可能なコンピューティングオプションを評価する

インスタンス、コンテナ、関数などのさまざまなコンピューティングオプションを使用することで、ワークロードにどのようなメリットがあるかを理解します。

期待される成果: 利用できるすべてのコンピューティングオプションを理解することにより、パフォーマンスを高め、不要なインフラストラクチャコストを削減し、ワークロードの維持に必要な運

用労力を低減するための機会に気付くことができます。また、新しいサービスや機能をデプロイする際に、市場投入までの時間を短縮できます。

一般的なアンチパターン:

- 移行後のワークロードで、オンプレミスで使用していたコンピューティングソリューションをそのまま使用する。
- クラウドコンピューティングソリューションや、そうしたソリューションがコンピューティング性能の向上にどのように役立つかについての認識が足りない。
- ワークロードの特性により的確に適合する代替のコンピューティングソリューションがあるにもかかわらず、スケーリングやパフォーマンスの要件を満たすために既存のコンピューティングソリューションのサイズを大きくしすぎる。

このベストプラクティスを活用するメリット: コンピューティング要件を特定し、利用できるコンピューティングソリューションを評価することにより、ビジネスステークホルダーやエンジニアリングチームが、選択したコンピューティングソリューションを使用することの利点や制限を理解できます。選択したコンピューティングソリューションは、ワークロードのパフォーマンス基準に適合している必要があります。主な基準には、処理の必要性、トラフィックパターン、データアクセスパターン、スケーリングの必要性、レイテンシー要件があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードにメリットをもたらし、パフォーマンス要件を満たす仮想化、コンテナ化、マネジメントのソリューションを理解します。1 つのワークロードには、複数の種類のコンピューティングソリューションを含めることができます。コンピューティングソリューションにはぞれぞれ、異なる特徴があります。ワークロードのスケールとコンピューティング要件を基に、コンピューティングソリューションを選択し、ニーズを満たすように構成できます。クラウドアーキテクトは、インスタンス、コンテナ、関数の長所と短所を学ぶ必要があります。次の手順では、ワークロードの特性とパフォーマンス要件に合ったコンピューティングソリューションを選択する方法を詳しく説明しています。

タイプ	サーバー	コンテナ	関数
AWS のサービス	Amazon Elastic Compute Cloud (Amazon EC2)	Amazon Elastic Container Service (Amazon ECS),	AWS Lambda

タイプ	サーバー	コンテナ	関数
		Amazon Elastic Kubernetes Service (Amazon EKS)	
主な特徴	ハードウェアライセンス要件、プレイスメントオプション、またコンピューティングメトリクスに基づくさまざまなインスタンスファミリーの幅広い選択肢のための専用オプションがある	デプロイが簡単、環境に一貫性がある、EC2 インスタンス上で運用、スケーリングが可能	ランタイムが短い (15 分以下)、メモリおよび CPU の上限は他のサービスほど高くない、マネージドハードウェア層、何百万の同時リクエストに対応してスケーリング
一般的なユースケース	リフトアンドシフトの移行、モノリシックなアプリケーション、ハイブリッド環境、エンタープライズアプリケーション	マイクロサービス、ハイブリッド環境	マイクロサービス、イベント駆動型アプリケーション

実装手順:

1. 「the section called “PERF05-BP06 ネットワーク要件に基づいてワークロードのロケーションを選択する”」セクションを評価して、コンピューティングソリューションを配置する場所を選択します。この場所により、使用できるコンピューティングソリューションのタイプが制限されます。
2. 場所の要件とアプリケーション要件に適したコンピューティングソリューションのタイプを特定します。
    a. Amazon Elastic Compute Cloud (Amazon EC2) 仮想サーバーインスタンスにはさまざまなファミリーとサイズがあります。またソリッドステートドライブ (SSD)、グラフィック処理ユニット (GPU) などさまざまな機能が使用できます。EC2 インスタンスは、インスタンスの選択において最大の柔軟性を提供します。EC2 インスタンスを起動する場合、指定するインスタン

スタイプによって、インスタンスのハードウェアが決まります。インスタンスタイプごとに、コンピューティング、メモリ、ストレージの機能が異なります。インスタンスタイプは、これらの機能に基づいてインスタンスファミリーにグループ分けされます。一般的なユースケースには、エンタープライズアプリケーションの運用、ハイパフォーマンスコンピューティング (HPC)、機械学習アプリケーションのトレーニングおよびデプロイ、クラウドネイティブアプリケーションの運用などがあります。

b. Amazon Elastic Container Service (Amazon ECS) はフルマネージド型のコンテナオーケストレーションサービスで、AWS Fargate を使用したサーバーレスインスタンスまたは EC2 インスタンスでクラスターを構成し、コンテナを自動的に実行および管理できるようにします。Amazon ECS は Amazon Route 53、Secrets Manager、AWS Identity and Access Management (IAM)、Amazon CloudWatch などのサービスと併せて使用できます。Amazon ECS は、アプリケーションがコンテナ化されており、エンジニアリングチームが Docker コンテナを好む場合に推奨されます。

c. Amazon Elastic Kubernetes Service (Amazon EKS) は、フルマネージド型の Kubernetes サービスです。AWS Fargate を使って EKS クラスターを実行することもできるため、サーバーのプロビジョニングと管理が不要になります。Amazon EKS の管理は、Amazon CloudWatch、Auto Scaling グループ、AWS Identity and Access Management (IAM)、Amazon Virtual Private Cloud (VPC) などの AWS のサービスとの統合によって簡素化されています。コンテナを使用する場合、EC2 インスタンスまたは AWS Fargate インスタンスのタイプを選択するためにコンピューティングメトリクスを使用するのと同様に、ワークロードに最も適したタイプを選択するためにコンピューティングメトリクスを使用する必要があります。Amazon EKS は、アプリケーションがコンテナ化されており、エンジニアリングチームが Docker コンテナよりも Kubernetes 好む場合に推奨されます。

d. 専用のインフラストラクチャで AWS Lambda を使用すると、許可されたランタイム、メモリ、CPU のオプションをサポートするコードを実行できます。コードをアップロードするだけで、AWS Lambda がそのコードの実行とスケーリングに必要となるものすべてを管理します。コードは、他の AWS のサービスから自動的にトリガーするように設定するか、直接呼び出すことができます。Lambda は、クラウド用に開発された実行時間の短いマイクロサービスアーキテクチャに推奨されます。

3. 新しいコンピューティングソリューションを試した後、移行を計画し、パフォーマンスメトリクスを検証します。これは継続的なプロセスです。詳細については、 the section called “PERF02-BP04 適切なサイジングによって必要な設定を決定する”を参照してください。

実装計画に必要な工数レベル: ワークロードがあるコンピューティングソリューションから別のコンピューティングソリューションに移行する場合は、アプリケーションのリファクタリングに 中 程度の工数が必要になる可能性があります。

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- EC2 インスタンスタイプ
- EC2 インスタンスのプロセッサのステート制御
- EKS コンテナ: EKS ワーカーノード
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- 関数: Lambda 関数の設定
- コンテナに関する規範ガイダンス
- サーバーレスに関する規範ガイダンス

関連動画:

- スタートアップ企業向けコンピューティングオプションの選択方法
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Amazon EC2 foundations (CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Deliver high-performance ML inference with AWS Inferentia (CMP324-R1)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

関連サンプル:

- Migrating the web application to containers
- Run a Serverless Hello World

PERF02-BP02 利用可能なコンピューティング設定オプションについて理解する

各コンピューティングソリューションには、ワークロードの特性をサポートするために使用できるオプションと設定があります。さまざまなオプションがワークロードをどのように補完するか、またア

プリケーションにどの設定オプションが最適かをご紹介します。これらのオプションの例には、インスタンスのファミリー、サイズ、機能 (GPU、I/O)、バースト、タイムアウト、関数サイズ、コンテナインスタンス、並行性などがあります。

期待される成果: CPU、メモリ、ネットワークスループット、GPU、IOPS、トラフィックパターン、データアクセスパターンなどのワークロードの特性を文書化し、ワークロードの特性に合わせてコンピューティングソリューションを設定するために使用されます。これらの各メトリクスと、お客様のワークロードに特化したカスタムメトリクスを記録、監視し、コンピューティング設定を最適化することで、要件に最適に対応します。

一般的なアンチパターン:

- オンプレミスで使用していたコンピューティングソリューションをそのまま使用する。
- ワークロードの特性に合ったコンピューティングオプションやインスタンスファミリーを見直さない。
- バースト能力を確保するためにコンピューティングを過剰にする。
- 同じワークロードに対して複数のコンピューティング管理プラットフォームを使用する。

このベストプラクティスを確立するメリット: AWS のコンピューティングサービスをよく理解し、それぞれのワークロードに適したソリューションを決定できるようにしましょう。ワークロードのコンピューティングサービスを選択したら、それらがワークロードのニーズをどの程度満たしているかを迅速に実験することができます。ワークロードの特性に合うように最適化されたコンピューティングソリューションを使用することで、パフォーマンス改善、コスト削減、信頼性向上を実現できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードで同じコンピューティングオプションを 4 週間以上使用しており、その特性が今後も変わらないと予想される場合は、 AWS Compute Optimizer を使用して、コンピューティング特性に基づいた推奨事項を提供することができます。メトリクスが足りない、インスタンスタイプがサポートされていない、特性が変わることが予想されるなどの理由から、AWS Compute Optimizer を使用できない場合は、負荷テストや 実験を基にメトリクスを予測する必要があります。

実装手順:

1. EC2 インスタンスで実行していますか、それとも EC2 起動タイプのコンテナで実行していますか。

a. ワークロードで GPU を使用してパフォーマンスを高めることができますか。

   i. 高速コンピューティング インスタンスとは GPU ベースのインスタンスで、機械学習のトレーニング、推論、ハイパフォーマンスコンピューティング向けに最も高いパフォーマンスを提供します。

b. ワークロードで機械学習推論アプリケーションを実行していますか。

   i. AWS Inferentia (Inf1) — Inf1 インスタンスは、機械学習推論アプリケーションをサポートするために構築されています。Inf1 インスタンスを使用すると、画像認識、音声認識、自然言語処理、パーソナライゼーション、不正行為検出といった大規模な機械学習推論アプリケーションを実行できます。モデルは、TensorFlow、PyTorch、MXNet などの一般的な機械学習フレームワークのいずれかで構築し、GPU インスタンスを使用してトレーニングできます。要件に合わせて機械学習モデルをトレーニングしたら、AWS Neuron を使用して Inf1 インスタンスにモデルをデプロイできます。 AWS Neuronは、Inferentia チップの機械学習推論パフォーマンスを最適化するコンパイラー、ランタイム、プロファイリングツールで構成される専用のソフトウェア開発キット (SDK) です。

c. パフォーマンス向上のために、ワークロードを低レベルのハードウェアと統合していますか。

   i. フィールドプログラマブルゲートアレイ (FPGA) — FPGA を使用することで、最も要求の厳しいワークロードをカスタムハードウェアで加速して実行することができ、ワークロードを最適化することができます。サポートされる、C や Go などの一般的なプログラミング言語や、Verilog や VHDL などのハードウェア指向の言語を利用してアルゴリズムを定義できます。

d. 少なくとも 4 週間分のメトリクスがあり、トラフィックパターンとメトリクスが今後もほぼ同じになると予測できますか。

   i. 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Compute Optimizer を使用して、コンピューティング特性に最適なコンピューティング構成に関する機械学習の推奨事項を取得します。

e. ワークロードのパフォーマンスは CPU のメトリクスによって制限されていますか。

   i. コンピューティング最適化 インスタンスは、高性能プロセッサーを必要とするワークロードに適しています。

f. ワークロードのパフォーマンスはメモリのメトリクスによって制限されていますか。

   i. メモリ最適化 インスタンスは大量のメモリを提供し、メモリを集中的に使用するワークロードをサポートします。

g. ワークロードのパフォーマンスは IOPS によって制限されていますか。

   i. ストレージ最適化 インスタンスは、ローカルストレージへの高いシーケンシャルな読み書きアクセス (IOPS) を必要とするワークロード向けに設計されています。

h. ワークロードの特性は、すべてのメトリクスにおいてバランスの取れたニーズを示していますか。

   i. ワークロードの CPU にはトラフィックの急上昇に対応するためのバーストが必要ですか。

      A. バーストパフォーマンス インスタンスは、コンピューティング最適化インスタンスに似ていますが、コンピューティング最適化インスタンスに見られる固定 CPU ベースラインを超えてバーストできるという点が異なります。

   ii. 汎用 インスタンスは、あらゆる特性においてバランスが取れており、さまざまなワークロードをサポートします。

i. コンピューティングインスタンスは Linux で実行されており、ネットワークインターフェイスカードのネットワークスループットによって制限されていますか。

   i. 「 パフォーマンスに関する質問 5、ベストプラクティス 2: 使用可能なネットワーク機能を評価する 」を確認し、パフォーマンスのニーズに合った適切なインスタンスのタイプおよびファミリーを特定します。

j. ワークロードは、特定のアベイラビリティゾーンで 1 年間コミットできる、一貫性のある予測可能なインスタンスを必要としますか。

   i. リザーブドインスタンス を使用すると、特定のアベイラビリティーゾーン内でキャパシティ予約が確定されます。リザーブドインスタンスは、特定のアベイラビリティゾーンでコンピューティング性能が必要な場合に最適です。

k. ワークロードには、専用ハードウェアを必要とするライセンスがありますか。

   i. 専有ホスト は、既存のソフトウェアライセンスをサポートし、コンプライアンス要件への準拠を支援します。

l. コンピューティングソリューションはバーストし、同期処理が必要ですか。

   i. オンデマンドインスタンス では、1 時間または 1 秒単位でコンピューティング性能を利用でき、長期的な契約は必要ありません。このインスタンスは、パフォーマンスのベースラインを超えるようなバースト的なニーズに適しています。

m. コンピューティングソリューションは、ステートレスでフォールト トレラント、かつ非同期ですか。

   i. スポットインスタンス では、未使用のインスタンスキャパシティをステートレスでフォールトトレラントなワークロードに活用できます。

2. コンテナを Fargateで実行していますか。

   a. タスクのパフォーマンスはメモリまたは CPU によって制限されていますか。

      i. メモリまたは CPU を調整するために、 タスクサイズ を使用します。

   b. トラフィックパターンのバーストによって、パフォーマンスが影響を受けていますか。

選択

    i. トラフィックパターンに合わせるために、 Auto Scaling の設定を使用します。

3. コンピューティングソリューションは Lambda上にありますか。
    a. 少なくとも 4 週間分のメトリクスがあり、トラフィックパターンとメトリクスが今後もほぼ同じになると予測できますか。
        i. 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Compute Optimizer を使用して、コンピューティング特性に最適なコンピューティング構成に関する機械学習の推奨事項を取得します。
    b. AWS Compute Optimizer を使用するのに十分なメトリクスがありますか。
        i. Compute Optimizer を使用するのに必要なメトリクスがない場合は、 AWS Lambda Power Tuning を使用して最適な設定を選択できます。
    c. 関数のパフォーマンスはメモリまたは CPU によって制限されていますか。
        i. パフォーマンスニーズのメトリクスを満たすように Lambda メモリ を設定します。
    d. 関数が実行時にタイムアウトしていますか。
        i. タイムアウトの設定 を変更します。
    e. 関数のパフォーマンスはアクティビティと並行処理のバーストによって制限されていますか。
        i. パフォーマンス要件を満たすように 並行処理の設定 を行います。
    f. 関数が非同期で実行され、再試行で失敗していますか。
        i. イベントの最大経過時間と最大再試行回数を 非同期設定 で指定します。

実装計画に必要な工数レベル:

このベストプラクティスを確立するには、現在のコンピューティングの特性とメトリクスを把握している必要があります。こうしたメトリクスを収集し、ベースラインを確立した上で、これらのメトリクスを使用して最適なコンピューティングオプションを特定するには、 低 ~ 中 程度の工数が必要です。これは、負荷テストや実験によって検証するのが最善です。

リソース

関連ドキュメント:

- Cloud Compute with AWS (AWS を使用したクラウドコンピューティング)
- AWS Compute Optimizer
- EC2 インスタンスタイプ
- EC2 インスタンスのプロセッサのステート制御

- EKS コンテナ: EKS ワーカーノード
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Functions: Lambda Function Configuration (関数: Lambda 関数の設定)

関連動画:

- Amazon EC2 foundations (CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Optimize performance and cost for your AWS compute (CMP323-R1) (AWS コンピューティングのパフォーマンスとコストを最適化する)

関連サンプル:

- Rightsizing with Compute Optimizer and Memory utilization enabled (Compute Optimizer とメモリ使用率を有効にした場合のライトサイジング)
- AWS Compute Optimizer Demo code (AWS Compute Optimizer デモコード)

PERF02-BP03 コンピューティング関連のメトリクスを収集する

コンピューティングリソースのパフォーマンスを理解するには、各種システムの実際の使用率を記録して追跡する必要があります。このデータは、リソース要件についてより正確な判断を行うために使用できます。

ワークロードでは、メトリクス、ログ、イベントなどのデータが大量に生成される可能性があります。既存のストレージ、モニタリング、可観測性のサービスが、生成されたデータを管理できるかどうかを判断してください。どのメトリクスがリソースの使用率を反映し、単一のプラットフォーム全体で収集、集計、相関できるかを特定します。このメトリクスは、システム全体を容易に可視化し、パフォーマンス改善の機会と問題を迅速に特定できるよう、すべてのワークロードリソース、アプリケーション、サービスを表している必要があります。

期待される成果: コンピューティング関連のリソースに関係するすべてのメトリクスが、単一のプラットフォーム上で特定、収集、集約されて関連付けが行われ、コストと運用の目標をサポートするために保持が実装されます。

一般的なアンチパターン:

- メトリクスの検索に手動ログファイルのみを使用している。

- 内部ツールにのみメトリクスを発行している。
- 一部のモニタリングソフトウェアで記録されるデフォルトのメトリクスのみを使用している。
- 問題が発生したときにだけメトリクスを確認している。

このベストプラクティスを活用するメリット: ワークロードのパフォーマンスをモニタリングするには、一定期間にわたって複数のパフォーマンスメトリクスを記録する必要があります。これらのメトリクスにより、パフォーマンスの異常を検出できます。また、ビジネスメトリクスに照らし合わせてパフォーマンスを測定することで、ワークロードのニーズを満たしているかどうかを確認できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

コンピューティング関連のメトリクスを特定、収集、集計し、関連付けを行います。Amazon CloudWatch などのサービスを使用すると、実装をより迅速かつ簡単に維持できます。デフォルトで記録されるメトリクスに加えて、ワークロード内のシステムレベルのメトリクスを追加で特定し、追跡します。CPU 使用率、メモリ、ディスク I/O、ネットワークのインバウンドおよびアウトバウンドメトリクスなどのデータを記録し、使用状況レベルやボトルネックを把握します。このデータは、ワークロードのパフォーマンスやコンピューティングソリューションの使用状況を理解するために不可欠です。これらのメトリクスをデータ駆動型のアプローチの一部として使用し、ワークロードのリソースを積極的に調整および最適化します。

実装手順:

1. 追跡するのが重要なコンピューティングソリューションメトリクスはどれですか。
    a. EC2 のデフォルトのメトリクス
    b. Amazon ECS のデフォルトのメトリクス
    c. EKS のデフォルトのメトリクス
    d. Lambda のデフォルトのメトリクス
    e. EC2 のメモリとディスクのメトリクス
2. 現在、承認済みの口ギングおよび監視ソリューションを使用していますか。
    a. Amazon CloudWatch
    b. AWS Distro for OpenTelemetry
    c. Amazon Managed Service for Prometheus
3. セキュリティおよび運用の目標に合ったデータ保持ポリシーを特定、構成しましたか。

   a. CloudWatch メトリクスのデフォルトのデータ保持
   b. CloudWatch Logs のデフォルトのデータ保持
4. メトリクスおよびログの集計エージェントをどのようにデプロイしますか。
   a. AWS Systems Manager オートメーション
   b. OpenTelemetry Collector

実装計画に必要な工数レベル: すべてのコンピューティングリソースからのメトリクスを特定、追跡、収集、集約し、関連付けるには、 中 程度の労力が必要です。

リソース

関連ドキュメント:

- Amazon CloudWatch のドキュメント
- CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログを収集する
- Accessing Amazon CloudWatch Logs for AWS Lambda
- コンテナインスタンスでの CloudWatch Logs の使用
- カスタムメトリクスをパブリッシュする
- AWS の回答: 集中ログ記録
- CloudWatch メトリクスを発行する AWS のサービス
- AWS Fargate での Amazon EKS のモニタリング

関連動画:

- Application Performance Management on AWS
- モニタリング計画を立てる

関連サンプル:

- Level 100: Monitoring with CloudWatch Dashboards
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards

PERF02-BP04 適切なサイジングによって必要な設定を決定する

ワークロードのさまざまなパフォーマンス特性と、それらの特性とメモリ、ネットワーク、CPU 使用率との関連を分析します。このデータは、ワークロードのプロファイルに最適なリソースを選択するために使用します。たとえば、メモリ集約型のワークロード (データベースなど) には r ファミリーのインスタンスが最適ですが、バーストするワークロードでは、伸縮自在なコンテナシステムからより多くのメリットを得ることができます。

一般的なアンチパターン:

- すべてのワークロードに対して使用できる最大のインスタンスを選択する。
- 管理しやすいように、すべてのインスタンスタイプを 1 つのタイプに標準化する。

このベストプラクティスを活用するメリット: AWS のコンピューティングサービスに精通していれば、さまざまなワークロードに適したソリューションを判断できます。ワークロードのさまざまなコンピューティングサービスを選択したら、それらのコンピューティングサービスを簡単に試し、ワークロードのニーズを満たすのはどれかをすばやく判断できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

適切なサイジングによってワークロード設定を変更する: パフォーマンスと全体的な効率性の両方を最適化するには、ワークロードに必要なリソースを特定します。CPU よりもメモリを必要とするシステムの場合は、メモリ最適化されたインスタンスを選択します。メモリ負荷の高くないデータ処理を実行するコンポーネントの場合は、コンピューティング最適化されたインスタンスを選択します。適切なサイジングを行うことで、ワークロードが要求されたリソースのみを使用しながら、可能な限り最高のパフォーマンスを達成することが可能になります。

リソース

関連ドキュメント:

- AWS Compute Optimizer
- AWS を使用したクラウドコンピューティング
- EC2 インスタンスタイプ
- ECS コンテナ: Amazon ECS コンテナインスタンス
- EKS コンテナ: EKS ワーカーノード

- 関数: Lambda 関数の設定
- EC2 インスタンスのプロセッサのステート制御

関連動画:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- スタートアップ企業向けコンピューティングオプションの選択方法
- Optimize performance and cost for your AWS compute (CMP323-R1)

関連サンプル:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer デモコード

PERF02-BP05 利用可能な伸縮性のあるリソースを使用する

クラウドは、需要の変化に対応するためのさまざまなメカニズムを通じて、リソースを動的に拡張または縮小する柔軟性を提供します。コンピューティング関連のメトリクスと組み合わせることによって、ワークロードは自動的に変化に対応し、最適な一連のリソースを利用して目標を達成できるようになります。

均衡の取れた需要と供給はワークロードコストを最小限に抑えますが、プロビジョニングの時間と個々のリソース障害に対応するために十分な供給を計画する必要もあります。需要は固定的である場合と変動する場合があるため、管理が負担になったり、不相応に多額なコストがかからないようにするためにも、メトリクスと自動化が必要になります。

AWS では、さまざまなアプローチで需要と供給を一致させることができます。コスト最適化の柱のホワイトペーパーでは、コストに関して以下のアプローチを使用する方法が説明されています。

- 需要ベースのアプローチ
- バッファーベースのアプローチ
- 時間ベースのアプローチ

ワークロードのデプロイメントで、確実にスケールアップおよびスケールダウンイベントを対処できるようにしてください。スケールダウンイベントのテストシナリオを作成して、ワークロードが期待どおりに動作することを確認します。

一般的なアンチパターン:

- アラームに対応するために手動でキャパシティーを増やす。
- スケーリングイベントの後、スケールダウンして元に戻すのではなく、キャパシティーを増加させたままにする。

このベストプラクティスを活用するメリット: ワークロードの伸縮性を設定してテストすることで、トラフィックの変化に応じたコストの削減、パフォーマンスベンチマークの維持、信頼性の向上を実現できます。本稼働用以外のほとんどのインスタンスは、使用されていないときに停止するべきです。未使用のインスタンスを手動でシャットダウンすることは可能ですが、大規模な場合は実用的ではありません。また、ボリュームベースの伸縮性を活用することもできます。これにより、需要の急上昇時にコンピューティングインスタンスの数を自動的に増やし、需要の減少時にキャパシティーを減らすことによって、パフォーマンスとコストを最適化できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

伸縮性を活用する: 伸縮性は、持っているリソースの供給を、それらのリソースに対する需要と一致させます。インスタンス、コンテナ、関数のいずれにも、伸縮性の仕組みが備わっています。サービスの機能として実装されている場合も、自動スケーリングと組み合わせて実現する場合もあります。アーキテクチャの伸縮性を活用して、あらゆる規模のパフォーマンス要件を満たすために十分なキャパシティーを確保してください。伸縮自在なリソースのスケールアップまたはスケールダウンのメトリクスが、デプロイされているワークロードのタイプに対して検証されていることを確認します。動画トランスコーディングアプリケーションをデプロイしようとする場合、100% の CPU 使用率が想定されるため、プライマリメトリクスにするべきではありません。代替手段として、インスタンスタイプのスケーリングを待機しているトランスコーディングジョブのキューの深さに対して測定することができます。ワークロードのデプロイメントがスケールアップおよびスケールダウンイベントの両方に対処できることを確実にします。ワークロードコンポーネントを安全にスケールダウンすることは、需要があるときにリソースをスケールアップするのと同じくらい重要です。スケールダウンイベントのテストシナリオを作成して、ワークロードが期待どおりに動作することを確認します。

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- EC2 インスタンスタイプ
- ECS コンテナ: Amazon ECS コンテナインスタンス
- EKS コンテナ: EKS ワーカーノード
- 関数: Lambda 関数の設定
- EC2 インスタンスのプロセッサのステート制御

関連動画:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system

関連サンプル:

- Amazon EC2 Auto Scaling グループの例
- Amazon EFS のチュートリアル

PERF02-BP06 メトリクスに基づいてコンピューティングニーズを再評価する

システムレベルのメトリクスを使用して、ワークロードの経時的な動作と要件を特定します。利用可能なリソースとこれらの要件を比較することによってワークロードのニーズを評価し、ワークロードのプロファイルに最も良く一致するようにコンピューティング環境を変更します。たとえば、時間がたつにつれて、システムが当初の想定よりもメモリ集約型であることがわかる場合があります。このため、別のインスタンスファミリー、またはインスタンスサイズに移行することでパフォーマンスと効率性の両方が向上する可能性があります。

一般的なアンチパターン:

- ワークロードに関する洞察を得るために、システムレベルのメトリクスのモニタリングのみを行う。
- ピーク時のワークロード要件に合わせてコンピューティングニーズを設計する。

- 新しいコンピューティングソリューションに移行する方がワークロードの特性に合っているにもかかわらず、スケーリングやパフォーマンスの要件を満たすためにコンピューティングソリューションのサイズを大きくしすぎる。

このベストプラクティスを活用するメリット: パフォーマンスとリソース使用率を最適化するには、統合された運用ビュー、リアルタイムの詳細なデータ、履歴参照が必要です。自動ダッシュボードを作成してこのデータを視覚化し、メトリクスの計算を実行して運用と利用に関する洞察を得ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

データ駆動型のアプローチを使用してリソースを最適化する: パフォーマンスと効率性を最大限に高めるには、ワークロードから継続的に収集したデータを使用してリソースを調整および最適化します。ワークロードの現在のリソース使用状況におけるトレンドを調べて、ワークロードのニーズにより適合させるために変更を加えることができる場所を特定します。リソースが過剰に使用されると、システムのパフォーマンスが低下します。その一方で、リソースの使用率が低いと、リソースの使用効率が低下し、コストがより高くなります。

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- AWS Compute Optimizer
- AWS を使用したクラウドコンピューティング
- EC2 インスタンスタイプ
- ECS コンテナ: Amazon ECS コンテナインスタンス
- EKS コンテナ: EKS ワーカーノード
- 関数: Lambda 関数の設定
- EC2 インスタンスのプロセッサのステート制御

関連動画:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system

関連サンプル:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer デモコード

PERF 3 ストレージソリューションはどのように選択すればよいですか?

システムにとって最適なストレージソリューションは、アクセス方法 (ブロック、ファイル、オブジェクト)、アクセスパターン (ランダム、シーケンシャル)、必要なスループットやアクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、可用性と耐久性に関する制約によって異なります。優れた設計のシステムでは、複数のストレージソリューションを使用し、さまざまな機能を有効にしてパフォーマンスとリソースの使用効率を高めています。

ベストプラクティス

- PERF03-BP01 ストレージ特性と要件を理解する
- PERF03-BP02 利用可能な設定オプションを評価する
- PERF03-BP03 アクセスパターンとメトリクスに基づいて意思決定を行う

PERF03-BP01 ストレージ特性と要件を理解する

ワークロードのストレージニーズを特定および文書化し、各ロケーションのストレージ特性を定義します。ストレージ特性の例としては、共有可能なアクセス、ファイルサイズ、成長率、スループット、IOPS、レイテンシー、アクセスパターン、およびデータ永続性などがあります。これらの特性を利用して、ブロック、ファイル、オブジェクト、インスタンスの各ストレージサービスが、ストレージのニーズに対して最も効率的なソリューションであるかどうかを評価します。

期待される成果: ストレージ要件ごとにストレージ要件を特定および文書化し、利用可能なストレージソリューションを評価します。主なストレージ特性に基づいて、チームは、選択したストレージサービスがワークロードのパフォーマンスのためにどのように役立つかを理解します。主な基準には、データアクセスパターン、成長率、スケーリングのニーズ、レイテンシー要件があります。

一般的なアンチパターン:

- すべてのワークロードに対して、Amazon Elastic Block Store (Amazon EBS) などの 1 つのストレージタイプのみを使用する。
- すべてのワークロードのストレージアクセスパフォーマンス要件が類似していることを前提としている。

このベストプラクティスを活用するメリット: 特定された必要な特性に基づいてストレージソリューションを選択することは、ワークロードのパフォーマンスを向上させ、コストを削減し、ワークロードを維持するための運用にかかる労力を軽減するのに役立ちます。ストレージサービスのソリューション、設定、およびロケーションは、ワークロードのパフォーマンスにメリットをもたらします。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードの最も重要なストレージパフォーマンスメトリクスを特定し、ベンチマークまたは負荷テストを使用して、データ駆動型アプローチの一環として改善を実施します。このデータを使用してストレージソリューションの制約が発生している場所を特定し、そのソリューションを改善する設定オプションを調べます。予想されるワークロードの成長率を判定し、この成長率に対応するストレージソリューションを選択します。AWS ストレージサービスを調査して、さまざまなワークロードのニーズに適したストレージソリューションを特定します。AWS でストレージソリューションをプロビジョニングすることにより、ストレージサービスをテストして、ワークロードのニーズに適しているかどうかを判断する機会が増えます。

AWS のサービス	主な特徴	一般的なユースケース
Amazon S3	99.999999999% の耐久性、拡大無制限、どこからでもアクセス可能、アクセスと回復力に基づいた複数のコストモデル	クラウドネイティブアプリケーションデータ、データのアーカイブおよびバックアップ、分析、データレイク、静的ウェブサイトホスティング、IoT データ
Amazon S3 Glacier	数秒から数時間のレイテンシー、拡大無制限、最安値のコスト、長期ストレージ	データアーカイブ、メディアアーカイブ、長期バックアップ保持
Amazon EBS	ストレージサイズには管理とモニタリングが必要、低レ	COTS アプリケーション、I/O 集約型アプリケーション、リ

AWS のサービス	主な特徴	一般的なユースケース
	イテンシー、永続的ストレージ、99.8% ~ 99.9% の耐久性、ほとんどのボリュームタイプは 1 つの EC2 インスタンスのみからアクセス可能	レーショナルデータベースおよび NoSQL データベース、バックアップとリカバリ
EC2 インスタンスストア	事前定義されたストレージサイズ、最も低いレイテンシー、永続化されない、1 つの EC2 インスタンスのみからアクセス可能	COTS アプリケーション、I/O インテンシブアプリケーション、インメモリデータストア
Amazon EFS	99.999999999% の耐久性、拡大無制限、複数のコンピューティングサービスによるアクセス可能	モダナイズされたアプリケーションが複数のコンピューティングサービスでファイルを共有、コンテンツ管理システムのスケーリングのためのファイルストレージ
Amazon FSx	4 つのファイルシステム (NetApp、OpenZFS、Windows File Server、および Amazon FSx for Lustre) をサポート、ファイルシステムごとに異なるストレージが利用可能、複数のコンピューティングサービスからのアクセス可能	クラウドネイティブなワークロード、プライベートクラウドでのバースト、特定のファイルシステムを必要とする移行ワークロード、VMC、ERP システム、オンプレミスファイルストレージおよびバックアップ
Snow ファミリー	ポータブルデバイス、256 ビットの暗号化、NFS エンドポイント、オンボードコンピューティング、TB 規模のストレージ	クラウド、ストレージ、および極端なオンプレミス条件下のコンピューティングへのデータの移行、ディザスタリカバリ、リモートデータ収集

AWS のサービス	主な特徴	一般的なユースケース
AWS Storage Gateway	クラウド対応ストレージに低レイテンシーなオンプレミスアクセスを提供、フルマネージド型オンプレミスキャッシュ	オンプレミスデータのクラウド移行、オンプレミスソースからクラウドデータレイクへの入力、モダナイズされたファイル共有

実装手順:

1. ベンチマークまたはロードテストを使用して、ストレージニーズの主な特性を収集します。主な特徴は次のとおりです。
   a. 共有可能 (どのコンポーネントがこのストレージにアクセスするか)
   b. 成長率
   c. スループット
   d. レイテンシー
   e. I/O サイズ
   f. 耐久性
   g. アクセスパターン (読み取りまたは書き取り、頻度、急増、または一貫性)
2. ストレージ特性をサポートするストレージのタイプを特定します。
   a. Amazon S3 には、無制限のスケーラビリティ、高可用性、およびアクセシビリティに関して複数のオプションがあります。Amazon S3 内外のオブジェクトの転送やアクセスには、 Transfer Acceleration または Access Points などのサービスを使用して、ロケーション、セキュリティニーズ、アクセスパターンをサポートします。この Amazon S3 パフォーマンスガイドラインを使用して ワークロードのパフォーマンスに関するニーズを満たすために、Amazon S3 設定を最適化します。
   b. Amazon S3 Glacier はデータアーカイブのために構築された Amazon S3 のストレージクラスです。ミリ秒単位のアクセスから 5 ~ 12 時間単位のアクセスまで、コストやセキュリティの異なる 3 つのアーカイブソリューションから選択できます。Amazon S3 Glacier は、ビジネス要件とデータ特性をサポートするデータライフサイクルを実装することで、パフォーマンス要件を満たすことができます。
   c. Amazon Elastic Block Store (Amazon EBS) は Amazon Elastic Compute Cloud (Amazon EC2) 向けに設計された、ハイパフォーマンスなブロックストレージサービスです。異なる特性を持つ SSD または HDD ベースの ソリューションから選択できます。これらは IOPS または ス

ループットを優先します。EBS ボリュームは、高性能なワークロード、ファイルシステム、データベースのプライマリストレージ、またはアタッチされたストレージシステムのみにアクセスできるアプリケーションに適しています。

d. Amazon EC2 インスタンスストア は、Amazon EC2 インスタンスにアタッチするという点で Amazon EBS と似ています。ただし、インスタンスストアは、バッファやキャッシュなどの一時的なコンテンツとして使用するのが理想的な、一時的なストレージに過ぎません。インスタンスストアをデタッチすることはできず、インスタンスがシャットダウンするとすべてのデータが失われます。インスタンスストアは、データの永続化を必要としない、高い I/O パフォーマンスと低レイテンシーのユースケースに使用することができます。

e. Amazon Elastic File System (Amazon EFS) は、さまざまなタイプのコンピューティングソリューションからアクセスできる、マウント可能なファイルシステムです。Amazon EFS は、ストレージを自動的に拡大/縮小し、パフォーマンスに最適化されているため、一貫した低レイテンシーを実現します。EFS には 2 つのパフォーマンス設定モードがあります。汎用と最大 I/O です。汎用は、読み出しレイテンシがミリ秒未満、書き込みレイテンシは 1 桁ミリ秒です。最大 I/O の機能により、共有ファイルシステムを必要とする何千ものコンピューティングインスタンスをサポートできます。Amazon EFS は以下をサポートしています。 2 つのスループットモード: バーストとプロビジョンド急増するアクセスパターンを経験するワークロードには、バーストスループットモードが役立ちます。一方、常に高いパフォーマンスを発揮するワークロードには、プロビジョンドスループットモードが適しています。

f. Amazon FSx は最新の AWS コンピューティングソリューションをベースに構築されており、一般的に使用されている 4 つのファイルシステム (NetApp ONTAP、OpenZFS、Windows File Server、Lustre) をサポートしています。Amazon FSx レイテンシー、スループット、および IOPS は ファイルシステムごとに異なるため、ワークロードのニーズに合わせて適切なファイルシステムを選択する際に考慮する必要があります。

g. AWS Snow Family は、オンラインおよびオフラインでのクラウドへのデータ移行、オンプレミスでのデータ保存やコンピューティングをサポートするストレージおよびコンピューティングデバイスです。AWS Snow デバイスは大量のオンプレミスデータの収集、データの処理、およびそれらのデータのクラウド移行をサポートします。ファイルの数、ファイルサイズ、および圧縮に関する 文書化されたパフォーマンスのベストプラクティス があります。

h. AWS Storage Gateway は、オンプレミスアプリケーションに、クラウドベースのストレージへのアクセスを提供します。AWS Storage Gateway は、Amazon S3、Amazon S3 Glacier、Amazon FSx、および Amazon EBS を含む、さまざまなクラウドストレージサービスをサポートしています。また、iSCSI、SMB、および NFS などの多くのプロトコルをサポートしています。アクセス頻度の高いデータをオンプレミスにキャッシュし、変更されたデータと

圧縮されたデータのみを AWS に送信することで、低レイテンシーのパフォーマンスを実現します。

3. 新しいストレージソリューションで実験を行い、最適な設定を確認したら、移行を計画し、パフォーマンスメトリクスを検証します。これは継続的なプロセスであり、主な特性の変更、または利用可能なサービスやオプションに変更があった場合に再評価される必要があります。

実装計画に必要な工数レベル: ワークロードが 1 つのストレージソリューションから別のものに移行する場合は 中 程度の工数が必要になる可能性があります。

リソース

関連ドキュメント:

- Amazon EBS Volume Types
- Amazon EC2 ストレージ
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon FSx for NetApp ONTAP performance
- Amazon FSx for OpenZFS performance
- Amazon S3 Glacier: Amazon S3 Glacier ドキュメント
- Amazon S3: リクエストレートとパフォーマンスに関する考慮事項
- AWS でのクラウドストレージ
- AWS Snow Family
- EBS I/O の特性

関連動画:

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

関連サンプル:

- Amazon EFS CSI Driver

- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 のサンプル
- Amazon FSx for Lustre Container Storage Interface (CSI) ドライバー

PERF03-BP02 利用可能な設定オプションを評価する

さまざまな特性や設定オプションとそれらがストレージにどのように関連するかを評価します。ワークロードのためのストレージ容量とパフォーマンスを最適化するために、プロビジョンド IOPS、SSD、磁気ストレージ、オブジェクトストレージ、アーカイブストレージ、またはエフェメラルストレージをどこでどのように使用するかを理解してください。

Amazon EBS には、ワークロードに対するストレージパフォーマンスとコストを最適にできる幅広いオプションがあります。これらのオプションは、データベースおよびブートボリュームなどのトランザクションワークロード向けの SSD を基盤とするストレージ (パフォーマンスは主に IOPS に依存) と、MapReduce および口グ処理などのスループット集約型ワークロード向けの HDD を基盤とするストレージ (パフォーマンスは主に MB/秒に依存) の 2 つの主なカテゴリに分けられます。

SSD タイプのボリュームには、レイテンシーに敏感なトランザクションワークロード向けのパフォーマンスの極めて高いプロビジョンド IOPS SSD と、さまざまなトランザクションデータで使用でき、価格とパフォーマンスのバランスが取れた汎用 SSD が含まれます。

Amazon S3 Transfer Acceleration を使用すると、クライアントと S3 バケットの間で、長距離にわたるファイル転送を高速で行えるようになります。Transfer Acceleration は、Amazon CloudFront のグローバルに分散したエッジロケーションを活用して、最適化されたネットワークパスでデータをルーティングします。集中的な GET リクエストがある S3 バケットのワークロードには、Amazon S3 と CloudFront を併用してください。大型のファイルをアップロードするときは、ネットワークスループットを最大化できるように、マルチパートアップロードを使用してください。

Amazon Elastic File System (Amazon EFS) は、AWS クラウドサービスおよびオンプレミスリソースでの使用のために、シンプルでスケーラブル、かつフルマネージド型の伸縮自在な NFS ファイルシステムを提供します。Amazon EFS は、多種多様なクラウドストレージワークロードをサポートするために、汎用パフォーマンスモードと最大 I/O パフォーマンスモードの 2 つのパフォーマンスモードを提供します。また、ファイルシステム向けに選択できる、バーストスループットとプロビジョンドスループットの 2 つのスループットモードもあります。ワークロードに使用する設定を決定するには、 Amazon EFS ユーザーガイド を参照してください。

Amazon FSx には 4 つのファイルシステムがあり、エンタープライズワークロード用の Amazon FSx for Windows File Server 、高パフォーマンスワークロード用の Amazon FSx for Lustre 、NetApp の ONTAP ファイルシステム用の Amazon FSx for NetApp ONTAP 、Linux ベースのファイルサーバー用の Amazon FSx for OpenZFS から選ぶことができます。FSx は SSD を基盤としており、高速、予測可能、スケーラブル、かつ一貫性のあるパフォーマンスを実現するように設計されています。Amazon FSx ファイルシステムは、持続的で高速な読み取り/書き込み速度と、一貫性のある低レイテンシーデータアクセスを提供します。スループットレベルは、ワークロードのニーズに合わせて必要なものを選択することができます。

一般的なアンチパターン:

- すべてのワークロードに対して、Amazon EBS などの 1 つのストレージタイプのみを使用する。
- すべてのストレージ層に対して実際のテストを行うことなく、すべてのワークロードにプロビジョンド IOPS を使用する。
- すべてのワークロードのストレージアクセスパフォーマンス要件が類似していることを前提としている。

このベストプラクティスを活用するメリット: すべてのストレージサービスオプションを評価することで、インフラストラクチャのコストとワークロードの維持に必要な労力を削減できます。これにより、新しいサービスや機能をデプロイするための市場投入までの時間を短縮できる可能性があります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ストレージ特性を特定する: ストレージソリューションを評価する場合には、必要なストレージ特性 (共有可能性、ファイルサイズ、キャッシュサイズ、レイテンシー、スループット、データの永続性など) を特定してください。その後、ニーズに最も適した AWS のサービスを見つけるために、要件と照らし合わせます。

リソース

関連ドキュメント:

- AWS でのクラウドストレージ
- Amazon EBS Volume Types
- Amazon EC2 ストレージ

- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier ドキュメント
- Amazon S3: リクエストレートとパフォーマンスに関する考慮事項
- AWS でのクラウドストレージ
- AWS でのクラウドストレージ
- EBS I/O の特性

関連動画:

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

関連サンプル:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 のサンプル

PERF03-BP03 アクセスパターンとメトリクスに基づいて意思決定を行う

ワークロードのアクセスパターンに基づいてストレージシステムを選択し、ワークロードがどのようにデータにアクセスするかを判断することによってそれらを設定します。ストレージの効率性を向上させるには、ブロックストレージではなくオブジェクトストレージを選択してください。選択したストレージオプションは、データのアクセスパターンに合わせて設定します。

データへのアクセス方法は、ストレージソリューションのパフォーマンスに影響します。アクセスパターンに最適なストレージソリューションを選択するか、パフォーマンスを最大にするためにストレージソリューションに合わせてアクセスパターンを変更することを検討してください。

RAID 0 アレイを作成することによって、単一のボリュームでプロビジョニングできるものよりも高レベルのパフォーマンスを、ファイルシステムのために実現することが可能になります。I/O パ

フォーマンスがフォルト トレランスよりも重要な場合は RAID 0 の使用を検討してください。例えば、データレプリケーションが既に個別にセットアップされている使用頻度が高いデータベースで使用できます。

お客様のワークロードが使用するすべてのストレージの選択肢について、ワークロードに適したストレージメトリクスを選択してください。バーストクレジットを使用するファイルシステムを利用する場合は、クレジット上限に近づいたときに通知するアラームを作成します。全体的なワークロードストレージの正常性を表示するには、ストレージダッシュボードを作成する必要があります。

Amazon EBS または Amazon FSx などの固定サイズのストレージシステムについては、使用済みのストレージの量を全体的なストレージサイズに照らしてモニタリングするようにし、可能であれば、しきい値に到達したときにストレージサイズを増加させるオートメーションを作成します。

一般的なアンチパターン:

- 顧客からの苦情がなければ、ストレージのパフォーマンスが十分であると考えている。
- ストレージ階層を 1 つだけ使用し、すべてのワークロードがその階層に適していると考えている。

このベストプラクティスを活用するメリット: パフォーマンスとリソース使用率を最適化するには、統合された運用ビュー、リアルタイムの詳細なデータ、履歴参照が必要です。1 秒間隔で自動ダッシュボードとデータを作成し、データに対してメトリクスの計算を実行し、ストレージニーズに対する運用と使用状況に関する洞察を得ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ストレージ使用量とアクセスパターンを最適化する: ワークロードのアクセスパターンと利用可能なストレージオプションの特性に基づいてストレージシステムを選択します。要件を満たしながら、オーバーヘッドを削減することを可能にする、最適なデータの保存場所を選択します。ストレージの特性に基づいてデータを設定および操作する際には、パフォーマンスの最適化とアクセスパターンを使用します (ボリュームのストライピング、データのパーティション化など)。

ストレージオプションに適切なメトリクスを選択する: ワークロードに適したストレージメトリクスを選択していることを確認します。各ストレージオプションには、時間の経過に伴うワークロードのパフォーマンス状況を追跡するためのさまざまなメトリクスが用意されています。ストレージバーストメトリクス (Amazon EFS のバーストクレジットのモニタリングなど) に照らして測定していることを確認します。Amazon Elastic Block Store やAmazon FSx など、固定サイズのストレージシステ

ムの場合は、使用したストレージの量と全体的なストレージサイズをモニタリングしていることを確認します。可能な場合はオートメーションを作成し、しきい値に達したときにストレージサイズを増やします。

メトリクスをモニタリングする: Amazon CloudWatch では、アーキテクチャ内のリソース全体のメトリクスを収集できます。また、カスタムメトリクスを収集および発行して、ビジネスメトリクスまたは導出メトリクスを表面化することも可能です。CloudWatch またはサードパーティーのソリューションを使用して、しきい値を超過したことを示すアラームを設定します。

リソース

関連ドキュメント:

- Amazon EBS Volume Types
- Amazon EC2 ストレージ
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier ドキュメント
- Amazon S3: リクエストレートとパフォーマンスに関する考慮事項
- AWS でのクラウドストレージ
- EBS I/O の特性
- Monitoring and understanding Amazon EBS performance using Amazon CloudWatch

関連動画:

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

関連サンプル:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale

- Amazon S3 のサンプル

PERF 4 データベースソリューションはどのように選択すればよいですか?

システムにとって最適なデータベースソリューションは、可用性、整合性、分断耐性、レイテンシー、耐久性、スケーラビリティ、クエリ機能などの要件に応じて異なります。多くのシステムでは、各種サブシステムに異なるデータベースソリューションを使用しているため、パフォーマンスを向上させるために活用する機能も異なります。システムに対して適切でないデータベースソリューションや機能を選択すると、パフォーマンス効率が低下する場合があります。

ベストプラクティス

- PERF04-BP01 データの特性を理解する
- PERF04-BP02 使用可能なオプションを評価する
- PERF04-BP03 データベースのパフォーマンスメトリクスを収集して記録する
- PERF04-BP04 アクセスパターンに基づいてデータストレージを選択する
- PERF04-BP05 アクセスパターンとメトリクスに基づいてデータストレージを最適化する

PERF04-BP01 データの特性を理解する

ワークロードのデータセットの特性、アクセスパターン、要件に最適なデータ管理ソリューションを選択します。データ管理ソリューションの選択および実装には、クエリ、スケーリング、ストレージの特性がワークロードのデータの要件をサポートしていることを確認する必要があります。さまざまなデータベースオプションがデータモデルにどのように適合するか、またユースケースに最適な構成オプションについて学びます。

AWS では、リレーショナル、key-value、ドキュメント、インメモリ、グラフ、時系列、台帳データベースなど多数のデータベースエンジンを提供しています。各データ管理ソリューションには、ユースケースとデータモデルをサポートするために使用できるオプションと設定があります。ワークロードでは、データの特性に応じて、いくつかの異なるデータベースソリューションを使用できる場合があります。特定の問題に最適なデータベースソリューションを選択することで、限定的な画一的アプローチのモノリシックなデータベースから脱却し、顧客のニーズに合わせたデータ管理に専念できます。

期待される成果: ワークロードのデータ特性が文書化され、サポートするデータベースソリューションの選択と設定を容易にし、考えられる代替手段についての洞察を得るのに十分な詳細が提供されます。

一般的なアンチパターン:

- 大規模なデータセットを同様の特性を持つ小さなデータコレクションにセグメント化する方法を考慮していないため、データと成長の特性により適した、目的に特化したデータベースを使用する機会が失われている。
- データアクセスパターンを前もって特定せず、複雑でコストのかかる作業をやり直すことになる。
- 必要に応じて迅速にスケールしないデータストレージ戦略を使用することで成長を制限している。
- すべてのワークロードに対して 1 つのデータベースタイプとベンダーを選択する。
- 特定のタイプのデータベースソリューションに関する社内知識と経験があるため、1 つのデータベースソリューションに固執する。
- オンプレミス環境でうまく機能したことを理由にデータベースソリューションをキープする。

このベストプラクティスを活用するメリット: さまざまなワークロードに適したデータベースソリューションを決定できるように、すべての AWS データベースソリューションについての知識を身に付けておきます。ワークロードに適したデータベースソリューションを選択したら、各データベースサービスを簡単に試し、ワークロードのニーズを満たし続けているかどうかを判断できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

- 実現可能なコスト削減が特定できない可能性がある。
- データを適切なレベルでセキュリティ保護できない可能性がある。
- データアクセスおよびストレージパフォーマンスが最適でない可能性がある。

実装のガイダンス

ワークロードのデータの特性とアクセスパターンを定義します。利用できるデータベースソリューションを確認し、どのソリューションがデータ要件をサポートするかを特定します。1 つのワークロードに対して、複数のデータベースを選択できます。各サービスまたはサービスのグループを審査し、個別に評価します。代替として利用可能なデータ管理ソリューションがデータの一部またはすべてに対して特定された場合は、コスト、セキュリティ、パフォーマンス、信頼性における利点を引き出す可能性のある代替の実装を試してみます。新しいデータ管理のアプローチを導入する場合は、既存のドキュメントを更新します。

タイプ	AWS のサービス	主な特徴	一般的なユースケース
リレーショナル	Amazon RDS、Amazon Aurora	参照整合性、ACID トランザクション、Schema-On-Write	ERP、CRM、市販のソフトウェア
key-value	Amazon DynamoDB	高スループット、低レイテンシー、ほぼ無限のスケーラビリティ	ショッピングカート (e コマース)、製品カタログ、チャットアプリケーション
ドキュメント	Amazon DocumentDB	JSON ドキュメントを保存し任意の属性でクエリを実行する	コンテンツ管理 (CMS)、顧客プロファイル、モバイルアプリケーション
インメモリ	Amazon ElastiCache、Amazon MemoryDB	ミリ秒のレイテンシー	キャッシュ、ゲームのリーダーボード
グラフ	Amazon Neptune	データ間のリレーションに意味がある関係性の高いデータ	ソーシャルネットワーク、パーソナライゼーションエンジン、不正検出
時系列	Amazon Timestream	プライマリディメンションが時刻のデータ	DevOps、IoT、モニタリング
ワイドカラム	Amazon Keyspaces	Cassandra ワークロード	産業機器のメンテナンス、ルートの最適化
台帳	Amazon QLDB	不変で暗号的に検証可能な変更の台帳	記録システム、医療、サプライチェーン、金融機関

実装手順

1. データはどのように構造化されていますか (非構造化、key-value、半構造化、リレーショナルなど)。
   a. データが構造化されていない場合は、 Amazon S3 などのオブジェクトストアか、 Amazon DocumentDB などの NoSQL データベースを検討します。
   b. key-value データの場合は、 DynamoDB、 ElastiCache for Redis または MemoryDB を検討します。
   c. データがリレーショナル構造を持っている場合、どのレベルの参照整合性が必要ですか。
      i. 外部キー制約の場合は、 Amazon RDS および Aurora などのリレーショナルデータベースがこのレベルの整合性を提供できます。
      ii. 通常、NoSQL データモデル内では、データをドキュメントまたはテーブルをまたいで結合するのではなく、単一のドキュメントまたはドキュメントのコレクションに非正規化して、単一のリクエストで取得します。
2. ACID (atomicity、consistency、isolation、durability) への準拠は必要ですか。
   a. リレーショナルデータベースに関連付けられている ACID プロパティが必要な場合は、 Amazon RDS および Aurora などのリレーショナルデータベースを検討します。
3. どのような一貫性モデルが必要ですか。
   a. アプリケーションが結果整合性を許容できる場合は、NoSQL の実装を検討します。他の特性を確認して、最適な NoSQL データベース を選びます。
   b. 強整合性が必要な場合は、 DynamoDB または Amazon RDSなどのリレーショナルデータベースで強力な整合性のある読み込みを使用できます。
4. どのような形式のクエリと結果をサポートする必要がありますか (SQL、CSV、Parque、Avro、JSON など)。
5. どのようなデータ型、フィールドサイズ、および全体の量が存在しますか (テキスト、数値、空間、時系列計算、バイナリまたはブロブ、ドキュメントなど)。
6. ストレージ要件は時間の経過とともにどのように変化しますか。 これにより、スケーラビリティにどのような影響がありますか。
   a. サーバーレスデータベース (DynamoDB および Amazon Quantum Ledger Database など) は、ほぼ無制限のストレージまで動的にスケールアップします。
   b. リレーショナルデータベースには、プロビジョニングされたストレージに上限があり、多くの場合、この上限に達すると、シャーディングなどのメカニズムを介して水平方向に分割する必要があります。

7. 書き込みクエリに対する読み取りクエリの割合はどのくらいですか。 キャッシングによってパフォーマンスが向上する可能性はありますか。

   a. 読み取り負荷の高いワークロードでは、キャッシングレイヤーを使用することでメリットが得られます。これには、 ElastiCache または DAX (データベースが DynamoDB の場合) などがあります。

   b. 読み取りは、 Amazon RDSなどのリレーショナルデータベースを使用して読み取りレプリカにオフロードすることもできます。

8. ストレージや変更 (OLTP - オンライントランザクション処理) または取得やレポート (OLAP - オンライン分析処理) のどちらが優先されますか。

   a. 高スループットのトランザクション処理については、DynamoDB や Amazon DocumentDB などの NoSQL データベースを検討します。

   b. 分析クエリの場合は、 Amazon Redshift などの列指向データベースを検討するか、データを Amazon S3 にエクスポートして Athena または QuickSight を使用して分析を実行することを検討します。

9. このデータの機密性はどの程度で、どのようなレベルの保護と暗号化が必要ですか。

   a. すべての Amazon RDS および Aurora エンジンは、AWS KMS を使用した保管時のデータ暗号化をサポートしています。Microsoft SQL Server と Oracle では、Amazon RDS を使用する場合、ネイティブの Transparent Data Encryption (TDE) もサポートします。

   b. DynamoDB の場合、 IAM できめ細かなアクセス制御 (FGAC) を使用し、誰がどのデータにアクセスできるかをキーレベルで制御できます。

10. データにはどのレベルの耐久性が必要ですか。

   a. Aurora は、リージョン内の 3 つのアベイラビリティーゾーンにわたってデータを自動的に複製します。これはつまり、データの耐久性が高く、データ損失の可能性が低くなることを意味します。

   b. DynamoDB は、複数のアベイラビリティーゾーンに自動的に複製され、高可用性とデータ耐久性を提供します。

   c. Amazon S3 は、99.9999999% (イレブンナイン) の耐久性を提供します。Amazon RDS や DynamoDB などの多くのデータベースサービスでは、長期的な保持とアーカイブの目的で、Amazon S3 へのデータのエクスポートをサポートしています。

11. すべきこと 目標復旧時間 (RTO) または目標復旧時点 (RPO) の要件はソリューションに影響しますか。

   a. Amazon RDS、Aurora、DynamoDB、Amazon DocumentDB、Neptune はすべて、ポイントインタイムリカバリとオンデマンドのバックアップと復元をサポートしています。

b. 高可用性の要件については、DynamoDB テーブルを グローバルテーブル 機能を使用してグローバルに複製でき、Aurora クラスターを Global Database 機能を使用して複数のリージョンでレプリケートできます。さらに、S3 バケットは、クロスリージョンレプリケーションを使用して AWS リージョン 間で複製できます。

12.商用データベースエンジンやライセンスコストから離れたいという希望はありますか。

a. Amazon RDS または Aurora で PostgreSQL や MySQL などのオープンソースのエンジンを検討します。

b. 商用データベースエンジンからオープンソースへの移行を行うには、 AWS DMS および AWS SCT を利用します。

13.データベースには運用上どのようなことが期待されますか。 マネージドサービスへの移行は主な懸念事項ですか。

a. Amazon EC2 の代わりに Amazon RDS を利用し、NoSQL データベースをセルフホスティングする代わりに DynamoDB または Amazon DocumentDB を利用することで、運用上の諸経費を削減できます。

14.データベースへのアクセスは現在どのように行われていますか。 アプリケーションアクセスのみですか、それともビジネスインテリジェンス (BI) ユーザーやその他の接続された既製アプリケーションが存在しますか。

a. 外部ツールに依存している場合は、ツールがサポートするデータベースとの互換性を維持することが必要になる場合があります。Amazon RDS は Microsoft SQL Server、Oracle、MySQL、PostgreSQL など、サポートするさまざまなエンジンバージョンとの完全な互換性があります。

15.以下は、利用可能なデータ管理サービスのリストと、その最適な使用場所です。

a. リレーショナルデータベースは、定義済みのスキーマとそれらの関係を使用してデータを格納します。これらのデータベースは、ACID (atomicity、consistency、isolation、durability) トランザクションをサポートし、参照整合性と強固なデータ整合性を維持するように設計されています。従来のアプリケーション、エンタープライズリソースプランニング (ERP)、顧客関係管理 (CRM)、e コマースの多くは、リレーショナルデータベースを使用してデータを保存します。これらのデータベースエンジンの多くは Amazon EC2 で実行することも、AWS のマネージド データベースサービス(Amazon Aurora、 Amazon RDS、または Amazon Redshift) から選ぶこともできます。

b. キー値データベースは、一般的に大量のデータを保存および取得するために、一般的なアクセスパターン用に最適化されています。これらのデータベースは、極端な量の同時リクエストでさえも、迅速な応答時間を実現します。高トラフィックのウェブアプリケーション、e コマースシステム、ゲーミングアプリケーションは、key-value データベースの典型的なユースケース

です。AWS では、 Amazon DynamoDBを利用することができます。これはマルチリージョンとマルチマスターに対応し、耐久性に優れたフルマネージドデータベースで、インターネットスケールのアプリケーションに対応した、組み込みのセキュリティ、バックアップと復元の機能、インメモリキャッシュを備えています

c. インメモリデータベースは、データへのリアルタイムアクセス、最小のレイテンシー、最大のスループットが必要なアプリケーションに使用されます。これらのデータベースは、データをメモリ内に直接保存することにより、ミリ秒単位のレイテンシーでは不十分なアプリケーションにマイクロ秒単位のレイテンシーを提供します。インメモリデータベースは、アプリケーションキャッシング、セッション管理、ゲームのリーダーボード、および地理空間アプリケーションに使用できます。 Amazon ElastiCache は、フルマネージド型のインメモリデータストアで、 Redis または Memcachedと互換性があります。アプリケーションの耐久性要件も高い場合、超高速パフォーマンス用の耐久性のあるインメモリデータベースサービスである Amazon MemoryDB for Redis がこれを組み合わせて提供します。

d. ドキュメントデータベースは、半構造化データを JSON 型のドキュメントとしとして保存するように設計されています。これらのデータベースは、開発者がコンテンツ管理、カタログ、およびユーザープロファイルなどのアプリケーションをすばやく構築し、更新するために役立ちます。 Amazon DocumentDB は、高速で、スケーラブル、かつ高可用性の完全マネージド型ドキュメントデータベースサービスで、MongoDB ワークロードに対応しています。

e. ワイドカラムデータストアは NoSQL データベースの一種です。テーブル、行、および列を使用しますが、リレーショナルデータベースとは異なり、同じテーブル内でも列の名前と形式が行ごとに異なる場合があります。ワイドカラムデータストアは通常、設備保全、フリート管理、およびルート最適化のための大規模な産業アプリケーションでの使用が見られます。 Amazon Keyspaces (Apache Cassandra 向け) は、ワイドカラムにスケールできる、可用性に優れたマネージド型の Apache Cassandra 互換データベースサービスです。

f. グラフデータベースは、関連性が高いグラフデータセット間における何百万もの関係を、大規模に、かつミリ秒単位のレイテンシーでナビゲートし、クエリする必要があるアプリケーション向けのデータベースです。多くの企業が、不正行為検出、ソーシャルネットワーキング、およびレコメンデーションエンジン向けにグラフデータベースを使用しています。 Amazon Neptune は、高速で信頼性に優れたフルマネージド型のグラフデータベースサービスで、関連性が高いデータセットを処理するアプリケーションの構築と実行を容易にします。

g. 時系列データベースは、時間の経過と共に変化するデータを効率的に収集、合成し、それらからインサイトを導き出します。時系列データベースは、IoT アプリケーション、DevOps、および産業用テレメトリに利用できます。 Amazon Timestream は、IoT および運用アプリケーション向けの高速でスケーラブルなフルマネージド型の時系列データベースサービスで、1 日あたり数兆件のイベントの保存と分析を容易にします。

h. 台帳データベースは、あらゆるアプリケーションについて、トランザクションのスケーラブルでイミュータブル、かつ暗号的な検証が可能なレコードを維持する信頼された中央機関を提供します。台帳データベースは、SoR、サプライチェーン、登録、および銀行取引にも使用されています。 Amazon Quantum Ledger Database (Amazon QLDB) はフルマネージド型の台帳データベースで、信頼された中央機関によって所有される、透過的でイミュータブル、かつ暗号的な検証が可能なトランザクションログを提供します。Amazon QLDB は、すべてのアプリケーションのデータ変更を追跡し、経時的な変更の完全で検証可能な履歴を維持します。

実装計画に必要な工数レベル: ワークロードがあるデータベースソリューションから別のデータベースソリューションに移行する場合は、アプリケーションのリファクタリングに 高 程度の工数が必要になる可能性があります。

リソース

関連ドキュメント:

- AWS でのクラウドデータベース
- AWS Database Caching
- Amazon DynamoDB Accelerator
- Amazon Aurora を使用する際のベストプラクティス
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum ベストプラクティス
- Amazon DynamoDB ベストプラクティス
- Choose between EC2 and Amazon RDS
- Best Practices for Implementing Amazon ElastiCache

関連動画:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

関連サンプル:

- Optimize Data Pattern using Amazon Redshift Data Sharing
- データベースの移行
- MS SQL Server - AWS Database Migration Service (DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune サンプル

PERF04-BP02 使用可能なオプションを評価する

データ管理ソリューションを選択する前に、利用可能なデータベースのオプションと、それぞれがどのようにパフォーマンスを最適化するかを理解します。負荷テストを使用して、ワークロードにとって重要なデータベースメトリクスを特定します。データベースのオプションを検討する際は、パラメータグループ、ストレージオプション、メモリ、コンピューティング、リードレプリカ、結果整合性、接続プーリング、キャッシュオプションなど、さまざまな側面を考慮します。メトリクスを改善するために、こうしたさまざまな設定オプションを試します。

期待される成果: ワークロードでは、データ型によって、1 つまたは複数のデータベースソリューションを使用できます。データベースの機能と利点が、データの特性、アクセスパターン、ワークロードの要件に最適に合致します。データベースのパフォーマンスとコストを最適化するには、データアクセスパターンを評価して適切なデータベースオプションを決定する必要があります。許容可能なクエリ時間を評価し、選択したデータベースオプションが要件を満たすことを確認します。

一般的なアンチパターン:

- データアクセスパターンを特定しない。
- 選択したデータ管理ソリューションの設定オプションを把握していない。
- 使用できる設定オプションを確認せずに、インスタンスサイズを増やすことのみに頼っている。
- 選んだソリューションのスケーリングに関する特性をテストしない。

このベストプラクティスを活用するメリット: データベースオプションを確認し、試してみることで、インフラストラクチャのコストを削減し、パフォーマンスとスケーラビリティを高め、ワークロードの維持に必要な労力を軽減できる場合があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

- 1 つのサイズで すべてに対応するよう データベースを最適化する必要があると、不必要な妥協をすることになる。

- データベースソリューションがトラフィックパターンに合わせて設定されていないため、コストが高くなる。
- スケーリングの問題から運用上の問題が発生する可能性がある。
- データを適切なレベルでセキュリティ保護できない可能性がある。

実装のガイダンス

データベースオプションを設定できるように、ワークロードデータの特性を理解します。負荷テストを行い、主要なパフォーマンスメトリクスとボトルネックを特定します。こうした特性およびメトリクスを使用して、データベースオプションを評価し、さまざまな設定を試します。

AWS のサービス	Amazon RDS、Amazon Aurora	Amazon DynamoDB	Amazon DocumentDB	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspaces	Amazon QLDB
コンピューティングのスケーリング	インスタンスサイズを増やす、Aurora サーバーレスインスタンスは負荷の変化に応じて自動スケーリングする	オンデマンドキャパシティモードによる自動の読み取り/書き込みスケーリング、またはプロビジョンドキャパシティモードでプロ	インスタンスサイズを増やす	インスタンスサイズを増やす、クラスターにノードを追加する	インスタンスサイズを増やす	自動的にスケーリングしてキャパシティを調整する	オンデマンドキャパシティモードによる自動の読み取り/書き込みスケーリング、またはプロビジョンドキャパシティモードでプロ	自動的にスケーリングしてキャパシティを調整する

AWS のサービス	Amazon RDS、Amn Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
		ビジョニングされた読み取り/書き込みキャパシティの自動スケーリング					ビジョニングされた読み取り/書き込みキャパシティの自動スケーリング	
読み取りのスケールアウト	すべてのエンジンでリードレプリカをサポート。Aurora はリードレプリカインスタンスの自動スケーリングをサポートする	プロビジョニングされた読み取りキャパシティユニットを増やす	リードレプリカ	リードレプリカ	リードレプリカ。リードレプリカインスタンスの自動スケーリングをサポート	自動的にスケーリングする	プロビジョニングされた読み取りキャパシティユニットを増やす	文書化された同時実行制限まで自動的にスケールアップ

AWS のサービス	Amazon RDS、An Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
書き込みのスケールアウト	インスタンスサイズを増やす、アプリケーション内の書き込みをバッチ処理する、またはデータベースの前にキューを追加する複数のインスタンスにまたがるアプリケーションレベルのシャーディン	プロビジョニングされた書き込みキャパシティユニットを増やす 最適なパーティションキーを確保し、パーティションレベルでの書き込みのスロットリングを防ぐ	プライマリインスタンスサイズを増やす	Redis をクラスターモードで使用して書き込みを複数のシャードに分散する	インスタンスサイズを増やす	書き込みリクエストがスケーリング中にスロットリングされる可能性がある。スロットリング例外が発生した場合は、同じ (またはそれ以上の) スループットでデータを送信し続け、自動的にスケーリング	プロビジョニングされた書き込みキャパシティユニットを増やす 最適なパーティションキーを確保し、パーティションレベルでの書き込みのスロットリングを防ぐ	文書化された同時実行制限まで自動的にスケールアップ

AWS のサービス	Amazon RDS、An Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
	グで水平スケーリング					する。書き込みをバッチ処理して同時実行される書き込みリクエストを減らす		
エンジンの設定	パラメータグループ	該当しない	パラメータグループ	パラメータグループ	パラメータグループ	該当しない	該当しない	該当しない

AWS のサービス	Amazon RDS、Amn Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
キャッシング	インメモリキャッシュ、パラメータグループを介して設定可能。Elasticache for Redis などの専用キャッシュと組み合わせて、よくアクセスされるアイテムに対するリクエストをオフロードする	DAX (DAX) 完全マネージド型キャッシュが利用可能	インメモリキャッシュ。必要に応じて、ElastiCache for Redis などの専用キャッシュと組み合わせて、よくアクセスされるアイテムに対するリクエストをオフロードすることも可能。	キャッシュが主な機能	クエリ結果キャッシュを使用して読み取り専用クエリの結果をキャッシュする	Timestream には 2 つのストレージ層があり、そのうち 1 つはハイパフォーマンスのインメモリ層	ElastiCache for Redis などの専用キャッシュを別にデプロイし、よくアクセスされるアイテムに対するリクエストをオフロードする	該当しない

AWS のサービス	Amazon RDS、Amn Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
高可用性 / ディザスタリカバリ	本番ワークロードで推奨される設定は、2 つ目のアベイラビリティーゾーンでスタンバイインスタンスを実行して、リージョン内で回復力を提供すること。 リージョン間の回復力については、Aurora	1 つのリージョン内で可用性が高い。テーブルは DynamoD グローバルテーブルを使用してリージョン間で複製できる	複数のインスタンスをアベイラビリティゾーンをまたいで作成して可用性を向上。 スナップショットはリージョンをまたいで共有でき、クラスタは DMS を使用して複製可能。これによりクロスリー	本番クラスタで推奨される設定は 2 つ目のアベイラビリティゾーンに少なくとも 1 つのノードを作成すること。 リージョン間でのクラスタの複製には ElastiCache Global Datastore を使用できる。	他のアベイラビリティゾーンのリードレプリカはフェイルオーバーターゲットとして機能する。 スナップショットはリージョンをまたいで共有でき、クラスタは Neptune ストリームを使	1 つのリージョンで可用性が高い。クロスリージョンレプリケーションには Timestream SDK を使用したカスタムアプリケーション開発が必要。	1 つのリージョン内で可用性が高い。 クロスリージョンレプリケーションにはカスタムアプリケーションロジックまたはサードパーティ製ツールが必要	1 つのリージョン内で可用性が高い。 リージョン間で複製するには、Amazon QLDB ジャーナルのコンテンツを S3 バケットにエクスポートし、クロスリージョンレプリケーション用にバケット

AWS のサービス	Amazon RDS、An Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
	Global Database を使用可能		ジョンレプリケーションやディザスタリカバリが提供される		用して複製可能。これにより、2 つの異なるリージョンの 2 つのクラスター間でデータを複製できる。			を設定する。

実装手順

1. 選択したデータベースで使用できる設定オプションは何ですか。
   a. Amazon RDS および Aurora のパラメータグループを使用すると、キャッシュに割り当てられたメモリやデータベースのタイムゾーンの調整など、一般的なデータベースエンジンレベルの設定を調整できます。
   b. Amazon RDS、Aurora、Neptune、Amazon DocumentDB などのプロビジョンドデータベースサービスや、Amazon EC2 にデプロイされたデータベースサービスでは、インスタンスタイプ、プロビジョニングされたストレージを変更し、リードレプリカを追加できます。
   c. DynamoDB では、オンデマンドとプロビジョンドの 2 つのキャパシティモードを指定できます。さまざまなワークロードに対応するために、随時モードを切り替えて、プロビジョニングドモードで割り当てられているキャパシティを増やすことができます。

2. ワークロードでは、読み取りまたは書き込みの負荷が高くなりますか。

    a. 読み取りをオフロードするためにどのようなソリューションを使用できますか (リードレプリカ、キャッシュなど)。

        i. DynamoDB テーブルの場合、キャッシュに DAX を使用して読み取りをオフロードできます。

        ii. リレーショナルデータベースの場合、ElastiCache for Redis クラスターを作成して、最初にキャッシュから読み取り、リクエストされたアイテムが存在しない場合にデータベースにフォールバックするようにアプリケーションを設定できます。

        iii. Amazon RDS および Aurora などのリレーショナルデータベース、Neptune などのプロビジョンド NoSQL データベース、Amazon DocumentDB はすべて、ワークロードの読み取り部分をオフロードするためのリードレプリカの追加をサポートします。

        iv. DynamoDB などのサーバーレスデータベースは自動的にスケーリングします。ワークロードを処理するのに十分な読み取りキャパシティユニット (RCU) がプロビジョニングされていることを確認します。

    b. 書き込みのスケーリングにはどのようなソリューションを使用できますか (パーティションキーのシャーディング、キューの導入など)。

        i. リレーショナルデータベースの場合、インスタンスのサイズを増やして増加したワークロードに対応するか、プロビジョンド IOPS を増やして、基盤となるストレージへのスループットを増やせるようにします。

            - また、データベースに直接書き込むのではなく、データベースの前にキューを導入することもできます。このパターンでは、データの取り込みをデータベースから切り離し、フローレートを制御することで、データベースが過負荷になるのを回避できます。
            - 存続時間の短いトランザクションを大量に作成するのではなく、書き込みリクエストをバッチ処理することで、書き込み量の多いリレーショナルデータベースのスループットを向上させることができます。

        ii. DynamoDB のようなサーバーレスデータベースは、キャパシティモードに応じて、自動的に、またはプロビジョニングされた書き込みキャパシティユニット (WCU) を調整することにより、書き込みスループットをスケーリングできます。

            - ただし、特定のパーティションキーのスループット上限に達すると、引き続き ホットパーティションの問題が発生する可能性があります。これは、より均等に分散されたパーティションキーを選択するか、パーティションキーを書き込みシャーディングすることで緩和できます。

3. 現在または予想される 1 秒あたりのピークトランザクション数はどのくらいですか。 このトラフィック量とこの量 +X% を使用してスケーリングの特性を理解します。

   a. PostgreSQL 用の pg_bench などのネイティブツールを使用して、データベースのストレステストを行い、ボトルネックとスケーリングの特性を理解できます。

   b. 合成ワークロードに加えて、実際の条件をシミュレートするために再現できるように、実稼働環境に似たトラフィックをキャプチャする必要があります。

4. サーバーレスまたは伸縮自在にスケーリングが可能なコンピューティングを使用している場合は、これをスケーリングした場合のデータベースへの影響をテストします。必要に応じて、接続管理またはプーリングを導入し、データベースへの影響を軽減します。

   a. Amazon RDS および Aurora でデータベースへの接続を管理するには、RDS Proxy を使用できます。

   b. DynamoDB などのサーバーレスデータベースには関連付けられている接続はありませんが、負荷の急増に対応するためにプロビジョンドキャパシティおよび自動スケーリングのポリシーを検討します。

5. 負荷は予測可能ですか。負荷の急増やアクティビティのない期間はありますか。

   a. アクティビティのない期間がある場合は、こうした期間中にプロビジョンドキャパシティまたはインスタンスサイズをスケールダウンすることを検討します。Aurora Serverless V2 は、負荷に応じて自動でスケールアップおよびスケールダウンします。

   b. 非本番インスタンスの場合は、業務時間外に一時停止または停止することを検討します。

6. アクセスパターンやデータの特性に応じてデータモデルをセグメント化および分割する必要がありますか。

   a. データを他のサービスに移動するには、AWS DMS または AWS SCT を使用することを検討します。

実装計画に必要な工数レベル:

このベストプラクティスを確立するには、現在のデータの特性とメトリクスを把握している必要があります。こうしたメトリクスを収集し、ベースラインを確立した上で、これらのメトリクスを使用して最適なデータベースの設定オプションを特定するには、 低 ~ 中 程度の工数が必要です。これは、負荷テストと実験によって検証するのが最善策です。

リソース

関連ドキュメント:

- AWS でのクラウドデータベース
- AWS Database Caching

- Amazon DynamoDB Accelerator
- Amazon Aurora を使用する際のベストプラクティス
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum ベストプラクティス
- Amazon DynamoDB ベストプラクティス

関連動画:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

関連サンプル:

- Amazon DynamoDB サンプル
- AWS データベース移行サンプル
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for Postgress DB

PERF04-BP03 データベースのパフォーマンスメトリクスを収集して記録する

データ管理システムのパフォーマンスを把握するには、関連性のあるメトリクスを追跡することが重要です。このようなメトリクスは、データ管理リソースを最適化し、ワークロード要件が満たされていることを確かめ、ワークロードのパフォーマンスの概要を明確に把握するのに役立ちます。データベースのパフォーマンスに関連するパフォーマンスの測定値を記録するツール、ライブラリ、システムを使用します。

メトリクスには、データベースがホストされているシステムに関連するメトリクス (CPU、ストレージ、メモリ、IOPS など) と、データ自体にアクセスするためのメトリクス (1 秒あたりのトランザクション数、クエリレート、応答時間、エラーなど) があります。これらのメトリクスは、すべてのサポートスタッフまたは運用スタッフが簡単にアクセスできる必要があります。また、傾向、異常、ボトルネックを特定できる十分な過去の記録が必要です。

期待される成果: データベースのワークロードのパフォーマンスをモニタリングするには、一定期間にわたって複数のパフォーマンスメトリクスを記録する必要があります。これにより、異常を検出し、ビジネスメトリクスに照らしてパフォーマンスを測定して、ワークロードのニーズを確実に満たすことができます。

一般的なアンチパターン:

- メトリクスの検索に手動ログファイルのみを使用している。
- チームが使用する内部ツールにのみメトリクスを発行しており、ワークロードの全体像を把握できていない。
- 一部のモニタリングソフトウェアで記録されるデフォルトのメトリクスのみを使用している。
- 問題が発生したときにだけメトリクスを確認している。
- システムレベルのメトリクスのみをモニタリングし、データアクセスや使用状況に関するメトリクスを把握していない。

このベストプラクティスを活用するメリット: パフォーマンスのベースラインを確立すると、ワークロードの通常の動作とワークロードの要件を理解するのに役立ちます。異常なパターンをより迅速に特定してデバッグできるため、データベースのパフォーマンスと信頼性が向上します。データベースのキャパシティは、パフォーマンスを犠牲にすることなくコストを最適化するように設定できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

- 異常なパフォーマンスレベルと通常のパフォーマンスレベルを区別できなければ、問題の特定とそれに伴う意思決定が困難になる。
- 実現可能なコスト削減が特定できない可能性がある。
- 成長パターンが特定されないため、信頼性やパフォーマンスの低下につながる可能性がある。

実装のガイダンス

データベースグ関連のメトリクスを特定、収集、集計し、関連付けを行います。メトリクスは、データベースをサポートする基盤となるシステムとデータベース自体の両方のメトリクスが含まれている必要があります。基盤となるシステムのメトリクスには、CPU 使用率、メモリ、使用可能なディスク容量、ディスク I/O、ネットワークのインバウンドとアウトバウンドに関するメトリクスなどがあり、データベースのメトリクスには 1 秒あたりのトランザクション数、上位のクエリ、平均クエリレート、応答時間、インデックス使用率、テーブルロック、クエリのタイムアウトの数、開いてい

る接続の数などがあります。このデータは、ワークロードのパフォーマンスやデータベースソリューションの使用状況を理解するために不可欠です。これらのメトリクスをデータ駆動型アプローチの一部として使用し、ワークロードのリソースを調整および最適化します。

実装手順:

1. 追跡するべき重要なデータベースメトリクスはどれですか。
    a. Amazon RDS のメトリクスのモニタリング
    b. Performance Insights を使用したモニタリング
    c. 拡張モニタリング
    d. DynamoDB メトリクス
    e. DynamoDB DAX のモニタリング
    f. MemoryDB のモニタリング
    g. Amazon Redshift のモニタリング
    h. 時系列のメトリクスとディメンション
    i. Aurora のクラスターレベルのメトリクス
    j. Amazon Keyspaces のモニタリング
    k. Amazon Neptune のモニタリング
2. 運用上の異常なパフォーマンスの問題を検出する機械学習ソリューションは、データベースのモニタリングに役立ちますか。
    a. Amazon DevOps Guru for Amazon RDS は、パフォーマンス上の問題を可視化し、是正措置についての推奨事項を提供します。
3. SQL の使用状況についてアプリケーションレベルの詳細が必要ですか。
    a. AWS X-Ray をアプリケーションに組み込むと、洞察を得て、単一のクエリのすべてのデータポイントをカプセル化できます。
4. 現在、承認済みのロギングおよび監視ソリューションがありますか。
    a. Amazon CloudWatch では、アーキテクチャ内のリソース全体のメトリクスを収集できます。また、カスタムメトリクスを収集および発行して、ビジネスメトリクスまたは導出メトリクスを表面化することも可能です。CloudWatch またはサードパーティーのソリューションを使用して、しきい値を超過したことを示すアラームを設定します。
5. セキュリティおよび運用の目標に合ったデータ保持ポリシーを特定、構成しましたか。
    a. CloudWatch メトリクスのデフォルトのデータ保持
    b. CloudWatch Logs のデフォルトのデータ保持

選択

実装計画に必要な工数レベル: すべてのデータベースリソースからのメトリクスを特定、追跡、収集、集約し、関連付けるには、 中 程度の労力が必要です。

リソース

関連ドキュメント:

- AWS Database Caching
- Amazon Athena top 10 performance tips
- Amazon Aurora を使用する際のベストプラクティス
- Amazon DynamoDB Accelerator
- Amazon DynamoDB ベストプラクティス
- Amazon Redshift Spectrum ベストプラクティス
- Amazon Redshift performance
- AWS でのクラウドデータベース
- Amazon RDS Performance Insights

関連動画:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

関連サンプル:

- Level 100: Monitoring with CloudWatch Dashboards
- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitoring Workshop

PERF04-BP04 アクセスパターンに基づいてデータストレージを選択する

ワークロードのアクセスパターンを使用して、使用するサービスとテクノロジーを決定します。パフォーマンスやスケールなどの機能以外の要件に加えて、アクセスパターンは、データベースおよびストレージのソリューションの選択に大きく影響します。まず重要な側面は、トランザクショ

ン、ACID コンプライアンス、一貫した読み取りの必要性です。すべてのデータベースがこれらをサポートしているわけではなく、ほとんどの NoSQL データベースは結果整合性モデルを提供しています。2 番目に重要な側面は、書き込みと読み取りの時間と空間における分散です。グローバルに分散されているアプリケーションでは、最適なストレージソリューションを特定するために、トラフィックパターン、レイテンシー、アクセス要件を考慮する必要があります。選択すべき 3 つ目の重要な側面は、クエリパターンの柔軟性、ランダムアクセスパターン、1 回限りのクエリです。テキストおよび自然言語の処理、時系列、グラフ向けの高度に専門化されたクエリ機能に関する条件についても考慮する必要があります。

期待される成果: 文書化されたデータアクセスパターンを基にデータストレージが選択されます。このデータアクセスパターンには、最も一般的な読み取り、書き込み、削除クエリ、アドホックな計算および集計の必要性、データの複雑性、データの独立性、必要な一貫性に関するニーズなどが含まれます。

一般的なアンチパターン:

- 運用管理を簡素化するため、データベースベンダーを 1 社だけ選択する。
- 時間が経過してもデータアクセスパターンが変わらないと考えている。
- 複雑なトランザクション、ロールバック、一貫性ロジックをアプリケーションに実装する。
- データベースがトラフィックの急増に対応できるように設定されており、データベースリソースがほとんどアイドル状態のままになる。
- トランザクションや分析の用途に共有データベースを使用する。

このベストプラクティスを活用するメリット: アクセスパターンを基にデータストレージを選択および最適化すると、開発の複雑さが軽減され、パフォーマンス改善の機会が最適化されます。リードレプリカ、グローバルテーブル、データのパーティショニング、キャッシュをいつ使用するべきかを理解することで、運用上の諸経費を減らし、ワークロードのニーズに応じてスケーリングできるようになります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

データアクセスパターンを特定、評価し、適切なストレージ設定を選択します。各データベースソリューションには、ストレージソリューションを設定および最適化するオプションがあります。収集したメトリクスとログを使用し、オプションを試してみることで、最適な設定を特定します。下記の表で、データベースサービスごとのストレージオプションを確認できます。

AWS のサービス	Amazon RDS、Amn Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
スケーリングするストレージ	ストレージの自動スケーリングオプションを使用して、プロビジョンされたストレージを自動的にスケーリングできる。プロビジョンド IOPS のストレージタイプを使用する場合は、プロビジョンされた	自動的にスケーリングする。テーブルはサイズに関して制約がない。	ストレージの自動スケーリングオプションを使用して、プロビジョンされたストレージをスケーリングできる	ストレージはインメモリで、インスタンスタイプまたはインスタンス数に関連付けられている	ストレージの自動スケーリングオプションを使用して、プロビジョンされたストレージを自動的にスケーリングできる	インメモリ層と磁気層の保持期間を日数で設定する。	テーブルのストレージを自動的にスケールアップまたはスケールダウンする	自動的にスケーリングする。テーブルはサイズに関して制約がない。

AWS のサービス	Amazon RDS、An Aurora	Amazon DynamoD	Amazon Document B	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	Amazon QLDB
	ストレージに関係なく IOPS をスケーリングすることもできる							

実装手順:

1. 予想されるデータとトラフィックの増加を特定して文書化します。
    a. Amazon RDS および Aurora は、文書化された制限までのストレージの自動スケーリングをサポートします。この制限を超える場合は、古いデータを Amazon S3 に移してアーカイブして過去のデータを集計するか、シャーディングを使って水平にスケーリングすることを検討します。
    b. DynamoDB および Amazon S3 は、ほぼ無限のストレージボリュームに自動的にスケーリングします。
    c. EC2 で実行されている Amazon RDS インスタンスとデータベースは、手動でサイズ変更でき、EC2 インスタンスには追加ストレージ用に新しい EBS ボリュームを後日追加できます。
    d. インスタンスタイプはアクティビティの変化に応じて変更できます。例えば、テスト中は小さいインスタンスから始め、サービスに本番のトラフィックが入ってくるようになったらインスタンスをスケーリングすることができます。Aurora Serverless V2 は負荷の変化に応じて自動でスケーリングします。

1. 通常のパフォーマンスおよびピークパフォーマンスに関する要件 (1 秒あたりのトランザクション数 TPS および 1 秒あたりのクエリ数 QPS) と一貫性に関する要件 (ACID および結果整合性) を文書化します。

2. ソリューションのデプロイに関する側面と、データベースのアクセス要件 (グローバル、マルチ AZ、リードレプリケーション、複数の書き込みノード) を文書化します。

実装計画に必要な工数レベル: データ管理ソリューションのログまたはメトリクスがない場合は、データアクセスパターンを特定して文書化する前にそれを準備する必要があります。データアクセスパターンを把握できたら、データストレージの選択および設定には 低 程度の工数が必要です。

リソース

関連ドキュメント:

- AWS Database Caching
- Amazon Athena top 10 performance tips
- Amazon Aurora を使用する際のベストプラクティス
- Amazon DynamoDB Accelerator
- Amazon DynamoDB ベストプラクティス
- Amazon Redshift Spectrum ベストプラクティス
- Amazon Redshift performance
- AWS でのクラウドデータベース
- Amazon RDS のストレージタイプ

関連動画:

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

関連サンプル:

- AWS での分散負荷テスト

PERF04-BP05 アクセスパターンとメトリクスに基づいてデータストレージを最適化する

データの保存方法とクエリ方法を最適化して可能な限り最高のパフォーマンスを達成するパフォーマンス特性とアクセスパターンを使用します。インデックス作成、キー分散、データウェアハウス設

計、またはキャッシング戦略などの最適化が、システムのパフォーマンスまたは全体的な効率性にどのように影響するかを測定してください。

一般的なアンチパターン:

- メトリクスの検索に手動ログファイルのみを使用している。
- 内部ツールにのみメトリクスを発行している。

このベストプラクティスを確立するメリット: ワークロードに必要なメトリクスを確実に満たすためには、読み取りと書き込みの両方に関連するデータベースパフォーマンスメトリクスをモニタリングする必要があります。このデータを使用して、データストレージレイヤーへの読み取りと書き込みの両方の新しい最適化を追加できます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

メトリクスとパターンに基づいてデータストレージを最適化する: レポートされたメトリクスを使用して、ワークロードのパフォーマンス不足の領域を特定し、データベースコンポーネントを最適化します。データのインデックス作成方法、キャッシュ方法、複数のシステム間での分散方法など、評価するパフォーマンス関連の特性は、データデータベースシステムごとに異なります。最適化の影響を測定します。

リソース

関連ドキュメント:

- AWS Database Caching (AWS データベースキャッシング)
- Amazon Athena top 10 performance tips
- Amazon Aurora を使用する際のベストプラクティス
- Amazon DynamoDB Accelerator
- Amazon DynamoDB ベストプラクティス
- Amazon Redshift Spectrum ベストプラクティス
- Amazon Redshift performance
- AWS でのクラウドデータベース
- DevOps Guru for RDS でパフォーマンスの異常を分析する
- DynamoDB の読み取り/書き込みキャパシティモード

関連動画:

- AWS purpose-built databases (DAT209-L) (AWS 目的別データベース)
- Amazon Aurora storage demystified: How it all works (DAT309-R) (Amazon Aurora ストレージの秘密: その仕組み)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

関連サンプル:

- Hands-on Labs for Amazon DynamoDB (Amazon DynamoDB 向けのハンズオンラボ)

PERF 5 ネットワーキングソリューションはどのように選択すればよいですか?

ワークロードに最適なネットワークソリューションは、レイテンシー、スループット要件、ジッター、および帯域幅に応じて異なります。ロケーションのオプションは、ユーザーまたはオンプレミスのリソースなどの物理的な制約に左右されます。これらの制約は、エッジロケーションまたはリソースの配置で相殺することができます。

ベストプラクティス

- PERF05-BP01 ネットワークがパフォーマンスに与える影響を理解する
- PERF05-BP02 使用可能なネットワーク機能を評価する
- PERF05-BP03 ハイブリッドワークロード用に適切なサイズの専用接続または VPN を選択する
- PERF05-BP04 ロードバランシングと暗号化のオフロードを活用する
- PERF05-BP05 パフォーマンスを高めるネットワークプロトコルを選択する
- PERF05-BP06 ネットワーク要件に基づいてワークロードのロケーションを選択する
- PERF05-BP07 メトリクスに基づいてネットワーク設定を最適化する

PERF05-BP01 ネットワークがパフォーマンスに与える影響を理解する

ネットワーク関連の意思決定がワークロードのパフォーマンスに与える影響を分析し、理解します。ネットワークは、アプリケーションコンポーネント、クラウドサービス、エッジネットワーク、オンプレミスデータ間の接続を担っているため、ワークロード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。ワークロードのパフォーマンスに加え、ユーザーエクスペリエンスも、ネットワークのレイテンシー、帯域幅、プロトコル、場所、ネットワークの混雑、ジッター、スループット、ルーティングルールの影響を受けます。

期待される成果: レイテンシー、パケットサイズ、ルーティングルール、プロトコル、サポートするトラフィックパターンなど、ワークロードのネットワーク要件をまとめて文書化します。利用可能なネットワークソリューションを確認し、ワークロードのネットワーク特性に適合するサービスを特定します。クラウドベースのネットワークは迅速に再構築できるため、パフォーマンス効率を向上させるためにもネットワーク アーキテクチャを時間とともに進化させる必要があります。

一般的なアンチパターン:

- すべてのトラフィックが既存のデータセンターを通過する。
- 実際の使用要件を把握せずに、Direct Connect セッションをオーバービルドする。
- ワークロードの特性および暗号化にかかるコストを考慮しない。
- クラウドのネットワーク戦略にオンプレミスのコンセプトと戦略を使用する。

このベストプラクティスを活用するメリット: ネットワークがワークロードのパフォーマンスに与える影響を理解することで、潜在的なボトルネックの特定、ユーザーエクスペリエンスの改善、信頼性の向上を実現しながら、ワークロードの変化に伴う運用保守業務を軽減できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードの重要なネットワークパフォーマンスメトリクスを特定し、ネットワークの特性を洗い出します。ベンチマークまたは負荷テストを使用して、データ駆動型アプローチの一部として要件を定義して文書化します。このデータを使用してネットワークソリューションの制約が発生している場所を特定し、ワークロードを改善できる設定オプションを調べます。クラウドネイティブネットワークで利用できる機能とオプションを理解し、要件に応じてワークロードのパフォーマンスにどのように影響するかを把握します。各ネットワーク機能には長所と短所があり、ワークロードの特性に適合し、ニーズに合わせてスケーリングするように設定できます。

実装手順:

1. ネットワークパフォーマンス要件を定義し、文書化します。
    a. ネットワークのレイテンシー、帯域幅、プロトコル、場所、トラフィックパターン (急増とその頻度)、スループット、暗号化、点検、ルーティングルールなどのメトリクスを含めます。
2. 基盤となるネットワークの特性を洗い出します。
    a. VPC フローログ
    b. AWS トランジットゲートウェイのメトリクス
    c. AWS PrivateLink のメトリクス

3. アプリケーションのネットワークの特性を洗い出します。
    a. Elastic Network Adaptor
    b. AWS App Mesh のメトリクス
    c. Amazon API Gateway のメトリクス
4. エッジネットワークの特性を洗い出します。
    a. Amazon CloudFront のメトリクス
    b. Amazon Route 53 のメトリクス
    c. AWS Global Accelerator のメトリクス
5. ハイブリッドネットワークの特性を洗い出します。
    a. Direct Connect のメトリクス
    b. AWS Site-to-Site VPN のメトリクス
    c. AWS Client VPN のメトリクス
    d. AWS クラウド WAN のメトリクス
6. セキュリティネットワークの特性を洗い出します。
    a. AWS Shield、WAF、Network Firewall のメトリクス
7. トレーシングツールでエンドツーエンドのパフォーマンスメトリクスを洗い出します。
    a. AWS X-Ray
    b. Amazon CloudWatch RUM
8. ネットワークパフォーマンスをベンチマーキングし、テストします。
    a. ネットワークループットをベンチマーキング する: インスタンスが同じ VPC 内にある場合、EC2 ネットワークパフォーマンスに影響する可能性のあるいくつかの要因。同じ VPC 内で EC2 Linux インスタンス間のネットワーク帯域幅を測定します。
    b. 負荷 テスト を実行し、ネットワークソリューションとオプションを試します。

実装計画に必要な工数レベル: ワークロードのネットワーク要件、オプション、利用可能なソリューションを文書化するには、 中 程度の労力が必要です。

リソース

関連ドキュメント:

- Application Load Balancer
- Linux での EC2 拡張ネットワーキング

- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS のネットワーク製品
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- Improve Global Network Performance for Applications
- EC2 Instances and Performance Optimization Best Practices
- Optimizing Network Performance for Amazon EC2 Instances
- Networking best practices and tips with the Well-Architected Framework
- AWS networking best practices in large-scale migrations

関連サンプル:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP02 使用可能なネットワーク機能を評価する

パフォーマンスの向上に役立つ可能性のあるクラウドのネットワーク機能を評価します。これらの機能の影響を、テスト、メトリクス、および分析を使って測定してください。たとえば、レイテンシー、パケット損失、ジッターを低減するために利用可能なネットワークレベルの機能を活用することができます。

多くのサービスはパフォーマンスを向上させるために作成され、他のサービスはネットワークパフォーマンスを最適化するための機能を提供するのが一般的です。AWS Global Accelerator およ

び Amazon CloudFront などのサービスは、パフォーマンスの向上のために用意されています。一方、ほとんどの他のサービスには、ネットワークトラフィックを最適化するという製品機能があります。ワークロードのパフォーマンス向上のために、EC2 インスタンスネットワーク機能、拡張ネットワーキングインスタンスタイプ、Amazon EBS 対応インスタンス、Amazon S3 Transfer Acceleration、および CloudFront などのサービス機能を確認してください。

期待される成果: ワークロード内のコンポーネントのインベントリを文書化し、コンポーネントごとにどのネットワーク構成がパフォーマンス要件を満たすのに役立つかを特定しました。ネットワーク機能を評価した後で、パフォーマンスメトリクスを実験および測定し、利用可能な機能をどのように使用するかを確認しました。。

一般的なアンチパターン:

- エンドユーザーに近い AWS リージョン ではなく、本社に最も近い AWS リージョン にワークロードを配置する。
- ワークロードパフォーマンスのベンチマークや、ベンチマークに対する継続的なワークロードパローマンスの評価に失敗する。
- パフォーマンス改善オプションのサービス設定を確認しない。

このベストプラクティスを活用するメリット: すべてのサービス機能とオプションを評価することにより、ワークロードパローマンスを向上させ、インフラストラクチャのコストを削減し、ワークロードを維持するために必要な労力を減らし、全体的なセキュリティ体制を強化できます。グローバルな AWS のバックボーンを活用すれば、最適なネットワークエクスペリエンスをお客様に提供することができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ネットワーク関連の設定オプションにはどのようなものがあるか、またそれらがワークロードにどのような影響を与えるかを確認します。これらのオプションがアーキテクチャとどのように相互作用し、測定されたパフォーマンスとユーザーが認識するパフォーマンスの両方に与える影響を理解することは、パフォーマンスの最適化にとって非常に重要です。

実装手順:

1. ワークロードコンポーネントのリストを作成します。
    a. 組織のネットワークを AWS クラウド WAN を使用して構築、管理、モニタリングします。

   b. ネットワークを可視化するために Network Manager を使用します。既存の設定管理データベース (CMDB) ツール (AWS Config など) を使用して、ワークロードや設定方法のインベントリを作成します。

2. これが既存のワークロードである場合、ボトルネックや改善すべき領域に特化して、パフォーマンスメトリクスのベンチマークを特定し、文書化します。パフォーマンスに関連するネットワークメトリクスは、ビジネス要件やワークロード特性により、ワークロードごとに異なります。最初のうちは、帯域幅、レイテンシー、パケットロス、ジッター、再送信などのメトリックスを確認することが重要かもしれません。
3. これが新しいワークロードである場合は、 負荷テスト を実施して、パフォーマンスのボトルネックを特定します。
4. 特定したパフォーマンスのボトルネックに対して、ソリューションの設定オプションを確認し、パフォーマンス改善の機会を見つけます。
5. ネットワークパスやルートが分からない場合は、 Network Access Analyzer を使用して特定します。
6. ネットワークプロトコルを確認して、さらにレイテンシーを減らします。
   - PERF05-BP05 パフォーマンスを高めるネットワークプロトコルを選択する
7. 複数の口ケーションで AWS Site-to-Site VPN を使用して AWS リージョン に接続している場合は、 高速 Site-to-Site VPN 接続 を確認して、ネットワークパフォーマンス向上のための機会を調べます。
8. ワークロードトラフィックが複数のアカウントにまたがっている場合は、ネットワークトポロジとサービスを評価して、レイテンシーを削減します。
   - 複数のアカウントに接続する際は、 VPC ピアリング と AWS Transit Gateway 間の、運用とパフォーマンスに関するトレードオフを評価します。AWS Transit Gateway は、AWS Site-to-Site VPN スループットをサポートしており、マルチパスを使用することにより、単一の 最大 IPsec 制限 を超えてスケーリングします。Amazon VPC と AWS Transit Gateway 間のトラフィックはプライベート AWS ネットワーク上に残り、インターネットには公開されません。AWS Transit Gateway は、数千の AWS アカウント やオンプレミスネットワークにまたがるすべての VPC を相互接続する方法を簡素化します。複数のアカウント間で AWS Transit Gateway を共有するには、 Resource Access Manager を使用します。グローバルネットワークトラフィックを可視化するには、 Network Manager を使用してネットワークメトリクスを一元的に把握することができます。
9. ユーザー口ケーションを確認し、ユーザーとワークロードとの間の距離を最短化します。
   a. AWS Global Accelerator は、Amazon Web Services グローバルネットワークインフラストラクチャを使用して、ユーザーのトラフィックのパフォーマンスを最大 60% まで向上するネット

ワークサービスです。インターネットが混雑している際には、AWS Global Accelerator は、アプリケーションへのパスを最適化して、パケット損失、ジッター、レイテンシーを一貫して低く保ちます。また、静的な IP アドレスを提供するため、DNS 設定の更新やクライアント向けアプリケーションの変更なしに、アベイラビリティゾーンや AWS リージョン 間でエンドポイントを簡単に移動させることができます。

b. Amazon CloudFront を利用することで、ワークロードのコンテンツ配信のパフォーマンスとレイテンシーをグローバルに向上させることができます。CloudFront は 410 以上のグローバルに分散したプレゼンスポイントがあり、コンテンツをキャッシュし、エンドユーザーに対してレイテンシーを減らすことができます。

c. Amazon Route 53 では、 レイテンシーベースルーティング、位置情報ルーティング、地理的近接性ルーティング、および IP ベースルーティング のオプションを提供しており、全世界ユーザーに対するワークロードのパフォーマンスを向上することができます。ワークロードトラフィックとユーザーロケーションを確認することにより、どのルーティングオプションによってワークロードパフォーマンスが最適化されるかを特定します。

10追加の Amazon S3 機能を評価してストレージ IOP を改善します。

a. Amazon S3 Transfer Acceleration により、外部ユーザーは、Amazon S3 にデータをアップロードする際に CloudFront のネットワーク最適化機能のメリットを享受できます。これにより、AWS クラウド への専用接続がないリモートロケーションから大量のデータを転送する機能が向上します。

b. Amazon S3 マルチリージョンアクセスポイント は、1 つのアクセスポイントを提供することにより、複数のリージョンへコンテンツをレプリケートし、ワークロードを簡素化します。マルチリージョンアクセスポイントが使用されている場合、低レイテンシーバケットを特定するサービスによって、データを要求したり Amazon S3 に書き込むことができます。

11コンピューティングリソースのネットワーク帯域幅を確認します。

a. EC2 インスタンス、コンテナ、および Lambda 機能が使用する Elastic Network Adapters (ENA) は、フロー単位で制限されています。プレイスメントグループを見直して、 EC2 ネットワークスループットを最適化します。フロー単位でのボトルネックを避けるために、複数フローを使用できるようアプリケーションを設計します。コンピューティング関連のネットワークメトリクスをモニタリングおよび可視化するために、 CloudWatch Metrics と `ethtool` を使用します。`ethtool` は ENA ドライバーに含まれており、追加のネットワーク関連のメトリクスを公開します。これらは、 カスタムメトリクス として CloudWatch に発行できます。

b. 新しい EC2 インスタンスでは、拡張ネットワーキングを利用できます。N シリーズの EC2 インスタンス(`M5n` と `M5dn` など) は、第 4 世代カスタム Nitro Card を活用して、最大 100 Gbps のネットワークスループットを 1 つのインスタンスに配信します。これらのインスタンスは 4

倍のネットワーク帯域幅とパケットプロセスを提供するため、ベース M5 インスタンスと比べて、ネットワーク集約型のアプリケーションに最適です。

c. Amazon Elastic Network Adapters (ENA) は、インスタンスに対してより良いスループットをクラスタープレイスメントグループ内で提供することにより、さらなる最適化をもたらします。

d. Elastic Fabric Adapter (EFA) は、高いレベルのインターノードコミュニケーションを AWS で大規模に要求するワークロードの実行を可能にする、Amazon EC2 インスタンスのネットワークインターフェイスです。EFA では、メッセージパッシングインターフェイス (MPI) を使用するハイパフォーマンスコンピューティング (HPC) アプリケーションと、NVIDIA Collective Communications Library (NCCL) を使用する機械学習 (ML) アプリケーションを、数千個に及ぶ CPU または GPU にスケールできます。

e. Amazon EBS 最適化 インスタンスは、最適化された設定スタックを使用し、Amazon EBS I/O を増加させるために専用の追加容量を提供します。この最適化により、Amazon EBS I/O とインスタンスからのその他のトラフィック間の競合を最小限に抑えることで、EBS ボリュームで最良のパフォーマンスを実現します。

実装計画に必要な工数レベル:

このベストプラクティスを確立するには、ネットワークパフォーマンスに影響を与える現在のワークロードコンポーネントのオプションを把握している必要があります。コンポーネントの収集、ネットワーク改善オプションの評価、実験、実装、およびこれらの改善の文書化には、 低 ~ 中 程度の工数が必要です。

リソース

関連ドキュメント:

- Amazon EBS – 最適化インスタンス
- Application Load Balancer
- Amazon EC2 インスタンスのネットワーク帯域幅
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS のネットワーク製品

- AWS Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ
- クラウド CMDB の構築
- AWS Transit Gateway を使用したVPN スループットのスケーリング

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- AWS Global Accelerator

関連する例:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP03 ハイブリッドワークロード用に適切なサイズの専用接続または VPN を選択する

AWS でオンプレミスとクラウドのリソースを接続するために一般的なネットワークが必要な場合は、パフォーマンス要件を満たす十分な帯域幅があることを確認します。ハイブリッドワークロードについては、帯域幅とレイテンシー要件を推定します。これらの数値によって、AWS Direct Connect、または VPN エンドポイントのサイジング要件が決まります。

期待される成果: ハイブリッドネットワーク接続を必要とするワークロードをデプロイする場合、マネージドおよび非マネージドの VPN や Direct Connect など、接続に関する複数の設定オプションがあります。各ワークロードに適切な接続タイプを選択し、ロケーションとクラウドの間に十分な帯域幅と暗号化要件への準拠を確保します。

一般的なアンチパターン:

- ネットワークの暗号化要件の VPN ソリューションのみを評価する。
- バックアップ接続または並列接続のオプションを評価しない。
- ルーター、トンネル、BGP セッションにデフォルトの設定を使用する。

- ワークロードのすべての要件 (暗号化、プロトコル、帯域幅、トラフィックのニーズ) を理解または特定できていない。

このベストプラクティスを活用するメリット: 適切にサイジングされたハイブリッドネットワークソリューションを選択し、設定することで、ワークロードの信頼性を高め、パフォーマンス改善の機会を最大限に増やすことができます。ワークロード要件を特定して前もって計画し、ハイブリッドソリューションを評価することで、市場投入までの時間を短縮しながら、コストの高いネットワークの物理的変更と運用上の諸経費を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

帯域幅要件に基づいてハイブリッドネットワーキングアーキテクチャを開発する: ハイブリッドアプリケーションの帯域幅とレイテンシー要件を見積もります。帯域幅の要件によっては、単一の VPN または Direct Connect 接続では十分でない場合があります。また、複数の接続間におけるトラフィック負荷分散を有効にするハイブリッドセットアップを設計する必要があります。プライベートネットワーク接続により、予測可能が高く一貫したパフォーマンスを提供する直接接続が必要になる場合があります。安定したレイテンシーとほぼジッターのない状態を必要とする本番ワークロードに最適です。

AWS Direct Connect は、50 Mbps から 10 Gbps までの AWS 環境への専用接続を実現します。これは、管理および制御されたレイテンシーとプロビジョニングされた帯域幅を提供することから、ワークロードが簡単かつ最も効率の良い方法で他の環境に接続できるようになります。AWS Direct Connect パートナーの 1 つを使用すると、複数の環境からエンドツーエンド接続が可能になるため、一貫したパフォーマンスでの拡張ネットワークが実現します。

AWS Site-to-Site VPN は、VPC のためのマネージド VPN サービスです。VPN 接続が確立されると、AWS は 2 つの異なる VPN エンドポイントへのトンネルを提供します。AWS Transit Gateway では、複数の VPC 間における接続をシンプル化でき、単一の VPN 接続を使って AWS Transit Gateway にアタッチされた任意の VPC に接続することもできます。AWS Transit Gateway は、複数の VPN トンネルで等コストマルチパス (ECMP) ルーティングのサポートを有効化することによって、1.25 Gbps IPsec の VPN スループット制限を超えたスケーリングを行うことも可能にします。

実装計画に必要な工数レベル: ハイブリッドネットワークのワークロードのニーズを評価し、ハイブリッドネットワークソリューションを実装するには、 高 程度の労力が必要です。

リソース

関連ドキュメント:

- Network Load Balancer
- AWS のネットワーク製品
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ
- Site-to-Site VPN
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Direct Connect
- Client VPN

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- AWS Global Accelerator
- Direct Connect
- Transit Gateway Connect
- VPN Solutions
- Security with VPN Solutions

関連サンプル:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP04 ロードバランシングと暗号化のオフロードを活用する

トラフィックを複数のリソースやサービスに分散させ、ワークロードがクラウドの伸縮性を活用できるようにします。また、パフォーマンスを向上させ、トラフィックを効率的に管理およびルーティングするために、ロードバランシングを使用して暗号化終了をオフロードすることもできます。

サービス内容に応じた複数のインスタンスを利用したいスケールアウトアーキテクチャを実装する場合、Amazon VPC 内でロードバランサーを使用できます。AWS は、ELB サービスでアプリケー

ション向けのモデルを複数提供しています。Application Load Balancer は、HTTP および HTTPS トラフィックのロードバランシングに最適で、マイクロサービスやコンテナなど、最新のアプリケーションアーキテクチャの実現を目標とした高度なリクエストルーティングを提供します。

Network Load Balancer は、きわめて高いパフォーマンスが要求される TCP トラフィックのロードバランシングに最適です。超低レイテンシーを維持しながら 1 秒あたり数百万件ものリクエストを処理することができ、突発的、または変動しやすいトラフィックパターンを処理するために最適化されています。

Elastic Load Balancing は、統合された証明書管理と SSL/TLS 復号化を提供するため、ロードバランサーの SSL 設定を一元的に管理し、CPU に負荷のかかる SSL/TLS 処理をお客様のワークロードからオフロードすることができます。

一般的なアンチパターン:

- 既存のロードバランサーを介して、すべてのインターネットトラフィックをルーティングしている。
- 汎用 TCP ロードバランシングを使用して、各コンピューティングノードが SSL 暗号化を処理するようにしている。

このベストプラクティスを確立するメリット: ロードバランサーは、単一のアベイラビリティーゾーンまたは複数のアベイラビリティーゾーンにおけるアプリケーショントラフィックのさまざまな負荷を処理します。ロードバランサーは、アプリケーションの耐障害性を高めるために必要な高可用性、自動スケーリング、堅牢なセキュリティを備えています。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードに適したロードバランサーを使用する: ワークロードに適したロードバランサーを選択します。HTTP リクエストをロードバランシングする必要がある場合は、Application Load Balancer をお勧めします。ネットワークおよびトランスポートプロトコル (layer4 – TCP、UDP) のロードバランシング、および極端なパフォーマンスと低レイテンシーのアプリケーションには Network Load Balancer をお勧めします。Application Load Balancers は HTTPS をサポートし、Network Load Balancer は TLS 暗号化オフロードをサポートします。

HTTPS または TLS 暗号化のオフロードを有効にする: Elastic Load Balancing には、証明書管理、ユーザー認証、SSL/TLS 復号が統合されています。TLS 設定を一元的に管理し、CPU 負荷の高いワークロードをアプリケーションからオフロードする柔軟性を提供します。ロードバランサーのデプロイの一部としてすべての HTTPS トラフィックを暗号化します。

リソース

関連ドキュメント:

- Amazon EBS 最適化インスタンスを使用する
- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- Networking Products with AWS (AWS のネットワーク製品)
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1) (AWS およびハイブリッド AWS ネットワークアーキテクチャへの接続性)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)

関連サンプル:

- AWS Transit Gateway and Scalable Security Solutions (AWS トランジットゲートウェイとスケーラブルセキュリティソリューション)
- AWS Networking Workshops (AWS ネットワーキングワークショップ)

PERF05-BP05 パフォーマンスを高めるネットワークプロトコルを選択する

ワークロードのパフォーマンスに対する影響に基づいて、システムとネットワーク間における通信のためのプロトコルを決定します。

スループットの達成には、レイテンシーと帯域幅間の関係が関与します。ファイル転送で TCP を使用している場合は、高レイテンシーが全体的なスループットを低下させます。これを修正するアプローチには、TCP チューニングと最適化された転送プロトコルを使うものがあり、UDP を使用するものもあります。

一般的なアンチパターン:

- パフォーマンス要件に関係なく、すべてのワークロードに TCP を使用する。

このベストプラクティスを活用するメリット: ワークロードコンポーネント間の通信に適切なプロトコルを選択すると、そのワークロードに対して最高のパフォーマンスを得ることができます。コネクションレス型 UDP は高速な対応を可能にしますが、再送信や高い信頼性は提供しません。TCP はフル機能のプロトコルですが、パケットの処理にはより大きなオーバーヘッドが必要です。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ネットワークトラフィックを最適化する; ワークロードのパフォーマンスを最適化するために適切なプロトコルを選択します。スループットの達成には、レイテンシーと帯域幅間の関係が関与します。ファイル転送で TCP を使用している場合は、高レイテンシーが全体的なスループットを低下させます。レイテンシーを修正するアプローチには、TCP チューニングと最適化された転送プロトコルを使うものがあるほか、UDP を使用するものもあります。

リソース

関連ドキュメント:

- Amazon EBS – 最適化インスタンス
- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS のネットワーク製品
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する

- VPC エンドポイント
- VPC フローログ

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)

関連サンプル:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP06 ネットワーク要件に基づいてワークロードのロケーションを選択する

利用可能なクラウドロケーションオプションを使用して、ネットワークレイテンシーを低減、またはスループットを向上させます。ネットワークレイテンシーの低減、またはスループットの向上には、AWS リージョン、アベイラビリティーゾーン、プレイスメントグループ、および AWS Outposts、AWS Local Zones、AWS Wavelength などのエッジロケーションを活用します。

AWS クラウドインフラストラクチャはリージョンとアベイラビリティーゾーンを中心として構築されます。AWS リージョンは世界中の物理的な場所であり、複数のアベイラビリティーゾーンで構成されています。

アベイラビリティーゾーンは 1 つ以上の独立したデータセンターで構成されます。各データセンターは、冗長性のある電源、ネットワーク、接続を備えており、別々の設備に収容されています。これらのアベイラビリティーゾーンは、単一のデータセンターで実現できるものよりも、可用性、耐障害性、および拡張性に優れた本番用のアプリケーションとデータベースを運用する能力を提供します。

以下の主な要素に基づいて、デプロイメントに適切なリージョン (単一または複数) を選択してください。

- ユーザーの所在地: お客様のワークロードを利用するユーザーに近いリージョンを選択することによって、ワークロードの使用時における低レイテンシーを確保します。
- データの場所: 大量のデータを使用するアプリケーションでは、データ転送がレイテンシーにおける主なボトルネックとなります。アプリケーションコードはできる限りデータに近い場所で実行してください。

- その他の制約: セキュリティおよびコンプライアンスなどの制約を考慮します。

Amazon EC2 にはネットワーク用のプレイスメントグループが用意されています。プレイスメントグループは、レイテンシーを減らしたり信頼性を高めたりするためのインスタンスの論理グループです。サポートされているインスタンスタイプを使ったプレイスメントグループと Elastic Network Adapter (ENA) を使用することにより、ワークロードを低レイテンシーの 25 Gbps ネットワークに参加させることができます。プレイスメントグループは、低ネットワークレイテンシー、高ネットワークスループット、またはその両方からメリットを得るワークロードに推奨されます。プレイスメントグループを使用すると、ネットワーク通信でジッターを低減できるという利点があります。

レイテンシーの影響を受けやすいサービスは、エッジロケーションのグローバルネットワークを使用し、エッジで提供されます。これらのエッジロケーションは一般に、コンテンツ配信ネットワーク (CDN) およびドメインネームシステム (DNS) などのサービスを提供します。これらのサービスをエッジで使用することにより、ワークロードがコンテンツまたは DNS 解決のリクエストに低いレイテンシーで応答できるようになります。これらのサービスは、コンテンツのジオターゲティング (エンドユーザーの位置に基づいて異なるコンテンツを提供) などの地理的なサービス、またはエンドユーザーを最寄りのリージョンに誘導するレイテンシーベースルーティング (最小レイテンシー) も提供します。

Amazon CloudFront はグローバル CDN で、画像やスクリプト、動画などの静的コンテンツと、API やウェブアプリケーションなどの動的コンテンツの両方を加速させることができます。これは、コンテンツをキャッシュし、ユーザーに高性能のネットワーク接続を提供するエッジロケーションのグローバルネットワークに依存しています。CloudFront は、コンテンツのアップロードや動的アプリケーションといった、他の多くの機能も加速させることができ、インターネット経由のトラフィックを処理するすべてのアプリケーションのパフォーマンスがさらに向上します。 Lambda@Edge は、ワークロードのユーザーに近い場所でコードを実行できるようにする Amazon CloudFront の機能で、パフォーマンスを向上させ、レイテンシーを低減します。

Amazon Route 53 は、可用性とスケーラビリティの高いクラウド DNS ウェブサービスです。www.example.com などの名前をコンピュータが互いに接続するための数字の IP アドレス (192.168.2.1 など) に変換することにより、開発者や会社のエンドユーザーをインターネットアプリケーションに転送させる、きわめて信頼性が高く、経済的な方法を提供します。Route 53 は IPv6 に完全対応しています。

AWS Outposts は、レイテンシー要件のためにオンプレミスの維持が必要となるワークロードで、AWS にあるその他のワークロードとシームレスに連携できるように設計されています。AWS Outposts は、AWS 設計のハードウェアで構築された完全マネージド型の設定可能なコンピューティ

ング/ストレージラックで、クラウド内にある AWS の多岐にわたるサービスにシームレスに接続しながら、オンプレミスでのコンピューティングとストレージの実行を可能にします。

AWS Local Zones は動画レンダリングやグラフィック集約型の仮想デスクトップアプリケーションなど、10 ミリ秒未満のレイテンシーを必要とするワークロードを実行するために設計されています。Local Zones では、エンドユーザーの近くにコンピューティングおよびストレージリソースを保有するメリットのすべてを得ることが可能になります。

AWS Wavelength は、AWS のインフラストラクチャ、サービス、API、ツールを 5G ネットワークに拡張することによって、5G デバイスに超低レイテンシーのアプリケーションを提供するために設計されたものです。Wavelength は、IoT デバイス、ゲームストリーミング、自律走行車、ライブメディア製作などの 10 ミリ秒未満のレイテンシーが必要な場合に 5G ワークロードを支援できるように、電気通信プロバイダーの 5G ネットワークにストレージとコンピューティングを埋め込みます。

エッジサービスを使用してレイテンシーを低減し、コンテンツキャッシングを有効化します。これらのアプローチから最大限のメリットを得るためにも、DNS と HTTP/HTTPS の両方にキャッシュ制御が正しく設定されていることを確認してください。

一般的なアンチパターン:

- すべてのワークロードリソースを 1 つの地理的場所に統合する。
- ワークロードのエンドユーザーではなく、自分の所在地に最も近いリージョンを選んでいる。

このベストプラクティスを活用するメリット: 顧客へのアクセスを希望するロケーションで、ネットワークが利用可能であることを確認する必要があります。AWS のプライベートグローバルネットワークを使用して、ワークロードを最も近いロケーションにデプロイすることで、最も低いレイテンシーのエクスペリエンスを顧客に提供できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

正しいロケーションを選択してレイテンシーを低減する: ユーザーがいる場所やデータがある場所を特定します。AWS リージョン、アベイラビリティーゾーン、プレイスメントグループ、エッジロケーションを利用してレイテンシーを低減します。

リソース

関連ドキュメント:

- Amazon EBS – 最適化インスタンス
- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS のネットワーク製品
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)

関連サンプル:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP07 メトリクスに基づいてネットワーク設定を最適化する

収集して分析したデータを使用して、ネットワーク設定の最適化に関する十分な知識に基づいた意思決定を行います。これらの変更の影響を測定して、その影響測定値を将来の意思決定に使用します。

ワークロードが使用するすべての VPC ネットワークに対して VPC フローログを有効化します。VPC フローログは、VPC 内のネットワークインターフェイスに出入りする IP トラフィックに関する情報を取得できるようにする機能です。VPC フローログは、特定のトラフィックがインスタンスに到達しない理由のトラブルシューティングといった多数のタスクの実行をサポートし、これは過剰に制限されたセキュリティグループルールの診断に役立ちます。フローログをセキュリティツー

ルとして使用して、インスタンスに到達するトラフィックのモニタリング、ネットワークトラフィックのプロファイリング、および異常なトラフィック動作の検出を行うことができます。

ネットワーキングメトリクスを使用して、ワークロードの進化に合わせてネットワーキング設定を変更します。クラウドベースのネットワークは迅速に再構築できるため、パフォーマンス効率を維持するためにもネットワークアーキテクチャを時間とともに進化させる必要があります。

一般的なアンチパターン:

- パフォーマンス関連の問題はすべてアプリケーション関連であると想定している。
- ワークロードをデプロイしたロケーションに近いロケーションからのみ、ネットワークパフォーマンスをテストする。

このベストプラクティスを活用するメリット:ワークロードに必要なメトリクスを確実に満たすには、ネットワークパフォーマンスメトリクスをモニタリングする必要があります。VPC のネットワークインターフェイスに出入りする IP トラフィックに関する情報をキャプチャし、このデータを使用して新しい最適化を追加したり、新しい地理的リージョンにワークロードをデプロイしたりできます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

VPC フローログを有効にする: VPC フローログにより、VPC のネットワークインターフェイスとの間で行き来する IP トラフィックに関する情報をキャプチャできます。VPC フローログは、特定のトラフィックがインスタンスに到達しない理由のトラブルシューティングといった多数のタスクの実行をサポートし、これは過剰に制限されたセキュリティグループルールの診断に役立ちます。フローログをセキュリティツールとして使用して、インスタンスに到達するトラフィックのモニタリング、ネットワークトラフィックのプロファイリング、および異常なトラフィック動作の検出を行うことができます。

ネットワークオプションに適切なメトリクスを有効化する: ワークロードに適したネットワークメトリクスを選択していることを確認します。VPC NAT ゲートウェイ、トランジットゲートウェイ、VPN トンネルのメトリクスを有効にできます。

リソース

関連ドキュメント:

- Amazon EBS – 最適化インスタンス

- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS のネットワーク製品
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ
- Monitoring your global and core networks with Amazon Cloudwatch metrics
- Continuously monitor network traffic and resources

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- Monitoring and troubleshooting network traffic
- Simplify Traffic Monitoring and Visibility with Amazon VPC Traffic Mirroring

関連サンプル:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops
- AWS Network Monitoring

レビュー

質問

- PERF 6 ワークロードを進化させるために、どのように新機能を取り込めばよいですか？

PERF 6 ワークロードを進化させるために、どのように新機能を取り込めばよいですか？

ワークロードを設計する際に選択できるオプションには限りがありますが、時間と共に、ワークロードのパフォーマンスを向上させることができる新しいテクノロジーとアプローチが利用できるようになります。

ベストプラクティス

- PERF06-BP01 新しいリソースとサービスに関する最新情報を常に入手する
- PERF06-BP02 ワークロードのパフォーマンス向上プロセスを定める
- PERF06-BP03 ワークロードのパフォーマンスを時間とともに進化させる

PERF06-BP01 新しいリソースとサービスに関する最新情報を常に入手する

新しいサービス、設計パターン、製品が利用可能になったら、パフォーマンスの向上方法を検討します。評価、社内でのディスカッション、または外部分析を通じて、これらのリリースとサービスのどれがワークロードのパフォーマンスまたは効率性を向上させるかを判断します。

ワークロードに関連するアップデート、新しい機能、サービスを評価するプロセスを定義します。例えば、新テクノロジーを使用する PoC (概念実証) の構築や内部グループとの協議などのプロセスが考えられます。新しいアイデアやサービスを試す場合、パフォーマンステストを実施して、ワークロードのパフォーマンスへの影響を測定します。Infrastructure as Code (IaC) と DevOps の利点を活用して、新しいアイデアやテクノロジーを頻繁に、最低限のコストやリスクでテストします。

期待される成果: コンポーネントのインベントリ、設計パターン、ワークロードの特徴を文書化しました。この文書を使用して、サブスクリプションの一覧を作成し、サービスのアップデート、機能、新製品の情報をチームに通知します。新しいリリースを評価しビジネスへの影響と優先順位に関するレコメンデーションを提供する、コンポーネントの関係者を特定しました。

一般的なアンチパターン:

- 新しいオプションやサービスを検討するのは、ワークロードがパフォーマンス要件を満たせなくなったときだけである。
- すべての新製品がワークロードにとって有益であるわけではないと考えている。
- ワークロードを改善する際は、製品を選ぶことはせず、常に内部で構築している。

このベストプラクティスを活用するメリット: 新しいサービスまたは製品を検討することで、ワークロードのパフォーマンスと効率の改善、インフラストラクチャコストの低減、およびサービスのメンテナンスに必要な工数の削減を行うことでできます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

AWS のアップデート、新しい機能、サービスを評価するプロセスを定めます。たとえば、新しいテクノロジーを使用した概念実証を構築します。新しいアイデアまたはサービスを試す際は、パフォーマンステストを実施して、ワークロードの効率性、またはパフォーマンスに対する影響を測定します。AWS が持つ柔軟性を活用し、最小限のコストとリスクで新しいアイデアやテクノロジーを頻繁にテストします。

実装手順

1. ワークロードソリューションを文書化します。設定管理データベース (CMDB) ソリューションを使用して、インベントリを文書化し、サービスと依存関係を分類します。AWS Config などのツールを使用して、 ワークロードで使用されている AWS のすべてのサービスを一覧化します。
2. タグ付け戦略を 使用して、 各ワークロードコンポーネントとカテゴリの所有者を文書化します。例えば、現在データベースソリューションとして Amazon RDS を使用している場合、データベース管理者 (DBA) を割り当てて、新しいサービスとアップデートの評価と調査の所有者として文書化します。
3. ワークロードコンポーネントに関連するニュースとアップデートソースを特定します。前述の Amazon RDS の例では、カテゴリの所有者は、 AWS の最新情報ブログを購読して、 ワークロードコンポーネントに適した製品の情報を入手する必要があります。RSS フィードを購読したり、E メールサブスクリプションを 管理したりして、購読できます。使用している Amazon RDS データベースのアップグレード、新しく導入される機能、リリースされるインスタンス、Amazon Aurora Serverless などの新製品をモニタリングします。業界ブログ、製品、コンポーネントが依存しているベンダーをモニタリングします。
4. アップデートと新しいサービスの評価プロセスを文書化します。アップデートと新しいサービスを調査、テスト、実験、検証する時間と場所をカテゴリの所有者に与えます。文書化したビジネスの要件と KPI を参照して、どのアップデートがビジネスにメリットをもたらすかの優先順位を付けます。

実装計画に必要な工数レベル: このベストプラクティスを採用するには、現在のワークロードコンポーネントを把握し、カテゴリ所有者およびサービスアップデートのソースを特定する必要がありま

す。実装開始時の工数レベルは低いものの、このベストプラクティスは長期間にわたって発展し改善しうる継続的なプロセスです。

リソース

関連するドキュメント:

- AWS ブログ
- AWS の最新情報

関連動画:

- AWS Events YouTube チャンネル
- AWS Online Tech Talks YouTube チャンネル
- Amazon Web Services YouTube チャンネル

関連サンプル:

- AWS Github
- AWS Skill Builder

PERF06-BP02 ワークロードのパフォーマンス向上プロセスを定める

新しいサービス、設計パターン、リソースの種類、設定が利用できるようになった時点で、これらを評価するプロセスを明確に定めます。たとえば、新しいインスタンス製品で既存のパフォーマンステストを実行して、ワークロードを向上させる可能性を判断します。

ワークロードのパフォーマンスには重要な制約がいくつかあります。その制約を文書化すれば、どのような種類のイノベーションがワークロードのパフォーマンス向上につながるかを把握できます。新しいサービスやテクノロジーが利用できるようになった場合、この情報を利用して、制約やボトルネックを軽減する方法を見つけます。

一般的なアンチパターン:

- 現在のアーキテクチャが静的なものとなり、時間が経過しても更新されることはないと想定している。
- メトリクスに基づく理由なしで、時間の経過とともにアーキテクチャの変更を導入する。

このベストプラクティスを活用するメリット: アーキテクチャの変更を行うためのプロセスを定義することで、収集されたデータが、時間の経過に伴い、ワークロード設計に影響を及ぼせるようにできます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ワークロードの主要なパフォーマンス上の制約を特定する: どのようなイノベーションがワークロードパフォーマンスの向上につながるかを知ることができるように、ワークロードのパフォーマンスの制約を文書化します。

リソース

関連ドキュメント:

- AWS ブログ
- AWS の最新情報

関連動画:

- AWS Events YouTube チャンネル
- AWS Online Tech Talks YouTube チャンネル
- Amazon Web Services YouTube チャンネル

関連サンプル:

- AWS Github
- AWS Skill Builder

PERF06-BP03 ワークロードのパフォーマンスを時間とともに進化させる

組織として、評価プロセスを通じて収集した情報を使用し、新しいサービスまたはリソースが利用可能になるときに、それらの導入を積極的に推進します。

新しいサービスやテクノロジーを評価する際に収集する情報を使用して、変化を促進します。ビジネスまたはワークロードの変化に伴い、パフォーマンスのニーズも変化します。ワークロードメトリクスから収集されたデータを利用して、効率性やパフォーマンスが最も改善する領域を見極め、新しいサービスとテクノロジーを積極的に採用して需要に対応します。

一般的なアンチパターン:

- 現在のアーキテクチャが静的なものとなり、時間が経過しても更新されることはないと想定している。
- メトリクスに基づく理由なしで、時間の経過とともにアーキテクチャの変更を導入する。
- 業界の他のすべての会社が使用しているというだけの理由で、アーキテクチャを変更する。

このベストプラクティスを活用するメリット: ワークロードのパフォーマンスとコストを最適化するには、利用可能なすべてのソフトウェアとサービスを評価して、ワークロードに適したソフトウェアとサービスを判断する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ワークロードを時間とともに進化させる: 新しいサービスやテクノロジーを評価する際に収集する情報を使用して、変化を促進します。ビジネスまたはワークロードの変化に伴い、パフォーマンスのニーズも変化します。ワークロードメトリクスから収集したデータを使用して、効率性またはパフォーマンスで最大の効果を得られる領域を見極め、新しいサービスとテクノロジーを積極的に取り入れて需要に対応します。

リソース

関連ドキュメント:

- AWS ブログ
- AWS の最新情報

関連動画:

- AWS Events YouTube チャンネル
- AWS Online Tech Talks YouTube チャンネル
- Amazon Web Services YouTube チャンネル

関連サンプル:

- AWS Github
- AWS Skill Builder

モニタリング

質問

- PERF 7 リソースが稼働していることを確認するには、リソースをどのようにモニタリングすればよいですか?

PERF 7 リソースが稼働していることを確認するには、リソースをどのようにモニタリングすればよいですか?

システムのパフォーマンスは徐々に低下することがあります。劣化を特定し、オペレーティングシステムまたはアプリケーション負荷などの内部および外部の要因を修正するために、システムのパフォーマンスをモニタリングします。

ベストプラクティス

- PERF07-BP01 パフォーマンスに関連するメトリクスを記録する
- PERF07-BP02 イベントやインシデントが発生したときにメトリクスを分析する
- PERF07-BP03 ワークロードのパフォーマンスを測定するための重要業績評価指標 (KPI) を設定する
- PERF07-BP04 モニタリングを使用してアラームベースの通知を生成する
- PERF07-BP05 メトリクスを定期的に見直す
- PERF07-BP06 モニタリングしてプロアクティブに警告する

PERF07-BP01 パフォーマンスに関連するメトリクスを記録する

モニタリングとオブザーバビリティサービスを使用して、パフォーマンス関連のメトリクスを記録します。メトリクスの例としては、データベーストランザクションの記録、低速クエリ、I/O レイテンシー、HTTP リクエストスループット、 サービスレイテンシー、その他の重要なデータなどがあります。

ワークロードにとって重要なパフォーマンスメトリクスを特定して記録します。このデータは、ワークロードの全体的なパフォーマンスや効率性に影響するコンポーネントを特定するための重要な要素です。

再度カスタマーエクスペリエンスを元に、重要なメトリクスを特定します。メトリクスごとに、ターゲット、測定アプローチ、および優先順位を特定します。これらを使用してアラームと通知を構築し、パフォーマンス関連の問題に積極的に対応します。

一般的なアンチパターン:

- ワークロードに関する洞察を得るために、オペレーティングシステムレベルのメトリクスのモニタリングのみを行う。
- ピーク時のワークロード要件に合わせてコンピューティングニーズを設計する。

このベストプラクティスを確立するメリット: パフォーマンスとリソース使用率を最適化するには、主要業績評価指標の統合された運用ビューが必要です。ダッシュボードを作成して、データに関するメトリクスの計算を実行して運用と利用に関する洞察を得ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードに関連するパフォーマンスメトリクスを特定し、それらを記録します。このデータは、どのコンポーネントがワークロードの全体的なパフォーマンスまたは効率性に影響しているかを特定するのに役立ちます。

パフォーマンスメトリクスを特定する: カスタマーエクスペリエンスを使用して、最も重要なメトリクスを特定します。メトリクスごとに、ターゲット、測定アプローチ、および優先順位を特定します。これらのデータポイントを使用してアラームと通知を構築し、パフォーマンス関連の問題に積極的に対応します。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する
- カスタムメトリクスの発行
- モニタリング、ログ記録、パフォーマンス APN パートナー
- X-Ray ドキュメント
- Amazon CloudWatch RUM

関連動画:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)

- Application Performance Management on AWS (AWS でのアプリケーションパフォーマンス管理)
- Build a Monitoring Plan

関連サンプル:

- Level 100: Monitoring with CloudWatch Dashboards (レベル 100: CloudWatch ダッシュボードを使ったモニタリング)
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards (レベル 100: ダッシュボードを使った Windows EC2 インスタンスのモニタリング)
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards (レベル 100: CloudWatch ダッシュボードを使った Amazon Linux EC2 インスタンスのモニタリング)

PERF07-BP02 イベントやインシデントが発生したときにメトリクスを分析する

イベントやインシデントが発生した後 (または発生中) に、モニタリングダッシュボードやレポートを使用してその影響を把握して診断します。これらのビューは、ワークロードのどの部分が期待通りに機能していないかに関するインサイトを提供します。

アーキテクチャに重要なユーザーストーリーを記述するときは、パフォーマンス要件を含めるようにし、それぞれの重要なストーリーをどの程度迅速に実行する必要があるかといった点を明記します。これらの重要なストーリーには、要件に対してこれらのストーリーがどのように実行されるかを知ることができるように、スクリプト化されたユーザージャーニーを追加で実装します。

一般的なアンチパターン:

- パフォーマンスイベントが一度限りのもので、異常にのみ関連するものであると考えている。
- パフォーマンスイベントに対応する場合にのみ、既存のパフォーマンスメトリクスを評価する。

このベストプラクティスを活用するメリット: ワークロードが想定レベルで動作しているかどうかを判断するには、分析のために追加のメトリクスデータを収集してパフォーマンスイベントに対応する必要があります。このデータは、パフォーマンスイベントの影響を理解し、ワークロードのパフォーマンスを改善するための変更を提案するために使用されます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

アーキテクチャに重要なユーザーストーリーを記述するときは、パフォーマンス要件を含めるようにし、それぞれの重要なストーリーをどの程度迅速に実行する必要があるかといった点を明記します。これらの重要なストーリーには、要件に対してユーザーストーリーがどのように実行されるかを知ることができるように、スクリプト化されたユーザージャーニーを追加で実装してください。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- Amazon CloudWatch Synthetics
- モニタリング、ログ記録、パフォーマンス APN パートナー
- X-Ray ドキュメント

関連動画:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Optimize applications through Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics のデモ

関連サンプル:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF07-BP03 ワークロードのパフォーマンスを測定するための重要業績評価指標 (KPI) を設定する

ワークロードのパフォーマンスを定量的および定性的に測定する KPI を特定します。KPI は、ビジネス目標に関連するワークロードの状態を測定するのに役立ちます。KPI を使用すると、ビジネスチームとエンジニアリングチームが、目標と戦略の測定と、それらがどのように組み合わさってビジネス成果を生み出すかについての認識をすり合わせることができます。ビジネスの目標および戦略、またはエンドユーザーの要件が変わった場合は、KPI を再検討する必要があります。

例えば、ウェブサイトのワークロードには、ページの読み込み時間を全体的なパフォーマンスの指標として使用する場合があります。このメトリクスは、エンドユーザーエクスペリエンスを測定する複

数のデータポイントのうちの 1 つとなります。ページの読み込み時間のしきい値を特定することに加えて、パフォーマンスが満たされない場合に期待される結果やビジネスリスクを文書化する必要があります。ページの読み込み時間が長いと、エンドユーザーに直接影響し、ユーザーエクスペリエンスの評価の低下、ひいては顧客の損失につながる可能性があります。KPI のしきい値を定義するときは、業界のベンチマークとエンドユーザーの期待値の両方を組み合わせます。例えば、現時点での業界のウェブページの読み込みベンチマークが 2 秒以内であっても、エンドユーザーが 1 秒以内での読み込みを期待する場合、KPI を設定する際にこれらのデータポイントの両方を考慮する必要があります。KPI の別の例として、内部パフォーマンスのニーズを満たすことに焦点を当てることもできます。本番データが生成されてから 1 営業日以内に売上レポートが作成されると、KPI のしきい値が設定される場合があります。これらのレポートは、日々の意思決定やビジネス成果に直接影響する可能性があります。

期待される成果: KPI の設定には、さまざまな部門やステークホルダーが関与します。チームは、リアルタイムの詳細なデータと参照用の履歴データを使用してワークロード KPI を評価し、KPI データにメトリクス計算を実行するダッシュボードを作成して、運用と使用状況に関する洞察を導き出す必要があります。KPI は、ビジネス目標や戦略をサポートするために合意された KPI としきい値を説明し、モニタリング対象のメトリクスに対応付け、文書化する必要があります。KPI は、パフォーマンス要件を特定し、意図的に見直され、すべてのチームと頻繁に共有され、理解されています。リスクとトレードオフが明確に特定され、KPI のしきい値が満たされない場合にビジネスにどのような影響があるかが理解されています。

一般的なアンチパターン:

- ワークロードについての洞察を得るためだけにシステムレベルのメトリクスをモニタリングし、これらのメトリクスがビジネスに与える影響を理解していない。
- KPI が標準的なメトリクスデータとして既に発行され、共有されていると思っている。
- KPI を定義したのにすべてのチームと共有していない。
- 定量的で測定可能な KPI を定めていない。
- KPI をビジネスの目標や戦略とすり合わせていない。

このベストプラクティスを確立するメリット: ワークロードの状態を表す具体的なメトリクスを特定することは、チームの優先事項を調整し、成功に向けたビジネス成果を定義するのに役立ちます。これらのメトリクスをすべての部門と共有することで、しきい値、期待値、ビジネスへの影響が可視化され、調整を図ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

KPI の設定には、ワークロードの状態によって影響を受けるすべての部門とビジネスチームが貢献する必要があります。組織の KPI に関するコラボレーション、タイムライン、ドキュメント、情報伝達の推進は 1 人の人物が担当するべきです。多くの場合、この人物が所有者となり、ビジネスの目標と戦略を共有し、ビジネスのステークホルダーにそれぞれの部門の KPI を策定するタスクを割り当てます。KPI が定義されたら、多くの場合、運用チームがさまざまな KPI を支援し、成功を示すメトリクスの定義をサポートします。KPI は、ワークロードを支えるチームの全員が KPI を認識している場合にのみ効果があります。

実装手順

1. ビジネスのステークホルダーを特定して文書化します。
2. 会社の目標と戦略を特定します。
3. 会社の目標と戦略と一致する一般的な業界 KPI を確認します。
4. ワークロードに対するエンドユーザーの期待を確認します。
5. 会社の目標と戦略をサポートする KPI を定義し、文書化します。
6. KPI を満たすための承認済みのトレードオフ戦略を特定し、文書化します。
7. KPI を表すメトリクスを特定し、文書化します。
8. 重大度またはアラームレベルの KPI しきい値を特定して文書化します。
9. KPI が満たされない場合のリスクと影響を特定して文書化します。
10. KPI ごとにレビューの頻度を特定します。
11. ワークロードをサポートするすべてのチームに KPI 関連の文書について連絡します。

実装ガイドに必要な工数レベル: KPI の定義と伝達にかかる労力は 低 レベルです。これは通常、ビジネス関係者と数週間にわたってミーティングを行い、目標、戦略、ワークロードのメトリクスを確認することで達成できます。

リソース

関連ドキュメント:

- CloudWatch documentation (CloudWatch ドキュメント)
- Monitoring, Logging, and Performance APN Partners (モニタリング、ログ記録、パフォーマンス APN パートナー)

- X-Ray Documentation (X-Ray ドキュメント)
- Amazon CloudWatch ダッシュボードの使用
- Amazon QuickSight KPI

関連動画:

- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R) (最初の 1,000 万ユーザーまでのスケールアップ)
- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- モニタリング計画を立てる

関連サンプル:

- Creating a dashboard with Amazon QuickSight (Amazon QuickSight でダッシュボードを作成する)

PERF07-BP04 モニタリングを使用してアラームベースの通知を生成する

モニタリングシステムを使用し、定義したパフォーマンス関連の主要業績評価指標 (KPI) の測定値が予想された境界線の外側にある場合に自動的にアラームを生成します。

Amazon CloudWatch では、アーキテクチャ内のリソース全体のメトリクスを収集できます。また、カスタムメトリクスを収集および発行して、ビジネスメトリクスまたは導出メトリクスを表面化することも可能です。CloudWatch またはサードパーティーのモニタリングサービスを使用して、しきい値を超えたことを示すアラームを設定します。このアラームは、メトリクスが期待される限度を超えたことを知らせます。

一般的なアンチパターン:

- メトリクスのモニタリングと、問題が発生したときの対応をスタッフに頼っている。
- 同じタスクを達成するためにサーバーレスワークフローがトリガーできるにもかかわらず、運用ランブックのみに頼っている。

このベストプラクティスを活用するメリット: 事前定義されたしきい値、またはメトリクスの異常な動作を識別する機械学習アルゴリズムに基づいて、アラームを設定してアクションを自動化できます。これらの同じアラームは、サーバーレスワークフローをトリガーし、ワークロードのパフォーマ

ンス特性を変更することもできます (例えば、コンピューティング性能の増加、データベース設定の変更など)。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

メトリクスをモニタリングする: Amazon CloudWatch では、アーキテクチャ内のリソース全体のメトリクスを収集できます。カスタムメトリクスを収集および発行して、ビジネスメトリクスまたは導出メトリクスを表面化することができます。CloudWatch またはサードパーティーのモニタリングサービスを使用して、しきい値を超過したことを示すアラームを設定します。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- モニタリング、ログ記録、パフォーマンス APN パートナー
- X-Ray ドキュメント
- Using Alarms and Alarm Actions in CloudWatch

関連動画:

- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)
- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Build a Monitoring Plan
- Using AWS Lambda with Amazon CloudWatch Events

関連サンプル:

- Cloudwatch Logs Customize Alarms

PERF07-BP05 メトリクスを定期的に見直す

定期的なメンテナンスとして、またはイベントやインシデントに応じて、収集対象のメトリクスを見直します。これらのレビューを使用して、どのメトリクスが問題対応の鍵となったか、またどのメトリクスを追加すると、それらが追跡される場合に問題の特定、対応、または防止に役立つと思われるかを特定します。

インシデントやイベントへの対応の一環として、問題解決に役立ったメトリクスと、問題解決に役立った可能性があるものの、現在は追跡されていないメトリクスを評価します。これを使用して、収集するメトリクスの品質を高め、今後のインシデントを防止、またはより迅速に解決できるようにします。

一般的なアンチパターン:

- メトリクスを長期間アラーム状態のままにする。
- 自動システムによって実行できないアラームを作成する。

このベストプラクティスを活用するメリット: 収集されているメトリクスを継続的に見直し、問題を適切に識別、対応、または防止します。また、メトリクスは、長期間アラーム状態のままとなった場合にも、陳腐化することがあります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

メトリクスの収集とモニタリングを継続的に改善する: インシデントまたはイベント対応の一環として、問題への対応に役立ったメトリクス、および現在は追跡されていないが役に立ったであろうと思われるメトリクスを評価します。この方法を使用して収集するメトリクスの品質を高め、今後のインシデントを防止、またはより迅速に解決できるようにします。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する
- モニタリング、ログ記録、パフォーマンス APN パートナー
- X-Ray ドキュメント

関連動画:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- モニタリング計画を立てる

関連サンプル:

- Creating a dashboard with Amazon QuickSight
- Level 100: Monitoring with CloudWatch Dashboards

PERF07-BP06 モニタリングしてプロアクティブに警告する

主要業績評価指標 (KPI) をモニタリングおよびアラート発行システムと組み合わせて使用し、パフォーマンス関連の問題に積極的に対処します。アラームを使用して、可能な場合に問題を修正する自動化されたアクションをトリガーします。自動化された対応が不可能な場合は、対応できるシステムにアラームをエスカレートします。たとえば、期待される主要業績評価指標 (KPI) 値を予測し、それらが特定のしきい値を超えた場合にアラームを発行できるシステム、または KPI が期待される値の範囲外である場合に、デプロイメントを自動的に停止、またはロールバックできるツールなどが考えられます。

実行中のワークロードのパフォーマンスを目で見て確認できるようにするプロセスを実装します。モニタリングダッシュボードを構築し、パフォーマンス期待のベースラインとなる基準を確立して、ワークロードが最適に機能しているかどうかを判断します。

一般的なアンチパターン:

- 運用スタッフのみに対して、ワークロードに運用上の変更を加えることを許可する。
- プロアクティブな修復を行うことなく、すべてのアラームが運用チームに届くようにしている。

このベストプラクティスを活用するメリット: アラームアクションをプロアクティブに修正することで、サポートスタッフは自動的に実行できない項目に集中できます。これにより、運用スタッフがすべてのアラームの対応に忙殺されることがなくなり、代わりに重要なアラームのみに集中できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

運用中にパフォーマンスをモニタリングする: 実行中のワークロードのパフォーマンスを目で見て確認できるプロセスを実装します。モニタリングダッシュボードを構築し、期待されるパフォーマンスのベースラインを確立します。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- モニタリング、ログ記録、パフォーマンス APN パートナー
- X-Ray ドキュメント
- Using Alarms and Alarm Actions in CloudWatch

関連動画:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- モニタリング計画を立てる
- Using AWS Lambda with Amazon CloudWatch Events

関連サンプル:

- Cloudwatch Logs Customize Alarms

トレードオフ

質問

- PERF 8 パフォーマンスを向上させるために、トレードオフをどのように利用すればよいですか?

PERF 8 パフォーマンスを向上させるために、トレードオフをどのように利用すればよいですか?

アーキテクチャの設計にあたって、最適なアプローチとなるトレードオフを特定します。多くの場合、整合性、耐久性、および時間とレイテンシーの余裕と引き換えに、パフォーマンスを向上させることができます。

ベストプラクティス

- PERF08-BP01 パフォーマンスが最も重要な分野を理解する
- PERF08-BP02 設計パターンとサービスについて理解する
- PERF08-BP03 トレードオフが顧客と効率性にどのように影響するかを明らかにする
- PERF08-BP04 パフォーマンス向上の影響を測定する

- PERF08-BP05 さまざまなパフォーマンス関連戦略を使用する

PERF08-BP01 パフォーマンスが最も重要な分野を理解する

ワークロードのパフォーマンスの向上が効率性やカスタマーエクスペリエンスにプラスの影響を与える分野を理解し、特定します。例えば、カスタマーインタラクションが多いウェブサイトは、エッジサービスを使用してコンテンツ配信をお客様に近い場所へ移動させることでメリットを得ることができます。

期待される成果: アーキテクチャ、トラフィックパターン、データアクセスパターンを理解し、レイテンシーと処理時間を特定することで、パフォーマンス効率を高めることができます。ワークロードが増加するにつれて、顧客エクスペリエンスに影響を及ぼす可能性のある潜在的なボトルネックを特定できます。これらの領域を特定したら、デプロイできるソリューションを調査し、パフォーマンスの懸念を取り除きます。

一般的なアンチパターン:

- パフォーマンスの問題の計測には、 CPU############## 標準的なコンピューティングメトリクスで十分だと考えている。
- 一部のモニタリングソフトウェアで記録されるデフォルトのメトリクスのみを使用している。
- 問題が発生したときにだけメトリクスを確認している。

このベストプラクティスを活用するメリット: パフォーマンスの重要な領域を理解することで、ワークロードの所有者は KPI をモニタリングし、影響の大きいパフォーマンスの改善に優先順位をつけることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

エンドツーエンドの追跡を構築して、トラフィックパターン、レイテンシー、重要なパフォーマンス領域を特定します。データアクセスパターンをモニタリングして、低速なクエリや不十分にフラグメント化されパーティション化されたデータを検出します。負荷テストまたはモニタリングを使用して、ワークロードのボトルネックを特定します。

実装手順

1. エンドツーエンドのモニタリングを構築して、すべてのワークロードコンポーネントおよびメトリクスをキャプチャします。

- Amazon CloudWatch Real-User Monitoring (RUM) を使用して、 実際のユーザークライアント側およびフロントエンドセッションからの アプリケーションパフォーマンスメトリクスをキャプチャします。
- AWS X-Ray を設定して、 アプリケーションレイヤー内のトラフィックを追跡し、 コンポーネントと依存関係間のレイテンシーを特定します。X-Ray サービスマップを使用して、リレーションシップとワークロードコンポーネント間のレイテンシーを確認します。
- Amazon CloudWatch Performance Insightsを使用して、 データベースパフォーマンスメトリクスを確認し、 パフォーマンスの改善を特定します。
- Amazon RDS 拡張モニタリングを使用して、 データベース OS の パフォーマンスメトリクスを確認します。
- ワークロードコンポーネントおよびサービスごとの CloudWatch メトリクスを収集し、 どのメトリクスがパフォーマンスの効率性に影響しているかを特定します。
- Amazon RDS を を設定して、 パフォーマンスのインサイトとレコメンデーションを追加します。

2. テストを実行してメトリクスを生成し、トラフィックパターン、ボトルネック、および重要なパフォーマンス領域を特定します。
   - CloudWatch Synthetic Canaries を設定して、 cron ジョブを使用したブラウザベースのユーザーアクティビティの プログラム的なシミュレーションや、 ########################## 表現の評価を行います。
   - AWS Distributed Load Testing ソリューションを使用して、 ピークトラフィックの生成や、 予想される増加率でのワークロードのテストを行います。
3. メトリクスとテレメトリを評価して、重要なパフォーマンス領域を特定します。これらの領域をチームと一緒にレビューして、モニタリングおよびボトルネックを防ぐためのソリューションについて話し合います。
4. パフォーマンスの改善をテストし、データを使用してこれらの変更を計測します。
   - CloudWatch Evidently を使用して、 新しい改善のテストや ワークロードパフォーマンスへの効果の確認を行います。

実装計画に必要な工数レベル: このベストプラクティスを確立するには、エンドツーエンドのメトリクスをレビューし、現在のワークロードパフォーマンスを把握する必要があります。エンドツーエンドのモニタリングの構築および重要なパフォーマンス領域の特定に必要な工数レベルは中程度です。

リソース

関連するドキュメント:

- Amazon Builders' Library
- X-Ray ドキュメント
- Amazon CloudWatch RUM
- を設定して、
- CloudWatch RUM および X-Ray

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- Demo of Amazon CloudWatch Synthetics (Amazon CloudWatch Synthetics のデモ)

関連サンプル:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Node.js
- X-Ray SDK for Python
- X-Ray SDK for Java
- X-Ray SDK for .Net
- X-Ray SDK for Ruby
- X-Ray Daemon
- AWS での分散負荷テスト

PERF08-BP02 設計パターンとサービスについて理解する

ワークロードのパフォーマンスの向上に役立つさまざまな設計パターンとサービスについて調査し、理解します。分析の一環として、より優れたパフォーマンスを達成するために何を引き替えにすることができるかを特定します。例えば、キャッシュサービスを使用することで、データベースシステムにかかる負荷を軽減できます。しかし、キャッシュは最終的な一貫性をもたらす可能性があり、ビジネス要件と顧客の期待の範囲内で実装するためにはエンジニアリングの労力が必要です。

期待される成果: 設計パターンを研究することは、最高のパフォーマンスを発揮するシステムを支えるアーキテクチャ設計を選択することにつながります。どのようなパフォーマンス設定オプションが利用できるか、およびそれらがワークロードにどのように影響し得るかについて学んでください。

ワークロードのパフォーマンスを最適化するには、これらのオプションがアーキテクチャとどのように相互作用し、測定されたパフォーマンスとエンドユーザーが知覚するパフォーマンスの両方に影響を与えるかを理解することに依存します。

一般的なアンチパターン:

- 従来のすべての IT ワークロードパフォーマンス戦略がクラウドワークロードに最適であると考えている。
- マネージドサービスを使用するのではなく、キャッシュソリューションを構築および管理する。
- どのパターンがワークロードのパフォーマンスを向上させるかを評価することなく、すべてのワークロードに同じデザインパターンを使用する。

このベストプラクティスを活用するメリット: ワークロードに適した設計パターンとサービスを選択することで、パフォーマンスを最適化し、運用性を向上させ、信頼性を高めることができます。適切な設計パターンは、現在のワークロード特性に適合し、将来の成長や変化に応じたスケールを容易にします。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

どのようなパフォーマンス設定オプションが利用できるか、およびそれらがワークロードにどのように影響し得るかについて学びます。ワークロードパフォーマンスの最適化は、これらのオプションがアーキテクチャとどのように相互作用するか、および実測パフォーマンスとユーザーが認識するパフォーマンスに及ぶ影響を理解することにかかっています。

実装手順:

1. ワークロードのパフォーマンスを向上させるような設計パターンを評価し、検討します。
    a. 最も Amazon Builders' Library では、Amazon がテクノロジーを構築し、運用する方法に関する詳しい説明を参照できます。これらの記事は Amazon のシニアエンジニアによって書かれたもので、アーキテクチャ、ソフトウェアデリバリー、および運用全般のトピックを取り上げています。
    b. AWS ソリューションライブラリ は、サービス、コード、および設定を集めた、すぐにデプロイできるソリューションのコレクションです。これらのソリューションは、業界別またはワークロードタイプ別にまとめられた一般的なユースケースと設計パターンに基づいて、AWS と AWS パートナーによって作成されたものです。例えば、ワークロードの 分散負荷テストソリューション をセットアップできます。

   c. AWS アーキテクチャセンター には、設計パターン、コンテンツタイプ、およびテクノロジー別にまとめられた参照アーキテクチャー図があります。
   d. AWS サンプル は、一般的なアーキテクチャパターン、ソリューション、およびサービスを試すことができる多くの実習例を含んだ GitHub リポジトリです。最新のサービスや事例など、頻繁に更新しています。
2. ワークロードを改善して、選択した設計パターンをモデル化し、サービスとサービス設定オプションを使用して、ワークロードパフォーマンスを高めます。
   a. 社内チームを以下で入手可能なリソースでトレーニングしましょう AWS Skills Guild.
   b. このセットアップに役立てるため AWS Partner Network を使用して、知識を迅速に提供し、改善能力を高めます。

実装計画に必要な工数レベル: このベストプラクティスを確立するには、ワークロードパフォーマンスの向上に役立つ設計パターンとサービスを認識する必要があります。設計パターンを評価した後、設計パターンを実装するには、 高 程度の工数が必要です。

リソース

関連するドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター
- Amazon Builders' Library
- 負荷制限を使用して過負荷を回避する
- Caching challenges and strategies

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- This is My Architecture

関連する例:

- AWS Samples

- AWS SDK サンプル

PERF08-BP03 トレードオフが顧客と効率性にどのように影響するかを明らかにする

パフォーマンス関連の向上を評価する場合、どの選択肢が顧客とワークロードの効率性に影響するかを特定します。たとえば、key-value データストアの使用がシステムパフォーマンスを向上させる場合、その結果整合性の性質がお客様に与える影響を評価することが大切です。

メトリクスとモニタリングを使用して、システムのパフォーマンスが低い領域を特定します。どのように改善できるか、その改善によってどのようなトレードオフがもたらされるか、およびそれらがシステムとユーザーエクスペリエンスにどのように影響するかを判断します。例えば、データのキャッシングを実装すると、パフォーマンスが劇的に向上しますが、システムの不正な動作を防止するためにキャッシュデータをいつどのように更新または無効化するかに関する明確な戦術が必要になります。

一般的なアンチパターン:

- 実装に結果整合性などのトレードオフがある場合でも、すべてのパフォーマンス向上が実装されるべきであると考えている。
- パフォーマンスの問題が限界に達した場合にのみ、ワークロードの変更を評価する。

このベストプラクティスを活用するメリット: パフォーマンス関連の改善の可能性を評価する場合は、変更のトレードオフがワークロード要件と整合的であるかどうかを判断する必要があります。場合によっては、トレードオフを補うために追加のコントロールを実装する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

トレードオフを特定する: メトリクスとモニタリングを使用して、システムのパフォーマンスが低い領域を特定します。改善を行う方法と、トレードオフがシステムとユーザーエクスペリエンスに与える影響を特定します。たとえば、データキャッシングの実装はパフォーマンスを劇的に向上させるうえで役立ちますが、システムの誤動作を防止するために、キャッシュされたデータをいつどのように更新または無効化するかに関する明確な戦略が必要になります。

リソース

関連ドキュメント:

- Amazon Builders' Library
- Amazon QuickSight KPI
- Amazon CloudWatch RUM
- X-Ray ドキュメント

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- モニタリング計画を立てる
- Optimize applications through Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics のデモ

関連サンプル:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF08-BP04 パフォーマンス向上の影響を測定する

パフォーマンスを向上させるための変更を行うと同時に、収集されたメトリクスとデータを評価します。この情報を使用して、そのパフォーマンス向上がワークロード、ワークロードのコンポーネント、および顧客に与えた影響を判断します。この測定は、トレードオフに由来するパフォーマンス向上を理解する、および副次的な悪影響が生じたかどうかを判断するために役立ちます。

適切に設計されたシステムでは、パフォーマンス関連の戦略を組み合わせて使用します。特定のホットスポットまたはボトルネックに最も大きなプラスの影響を与える戦略を特定します。たとえば、複数のリレーショナルデータベースシステム全体でのデータシャーディングは、トランザクションに対するサポートを維持しながら、全体的なスループットを向上させることができ、各シャード内ではキャッシングが負荷の低減に役立ちます。

一般的なアンチパターン:

- マネージドサービスとして使用できるテクノロジーを手動でデプロイおよび管理する。
- ワークロードのパフォーマンス向上に複数のコンポーネントを使用できる場合に、ネットワーキングなど、1 つのコンポーネントにのみ重点を置く。

- 顧客からのフィードバックと認識を唯一のベンチマークとして使用している。

このベストプラクティスを活用するメリット: パフォーマンス戦略を実装するには、複数のサービスと機能を選択する必要があります。これらを組み合わせると、パフォーマンスに関するワークロード要件を満たすことができます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

適切に設計されたシステムでは、パフォーマンス関連の戦略を組み合わせて使用します。特定のホットスポットまたはボトルネックに最も大きなプラスの影響を与える戦略を特定します。たとえば、複数のリレーショナルデータベースシステム全体でのデータシャーディングは、トランザクションに対するサポートを維持しながら、全体的なスループットを向上させることができ、各シャード内ではキャッシングが負荷の低減に役立ちます。

リソース

関連ドキュメント:

- Amazon Builders' Library
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- AWS での分散負荷テスト

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- Optimize applications through Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics のデモ

関連サンプル:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- AWS での分散負荷テスト

PERF08-BP05 さまざまなパフォーマンス関連戦略を使用する

状況に応じて、複数の戦略を活用してパフォーマンスを向上させます。たとえば、過剰なネットワーク呼び出しやデータベース呼び出しを防止するためにデータキャッシングなどの戦略を使用する、データベースエンジンの読み込み速度を向上させるためにリードレプリカを使用する、データボリュームを削減するためにデータのシャーディングまたは圧縮を可能な限り使用する、ブロッキングを回避するために利用可能になった結果をバッファし、ストリーミングするなどの戦略を使用します。

ワークロードに変更を加えた後、メトリクスを収集して評価し、それらの変更の影響を判断します。システムおよびエンドユーザーに対する影響を測定して、トレードオフがワークロードにどのような影響を与えるかを理解します。負荷テストなどの体系的なアプローチを使用して、トレードオフによってパフォーマンスが向上するかどうかを調べます。

一般的なアンチパターン:

- 顧客からの苦情がなければ、ワークロードのパフォーマンスが十分であると考えている。
- パフォーマンス関連の変更を行った後にのみ、パフォーマンスに関するデータを収集する。

このベストプラクティスを活用するメリット: パフォーマンスとリソース使用率を最適化するには、統合された運用ビュー、リアルタイムの詳細なデータ、履歴参照が必要です。ダッシュボードを作成し、データに関するメトリクスの計算を実行して、ワークロードが時間の経過とともに変化するのに伴い、ワークロードの運用と利用に関する洞察を得ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

データ駆動型のアプローチでアーキテクチャを進化させる: ワークロードに変更を加えた後、メトリクスを収集して評価し、変更の影響を判断します。システムおよびエンドユーザーに対する影響を測定して、トレードオフがワークロードにどのような影響を与えるかを理解します。負荷テストなどの体系的なアプローチを使用して、トレードオフによってパフォーマンスが向上するかどうかを調べます。

リソース

関連ドキュメント:

- Amazon Builders' Library

- Best Practices for Implementing Amazon ElastiCache
- AWS Database Caching
- Amazon CloudWatch RUM
- AWS での分散負荷テスト

関連動画:

- Introducing The Amazon Builders' Library (DOP328)
- AWS purpose-built databases (DAT209-L)
- Optimize applications through Amazon CloudWatch RUM

関連サンプル:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- AWS での分散負荷テスト

コスト最適化

トピック

- クラウド財務管理を実践する
- 経費支出と使用量の認識
- 費用対効果の高いリソース
- 需要を管理しリソースを供給する
- 継続的最適化

クラウド財務管理を実践する

質問

- COST 1 クラウド財務管理はどのように実装すればよいですか?

COST 1 クラウド財務管理はどのように実装すればよいですか?

組織はクラウド財務管理の実装により、AWS によってコストと使用状況の最適化とスケーリングをすることで、ビジネス価値と財務上の成功を実現できます。

ベストプラクティス

- COST01-BP01 コスト最適化担当者を設定する
- COST01-BP02 財務とテクノロジーの連携を確立する
- COST01-BP03 クラウドの予算と予測を確立する
- COST01-BP04 組織のプロセスにコスト意識を採り入れる
- COST01-BP05 コスト最適化に関して報告および通知する
- COST01-BP06 コストをプロアクティブにモニタリングする
- COST01-BP07 新しいサービスリリースに関する最新情報を把握しておく

COST01-BP01 コスト最適化担当者を設定する

組織全体のコスト認識を確立し、維持する責任を持つチーム (クラウドビジネスオフィスまたは Cloud Center of Excellence) を作成します。このチームには、組織全体の財務、テクノロジー、およびビジネスの役割を持つ人材が必要です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

クラウドコンピューティングにおけるコスト意識の文化を確立し、維持する責任を負うクラウドビジネスオフィス (CBO) または Cloud Center of Excellence (CCOE) チームを確立します。これは社内の個人でも、チームでもかまいません。組織全体から財務、テクノロジーなどの主なステークホルダーを集めてチームを新規編成することもできます。

担当者 (個人またはチーム) は、コスト管理とコスト最適化活動に必要な時間を優先順序を付けて配分します。小規模な組織の場合、大企業のフルタイムの担当と比較すると、費やす時間の割合は少ない場合があります。

プロジェクトマネジメント、データサイエンス、財務分析、ソフトウェアやインフラストラクチャ開発など、複合的な能力が求められます。す。担当者は次の 3 つの異なる所有権内でコスト最適化を実行することにより、ワークロードの効率を高めることができます。

- 一元化: 財務オペレーション、コスト最適化、CBO、CCOE などの指定チームを通じて、お客様はガバナンスの仕組みを設計、導入し、ベストプラクティスを全社的に推進することができます。
- 分散型: テクノロジーチームに影響を与え、最適化を実行させる。
- ハイブリッド: 一元化されたチームと分散されたチームの両方が協力して、コストの最適化を実行することができます。

この担当者は、コスト最適化目標 (ワークロード効率メトリクスなど) に対する実行および提供能力を評価されることになります。

この担当者が変化を起こすためには、エグゼクティブスポンサーシップを確保しなければならず、これが重要な成功要因です。エグゼクティブスポンサーは、クラウド利用のコスト効率を判断する最高責任者として、担当者の考え方を上長にエスカレーションして、組織が定める優先事項としてコスト最適化活動が扱われるようサポートします。そうでなければ、ガイダンスは無視され、コスト削減の機会が優先されなくなります。エグゼクティブスポンサーと担当者は協力して、組織のクラウド利用を効率化し、ビジネスバリューの実現を継続できるようにします。

ビジネス、Enterprise-On-Ramp、またはエンタープライズサポートプランを利用していて、このチームまたは担当者の構築に支援が必要な場合は、アカウントチームを通じてクラウド財務管理 (CFM) のエキスパートにご連絡ください。

実装手順

- 主要なメンバーを定義する: 組織のすべての関連部分が貢献し、コスト管理に関与している状況を確保する必要があります。一般的な組織内チームには、通常、財務、アプリケーションまたはプロダクトの所有者、管理、技術チーム (DevOps) が含まれています。一部は専属 (財務、技術) で、その他は必要に応じて定期的に関与します。CFM を実行する個人またはチームには、一般に以下のスキルセットが必要です。
  - ソフトウェア開発スキル - スクリプトとオートメーションが構築される場合。
  - インフラストラクチャエンジニアリングスキル - スクリプトまたはオートメーションをデプロイし、サービスまたはリソースのプロビジョニング方法を理解するためです。
  - 運用の洞察力 - CFM とは、クラウドの効率的な利用を測定、モニタリング、修正、計画、拡張することで、クラウドで効率的に運用することです。
- 目標とメトリクスを定義する: この担当者は、さまざまな方法で組織に価値をもたらす必要があります。これらの目標は定義され、組織が進化するにつれて継続的に進化します。一般的な活動には、組織全体のコスト最適化に関する教育プログラムの作成と実行、コスト最適化のためのモニタリングやレポート作成などの組織全体の標準策定、最適化に関するワークロード目標の設定などが

あります。この担当者は、組織のコスト最適化機能について定期的に組織に報告する必要もあります。

価値ベースの重要業績指標 (KPI) を定義できます。KPI には、コストベースと価値ベースがあります。KPI を定義すると、効率性と予想されるビジネス成果の観点から、予想されるコストを計算できます。価値ベースの KPI は、コストおよび使用量のメトリクスをビジネスバリュー因子に結び付け、AWS 経費の変更の合理化に役立ちます。価値ベースの KPI を導き出す最初のステップは、組織横断的に協力し、KPI の標準セットを選択し、合意することです。

- 定期的なミーティングを設定する: グループ (財務、技術、およびビジネスチーム) は定期的に会合を開いて、目標とメトリクスをレビューする必要があります。一般的なミーティングでは、組織の状態の確認、現在実行中のプログラムの確認、全体的な財務および最適化メトリクスの確認を行います。その後、主要なワークロードが詳細に報告されます。

  このような定期的なミーティングの際に、ワークロードの効率性 (コスト) とビジネス成果をレビューできます。例えば、ワークロードのコストが 20% 増加したが、顧客使用量も増加したかもしれません。この場合、この 20% のコスト増加を投資と解釈できます。このような定期的なミーティングにより、チームは組織全体にとって有意義な価値ベースの KPI を特定できます。

リソース

関連するドキュメント:

- AWS CCOE ブログ
- クラウドビジネスオフィスの作成
- CCOE - Cloud Center of Excellence

関連動画:

- Vanguard CCOE 成功事例

関連する例:

- Could Center of Excellence (CCoE) を活用した企業全体の変革
- 企業全体を変革する CCOE の構築
- CCOE を構築するときに回避すべき 7 つの落し穴

COST01-BP02 財務とテクノロジーの連携を確立する

クラウドジャーニーのすべての段階で、コストと使用状況に関するディスカッションに財務チームとテクノロジーチームを参加させます。チームは、定期的に集まり、組織の最終目的や目標、コストと使用状況の現状、財務や会計のプラクティスなどのトピックについて話し合います。

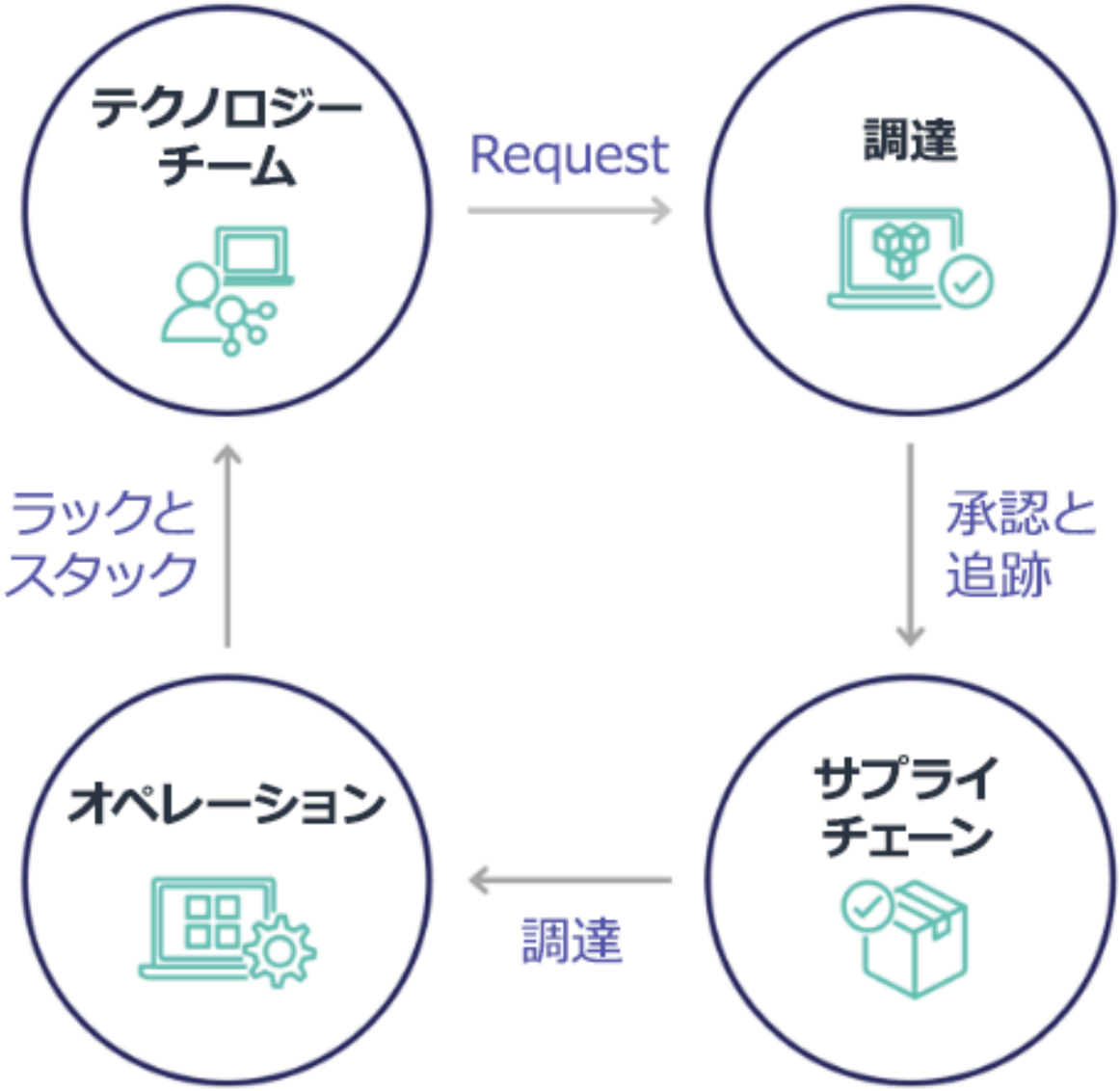
このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

承認、調達、インフラストラクチャのデプロイサイクルが短縮されるため、テクノロジーチームは、クラウドでのイノベーションを迅速に行うことができます。ファイナンス組織はこれまでプロジェクト承認時のデータセンターやオンプレミス環境の調達に大量に使用されていた時間とリソースを調整することができます。

財務および調達組織の観点から見ると、資本予算、資本要求、承認、調達、物理的インフラストラクチャの設置のプロセスは、何十年にもわたって学習され、標準化されてきたプロセスの一つです。

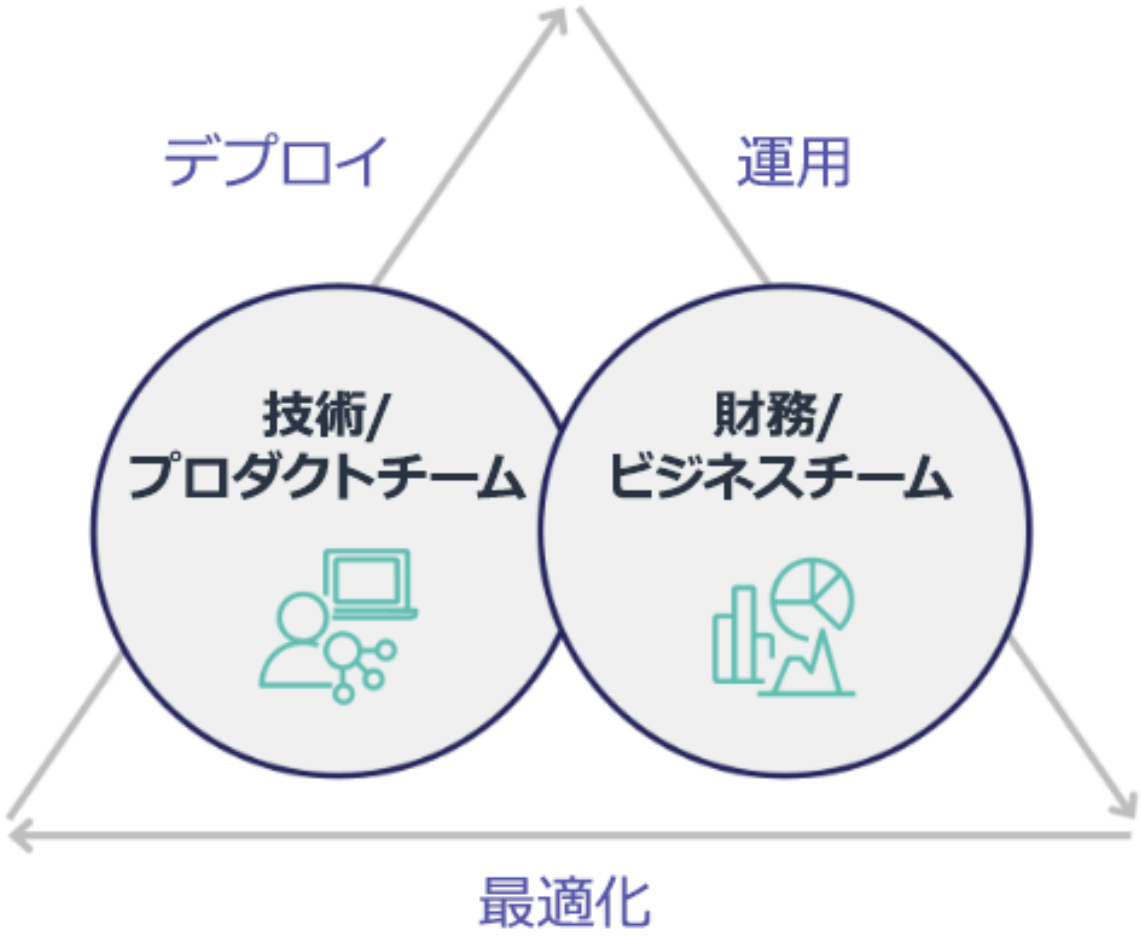
- エンジニアリングチームや IT チームは、通常、依頼主である
- さまざまな財務チームが承認者、調達者として機能する
- 運用チームは、すぐに使えるインフラストラクチャのラック、スタック、および引き渡しを行う



クラウドの採用により、インフラストラクチャの調達と消費は依存関係の連鎖と切り離されます。クラウドモデルでは、テクノロジーチームと製品チームは単なる構築者ではなく、製品の運用者兼所有者となり、調達とデプロイなど、従来は財務および運用チームに割り当てられていたアクティビティのほとんどを担当します。

クラウドリソースのプロビジョニングに必要なものは、ユーザーアカウントと適切なアクセス許可のセットだけです。これは IT および財務リスクを軽減することにもなります。つまり、チームは常に数回のクリックまたは API コールで、アイドル状態または不要なクラウドリソースを停止することができるのです。また、テクノロジーチームがより速くイノベーションを起こすことができるのも、実験を立ち上げては破棄する俊敏性と能力があってこそです。クラウド消費には変動的な性質があるため、資本支出予算と予測の観点から予測可能性に影響することがある一方で、クラウドによって組

織は、オーバープロビジョニングのコストを削減し、控えめなアンダープロビジョニングに伴う機会コストも削減できます。



主要な財務部門とテクノロジー部門のステークホルダー同士のパートナーシップを確立し、組織の目標の共通理解を深め、クラウドコンピューティングのさまざまな消費モデルにおいて財務上の成功を実現するためのメカニズムを開発します。クラウドジャーニーのすべてのステージにおいて、コストと使用量に関するディスカッションに参加する必要がある組織内の関連するチームは、以下の。

- ファイナンシャルリード: CFO、財務管理者、フィナンシャルプランナー、ビジネスアナリスト、調達、ソーシング、支払担当は、クラウドの消費モデル、購入オプション、月次請求プロセスを理解する必要があります。財務部門は、テクノロジーチームと連携して、IT バリューの事例を作成およびソーシャル化し、テクノロジー支出がビジネスの成果にどのように関連しているかをビジネスチームが理解できるようにする必要があります。このように、技術支出はコストではなく投資と見なされます。クラウド運用にはオンプレミスのオペレーションと比べて根本的な違い (使用量の変動率、従量課金制やティア別課金制、料金モデル、請求明細と使用量情報など) があるため、ク

ラウド利用が調達プロセス、インセンティブ追跡、コスト配分、財務諸表などのビジネス局面に与えるインパクトをファイナンス部門で理解することが不可欠です。

- テクノロジーリード: テクノロジーリード (製品およびアプリケーションの所有者を含む) は、財務要件 (予算の制約など) やビジネス要件 (サービスレベルアグリーメントなど) を認識する必要があります。これにより、組織が目指すビジネス目標を達成するワークロードの導入が可能になります。

財務とテクノロジーのパートナーシップには、以下のような利点があります。

- 財務チームとテクノロジーチームは、コストと使用量をほぼリアルタイムで把握できます。
- 財務チームとテクノロジーチームは、クラウドへの支出の変動に対応するための標準となる運用手順を確立します。
- 財務部門のステークホルダーは、コミットメント割引 (リザーブドインスタンスや AWS Savings Plans など) の購入に資金がどう使用されるか、また組織を拡大するためにクラウドがどのように利用されるかに関して、戦略アドバイザーとして行動します。
- 既存の支払いアカウントと調達プロセスは、クラウドとともに使用されます。
- 財務チームとテクノロジーチームは、協力して将来的な AWS のコストと使用量を予測し、組織の予算を調整および構築します。
- 両者の共通言語により組織間のコミュニケーションが向上し、財務の概念への共通理解が得られます。

コストと使用量のディスカッションについて、組織内で関わるべきその他のステークホルダーは以下のとおりです。

- 事業部門オーナー: 事業部門オーナーは、事業部門と会社全体の両方に方向性を提供できるように、クラウドのビジネスモデルを理解する必要があります。こうしたクラウド知識は成長とワークロード使用量を予測する際に、またリザーブドインスタンスや Savings Plans などの長期購入オプションを検討する際に重要な役割を果たします。
- エンジニアリングチーム: エンジニアがクラウド財務管理 (CFM) に取り組むよう促す、コストを意識した企業文化の構築には、財務チームとテクノロジーチームのパートナーシップの確立が欠かせません。CFM や財務業務の実務担当者や財務チームが共通して抱える問題の 1 つは、エンジニアにクラウド上のビジネス全体を理解させ、ベストプラクティスに従わせ、推奨されるアクションを取らせることです。
- サードパーティー: サードパーティー (コンサルタントやツールなど) を利用する場合、こうしたサードパーティーが財務目標に適合し、エンゲージメントモデルと投資収益率 (ROI) を通じて、ど

ちらの整合性も実証できるようにします。通常、サードパーティーは自社管理のワークロードのレポーティングと分析を担当したり、自社設計のワークロードのコストを分析したりします。

CFM を導入し、成功させるには、財務、テクノロジー、ビジネスの各チームが協力し、組織全体におけるクラウド費用の伝達と評価の方法を変える必要があります。エンジニアリングチームを巻き込み、あらゆる段階でコストと使用に関する議論に参加させ、ベストプラクティスに従って合意されたアクションを取るよう奨励します。

実装手順

- 主要なメンバーを定義する: 財務チームとテクノロジーチームのすべての関連メンバーがこの連携に関与していることを確認します。関連する財務メンバーは、クラウドの請求書に関する業務に従事するメンバーです。通常は、CFO、財務コントローラー、財務プランナー、ビジネスアナリスト、購買管理です。テクノロジーチームのメンバーは通常、プロダクトおよびアプリケーションの所有者、テクニカルマネージャー、およびクラウド上に構築するすべてのチームの代表者です。他のメンバーとしては、製品の使用に影響するマーケティングなどのビジネスユニットの所有者、および貴社の目標やメカニズムとの整合性を確保し、報告をサポートするコンサルタントなどのサードパーティーが含まれることがあります。
- ディスカッションのためのトピックを定義する: チーム間で共通する、または理解を共有する必要があるトピックを定義します。作成から支払いまでにかかるコストを追います。関連するメンバー、および適用する必要がある組織のプロセスを書き留めます。各ステップまたはプロセスのほか、利用可能な料金モデル、階層化された料金、割引モデル、予算、財務要件などの関連情報を理解します。
- 定期的なミーティングを設定する: 財務とテクノロジーのパートナーシップを構築するために、定期的なコミュニケーションの機会を設け、連携を維持します。グループは目標とメトリクスに照らして定期的に集まる必要があります。一般的なミーティングでは、組織の状態の確認、現在実行中のプログラムの確認、全体的な財務および最適化メトリクスの確認を行います。その後、主要なワークロードが詳細に報告されます。

リソース

関連するドキュメント:

- AWS ニュースブログ

COST01-BP03 クラウドの予算と予測を確立する

既存の組織の予算作成および予測プロセスを調整し、非常に変動しやすいクラウドのコストと使用状況の性質に対応できるようにします。プロセスは、トレンドベースまたはビジネスドライバーベースのアルゴリズム、またはそれらの組み合わせを使用して、動的なものとする必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

お客様は効率性、スピード、俊敏性を求めてクラウドを利用しますが、コストと使用量は大きく変動するものです。コストは、ワークロードの効率性の向上や、新規ワークロードや新機能のデプロイにより削減可能です。ワークロードの効率性が向上したときや、新しいワークロードと機能がデプロイされたときにコストが増加することがあります。ワークロードをスケーリングすると、サービスを提供する顧客が増えますが、その分クラウドの使用量とコストが増加します。リソースは以前より容易にアクセスできるようになります。クラウドの伸縮性は、コストと予測の伸縮性にもつながります。こうした変動を折り込めるように、組織の既存の予算編成プロセスを変える必要があります。

トレンドベースのアルゴリズム (コスト履歴を入力値として使用)、ビジネスドライバーベースのアルゴリズム (新製品の発売や営業地域の拡大など)、またはこの 2 つのアルゴリズムを組み合わせて、既存の予算編成と予測プロセスをより動的なものに調整します。

予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Budgets を使用して、期間、繰り返し、または金額 (固定費または可変費) を指定し、サービス、AWS リージョン、タグなどのフィルターを追加することで、カスタム予算を詳細レベルで設定します。既存予算のパフォーマンスについて常に情報を入手するには、 AWS Budgets Reports を作成して、自分と関係者に定期的に E メールで送信されるようにスケジュールします。また、 AWS Budgets アラート は、実際のコストに基づいて作成することもできます (これは反応型です) し、予測コストに基づいて作成することも可能で、潜在的なコスト超過に対する緩和策を実施する時間を確保することができます。コストまたは使用量が予算額を超えるか、超えると予測された場合、アラートが表示されます。

AWS は、動的な予測と予算編成のプロセスを柔軟に構築できるため、コストが予算の上限を守っているか、あるいは超えているかを常に把握することができます。

予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Cost Explorer を使用し、過去の支出に基づいて、定義された期間のコストを予測します。AWS Cost Explorer の予測エンジンは、料金タイプ (リザーブドインスタンスなど) に基づいて履歴データをセグメント化し、機械学習とルールベースのモデルの組み合わせを使用して、すべての料金タイプにかかる費用を個別に予測します。予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Cost

Explorer を使用して、過去のコスト (トレンドベース) に適用される機械学習アルゴリズムに基づいて、日次 (最大 3 か月) または月次 (最大 12 か月) のクラウドコストを予測します。

Cost Explorer を使用してトレンドベースの予測が出来たら、 AWS Pricing Calculator を使用し、予想される使用量 (トラフィック、1 秒あたりのリクエスト数、必要な Amazon Elastic Compute Cloud (Amazon EC2) インスタンスなど) に基づいて、AWS のユースケースと今後のコストを見積もります。これは、AWS を利用する際に、支出方法のプランニング、コスト節減機会の発見、情報に基づいた意思決定にも役立てることができます。

予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Cost Anomaly Detection を使用して、予想外のコストを防止または削減し、イノベーションを遅らせることなく制御を強化します。AWS Cost Anomaly Detection は、高度な機械学習テクノロジーを利用して、異常な支出と根本原因を特定するため、迅速に対応できます。3 つのシンプルなステップで、状況に応じて独自のモニターを作成し、異常な支出が検出されたときにアラートを受け取ることができます。構築はビルダーに任せ、AWS Cost Anomaly Detection は支出をモニタリングし、予期せぬ請求のリスクを軽減します。

次のサブセクション「Well-Architected コスト最適化の柱の財務とテクノロジーのパートナーシップ」セクションで述べられているように、IT、財務、その他の関係者が同じツールやプロセスを使用し、一貫性を保つためのパートナーシップと連携が重要です。予算を変更する必要がある場合、ミーティングの回数を増やすと、それらの変更により迅速に対応できます。

実装手順

- 既存の予算作成および予測プロセスを更新する: 予算作成プロセスと予測プロセスにおいて、トレンドベース、ビジネスドライバーベース、または両方の組み合わせを採り入れます。
- アラートと通知を設定する: AWS Budgets アラートと Cost Anomaly Detection を使用します。
- 主要関係者と定期的なレビューを行う: 例えば、IT、財務、プラットフォーム、およびその他のビジネス分野の関係者が、ビジネスの方向性と使用状況の変化を認識し、連携を図ります。

リソース

関連するドキュメント:

- AWS Cost Explorer
- AWS Budgets
- AWS Pricing Calculator

- AWS Cost Anomaly Detection
- AWS License Manager

関連する例:

- ローンチ: 使用量ベースの予測が AWS Cost Explorer で使用可能に
- AWS Well-Architected ラボ: コストと使用量のガバナンス

COST01-BP04 組織のプロセスにコスト意識を採り入れる

コスト意識を高め、透明性を強化し、使用量に影響する新規または既存のプロセスにコストの説明責任を採り入れ、コスト意識に関する既存のプロセスを活用します。従業員のトレーニングにコスト意識の要素を採り入れます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

コスト意識は、組織の新規および既存のプロセスに採り入れる必要があります。他のベストプラクティスの基盤となる前提条件の機能の 1 つです。可能な限り既存のプロセスを再利用し、修正することが推奨されます。これにより、俊敏性と速度への影響を最小化することができます。クラウドのコストをテクノロジーチームと、ビジネスチームおよび財務チームの意思決定者に報告して、コスト意識を高め、効率性についての重要業績評価指標 (KPI) を財務およびビジネス部門関係者向けに確立します。次の推奨事項は、ワークロードにコスト意識を実装するのに役立ちます。

- 変更管理に、変更による財務への影響を数値化するコスト測定が含まれていることを確認します。これは、コスト関連の懸念に積極的に対処し、コスト削減を強調するのに役立ちます。
- コストの最適化が、業務能力の中核をなす要素であることを確認します。例えば、既存のインシデントマネジメントプロセスを活用して、コストと使用量に関する異常値 (コスト超過) の根本原因を調査、特定することができます。
- オートメーションやツールにより、コスト削減とビジネス価値の実現を加速します。導入コストを考える場合、時間や費用の投資を正当化するために、投資収益率 (ROI)の要素を含むように話を組み立てます。
- コミットメントベースの購入オプション、共有サービス、マーケットプレイスでの購入を含むクラウド使用に対してショーバックまたはチャージバックを実施し、最もコストを意識したクラウド消費を促進することで、クラウドのコストを配分します。

- 既存のトレーニングおよび開発プログラムを拡張し、コスト意識向上のためのトレーニングを組織全体で実施します。これには継続的なトレーニングと認定が含まれることをお勧めします。これにより、コストと使用量を自己管理できる組織が育成されます。
- 次のような無料の AWS ネイティブツールを利用します。 AWS Cost Anomaly Detectionio1AWS Budgets、および AWS Budgets Reports.

組織が一貫して クラウド財務管理 (CFM) 実践を採用すると、そのような行動は、仕事の進め方や意思決定方法に定着していきます。その結果、新しいクラウドで生まれたアプリケーションを設計するデベロッパーから、これらの新しいクラウド投資の ROI を分析する財務マネージャーまで、よりコストを意識した文化が生まれます。

実装手順

- 関連する組織のプロセスを把握する: 各組織単位は、そのプロセスをレビューし、コストと使用状況に影響を与えるプロセスを把握します。リソースの作成または終了につながるすべてのプロセスは、レビューの対象とする必要があります。インシデント管理やトレーニングなど、ビジネスにおけるコスト認識の維持につながるプロセスを探します。
- コストを意識した自立的な企業文化を確立する: すべての関係者がクラウドのコストを理解できるように、変更原因と影響をコストとして認識するようにします。そうすることで、組織がコストを意識したイノベーションの文化を自立的に確立することができます。
- コスト意識の要素を採り入れてプロセスを更新する: 各プロセスは、コスト意識が浸透するように変更されます。このプロセスでは、コストの影響の評価などの追加の事前チェック、またはコストと使用状況の予想された変化が発生したかどうかを検証する事後チェックが必要になる場合があります。トレーニングやインシデント管理などのサポートプロセスは、コストと使用状況の項目を含むように拡張できます。

ご不明な点がありましたら、アカウントチームを通じて CFM のエキスパートにお問い合わせいただくか、以下のリソースや関連ドキュメントをご覧ください。

リソース

関連するドキュメント:

- AWS によるクラウド財務管理

関連する例:

- 効率的なクラウドコスト管理の戦略
- Cost Control Blog Series #3: How to Handle Cost Shock (コスト管理ブログシリーズ #3: コストショックの扱い方)
- AWS Cost Management 初心者ガイド

COST01-BP05 コスト最適化に関して報告および通知する

AWS Budgets と AWS Cost Anomaly Detection を設定して、目標に対するコストと使用量に関する通知を提供します。定例会議を開催して、このワークロードのコスト効率を分析し、社内にコスト意識を浸透させます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

組織内のコストと使用量の最適化について、定期的に報告する必要があります。コスト最適化のための専用セッションの運用や、ワークロードの通常の運用レポートサイクルにコスト最適化を盛り込むことも意味があるでしょう。サービスとツールを使用して、コスト節減機会を特定し、実現します。AWS Cost Explorer では、ダッシュボードとレポートを提供します。設定された予算に対するコストと使用量の進行状況を追跡するには、 AWS Budgets Reports.

予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Budgets を使用してカスタム予算を設定し、コストと使用量を追跡し、しきい値を超えた場合は E メールまたは Amazon Simple Notification Service (Amazon SNS) 通知でアラートを受信し、迅速に対応します。優先予算 期間を日次、月次、四半期ごと、または年次に設定し、特定の予算限度を設けて、予算しきい値に対する実際または予測されたコストと使用量の進捗状況に関して、常に情報を入手します。また、 アラート と アクション を設定し、アクションがアラートに対して自動的に実行するか、予算目標を超えたときに承認プロセスを通じて実行するようにします。

コストと使用量に関する通知を実装して、コストと使用量の変化が予想外の場合はすばやく対応できるようにします。AWS Cost Anomaly Detection では、イノベーションを遅らせることなく、予想外のコストを削減し、管理を強化できます。AWS Cost Anomaly Detection は異常な費用と根本原因を特定するので、予想外の請求のリスクを削減できます。3 つのシンプルなステップで、状況応じて独自のモニターを作成し、異常な費用が検出されたときにアラートを受け取ることができます。

また、 Amazon QuickSight と AWS Cost and Usage Report (CUR) のデータを使用して、高度にカスタマイズされ、きめ細かなデータを含んだレポートを提供できます。Amazon QuickSight では、過去のコストと使用量またはコスト節減機会に関するレポートをスケジュールし、定期的なコストレポート E メールを受信できます。

予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Trusted Advisorを使用すると、プロビジョニングされたリソースがコスト最適化のための AWS のベストプラクティスに準拠しているかどうかを確認するためのガイダンスが提供されます。

Savings Plans、リザーブドインスタンス、および Amazon Elastic Compute Cloud (Amazon EC2) の適切なサイズ設計に関する AWS Cost Explorer からのレコメンデーションを含んだレポートを定期的に作成して、定常状態のワークロード、アイドルおよび使用量の少ないリソースに関するコストの削減を開始します。デプロイされているリソースのうち、クラウドの無駄に関する費用を特定し、回収します。クラウドの無駄は、サイズ設定が正しくないリソースが作成されたときや、使用量のパターンが予想とは異なるときに発生します。AWS のベストプラクティスに従って無駄を少なくし、クラウドコストを最適化し、 節減します。

定期的にレポートを作成し、リソースの購入オプションを改善することで、ワークロードの単価を下げることができます。Savings Plans、リザーブドインスタンス、Amazon EC2 スポットインスタンスなどの購入オプションは、耐障害性の高いワークロードのコストを最も低く抑え、関係者 (ビジネス所有者、財務チーム、テクノロジーチーム) がこれらのコミットメントの議論に参加できるようにするものです。

クラウドの総保有コスト (TCO) の削減に役立つ可能性のある機会または新しいリリースの発表を含んだレポートを共有します。新しいサービス、リージョン、機能、ソリューション、またはさらにコスト削減を達成する新しい方法を採用します。

実装手順

- AWS Budgets を設定する: ワークロードのすべてのアカウントで AWS Budgets を設定します。タグを使用して、アカウント全体の支出の予算とワークロードの予算を設定します。
  - Well-Architected ラボ: コストと使用量のガバナンス
- コスト最適化に関して報告する: ワークロードの効率について話し合い、分析する定期的なミーティングを設定します。確立されたメトリクスを使用して、達成されたメトリクスとそれを達成するためにかかったコストを報告します。好ましくない傾向を特定して修正するとともに、組織全体で推進できるような改善傾向を特定します。報告には、アプリケーションチームおよび所有者、財務、および管理の担当者が関与する必要があります。
  - Well-Architected ラボ: 可視化

リソース

関連するドキュメント:

- AWS Cost Explorer

- AWS Trusted Advisor
- AWS Budgets
- AWS Budgets のベストプラクティス
- Amazon CloudWatch
- AWS CloudTrail
- Amazon S3 分析
- AWS Cost and Usage Report

関連する例:

- Well-Architected ラボ: コストと使用量のガバナンス
- Well-Architected ラボ: 可視化
- AWS クラウドコストの最適化を開始する主な方法

COST01-BP06 コストをプロアクティブにモニタリングする

ツールとダッシュボードを実装して、ワークロードのコストをプロアクティブにモニタリングします。通知を受けたときだけコストやカテゴリを見るのではなく、設定されたツールや既存のツールで定期的にコストを見直しましょう。コストをプロアクティブにモニタリングし、分析することで、ポジティブな傾向を把握し、組織全体で推進することが可能になります。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

例外や異常がある場合に限らず、組織内のコストと使用量を事前にモニタリングすることを推奨します。オフィスや職場環境全体を高度に可視化したダッシュボードにより、主な担当者が必要とする情報にアクセスできるようになります。また組織がコスト最適化を重視していることを示すことができます。可視化されたダッシュボードにより、 成功した事例を積極的に推進し、組織全体で実践することができます。

日次または頻繁なルーチンを作成するには、 AWS Cost Explorer または Amazon QuickSight など、その他のダッシュボードを使用して、コストを確認し、プロアクティブに分析します。AWS サービスの使用量とコストを AWS アカウントレベル、ワークロードレベル、または特定の AWS サービスレベルでグループ化とフィルタリングを使用して分析し、予想通りかどうかを検証します。時間単位、リソース単位の詳細度やタグを使用して、上位リソースの発生コストをフィルタリングし、特

定します。また、 Cost Intelligence Dashboard という Amazon QuickSight ソリューション (AWS ソリューションアーキテクトが構築) を使用して独自のレポートを作成して、予算と実際のコストおよび使用量を比較することもできます。

実装手順

- コスト最適化に関して報告する: ワークロードの効率について話し合い、分析する定期的なミーティングを設定します。確立されたメトリクスを使用して、達成されたメトリクスとそれを達成するためにかかったコストを報告します。ネガティブな傾向を特定して修正し、ポジティブな傾向を特定して組織全体に普及させます。報告には、アプリケーションチームおよび所有者、財務、および管理の担当者が関与する必要があります。
- コストと使用量について日次の詳細度の AWS Budgets を作成し、有効にすることで、潜在的なコスト超過を防ぐためのタイムリーなアクションを取ることができます。 AWS Budgets ではアラート通知を設定できるため、いずれかの予算タイプが事前設定のしきい値を超えた場合に通知を受けることができます。AWS Budgets を活用する最善の方法は、予想されるコストと使用量を限度として設定することです。そうすれば、予算を超えたものは使い過ぎとみなすことができます。
- コストモニターとして AWS Cost Anomaly Detection を作成します。 AWS Cost Anomaly Detection は、高度な機械学習テクノロジーを使用して、異常な支出と根本原因を特定するため、迅速に対策を講じることができます。評価したい支出セグメント (例えば、個々の AWS サービス、メンバーアカウント、コスト配分タグ、コストカテゴリ) を定義するコストモニターを設定することができ、アラート通知をいつ、どこで、どのように受け取るかを設定することが可能です。各モニターには、ビジネスオーナーや テクノロジーチーム向けの複数のアラートサブスクリプションをアタッチし、各サブスクリプションの名前、コスト影響しきい値、アラート頻度 (個別アラート、日次サマリー、週次サマリー) などを設定します。
- AWS Cost Explorer を使用するか、AWS Cost and Usage Report (CUR) データを Amazon QuickSight ダッシュボードに統合して、組織のコストを視覚化します。 AWS Cost Explorer には、使いやすいインターフェイスがあり、AWS のコストと使用量を時系列で可視化し、理解し、管理することができます。最も Cost Intelligence Dashboard カスタマイズ可能でアクセスしやすいダッシュボードで、独自のコスト管理、最適化ツールの基礎作りを支援します。

リソース

関連するドキュメント:

- AWS Budgets
- AWS Cost Explorer
- 日次コストおよび使用量の予算

- AWS Cost Anomaly Detection

関連する例:

- Well-Architected ラボ: 可視化
- Well-Architected ラボ: 高度な可視化
- Well-Architected ラボ: Cloud Intelligence Dashboards
- Well-Architected ラボ: コストの可視化
- Slack による AWS Cost Anomaly Detection アラート

COST01-BP07 新しいサービスリリースに関する最新情報を把握しておく

エキスパートや AWS パートナーに定期的に相談して、コストの低いサービスと機能を検討します。AWS のブログやその他の情報ソースを確認します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

AWS は常に新しい機能を追加しているため、最新のテクノロジーを利用して、実験とイノベーションをより迅速に行うことができます。新しい AWS のサービスや機能を実装することでワークロードのコスト効率を改善できる場合があります。新しいサービスと機能のリリースに関する情報を入手するには、AWS コスト管理、AWS ニュースブログ、AWS コスト管理ブログ、および AWS の最新情報 を定期的に確認してください。「最新情報」では、AWS サービス、機能、リージョン拡大の発表があった際に、その概要をお知らせしています。

実装手順

- ブログをサブスクライブする: AWS ブログのページにアクセスし、最新情報ブログやその他の関連ブログをサブスクライブします。E メールアドレスを記載して、通信方法 ページでサインアップできます。
- AWS ニュースを購読する: 新しいサービスと機能のリリースに関する情報を入手するには、AWS ニュースブログ と AWS の最新情報 を定期的に確認してください。RSS フィード、または E メールでお知らせやリリースを購読することができます。
- AWS の値下げをフォローする: すべてのサービスにおいて定期的な値下げを行うことは、AWS がその規模から得られる経済的な効率性をお客様に還元するための標準的な方法となっています。2022 年 4 月時点で、AWS は 2006 年のサービス提供開始以来、115 回の料金引き下げを行ってきました。料金面の懸念から経営判断を保留しているものがあれば、値下げや新サービス統合

後に再度見直すことも可能です。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスも含め、以前の値下げについては、 AWS ニュースブログの料金引き下げカテゴリを参照してください.

- AWS のイベントおよび交流: ローカルの AWS サミットや、地域内の他の組織との交流に参加しましょう。直接参加できない場合は、バーチャルイベントに参加して、AWS のエキスパートや他のお客様のビジネスケースから情報を得るようにしてください。
- アカウントチームとのミーティングを設ける: アカウントチームとの定期的なミーティングを設定し、チームと会い、業界の動向と AWS のサービスについて話し合います。アカウントマネージャー、ソリューションアーキテクト、サポートチームに相談する。

リソース

関連するドキュメント:

- AWS コスト管理
- AWS の最新情報
- AWS ニュースブログ

関連する例:

- Amazon EC2 - IT コストの最適化と削減に取り組んだ 15 年間
- AWS ニュースブログ - 料金引き下げ

経費支出と使用量の認識

質問

- COST 2 使用状況はどのように管理すればよいですか?
- COST 3 使用状況とコストはどのようにモニタリングすればよいですか?
- COST 4 不要なリソースはどのように削除すればよいですか ?

COST 2 使用状況はどのように管理すればよいですか?

発生コストを適正な範囲内に抑えつつ、目的を確実に達成するためのポリシーとメカニズムを設定します。チェックアンドバランスのアプローチを採用することで、無駄なコストを費やすことなくイノベーションが可能です。

ベストプラクティス

- COST02-BP01 組織の要件に基づいてポリシーを策定する
- COST02-BP02 目標およびターゲットを策定する
- COST02-BP03 アカウント構造を実装する
- COST02-BP04 グループとロールを実装する
- COST02-BP05 コストコントロールを実装する
- COST02-BP06 プロジェクトのライフサイクルを追跡する

COST02-BP01 組織の要件に基づいてポリシーを策定する

組織のリソースの管理方法を定義するポリシーを策定します。ポリシーでは、リソースのライフサイクル全体にわたる作成、変更、削除を含む、リソースとワークロードのコスト面をカバーする必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

組織のコストおよびコスト要因を把握することは、コストと使用量を効果的に管理し、コスト削減の機会を特定するうえできわめて重要です。組織では一般に、複数のワークロードが複数のチームによってオペレーションされています。各チームはさまざまな組織単位に属する可能性があり、そのそれぞれに独自の収益の流れがあります。リソースのコストをワークロード、それぞれの組織、製品オーナーに帰属させることができると、リソースを効率的に使用し、無駄を削減できます。コストと使用量を正確にモニタリングすることで、各組織単位や製品の収益性が把握できるようになり、より確かな情報に基づいて組織内のリソース配分を決定できます。使用量が変化するとコストも変動するため、組織内のあらゆるレベルの使用量を認識することは、変化を促進する鍵となります。使用方法と支出を認識するために多面的なアプローチを取ることを検討してください。

ガバナンスを実行するための最初のステップは、組織の要件を使用して、クラウド使用に関するポリシーを作成することです。ポリシーでは、組織がクラウドをどのように使用するかや、リソースをどのように管理するかを定義します。ポリシーではコストや使用量に関係するリソースとワークロードのあらゆる局面、つまりリソースのライフタイム全体にわたる作成、変更、削除をカバーする必要があります。

ポリシーを簡単に理解し、組織全体で効果的に実装するには、シンプルなものにする必要があります。使用許可する地理的リージョンや、リソースを実行する時間帯など、幅広い高レベルのポリシーから始めます。続いてポリシーを徐々に絞り込み、さまざまな組織単位やワークロードに対応させます。一般的なポリシーの例としては、どのサービスと機能が利用できるか (例えば、テスト環境や開

発環境ではパフォーマンスが低下するストレージ)、どのタイプのリソースが各グループで使用できるか (例えば、開発用アカウントのリソースの最大サイズはミディアム) などがあります。

実装手順

- チームメンバーとのミーティングを設ける: ポリシーを開発するために、組織からすべてのチームメンバーを集め、要件を指定し、適切に文書化します。幅広く開始し、各ステップで最小単位まで継続的に絞り込んでいくという反復型アプローチを採用します。チームメンバーには、組織単位やアプリケーションの所有者など、ワークロードの直接の関係者に加え、セキュリティチームや財務チームなどのサポートグループを含みます。
- ワークロードの場所を定義する: ワークロードの運用場所 (国や国内のエリアなど) を定義します。この情報は、AWS リージョンとアベイラビリティーゾーンへのマッピングに使用されます。
- サービスとリソースを定義およびグループ化する: ワークロードに必要なサービスを定義します。サービスごとに、タイプ、サイズ、必要なリソースの数を指定します。アプリケーションサーバーやデータベースストレージなどの機能別にリソースのグループを定義します。リソースは複数のグループに属することができます。
- 機能別にユーザーを定義およびグループ化する: ワークロードに関係するユーザーについて、当該ユーザーが誰かまたは組織内での地位に焦点を当てるのではなく、何を行うか、またはどのようにワークロードを使用するかに焦点を当てて定義します。類似したユーザーまたは機能をグループ化します。AWS 管理ポリシーをガイドとして使用できます。
- アクションを定義する: 以前に特定した場所、リソース、およびユーザーを使用して、そのライフタイム (開発、運用、削除) にわたってワークロードの成果を得るために、それぞれが必要とするアクションを定義します。各場所で、グループ内の個々の要素ではなく、グループに基づいてアクションを特定します。開始時には読み取りまたは書き込みを幅広く設定し、それぞれのサービスについて、特定のアクションへと絞り込んでいきます。
- レビュー期間を定義する: ワークロードと組織の要件は、時間の経過とともに変化する可能性があります。ワークロードのレビュースケジュールを定義して、組織の優先順位に合わせた状態を維持します。
- ポリシーを文書化する: 定義されたポリシーが、組織の必要に応じてアクセス可能であることを確認します。これらのポリシーは、環境へのアクセスを実装、保守、監査するために使用されます。

リソース

関連するドキュメント:

- AWS Managed Policies for Job Functions

- AWS 複数アカウントの請求戦略
- Actions, Resources, and Condition Keys for AWS Services
- クラウド製品
- Control access to AWS リージョン using IAM policies
- グローバルインフラストラクチャリージョンと AZ

COST02-BP02 目標およびターゲットを策定する

ワークロードのコストおよび使用量の両方について、目標を策定します。目標はコストと使用状況について組織に方向性を提供し、ターゲットはワークロードについての測定可能な結果を提供します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

組織のコスト、目標使用量、ターゲットを設定します。目標は、期待される成果に関するガイダンスと指示を組織にもたらします。ターゲットは、具体的かつ測定可能な達成すべき結果をもたらします。目標の一例: プラットフォームの使用量を大幅に増加させ、コストは微増 (非線形) にとどまるようにする。ターゲットの一例:プラットフォームの使用率を 20% 増加させ、コスト増は 5% 未満。もう 1 つの一般的な目標となるのは、ワークロードを 6 か月ごとにより効率的にするという必要性です。この種のターゲットとして、ワークロードの結果あたりのコストを 6 か月ごとに 5% 削減する必要があるというケースも考えられます。

クラウドワークロードの一般的な目標は、ワークロードの効率を高めることです。これは、ワークロードのビジネス成果あたりのコストを経時的に削減することです。この目標と合わせて、6～12 か月ごとに効率を 5% 向上させるなどのターゲットをすべてのワークロードに設定することを推奨します。これは、クラウド内でコスト最適化の機能を構築し、新しいサービスやサービス機能のリリースを行うことで達成できます。

実装手順

- 予想される使用レベルを定義する: まず、使用レベルに焦点を当てます。アプリケーションの所有者、マーケティング、およびより大きなビジネスチームと協力して、ワークロードに対して予想される使用レベルを把握します。顧客の需要が時間の経過とともにどのように変化するか、季節的な増加やマーケティングキャンペーンによって変化が生じるかどうか、などを考慮します。
- ワークロードのリソースとコストを定義する: 使用レベルを定義した上で、これらの使用レベルを満たすために必要なワークロードリソースの変化を数値化します。ワークロードコンポーネントのサイズまたはリソースの数を増やしたり、データ転送を増やしたり、特定のレベルでワークロード

コンポーネントを別のサービスに変更したりすることが必要な場合があります。これらの主な各ポイントにおけるコストと、使用状況が変更された場合のコストの変化を指定します。

- ビジネス目標を定義する: 予想される使用量とコストの変化から結果を取得し、これを、予想されるテクノロジーや実行中のプログラムの変化と組み合わせて、ワークロードの目標を策定します。目標は、使用状況、コスト、および両者の関係を対象に含めたものである必要があります。コストは変化するが使用状況に変化がないことが予想される場合は、トレーニングや教育などの機能構築などの組織プログラムが存在していることを確認します。
- ターゲットを定義する: 定義された目標ごとに、測定可能なターゲットを指定します。目標がワークロードの効率を高めることである場合、ターゲットは、改善 (通常は 1 ドルあたりのビジネス出力量の改善) の量と、改善がいつ実現されるかを数値化します。

リソース

関連するドキュメント:

- AWS ジョブ機能の管理ポリシー
- AWS 複数アカウントの請求戦略
- Control access to AWS リージョン using IAM policies

COST02-BP03 アカウント構造を実装する

組織にマッピングされるアカウントの構造を実装します。これは組織全体でのコストの割り当てと管理に役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS は 1 つの親アカウントと複数の子アカウントからなる構造です。このアカウント構造は、一般に管理 (親、旧称は支払者) アカウント、メンバー (子、旧称はリンク) アカウントと呼ばれます。ベストプラクティスは、組織の規模や使用量にかかわらず、1 つのメンバーアカウントを持つマスターを少なくとも常に 1 つ持つことです。すべてのワークロードリソースが存在するのは、メンバーアカウント内のみとする必要があります。

AWS アカウントの最適な数は状況に応じて異なります。現在と将来の運用モデルとコストモデルを見積もり、AWS アカウントの構造が組織の目標を反映するようにします。ビジネス上の理由から複数の AWS アカウントを作成する企業もあります。次に例を示します。

- 組織単位、コストセンター、特定のワークロード間で、管理、会計、請求の職務機能を切り離す必要がある場合。
- AWS のサービスの制限が特定のワークロードのみに設定される場合。
- ワークロードとリソース間の隔離と分離には要件があります。

では AWS Organizationsio1一括請求 (コンソリデーティッドビリング) により、1 つ以上のメンバーアカウントと管理アカウントとの間に構造が作成されます。メンバーアカウントを使用すると、コストと使用量をグループ別に区別できます。一般的には、各組織単位 (財務、マーケティング、営業など)、各環境ライフサイクル (開発、テスト、本番など)、各ワークロード (ワークロード a、b、c) にメンバーアカウントをいったん分離したうえで、一括請求 (コンソリデーティッドビリング) を使用してこれらのアカウントを集約します。

一括請求 (コンソリデーティッドビリング) 機能を使用すると、複数のメンバー AWS アカウントの支払いを単一の管理アカウントにまとめつつ、各リンクアカウントのアクティビティの可視性も維持できます。コストと使用量が管理アカウントに集計されると、サービスの従量制割引とコミットメント割引 (Savings Plans とリザーブドインスタンス) を最大限に活用し、割引額を最大化できます。

AWS Control Tower では、複数の AWS アカウントのセットアップと構成をすばやく行い、ガバナンスが組織の要件に適合していることを確認できます。

実装手順

- 分離要件を定義する: 分離の要件は、セキュリティ、信頼性、財務構造など、複数の要因の組み合わせです。各要因を順番に確認し、ワークロードまたはワークロード環境を他のワークロードから分離するかどうかを指定します。セキュリティは、アクセスとデータ要件を確実に守るものです。信頼性は、環境やワークロードが他の環境に影響を与えないように制限を確実に管理するものです。財務構造は、厳格な財務分離と説明責任を確保するものです。分離の一般的な例としては、本番稼働用ワークロードとテストワークロードが別々のアカウントで実行されることや、請求書と請求データをサードパーティー組織に提供できるように別のアカウントを使用することが挙げられます。
- グループ化要件を定義する: グループ化要件は分離要件を上書きしませんが、管理を支援するために使用されます。分離を必要としない同様の環境またはワークロードをグループ化します。この例としては、1 つ以上のワークロードから複数のテスト環境または開発環境をグループ化することが挙げられます。
- アカウント構造を定義する: これらの分離およびグループ化を使用して、各グループのアカウントを指定し、分離要件が維持されるようにします。これらのアカウントは、メンバーアカウントまたは連結アカウントです。これらのメンバーアカウントを単一の管理アカウントまたは支払者ア

カウントでグループ化することで、使用量が合算されるので、すべてのアカウントでの従量制割引がより大きくなり、すべてのアカウントに対して単一の請求書が提供されます。請求データを分離し、各メンバーアカウントに請求データの個別のビューを提供することができます。メンバーアカウントが使用量や請求データを他のアカウントに表示してはならない場合、または AWS から別々の請求書を必要とする場合は、複数の管理アカウントまたは支払者アカウントを定義します。この場合、各メンバーアカウントは独自の管理アカウントまたは支払者アカウントを持つことになります。リソースは常にメンバーアカウントまたは連結アカウントに配置する必要があります。管理アカウントまたは支払者アカウントは、管理のためにのみ使用してください。

リソース

関連するドキュメント:

- AWS ジョブ機能の管理ポリシー
- AWS 複数アカウントの請求戦略
- Control access to AWS リージョン using IAM policies
- AWS Control Tower
- AWS Organizations
- 一括請求

関連する例:

- CUR の分割とアクセスの共有

COST02-BP04 グループとロールを実装する

ポリシーに沿ったグループおよびロールを実装し、各グループのインスタンスおよびリソースを作成、変更、削除できるユーザーを管理します。たとえば、開発、テスト、本番グループを実装します。これは、AWS のサービスやサードパーティーのソリューションに適用されます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ポリシーを作成すると、組織内のユーザーの論理グループとロールを作成できます。これにより、アクセス許可の割り当てと使用量の制御が可能になります。人材のおおまかなグループ化から始めます。通常これは、組織単位と役職 (IT 部門のシステム管理者、会計監査担当者など) と合致します。

グループとは、類似したタスクに従事し、類似したアクセスを必要とするユーザーの集団を指します。ロールとは、グループとして義務付けられた仕事の定義を指します。たとえば、IT のシステム管理者はすべてのリソースを作成するためのアクセスが必要ですが、分析チームのメンバーは分析リソースを作成するアクセスのみで十分です。

実装手順

- グループを実装する: 必要に応じて、組織のポリシーで定義されているユーザーのグループを使用して、対応するグループを実装します。ユーザー、グループ、および認証のベストプラクティスについては、セキュリティの柱を参照してください。
- ロールとポリシーを実装する: 組織のポリシーで定義されているアクションを使用して、必要なロールとアクセスポリシーを作成します。ロールとポリシーのベストプラクティスについては、セキュリティの柱を参照してください。

リソース

関連するドキュメント:

- AWS ジョブ機能の管理ポリシー
- AWS 複数アカウントの請求戦略
- Control access to AWS リージョン using IAM policies
- Well-Architected セキュリティの柱

関連する例:

- Well-Architected ラボ: 基本的なアイデンティティとアクセス

COST02-BP05 コストコントロールを実装する

組織のポリシーと定義済みのグループおよびロールに基づいてコントロールを実装します。これにより、組織の要件で定義されているとおりにコストが発生することが保証されます。例えば、AWS Identity and Access Management (IAM) ポリシーでリージョンまたはリソースタイプへのアクセスをコントロールできます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

コストコントロールを実装する際の一般的な最初のステップは、ポリシー外のコストまたは使用量イベントが発生した場合に通知するように設定することです。これにより、ワークロードや新しいアクティビティを制限したり悪影響を与えたりすることなく、迅速に行動し、是正措置の必要性の有無を確認できます。ワークロードと環境の制限を理解したら、ガバナンスを適用できます。AWS では、通知を実行する AWS Budgets により、AWS のコスト、使用量、コミットメント割引 (Savings Plans とリザーブドインスタンス) の月次予算を定義できます。予算は、集計コストのレベル (たとえば、全コスト)、またはリンクアカウント、サービス、タグ、アベイラビリティーゾーンなどの特定のディメンションのみを含む詳細レベルで作成できます。

次のステップとして、AWS では AWS Identity and Access Management (IAM) と AWS Organizations サービスコントロールポリシー (SCP)を介して、ガバナンスポリシーを適用できます。IAM により、AWS のサービスとリソースへのアクセスを安全に管理できます。IAM を使用すると、AWS のリソースを作成および管理できるユーザー、作成できるリソースのタイプ、リソースを作成できる場所を制御できます。これにより、不要なリソースの作成が最小限に抑えられます。以前に作成したロールとグループを使用して IAM ポリシーを割り当て、 適切な使用量を適用します。SCP は組織内のすべてのアカウントで利用可能なアクセス許可の上限を一元的に制御し、アカウントがアクセスコントロールのガイドライン内に収まるようにします。SCP はすべての機能が有効になっている組織でのみ使用可能で、デフォルトでメンバーアカウントのアクションの可否を SCP を設定できます。アクセス管理の実装の詳細については、 Well-Architected セキュリティの柱のホワイトペーパー を参照してください。

Service Quotas を管理することで、ガバナンスを導入することもできます。サービスクォータを最小オーバーヘッドに設定し、正確に維持するよう徹底することで、組織に不要なリソースの作成を最小限に抑えることができます。これを実現するには、要件がどれだけ速く変化するかを理解し、進行中のプロジェクト (リソースの作成と削除の両方) を理解し、クォータ変更をどれだけすばやく実装できるかを考慮する必要があります。Service Quotas を使用して、必要に応じてクォータを増加させることができます。

実装手順

- 支出に関する通知を実装する: 定義した組織のポリシーを使用して、AWS Budgets を作成し、支出がポリシーを外れた場合に通知を提供するようにします。アカウントごとに 1 つずつ、複数のコスト予算を設定し、アカウントの支出全体について通知するようにします。次に、アカウント内のより小さな単位について、各アカウント内にコスト予算を追加で設定します。これらの単位は、アカウント構造によって異なります。一般的な例としては、AWS リージョン、ワークロード (タグを使用)、または AWS のサービスがあります。E メール配信リストが通知の受取人として設定

されており、また、個人の E メールアカウントではないことを確認します。金額を超えたときの実際の予算を設定するか、予測された使用量が通知されたときの予測された予算を使用します。

- 使用量のコントロールを実装する: 定義した組織のポリシーを使用して、IAM ポリシーとロールを実装し、ユーザーが実行できるアクションと実行できないアクションを指定します。AWS ポリシーには、複数の組織ポリシーを含めることができます。ポリシーを定義するのと同じ方法で、幅広く開始し、各ステップでより詳細なコントロールを適用します。サービスの制限も、使用量に対する効果的なコントロールです。すべてのアカウントに正しいサービス制限を実装します。

リソース

関連するドキュメント:

- AWS ジョブ機能の管理ポリシー
- AWS 複数アカウントの請求戦略
- Control access to AWS リージョン using IAM policies

関連する例:

- Well-Architected ラボ: コストと使用量のガバナンス
- Well-Architected ラボ: コストと使用量のガバナンス

COST02-BP06 プロジェクトのライフサイクルを追跡する

プロジェクト、チーム、環境のライフサイクルを追跡、計測、監査して、不要なリソースの使用やそれに伴う支払いを回避できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ワークロードのライフサイクル全体を確実に追跡します。これにより、ワークロードやワークロードコンポーネントが不要になった場合、削除や変更が可能になります。これは、新しいサービスや機能をリリースする際に特に便利です。既存のワークロードとコンポーネントは使用されているように見えても、顧客を新しいサービスにリダイレクトするために使用を停止する必要があります。ワークロードの以前のステージに注目してください。ワークロードが本番稼働状態になったら、以前の環境は廃棄することと、再び必要になるまでキャパシティーを大幅に削減することも可能です。

AWS には、エンティティのライフサイクル追跡に使用できる管理およびガバナンスサービスが多数用意されています。専用のインフラストラクチャで AWS Config または AWS Systems Manager を使用すると、AWS リソースと設定の詳細なインベントリが入手可能です。プロジェクトやアセットを管理する既存のシステムを統合して、組織内のアクティブなプロジェクトや製品を追跡することを推奨します。現在のシステムを AWS が提供する豊富なイベントやメトリクスと組み合わせることにより、重要なライフサイクルイベントのビューを作成し、前もってリソースを管理し、不要なコストを削減できます。

ウェブアプリケーションのバックエンドに関する推奨事項については、 Well-Architected 運用上の優秀性の柱のホワイトペーパー を参照してください。

実装手順

- ワークロードの確認を実行する: 組織のポリシーで定義されているところに従って、既存のプロジェクトを監査します。監査に費やされる労力の量は、組織のおおよそのリスク、価値、またはコストに比例する必要があります。監査に含めるべき主な領域は、インシデントまたは機能停止の組織に対するリスク、価値、組織への寄与 (収益またはブランドに対する評価で測定)、ワークロードのコスト (リソースおよび運用の合計コストとして測定)、およびワークロードの使用量 (時間単位ごとの組織の成果の数で測定) です。これらの領域がライフサイクルを通じて変化する場合、完全または部分的な削除など、ワークロードの調整が必要です。

リソース

関連するドキュメント:

- AWS Config
- AWS Systems Manager
- AWS managed policies for job functions
- AWS 複数アカウントの請求戦略
- Control access to AWS リージョン using IAM policies

COST 3 使用状況とコストはどのようにモニタリングすればよいですか?

コストをモニタリングし、適切に配分するためのポリシー手順を定めます。これにより、ワークロードのコスト効率を測定し、向上させることができます。

ベストプラクティス

- COST03-BP01 詳細情報ソースを設定する
- COST03-BP02 コスト属性カテゴリを特定する
- COST03-BP03 組織のメトリクスを確立する
- COST03-BP04 請求およびコスト管理ツールを設定する
- COST03-BP05 コストと使用状況に組織情報を追加する
- COST03-BP06 ワークロードメトリクスに基づいてコストを配分する

COST03-BP01 詳細情報ソースを設定する

AWS のコストと使用状況レポートおよび Cost Explorer の時間単位の粒度を設定し、コストと使用状況の詳細情報を提供します。ワークロードが、もたらされるすべてのビジネス成果のログエントリを持つように設定します。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

AWS Cost Explorerで時間単位の粒度を有効にし、 AWS Cost and Usage Report (CUR)を作成します。これらのデータソースは、組織全体のコストと使用量の最も正確なビューを提供します。CUR では、課金されるすべての AWS のサービスについて、日単位または時間単位の使用量の粒度、料金、コスト、使用属性が提供されます。CUR には、タグ付け、場所、リソース属性、アカウント ID など想定可能なすべてのディメンションがあります。

以下のカスタマイズ項目で CUR を設定します。

- リソース ID を含める
- CUR を自動更新する
- 時間単位の詳細
- バージョニング: 既存のレポートを上書きする
- データ統合: Amazon Athena (Parquet 形式、圧縮)

予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS Glue を使用して分析用のデータを準備し、 Amazon Athena を使用して、データ分析を実行し、SQL を使用してデータをクエリします。また、 Amazon QuickSight を使用して、カスタムの可視化や複雑な可視化を行い、組織全体に配布することもできます。

実装手順

- コストと使用状況レポートを設定する: 請求コンソールを使用して、少なくとも 1 つのコストと使用状況レポートを設定します。すべての識別子とリソース ID を含む時間単位の粒度でレポートを設定します。粒度が異なる他のレポートを作成して、概要情報を提供することもできます。
- Cost Explorer で時間単位の粒度を設定する: 請求コンソールを使用して、[時間およびリソースレベルのデータ] を有効にします。

> **Note**
>
> この機能を有効にすると費用が発生します。詳細については、料金を参照してください。

- アプリケーションログ記録を設定する: アプリケーションがもたらすビジネスの各成果が ログに記録され、追跡および測定が可能であることを確認します。このデータの粒度が少なくとも 1 時間単位であることを確認し、コストと使用状況のデータと一致するようにします。ログ記録とモニタリングの詳細については、「 Well-Architected 運用上の優秀性の柱 」を参照してください。

リソース

関連するドキュメント:

- AWS アカウント設定
- AWS Cost and Usage Report (CUR)
- AWS Glue
- Amazon QuickSight
- AWS コスト管理の料金
- AWS リソースのタグ付け
- AWS Budgets を用いてコストを分析する
- Cost Explorer を用いてコストを分析する
- Managing AWS Cost and Usage Reports (AWS コストと使用状況レポートの管理)
- Well-Architected 運用上の優秀性の柱

関連する例:

- AWS アカウント設定

COST03-BP02 コスト属性カテゴリを特定する

組織内でのコストの配分に使用可能な組織カテゴリを特定します。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

財務チームやその他の関係者と協力し、組織内でコストを配分する方法の要件を理解します。ワークロードのコストは、開発、テスト、本稼働、廃止などライフサイクル全体にわたって配分する必要があります。学習、スタッフ育成、アイデア創出に要したコストが、どのように組織に帰属するかを理解します。この目的で使用される金額を一般的な IT コスト予算ではなく、トレーニング予算や開発の予算に正しく割り当てるのに便利です。

実装手順

- 組織カテゴリを定義する: ステークホルダーとミーティングをして、組織の構造と要件を反映するカテゴリを定義します。これらは、ビジネス単位、予算、コストセンター、部門など、既存の財務カテゴリの構造に直接マッピングされます。トレーニングや教育など、クラウドがもたらすビジネスの成果を確認します。これらは組織のカテゴリでもあります。複数のカテゴリをリソースに割り当てることができます。また、リソースは複数の異なるカテゴリに存在することができるため、必要な数のカテゴリを定義します。
- 機能カテゴリを定義する: 利害関係者とミーティングをして、ビジネス内の機能を反映するカテゴリを定義します。これは、ワークロードまたはアプリケーション名、および実稼働、テスト、開発などの環境のタイプである場合があります。複数のカテゴリをリソースに割り当てることができます。また、リソースは複数の異なるカテゴリに存在することができるため、必要な数のカテゴリを定義します。

リソース

関連するドキュメント:

- AWS リソースのタグ付け
- AWS Budgets を用いてコストを分析する
- Cost Explorer を用いてコストを分析する
- Managing AWS Cost and Usage Reports (AWS コストと使用状況レポートの管理)

COST03-BP03 組織のメトリクスを確立する

このワークロード用のメトリクスを組織内で定めます。ワークロードのメトリクスの例として、作成された顧客レポートや顧客に提供されるウェブページが挙げられます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードのアウトプットがビジネスの成功に対してどのように測定されるかを理解します。通常、各ワークロードには、パフォーマンスを示す主な成果の小さな組み合わせがあります。多数のコンポーネントを含む高度なワークロードがある場合は、リストに優先順位を付けるか、各コンポーネントのメトリクスを定義して追跡できます。チームと協力して、どのメトリクスを使用するか理解します。この単位は、ワークロードの効率または各ビジネス成果のコストを把握するために使用されます。

実装手順

- ワークロードの成果を定義する: ビジネスの利害関係者とミーティングをして、ワークロードの成果を定義します。これらは顧客の使用状況の主要な測定指標であり、技術的メトリクスではなく、ビジネスメトリクスである必要があります。ワークロードごとに少数の概要的なメトリクス (5 つ未満) が存在する必要があります。ワークロードが異なるユースケースで複数の成果を生成する場合は、それらを単一のメトリクスにグループ化してください。
- ワークロードコンポーネントの成果を定義する: 必要に応じて、大規模で複雑なワークロードがある場合、または明確に定義された入出力を使用してワークロードをコンポーネント (マイクロサービスなど) に簡単に分割できる場合は、各コンポーネントのメトリクスを定義します。この作業では、コンポーネントの価値とコストを反映する必要があります。最大のコンポーネントから開始し、大きさの順番で、最小のコンポーネントまで作業します。

リソース

関連するドキュメント:

- AWS リソースのタグ付け
- Analyzing your costs with AWS Budgets
- Cost Explorer によるコストの分析
- Managing AWS Cost and Usage Reports

COST03-BP04 請求およびコスト管理ツールを設定する

AWS Cost Explorer と AWS Budgets を組織のポリシーに沿って設定します。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

使用量を変更してコストを調整するには、組織内の各ユーザーがそれぞれのコストと使用量の情報にアクセスできる必要があります。クラウドを使用する場合、すべてのワークロードとチームに次のツールを設定することを推奨します。

- レポート: すべてのコストと使用情報を要約する。
- 通知: コストまたは使用量が設定された制限を超えた場合に通知する。
- 現在の状態: 現在のコストと使用量を示すダッシュボードを設定する。ダッシュボードはオペレーションダッシュボードと同様に、作業環境内の目に付きやすい場所で使用できるようにする必要があります。
- 傾向: 要求した期間におけるコストと使用量の変動を、必要な詳細度で示す。
- 予測: 将来の推定コストを示す。
- 追跡: 設定された目標またはターゲットに対する現在のコストと使用量を表示する。
- 分析: チームメンバーが、すべての可能なディメンションを使用して、時間単位の詳細度までカスタムおよび詳細な分析を実行する機能を提供する。

AWS のネイティブツール (AWS Cost Explorer、 AWS Budgets、および Amazon Athena と Amazon QuickSight など) を使用して、この機能を提供できます。サードパーティー製のツールを使用することもできますが、このツールのコストが組織に価値をもたらすことを確認する必要があります。

実装手順

- コスト最適化グループを作成する: アカウントを設定し、必要なコストと使用状況レポートにアクセスできるグループを作成します。このグループには、アプリケーションを所有または管理するすべてのチームの代表者を含める必要があります。これにより、すべてのチームがコストと使用情報にアクセスできることが保証されます。
- AWS Budgets を設定する: ワークロードのすべてのアカウントで AWS Budgets を設定します。タグを使用して、アカウント全体の支出の予算とワークロードの予算を設定します。

- AWS Cost Explorer を設定する: ワークロードとアカウントで AWS Cost Explorer を設定します。全体的な支出を追跡するワークロードのダッシュボードと、ワークロードの主要な使用状況メトリクスを作成します。
- 高度なツールを設定する: 必要に応じて、さらなる詳細と粒度を提供する組織用のカスタムツールを作成できます。高度な分析機能を実装するには Amazon Athenaを使用し、ダッシュボードを実装するには Amazon QuickSightを使用します。

リソース

関連するドキュメント:

- AWS リソースのタグ付け
- AWS Budgets を用いてコストを分析する
- Cost Explorer を用いてコストを分析する
- Managing AWS Cost and Usage Reports (AWS コストと使用状況レポートの管理)

関連する例:

- Well-Architected ラボ - AWS アカウント設定
- Well-Architected ラボ: 請求の可視化
- Well-Architected ラボ: コストと使用に関するガバナンス
- Well-Architected ラボ: コストと使用状況の分析
- Well-Architected ラボ: コストと使用状況の可視化

COST03-BP05 コストと使用状況に組織情報を追加する

組織、ワークロード属性、コスト配分カテゴリに基づいてタグ付けスキーマを定義します。すべてのリソースにタグを付けます。Cost Categories を使用して、組織の属性に従ってコストと使用状況をグループ化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

タグ付けを AWS で実装し、 リソースに組織の情報を追加します。追加した情報は、コストと使用状況の情報に追加されます。タグはキーと値のペアです。キーは定義されており、組織全体で一意である必要があります。値はリソースのグループに対して一意です。キーと値のペアの一例としては、

キーが Environment で、値は Production となります。本稼働環境のすべてのリソースには、キーと値のペアがあります。タグ付けにより、関連性の高い組織情報を使用して、コストを分類、追跡できます。組織のカテゴリ (コストセンター、アプリケーション名、プロジェクト、オーナーなど) を表すタグを適用し、ワークロードやワークロードの特性 (テストや本番など) を識別して、組織全体のコストと使用状況の帰属先を付与できます。

AWS リソース (Amazon Elastic Compute Cloud インスタンスや Amazon Simple Storage Service バケットなど) にタグを付け、そのタグをアクティブ化すると、AWS はこの情報をコストと使用状況レポートに追加します。タグ付けされたリソースとタグ付けされていないリソースに対してレポートを実行し、分析を実行することで、内部のコスト管理ポリシーへの準拠を強化し、正確な帰属を保証できます。

組織のアカウント全体に AWS タグ付け基準を作成、導入することで、AWS 環境を一貫性のある統一された方法で管理することができます。タグ使用のルールを定義するには、タグポリシー を AWS Organizations で使用します。このルールは、AWS Organizations のアカウントの AWS リソースに対してタグをどのように使用できるかを定めたものです。タグポリシーを使用すると、AWS リソースにタグを付ける標準アプローチを簡単に導入できます。

AWS Tag Editor では、複数のリソースのタグを追加、削除、管理できます。

AWS Cost Categories を使用すると、リソースにタグを付けることなく組織としての意味をコストに割り当てることができます。コストと使用量に関する情報を、一意の内部組織構造にマッピングできます。アカウントやタグなどの請求ディメンションを使用して、コストをマッピングおよび分類するカテゴリルールを定義します。これにより、タグ付けに加えて、別のレベルの管理機能が提供されます。また、特定のアカウントとタグを複数のプロジェクトにマッピングすることもできます。

実装手順

- タグ付けスキーマを定義する: すべての利害関係者をビジネス全体から集めて、スキーマを定義します。これには通常、技術、財務、および管理ロールの担当者が含まれます。すべてのリソースに必要なタグのリストと、リソースに必要なタグのリストを定義します。タグの名前と値が組織全体で一貫していることを確認します。
- リソースにタグを付ける: 定義したコスト帰属カテゴリを使用して、カテゴリに従ってワークロードのすべてのリソースにタグを付けます。CLI、Tag Editor、Systems Manager などのツールを使用して、効率を向上させます。
- Cost Categories を実装する: タグ付けを実装せずに Cost Categories を作成できます。Cost Categories は既存のコストと使用状況ディメンションを使用します。スキーマからカテゴリルールを作成し、それを Cost Categories に実装します。

- タグ付けを自動化する: すべてのリソースにわたってタグ付けの高いレベルを維持していることを確認するには、タグ付けを自動化して、リソースの作成時に自動的にタグ付けされるようにします。サービス内の機能、または AWS CloudFormation などのサービスを使用して、リソースの作成時にタグ付けされるようにします。ワークロードを定期的にスキャンし、タグ付けされていないリソースをすべて削除するカスタムマイクロサービスを作成することもできます。これは、テスト環境および開発環境に最適です。
- タグ付けをモニタリングおよびレポートする: 組織全体でタグ付けの高いレベルを維持していることを確認するには、ワークロード全体でタグをレポートおよびモニタリングします。AWS Cost Explorer を使用して、タグ付けされたリソースとタグ付けされていないリソースのコストを表示したり、Tag Editor などのサービスを使用したりできます。タグ付けされていないリソースの数を定期的に確認し、必要なレベルのタグ付けになるまでタグを追加するアクションを実行します。

リソース

関連するドキュメント:

- AWS CloudFormation Resource Tag
- AWS Cost Categories
- AWS リソースのタグ付け
- Amazon EC2 および Amazon EBS でリソース作成時のタグ付けのサポートを追加
- AWS Budgets を用いてコストを管理する
- Cost Explorer によるコストの分析
- Managing AWS Cost and Usage Reports

COST03-BP06 ワークロードメトリクスに基づいてコストを配分する

メトリクスや業績に基づいてワークロードのコストを配分し、ワークロードのコスト効率を測定します。AWS のコストと使用状況レポートを分析するプロセスを実装します。これには、洞察力とチャージバック機能を提供する Amazon Athenaを使用します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

コスト最適化によって、最低価格でビジネス成果が提供されます。これは、ワークロードメトリクス (ワークロードの効率で測定) ごとにワークロードのコストを配分することによってのみ達成でき

ます。定義されたワークロードメトリクスを、ログファイルまたは他のアプリケーションのモニタリングを使用してモニタリングします。このデータをワークロードのコストと組み合わせます。ワークロードのコストは、特定のタグ値またはアカウント ID でコストを確認することで取得できます。この分析は時間単位で実行することをお勧めします。リクエストレートが変化する静的なコストコンポーネント (24 時間年中無休で実行されるバックエンドデータベースなど) がある場合、通常、効率性は変化します (たとえば、使用量のピークは午前 9 時から午後 5 時で、夜間のリクエストはほとんどありません)。静的コストと変動コストの関係を理解しておくと、最適化のアクティビティに集中する一助となります。

実装手順

- ワークロードメトリクスにコストを割り当てる: 定義されたメトリクスと設定されたタグ付けを使用して、ワークロードの結果とワークロードのコストを組み合わせたメトリクスを作成します。Amazon Athena や Amazon QuickSight などの分析サービスを使用して、ワークロード全体や任意のコンポーネントに対する効率性ダッシュボードを作成します。

リソース

関連するドキュメント:

- AWS リソースのタグ付け
- AWS Budgets を用いてコストを分析する
- Cost Explorer を用いてコストを分析する
- Managing AWS Cost and Usage Reports (AWS コストと使用状況レポートの管理)

COST 4 不要なリソースはどのように削除すればよいですか？

プロジェクトの開始から終了まで変更管理とリソース管理を実装します。これにより、使用されていないリソースをシャットダウンまたは終了して、無駄を減らします。

ベストプラクティス

- COST04-BP01 ライフタイム全体にわたってリソースを追跡する
- COST04-BP02 削除プロセスを実装する
- COST04-BP03 リソースを削除する
- COST04-BP04 自動的にリソースを削除する

COST04-BP01 ライフタイム全体にわたってリソースを追跡する

ライフタイム全体にわたって、リソースや、リソースとシステムとの関係を追跡するメソッドを定義し、実装します。タグ付けにより、リソースのワークロードまたは機能を特定できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

不要になったワークロードリソースを廃止します。一般的な例としては、テスト用途のリソースがあります。テストが完了したら、リソースは削除できます。タグでリソースを追跡する (およびそのタグのレポートを作成する) ことは、廃棄するアセットの特定に役立ちます。リソース追跡には、タグの使用が効果的な方法です。リソースにその機能か、または廃止可能になる既知の日付をラベリングできます。そうすると、これらのタグでレポートを作成できます。機能タグを付ける場合の一例として、 `feature-X testing` という値であれば、ワークロードのライフサイクルの観点からリソースの目的を識別できます。

実装手順

- タグ付けスキームを実装する: リソースが属するワークロードを識別するタグ付けスキームを実装し、ワークロード内のすべてのリソースが適切にタグ付けされることを確認します。
- ワークロードのスループットまたは出力モニタリングを実装する: ワークロードのスループットのモニタリングまたはアラームを実装し、入力リクエストまたは出力の完了時にトリガーします。ワークロードのリクエストまたは出力がゼロになり、ワークロードのリソースが使用されなくなったことを示す通知を提供するように設定します。ワークロードが通常の条件下で定期的にゼロに低下する場合は、時間要因を組み込みます。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS リソースのタグ付け
- カスタムメトリクスの発行

COST04-BP02 削除プロセスを実装する

孤立したリソースを特定して削除するためのプロセスを実装します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

組織全体の標準化プロセスで、使用されていないリソースを特定して廃棄します。検索の実行頻度および組織のすべての要件を確実に満たすために、このプロセスではリソースを削除するプロセスを定義する必要があります。

実装手順

- 削除プロセスを作成して実装する: ワークロードの開発者や所有者と協力して、ワークロードとそのリソースの削除プロセスを構築します。このプロセスでは、ワークロードが使用中であるかどうか、および各ワークロードリソースが使用中であるかどうかを確認する方法を採り入れる必要があります。このプロセスでは、リソースを削除するために必要なステップを採り入れ、サービスから削除すると同時に、規制要件の遵守を確保する必要もあります。ライセンスやアタッチされたストレージなど、関連するリソースも対象となります。このプロセスでは、削除プロセスが実行されたことをワークロードの所有者に通知する必要があります。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor

COST04-BP03 リソースを削除する

定期監査や使用状況の変化などのイベントを契機として、リソースを削除します。通常、削除は定期的に行われ、手動または自動で行われます。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

使用していないリソースを検索する場合は節減額の程度によって検索頻度と投入する労力を決定する必要があるため、コスト発生額の小さいアカウントの分析は、コスト発生額が高額のアカウントよりも頻度を下げるべきです。イベントの検索および廃止は、製品が寿命を迎えた場合や交換する場合など、ワークロードの状態の変化によってトリガーされます。イベントの検索および廃止は、市況の変化や製品終了などの外部イベントによってトリガーされる場合もあります。

実装手順

- リソースを削除する: 削除プロセスを使用して、孤立したと識別された各リソースを削除します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor

COST04-BP04 自動的にリソースを削除する

重要度が低いリソース、不要なリソース、使用率が低いリソースを特定して削除する作業を適切に行えるようにワークロードを設計します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

オートメーションを使用して、廃棄プロセスの関連コストを削減または削除します。自動削除するようにワークロードを設計すると、そのライフタイム全体にわたるワークロードコストを削減できます。AWS リソースと設定の詳細なインベントリは、 AWS Auto Scaling を使用して実行します。また、APIまたはSDKを使用して カスタムコードを実装し、 ワークロードリソースを自動的に削除することもできます。

実装手順

- AWS Auto Scaling を実装する: サポートされているリソースについては、AWS Auto Scaling で設定します。
- CloudWatch を設定して、インスタンスを終了させる: インスタンスは、CloudWatch アラームを使用して終了するように設定できます。削除プロセスのメトリクスを使用して、Amazon Elastic Compute Cloud (Amazon EC2) アクションでアラームを実装します。ロールアウトする前に、非本番環境でオペレーションを検証します。
- ワークロード内にコードを実装する: AWS SDK または AWS CLI を使用して、ワークロードリソースを削除できます。AWS と統合されるアプリケーション内にコードを実装し、使用されなくなったリソースを終了または削除します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- インスタンスを停止、終了、再起動、または復旧するアラームを作成する
- Getting Started With Amazon EC2 Auto Scaling

費用対効果の高いリソース

質問

- COST 5 サービスを選択する際、どのようにコストを評価すればよいですか?
- COST 6 リソースタイプ、リソースサイズ、およびリソース数を選択する際、コスト目標を達成するにはどうすればよいですか?
- COST 7 コスト削減のために、どのように料金モデルを使用すればよいですか?
- COST 8 データ転送料金はどのように計画すればよいですか ?

COST 5 サービスを選択する際、どのようにコストを評価すればよいですか?

Amazon EC2、Amazon EBS、Amazon S3 は、基盤となる AWS のサービスです。Amazon RDS や Amazon DynamoDB などのマネージドサービスは 、より高レベル、つまりアプリケーションレベルの AWS のサービスです。基盤となるサービスやマネージドサービスを適切に選択することで、このワークロードのコストを最適化できます。例えば、マネージドサービスを使用することで、管理や運用によって発生するオーバーヘッドを削減またはゼロにでき、アプリケーション開発やビジネス上の他の活動に注力できるようになります。

ベストプラクティス

- COST05-BP01 組織のコスト要件を特定する
- COST05-BP02 このワークロードのすべてのコンポーネントを分析する
- COST05-BP03 各コンポーネントの詳細な分析を実行する
- COST05-BP04 コスト効率の高いライセンスを提供するソフトウェアを選択する
- COST05-BP04 組織の優先順位に従ってコストが最適化されるようにこのワークロードのコンポーネントを選択する

- COST05-BP06 異なる使用量について経時的なコスト分析を実行する

COST05-BP01 組織のコスト要件を特定する

チームメンバーと協力して、コストの最適化とこのワークロードのその他の柱とのバランス (パフォーマンスや信頼性など) を定義します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードのサービスを選択する場合は、組織の優先順位を理解することが重要です。パフォーマンスや信頼性などの Well-Architected の柱と、コストとのバランスが取れているようにします。十分にコスト最適化されたワークロードとは組織の要件に最も適合するソリューションであって、必ずしも最低コストのソリューションとは限りません。組織内のすべてのチームと会合し、製品、ビジネス、技術、財務などの情報を収集します。

実装手順

- 組織のコスト要件を特定する: 製品管理、アプリケーション所有者、開発および運用チーム、管理、財務ロールのメンバーなど、組織のチームメンバーとミーティングを行います。このワークロードとそのコンポーネントに対して Well-Architected の柱に優先順位を付けると、柱が順番に表示されたリストが出力されます。また、それぞれに重み付けを追加することもできます。これにより、柱がどの程度余分に重点化されているか、2 つの柱間の重点度がどの程度類似しているかを示すことができます。

リソース

関連するドキュメント:

- AWS 総保有コスト (TCO) 計算ツール
- Amazon S3 ストレージクラス
- クラウド製品

COST05-BP02 このワークロードのすべてのコンポーネントを分析する

現在のサイズやコストに関係なく、すべてのワークロードが分析されることを確認します。見直しを行う際には、現在のコストや予想コストなどの潜在的利益を織り込む必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ワークロードのすべてのコンポーネントについて詳細な分析を実行します。分析コストと、そのライフサイクルで可能と考えられるワークロードの節減額のバランスを取るようにします。コンポーネントの現在の影響、および将来的に与えると考えられる影響を洗い出す必要があります。例えば、提案されたリソースのコストが月額 10 ドルで、予測負荷が月額 15 ドルを超えない場合に、コストを 50% (月額 5 ドル) 削減するために 1 日分の労力を費やすようでは、システムの寿命全体にわたって得られると考えられる利益を超えることになるかもしれません。データに基づくより高速でより効率的な予測を使用すると、このコンポーネントの全体的な成果を最善のものにできます。

ワークロードは時間の経過とともに変化する可能性があり、ワークロードのアーキテクチャや使用方法が変化すると、適切だったサービスの組み合わせが最適ではなくなってしまうことがあります。サービスの選択に関する分析には、現在および将来のワークロードの状態と使用量レベルが組み込まれる必要があります。将来のワークロードの状態や使用量に合わせてサービスを運用すると、今後の変更に必要な労力を軽減または削除できることになり、全体的なコストを削減できます。

AWS Cost Explorer および AWS Cost and Usage Report (CUR) では、PoC (概念実証) または実行中の環境のコストを分析できます。また、 AWS Pricing Calculator を使用して、ワークロードのコストを見積もることもできます。

実装手順

- ワークロードコンポーネントをリスト化する: すべてのワークロードコンポーネントのリストを作成します。これは、各コンポーネントが分析されたことを確認するための検証として使用されます。費やされる労力は、組織の優先順位によって定義されたワークロードの重要性を反映している必要があります。リソースをグループ化すると、機能的に効率が向上します (例えば、複数のデータベースがある場合は本番データベースストレージ)。
- コンポーネントリストに優先順位を付ける: コンポーネントリストを取得して、労力をかける順で優先順位を付けます。これは通常、コンポーネントのコストが最も高価なものから最も安価なものへ、または組織の優先順位で定義されている重要度の順に並べられます。
- 分析を実行する: リストの各コンポーネントについて、使用可能なオプションとサービスを確認し、組織の優先順位に最適なオプションを選択します。

リソース

関連するドキュメント:

- AWS Pricing Calculator
- AWS Cost Explorer
- Amazon S3 ストレージクラス
- クラウド製品

COST05-BP03 各コンポーネントの詳細な分析を実行する

各コンポーネントの、組織にとっての全体的なコストを調べます。運用および管理のコスト、特にマネージドサービスを使用するコストを考慮して総所有コストを調べます。見直しを行う際には、分析に費やされた時間がコンポーネントのコストに比例しているなどの潜在的利益を織り込む必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

時間短縮を検討して、チームが技術的な負債の返済、イノベーション、付加価値機能に集中できるようにします。例えば、オンプレミス環境からクラウド環境に「リフトアンドシフト」式にできるだけ早急に移行して、最適化は後回しにする必要性が生じる場合があります。ライセンスコストを排除または削減するマネージドサービスを利用して、コスト削減が可能なところを模索することには時間をかけるだけの価値があります。マネージドサービスによってサービス維持に伴う運用上および管理上の負担が軽減されるため、イノベーションに集中できます。さらに、マネージドサービスはクラウドのスケールで運用されるため、トランザクション単位またはサービス単位でコストを削減できます。

通常、マネージドサービスは、十分なキャパシティーを確保するために設定できる属性を備えています。この属性を設定およびモニタリングして、余剰キャパシティーを最小限に抑え、パフォーマンスを最大化する必要があります。AWS Management Console や AWS API および SDK を使用して AWS Managed Services の属性を変更し、需要の変化に合わせてリソースのニーズを調整できます。例えば、Amazon EMR クラスターまたは Amazon Redshift クラスターのノード数を増減して、規模をスケールアウトまたはスケールインできます。

また、AWS リソースの複数のインスタンスを圧縮して、高密度に使用することもできます。例えば、単一の Amazon Relational Database Service (Amazon RDS) データベースインスタンスで、複数の小さなデータベースをプロビジョニングできます。使用量が増えたら、スナップショットや復元プロセスを使用して、そのデータベースの 1 つを専用 Amazon RDS データベースインスタンスに移行できます。

マネージドサービスでワークロードをプロビジョニングする際は、サービスキャパシティーの調整要件を理解する必要があります。主な要件としては、時間、労力、通常のワークロードオペレーション

への影響などが一般には考えられます。プロビジョニングされたリソースでは変更が発生するまでの時間が許容され、このために必要なオーバーヘッドをプロビジョニングする必要があります。サービス変更に必要な継続的労力は、システムに統合する API と SDK やAmazon CloudWatchなどのモニタリングツールを使用することで、実質ゼロまで減らすことができます。

Amazon RDSio1Amazon Redshift、および Amazon ElastiCache はマネージドデータベースサービスを提供します。Amazon Athenaio1Amazon EMR、および Amazon OpenSearch Service はマネージド型分析サービスを提供します。

AMS は、エンタープライズのお客様やパートナーに代わって AWS インフラストラクチャを運用するサービスです。コンプライアンスに準拠したセキュアな環境で、ワークロードをデプロイできます。AMS では、エンタープライズクラウド運用モデルとオートメーションを使用して、組織の要件を満たし、クラウド移行を高速化し、継続的な管理コストを削減できます。

実装手順

- 詳細な分析を実行する: コンポーネントリストを使用して、各コンポーネントを優先度が高いものから処理します。優先度がより高く、より多くのコストがかかるコンポーネントについては、追加の分析を実行し、利用可能なすべてのオプションとその長期的な影響を評価します。優先度の低いコンポーネントの場合、使用状況の変化によってコンポーネントの優先度が変更するかどうかを評価し、かける労力の適切性の分析を実行します。

リソース

関連するドキュメント:

- AWS 総保有コスト (TCO) 計算ツール
- Amazon S3 ストレージクラス
- クラウド製品

COST05-BP04 コスト効率の高いライセンスを提供するソフトウェアを選択する

オープンソースソフトウェアは、ワークロードに多大なコストをもたらすソフトウェアライセンスコストを排除することができます。ライセンスされたソフトウェアが必要な場合は、CPU などの任意の属性に結びついたライセンスは避け、出力または結果に結びついたライセンスを探します。これらのライセンスのコストは、提供するメリットに応じてより密にスケールされます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

ソフトウェアライセンスのコストは、オープンソースソフトウェアを使用することで削減できます。オープンソースソフトウェアへの変更は、ワークロードサイズが拡大するにつれ、ワークロードコストに大きな影響を与える可能性があります。ライセンスを取得したソフトウェアの利点を総コストと比較して測定し、ワークロードを最適化します。ライセンス変更とその変更がワークロードコストに与える影響をモデリングします。あるベンダーがデータベースライセンスのコストを変更したなら、それがワークロードの全体的な効率にどのような影響を与えるかを調査します。ベンダーの過去の価格アナウンスを検討して、ベンダー製品全体のライセンス変更の傾向を検討してください。ライセンスコストは、ハードウェアごとにスケールするライセンス (CPU バウンドライセンス) など、スループットや使用量とは関係なくスケールされる場合があります。こうしたライセンスは、それに伴う成果が見られないままコストが急増する可能性があるため、避けてください。

実装手順

- ライセンスオプションを分析する: 利用可能なソフトウェアのライセンス条項を確認します。必要な機能を備えたオープンソースのバージョンを探し、ライセンスされたソフトウェアの利点がコストを上回っているかどうかを確認します。有利な条件は、それが提供するメリットに照らして、コストを正当化します。
- ソフトウェアプロバイダーを分析する: ベンダーからの料金またはライセンスの変更履歴を確認します。特定のベンダーのハードウェアまたはプラットフォームで実行することについての懲罰的な条件など、結果に見合わない変更を調べます。また、監査の実行方法や課される可能性のある罰則についても確認します。

リソース

関連するドキュメント:

- AWS 総保有コスト (TCO) 計算ツール
- Amazon S3 ストレージクラス
- クラウド製品

COST05-BP04 組織の優先順位に従ってコストが最適化されるようにこのワークロードのコンポーネントを選択する

すべてのコンポーネントを選択したときのコストを考慮します。これには、アプリケーションレベルのサービスとマネージドサービス (Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB、Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Email Service

(Amazon SES) など) を使用して組織の全体的なコストを削減することが含まれます。サーバーレスやコンテナをコンピューティングに使用します。AWS Lambda、静的ウェブサイト用の Amazon Simple Storage Service (Amazon S3)、Amazon Elastic Container Service (Amazon ECS) などです。オープンソースソフトウェア、またはライセンス料金のないソフトウェア (コンピューティングワークロード用の Amazon Linux、データベースを Amazon Auroraに移行するなど) を使用して、ライセンスコストを最小限に抑えます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

以下のサーバーレスまたはアプリケーションレベルのサービスを使用できます。 AWS Lambdaio1Amazon Simple Queue Service (Amazon SQS)io1 Amazon SNS、および Amazon SES.これらのサービスではリソースを管理する必要がなく、コード実行、キューサービス、メッセージ配信の機能を利用できます。もう 1 つの利点は、使用量に応じてパフォーマンスとコストをスケールするため、コスト配分とコストの帰属が効率的になることです。

サーバーレスの詳細については、 Well-Architected サーバーレスアプリケーションレンズのホワイトペーパーを参照してください。

実装手順

- 各サービスを選択してコストを最適化する: 優先順位リストと分析を使用して、組織の優先順位に最も合致する各オプションを選択します。

リソース

関連するドキュメント:

- AWS 総保有コスト (TCO) 計算ツール
- Amazon S3 ストレージクラス
- クラウド製品

COST05-BP06 異なる使用量について経時的なコスト分析を実行する

ワークロードは時間の経過とともに変化することがあります。それぞれのサービスまたは機能のコスト効率は、使用レベルによって異なります。各コンポーネントについて予想使用量に基づく経時的な分析を実行することで、ワークロードのコスト効率性がそのライフタイム全体にわたって維持されます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

AWS で新しいサービスや機能がリリースされると、ワークロードに最適なサービスが変化する可能性があります。求められる労力は、潜在的な利点が反映されたものである必要があります。ワークロードレビューの頻度は、組織の要件によって異なります。ワークロードにかなりのコストがかかっている場合、新しいサービスの運用が早いほどコスト削減が最大になるため、レビュー頻度が高い方が有利です。レビューのトリガーとしては、使用パターンの変化も挙げられます。使用量が大幅に変化した場合は、別のサービスを使った方がよい場合もあります。たとえば、データ転送速度が高い場合、Direct Connect サービスのほうが VPN よりも安価で、必要な接続性能を提供できます。サービス変更時に起こりうる影響を予測すると、使用量レベルのトリガーをモニタリングできるため、費用対効果が最も高いサービスを速やかに運用できます。

実装手順

- 予測された使用パターンを定義する: マーケティングや製品所有者などの組織と協力して、ワークロードに対して期待および予測される使用パターンを文書化します。
- 予測された使用量に基づくコスト分析を実行する: 定義された使用パターンを使用して、これらの各ポイントで分析を実行します。分析作業は、潜在的な結果を反映する必要があります。例えば、使用量の変化が大きい場合は、コストと変更を確認するために詳細な分析を実行する必要があります。

リソース

関連するドキュメント:

- AWS 総保有コスト (TCO) 計算ツール
- Amazon S3 ストレージクラス
- クラウド製品

COST 6 リソースタイプ、リソースサイズ、およびリソース数を選択する際、コスト目標を達成するにはどうすればよいですか?

対象タスクについて適切なリソースサイズおよびリソース数を選択していることを確認します。最もコスト効率の高いタイプ、サイズ、数を選択することで、無駄を最小限に抑えます。

ベストプラクティス

- COST06-BP01 コストモデリングを実行する
- COST06-BP02 データに基づいてリソースタイプ、リソースサイズ、リソース数を選択する
- COST06-BP03 メトリクスに基づいて自動的にリソースタイプ、リソースサイズ、リソース数を選択する

COST06-BP01 コストモデリングを実行する

組織の要件を特定し、ワークロードとその各コンポーネントのコストモデリングを実行します。さまざまな予測負荷におけるワークロードのベンチマークアクティビティを実行し、コストを比較します。モデリングの際には、潜在的な利点を織り込む必要があります。例えば、費やされた時間がコンポーネントのコストに比例しているなどです。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードと各コンポーネントのコストモデリングを実行してリソース間のバランスを把握し、特定のパフォーマンスレベルに応じてワークロード内の各リソースの適切なサイズを見つけます。さまざまな予測負荷におけるワークロードのベンチマークアクティビティを実行し、コストを比較します。モデリングの際には、費やした時間がコンポーネントのコストまたは予想される削減額に比例しているといった潜在的な利点を織り込む必要があります。ベストプラクティスについては、「 レビュー 」セクション (パフォーマンス効率の柱に関するホワイトペーパー内) を参照してください。

AWS Compute Optimizer は、ワークロードの実行におけるコストモデリングに役立ちます。使用履歴に基づき、コンピューティングリソースの正しいサイズ設定に関するレコメンデーションを提供します。これがコンピューティングリソースにとって理想的なデータソースである理由は、リスクレベルに応じて複数のレコメンデーションを作成できる機械学習が使われている無料サービスだからです。また、 Amazon CloudWatch および Amazon CloudWatch Logs をデータソースとしてカスタムログと併用して、他のサービスやワークロードコンポーネントの適切なサイズ設定のオペレーションを行うこともできます。

コストモデリングのデータとメトリクスに関する推奨事項は以下の通りです。

- モニタリングにはエンドユーザーのエクスペリエンスを正確に反映する必要があります。対象期間に適切な間隔を選択して、平均の変わりに最大値や 99 パーセンタイル値をじっくり見極めます。
- すべてのワークロードのサイクルをカバーするために必要な分析期間の適切な間隔を選択します。たとえば、分析を 2 週間間隔で実行する場合、1 か月サイクルで使用率が高くても見逃す場合があり、過小プロビジョニングにつながる可能性があります。

実装手順

- コストモデリングを実行する: ワークロードまたは概念実証を、テストする特定のリソースタイプとサイズを持つ別のアカウントにデプロイします。テストデータを使用してワークロードを実行し、出力結果のほか、テスト実行時のコストデータを記録します。次に、ワークロードを再デプロイするか、リソースタイプとサイズを変更して、テストを再実行します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- Amazon CloudWatch の特徴
- Amazon EC2 Right Sizing によるコスト最適化
- AWS Compute Optimizer

COST06-BP02 データに基づいてリソースタイプ、リソースサイズ、リソース数を選択する

ワークロードとリソースの特性に関するデータに基づいて、リソースのサイズやタイプを選択します。例えば、コンピューティング、メモリ、スループット、書き込み頻度などです。この選択は通常、以前の (オンプレミス) バージョンのワークロード、ドキュメント、ワークロードに関する他の情報ソースを用いて行います。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

ワークロードとリソースの特性 (例えば、コンピューティング、メモリ、スループット、書き込み頻度) に基づいて、リソースのサイズやタイプを選択します。この選択は通常、コストモデリング、以前のバージョンのワークロード (オンプレミスバージョンなど)、ドキュメント、ワークロードに関する他の情報ソース (ホワイトペーパー、公開ソリューション) を用いて行います。

実装手順

- データに基づいてリソースを選択する: コストモデリングデータを使用して、予想されるワークロードの使用レベルを選択し、指定したリソースタイプとサイズを選択します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- Amazon CloudWatch の特徴
- EC2 Right Sizing によるコスト最適化

COST06-BP03 メトリクスに基づいて自動的にリソースタイプ、リソースサイズ、リソース数を選択する

現在実行しているワークロードからのメトリクスを用いて、コストを最適化する適切なサイズやタイプを選択します。Amazon Elastic Compute Cloud (Amazon EC2)、Amazon DynamoDB、Amazon Elastic Block Store (Amazon EBS) (PIOPS)、Amazon Relational Database Service (Amazon RDS)、Amazon EMR、ネットワークなどのサービスに、適切なスループット、サイジング、ストレージをプロビジョニングします。これは、自動スケーリングなどのフィードバックループまたはワークロードのカスタムコードで行うことができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ワークロード内に、実行中のワークロードのアクティブなメトリクスを使用してそのワークロードを変更するフィードバックループを作成します。また、 AWS Auto Scalingなどのマネージドサービスを使用して、正しいサイズ変更オペレーションを実行できるように設定します。AWS は、API、SDKのほか、最低限の労力でリソースを変更するための機能も提供しています。Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの停止と起動のワークロードをプログラムして、インスタンスサイズやインスタンスタイプを変更できます。これにより、正しいサイズ設定による利点が得られるだけでなく、変更に必要なほぼすべての運用コストを削減することもできます。

一部の AWS のサービスには、 Amazon Simple Storage Service (Amazon S3) Intelligent-Tieringなどのタイプやサイズの自動選択が組み込まれています。Amazon S3 Intelligent-Tiering では、使用パターンに基づいて、高頻度アクセスと低頻度アクセスの 2 つのアクセスティア間でデータが自動的に移動します。

実装手順

- ワークロードメトリクスを設定する: ワークロードの主要メトリクスをキャプチャしていることを確認します。これらのメトリクスは、ワークロード出力などのカスタマーエクスペリエンスに関す

る示唆を提供し、CPU やメモリの使用状況などのリソースのタイプとサイズの違いに合わせて調整されます。

- 適切なサイズのレコメンデーションを表示する: AWS Compute Optimizer の適切なサイズのレコメンデーションを使用して、ワークロードを調整します。
- メトリクスに基づいて自動的にリソースタイプとリソースサイズを選択する: ワークロードメトリクスを使用して、ワークロードリソースを手動または自動的に選択します。AWS Auto Scaling を設定したり、アプリケーション内でコードを実装したりすると、頻繁な変更が要求される場合に必要となる労力を減らすことができるほか、手動プロセスより早く変更を実装できる可能性もあります。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Compute Optimizer
- Amazon CloudWatch の特徴
- CloudWatch 準備作業
- CloudWatch カスタムメトリクスの発行
- Cost Optimization: Amazon EC2 Right Sizing
- Getting Started With Amazon EC2 Auto Scaling
- Amazon S3 Intelligent-Tiering
- Launch an EC2 Instance Using the SDK

COST 7 コスト削減のために、どのように料金モデルを使用すればよいですか?

リソースのコストを最小限に抑えるのに最も適した料金モデルを使用します。

ベストプラクティス

- COST07-BP01 料金モデルの分析を実行する
- COST07-BP02 コストに基づいてリージョンを選択する
- COST07-BP03 費用対効果の高い条件を提供するサードパーティーの契約を選択する
- COST07-BP04 このワークロードのすべてのコンポーネントに対して料金モデルを実装する
- COST07-BP05 管理アカウントレベルで料金モデル分析を実行する

COST07-BP01 料金モデルの分析を実行する

ワークロードの各コンポーネントを分析します。コンポーネントとリソースが長期間実行されるか (コミットメント割引)、動的および短期実行 (スポットインスタンスまたはオンデマンドインスタンス) とするかを決定します。AWS Cost Explorer のレコメンデーション機能を使用して、ワークロードに関する分析を実行します。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

AWS には複数の 料金モデルがあり、 組織のニーズに合った最も費用対効果の高い方法でリソースの料金をお支払いいただくことができます。

実装手順

- コミットメント割引の分析を実行する: アカウントで Cost Explorer を使用して、Savings Plans とリザーブドインスタンスの推奨事項を確認します。必要な割引を適用し、リスクを認識した上で、正しい推奨事項を実装していることを確認するには、 Well-Architected ラボに従ってください。
- ワークロードの伸縮性を分析する: Cost Explorer の時間単位の粒度またはカスタムダッシュボードを使用します。ワークロードの伸縮性を分析します。実行中のインスタンス数の定期的な変更を調べます。短期間のインスタンスはスポットインスタンスまたはスポットフリートの候補です。
  - Well-Architected ラボ: Cost Explorer
  - Well-Architected ラボ: コストの可視化

リソース

関連するドキュメント:

- リザーブドインスタンスの推奨事項へのアクセス
- インスタンス購入オプション

関連動画:

- Save up to 90% and run production workloads on Spot (最大 90% 節約し、Spot で本番環境のワークロードを稼動)

関連する例:

- Well-Architected ラボ: Cost Explorer
- Well-Architected ラボ: コストの可視化
- Well-Architected ラボ: 料金モデル

COST07-BP02 コストに基づいてリージョンを選択する

リソースの料金は各リージョンで異なる場合があります。リージョンコストを織り込むことで、このワークロードに対して支払う料金の合計を最低限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

ソリューションを設計する際、ユーザーに近いコンピューティングリソースの場所を探して、レイテンシー低下とデータ主権の強化を図ることが推奨されます。グローバルな利用者のニーズに応えるためには、複数の口ケーションを使用する必要があります。コストを最低限に抑えられる地理的口ケーションを選択する必要もあります。

AWS クラウドクラウドインフラストラクチャは、 リージョンとアベイラビリティーゾーンを中心に構築されています。リージョンは世界中の物理的場所であり、複数のアベイラビリティーゾーンが配置されています。アベイラビリティーゾーンは 1 つ以上の独立したデータセンターで構成されます。各データセンターは、冗長性のある電源、ネットワーク、接続を備えており、別々の設備に収容されています。

各 AWS リージョンは、地域の市場の条件下で運用されており、リソースの料金はリージョンごとに異なります。世界的に最小料金で稼働できるように、ソリューションのコンポーネントまたは全体を運用する特定のリージョンを選択します。さまざまなリージョンのワークロードのコストを見積もるには、 AWS Pricing Calculator を使用します。

実装手順

- リージョンの料金を確認する: 現在のリージョンのワークロードコストを分析します。サービスおよび使用タイプ別の最も高いコストから、利用可能な他のリージョンのコストを計算します。予測される費用削減効果がコンポーネントまたはワークロードの移動コストを上回っている場合は、新しいリージョンに移行します。

リソース

関連するドキュメント:

- リザーブドインスタンスのレコメンデーションへのアクセス
- Amazon EC2 料金
- インスタンス購入オプション
- リージョン別の表

関連動画:

- Save up to 90% and run production workloads on Spot

COST07-BP03 費用対効果の高い条件を提供するサードパーティーの契約を選択する

コスト効率に優れた契約と条件により、これらのサービスのコストが、提供されるメリットに見合ったものとなります。組織に追加のメリットを提供するときに、それに合わせてスケーリングする契約と料金を選択します。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

クラウドでサードパーティー製のソリューションまたはサービスを利用する場合、料金構造とコスト最適化の結果を連動させることが重要です。料金は、コスト最適化の結果とサービスの価値に合わせてスケールする必要があります。この一例として、削減率を使用したソフトウェアがあります。削減率 (結果) が高くなるほど請求額も上がるというものです。請求額に合わせてスケールする契約は、特定の請求書のあらゆる部分で結果が得られない限り、ほとんどの場合コストの最適化と連動していません。例えば、Amazon Elastic Compute Cloud (Amazon EC2) の推奨事項を提供し、請求全体のある割合が課金されるソリューションでは、メリットを提供しない他のサービスを利用すると、その分増加します。もう 1 つの例は、管理するリソースのコストの割合に応じて課金されるマネージドサービスです。インスタンスサイズを大きくすると常に管理作業が増えるわけではありませんが、請求額が高くなります。これらのサービス料金設定に、コスト最適化プログラムまたはサービス機能が含まれていることを承知のうえで効率性を向上させます。

実装手順

- サードパーティーの契約と諸条件を分析する: サードパーティー契約の料金を確認します。さまざまな使用レベルに対応したモデリングを行い、新しいサービスの使用や、ワークロードの増加による現在のサービスの増加など、新たなコストを考慮します。追加コストによってビジネスに必要なメリットが得られるかどうかを判断します。

リソース

関連するドキュメント:

- リザーブドインスタンスの推奨事項へのアクセス
- インスタンス購入オプション

関連動画:

- Save up to 90% and run production workloads on Spot (最大 90% 節約し、Spot で本番環境のワークロードを稼動)

COST07-BP04 このワークロードのすべてのコンポーネントに対して料金モデルを実装する

永続的に実行されるリソースでは、Savings Plans やリザーブドインスタンスなどのリザーブドキャパシティを利用する必要があります。短期的な使用には、スポットインスタンスまたはスポットフリートを使用するように設定します。オンデマンドインスタンスは、中断することのできない、かつリザーブドキャパシティに対して長時間稼働しない短期ワークロードに対してのみ使用されます (リソースタイプに応じて、期間の 25% から 75%)。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

ワークロードコンポーネントの要件を考慮し、料金のポテンシャルモデルを理解します。コンポーネントの可用性要件を定義します。ワークロードで関数を実行する複数の独立したリソースの有無、ワークロードの継続的に必要となる要件を確認します。デフォルトのオンデマンド料金モデルと他の適用可能なモデルを使用して、リソースのコストを比較します。リソースまたはワークロードコンポーネントで変更可能なものはすべて考慮します。

実装手順

- 料金モデルを実装する: 分析結果を使用して、Savings Plans (SP) やリザーブドインスタンス (RI) を購入するか、スポットインスタンスを実装します。RI を初めて購入する場合は、リストで上位 5 件または 10 件のレコメンデーションを選択し、翌月または翌々月までの結果をモニタリングして分析します。少数のコミットメント割引を定期的に購入します (例: 2 週間ごと、または 1 か月ごと)。中断可能またはステートレスなワークロードにスポットインスタンスを実装します。

- ワークロードレビューサイクル: 特に料金モデルカバレッジを分析するワークロードのレビューサイクルを実装します。ワークロードが必要な範囲に達したら、2～4 週間ごと、または組織の使用状況の変化に応じて、追加のコミットメント割引を購入します。

リソース

関連するドキュメント:

- リザーブドインスタンスの推奨事項へのアクセス
- EC2 フリート
- リザーブドインスタンスの購入方法
- インスタンス購入オプション
- スポットインスタンス

関連動画:

- Save up to 90% and run production workloads on Spot (最大 90% 節約し、Spot で本番環境のワークロードを稼動)

COST07-BP05 管理アカウントレベルで料金モデル分析を実行する

Cost Explorer の Savings Plans およびリザーブドインスタンスのレコメンデーションを使用して、コミットメント割引の管理アカウントレベルでの定期的な分析を実行します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

コストモデリングを定期的に実行すると、複数のワークロードにまたがって最適化する機会が確実に得られます。例えば、複数のワークロードでオンデマンドインスタンスを使用している場合、集計レベルでは変更リスクが低くなり、コミットメントベースの割引を運用すると全体的なコストが低くなります。2 週間から 1 か月の定期的なサイクルで分析を実行することを推奨します。これにより、調整のための小口購入が可能になり、ワークロードやコンポーネントの変更に合わせて料金モデルの調整を続けることができます。

コミットメント割引を適用する機会を見つけるために、 AWS Cost Explorer を使用します。

スポットのワークロードを実行する機会を見つけるには、使用量全体の 1 時間ごとのビューを使用して、使用量や伸縮性の変化を定期に探します。

実装手順

- コミットメント割引分析を実行する: アカウントで Cost Explorer を使用して、Savings Plans とリザーブドインスタンスのレコメンデーションを確認します。必要な割引を適用し、リスクを認識した上で、正しいレコメンデーションを実装していることを確認するには、Well-Architected ラボに従ってください。

リソース

関連するドキュメント:

- リザーブドインスタンスのレコメンデーションへのアクセス
- インスタンス購入オプション

関連動画:

- Save up to 90% and run production workloads on Spot

関連する例:

- Well-Architected ラボ: 料金モデル

COST 8 データ転送料金はどのように計画すればよいですか？

データ転送料金を計画し、モニタリングすることで、これらのコストを最小化するためのアーキテクチャ上の決定を下すことができます。小規模でも効果的なアーキテクチャ変更により、長期的な運用コストを大幅に削減できる場合があります。

ベストプラクティス

- COST08-BP01 データ転送モデリングを実行する
- COST08-BP02 データ転送コストを最適化するコンポーネントを選択する
- COST08-BP03 データ転送コストを削減するサービスを実装する

COST08-BP01 データ転送モデリングを実行する

組織の要件を取りまとめ、ワークロードとその各コンポーネントのデータ転送モデリングを実行します。これにより、現在のデータ転送要件に対する最低コストを特定できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードでデータ転送が行われる場所、転送コスト、およびそれに関連する利点を理解します。これにより、十分な情報に基づいてアーキテクチャ設計上の変更や承諾の決定ができます。たとえば、アベイラビリティーゾーン間でデータをレプリケートするマルチアベイラビリティーゾーンを設定したとします。構造のコストをモデリングし、これが許容可能なコスト (両方のアベイラビリティーゾーンにおけるコンピューティングコストとストレージコストと同様のもの) であると判断されると、必要な信頼性と耐障害性が達成されます。

さまざまな使用量のレベルでコストをモデリングします。ワークロード使用量は経時変化します。また、サービスの種類ごとに異なるレベルで費用対効果が向上する場合があります。

データ転送コストを理解し、モデリングするには、 AWS Cost Explorer や AWS Cost and Usage Report (CUR) を使用します。PoC (概念実証) を設定するか、またはワークロードをテストして、現実的な条件でシミュレートされた負荷を用いてテストを実行します。ワークロードのさまざまな需要に応じてコストをモデルリングできます。

実装手順

- データ転送コストを計算する: ワークロードのデータ転送コストを計算するには、 AWS の料金ページ を使用します。ワークロードの使用量が増減した場合の、使用量別のデータ転送コストを計算します。ワークロードアーキテクチャに複数のオプションがある場合は、比較のために各オプションのコストを計算します。
- コストを結果にリンクする: 発生したデータ転送コストごとに、ワークロードで達成した結果を指定します。コンポーネント間の転送であればデカップリングのため、アベイラビリティゾーン間の転送であれば冗長性のためかもしれません。

リソース

関連するドキュメント:

- AWS caching solutions
- AWS の料金
- Amazon EC2 の料金
- Amazon VPC の料金
- Amazon CloudFront を使用してコンテンツを迅速に配信する

COST08-BP02 データ転送コストを最適化するコンポーネントを選択する

すべてのコンポーネントが選択され、データ転送コストを低減するようアーキテクチャが設計されます。これには、ワイドエリアネットワーク (WAN) 最適化やマルチアベイラビリティゾーン (AZ) 設定などのコンポーネントの使用が含まれます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

データ転送向けのアーキテクチャでは、データ転送コストが最小化されます。このアーキテクチャでは、コンテンツ配信ネットワークを使用してユーザーに近いデータを特定したり、お客様のプレミスと AWS をつなぐ専用ネットワーク接続が使用される場合があります。WAN の最適化やアプリケーションの最適化によって、コンポーネント間で転送されるデータ量を減らすこともできます。

実装手順

- データ転送用にコンポーネントを選択する: データ転送モデリングを使用して、データ転送コストが最も大きい場所や、ワークロードの使用状況が変わった場合の場所に焦点を当てます。データ転送の必要性を排除または削減したり、コストを削減したりする代替アーキテクチャや追加のコンポーネントを探します。

リソース

関連するドキュメント:

- AWS caching solutions
- Amazon CloudFront を使用してコンテンツを迅速に配信する

COST08-BP03 データ転送コストを削減するサービスを実装する

データ転送コストを削減するサービスを実装します。例えば、Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) を使用してエンドユーザーにコンテンツを配信する、Amazon ElastiCache を使用してレイヤーをキャッシュする、AWS への接続用に VPN の代わりに AWS Direct Connect を使用するなどです。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

Amazon CloudFront は、低レイテンシーかつ高速な転送速度でデータを転送するグローバルなコンテンツ配信ネットワークです。世界中のエッジロケーションでデータをキャッシュすることで、お客様のリソースの負荷を軽減します。CloudFront を使用してレイテンシーを最低限に抑え、世界中の多数のユーザーにコンテンツを配信するための管理労力を軽減できます。

AWS Direct Connect では、AWS への専用ネットワーク接続を確立できます。このサービスによって、ネットワークコストの削減、帯域幅の増加、インターネット経由の接続よりも安定したネットワーク接続が可能になります。

AWS VPN を使用すると、プライベートネットワークと AWS グローバルネットワークとの間に安全なプライベート接続を確立できます。迅速かつ簡単な接続とフルマネージド型の伸縮自在なサービスは、小規模なオフィスやビジネスパートナーに最適です。

VPC エンドポイント により、プライベートネットワークを利用して AWS のサービス間の接続が可能になり、パブリックデータ転送と NAT ゲートウェイ のコストを削減できます。ゲートウェイ VPC エンドポイント では、時間単位の料金は発生せず、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB がサポートされます。インターフェイス VPC エンドポイント は、 AWS PrivateLink によって提供され、時間単位の料金と GB あたりの使用料がかかります。

実装手順

- サービスを実装する: データ転送モデリングを使用して、最大のコストと最大のボリュームフローがどこにあるかを調べます。AWS のサービスを確認し、転送を減らすか排除するサービス (特にネットワークとコンテンツ配信) があるかどうかを評価します。また、データへの繰り返しのアクセス、または大量のデータがあるキャッシュサービスを探します。

リソース

関連するドキュメント:

- AWS Direct Connect
- AWS の製品を見る
- AWS caching solutions
- Amazon CloudFront
- Amazon CloudFront を使用してコンテンツを迅速に配信する

需要を管理しリソースを供給する

質問

- COST 9 どのように需要を管理し、リソースを供給すればよいですか?

COST 9 どのように需要を管理し、リソースを供給すればよいですか?

費用とパフォーマンスのバランスが取れたワークロードを作成するには、費用を掛けたすべてのものが活用されるようにし、使用率が著しく低いインスタンスが生じるのを回避します。利用が過剰でも過少でも偏りが生じると、運用コスト (利用過剰によるパフォーマンスの低下) または無駄な AWS 費用 (過剰なプロビジョニング) のいずれかで、組織に悪影響が及びます。

ベストプラクティス

- COST09-BP01 ワークロードの需要に関する分析を実行する
- COST09-BP01 需要を管理するためのバッファまたはスロットルを実装する
- COST09-BP03 リソースを動的に供給する

COST09-BP01 ワークロードの需要に関する分析を実行する

ワークロードの需要を経時的に分析します。分析が季節的傾向を考慮し、ワークロードのライフタイム全体にわたる動作条件を正確に反映したものであることを確認します。分析を行う際には、費やされた時間がワークロードのコストに比例しているなどの潜在的利益を織り込む必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードの要件を把握します。組織の要件に、リクエストに対するワークロードの応答時間を含める必要があります。応答時間は、需要が管理されているかどうか、または需要を満たすためにリソースの供給が変化するかどうかを判断するために使用できます。

分析には、需要の予測可能性と再現性、需要の変化率、需要の変化量を含める必要があります。分析は、月末処理や休日のピークなどの時季的な変動が組み込まれるように、十分な期間にわたって実行されるようにします。

分析作業では、スケーリングの実装による潜在的な利点が反映されていることを確認します。コンポーネントの予想される合計コスト、ワークロードのライフタイムにおける使用量の増減およびコストの増減に注目します。

ワークロードの需要は AWS Cost Explorer または Amazon QuickSight を AWS Cost and Usage Report (CUR) またはアプリケーションログとともに使用して、視覚的に分析できます。

実装手順

- 既存のワークロードデータを分析する: 既存のワークロード、以前のバージョンのワークロード、または予測された使用パターンのデータを分析します。ログファイルとモニタリングデータを使用して、顧客がワークロードをどのように使用しているかについての洞察を得ます。一般的なメトリクスは、実際の需要 (1 秒あたりのリクエスト数)、需要率が変化するか異なるレベルとなるタイミング、および需要の変化率です。ワークロードの全サイクルを分析し、月末や年末のイベントなどの季節的な変化のデータを収集します。分析に反映される労力は、ワークロードの特性を反映する必要があります。最大の労力は、需要に最も大きな変化がある価値の高いワークロードに割り当てられる必要があります。需要の変化が最小である低価値のワークロードには、最小の労力を割り当てる必要があります。価値の一般的なメトリクスは、リスク、ブランド認知、収益、ワークロードのコストです。
- 外部の影響を予測する: 組織全体において、ワークロードの需要に影響を与え、または変化させる可能性のあるチームメンバーとミーティングを行います。一般的なチームは販売、マーケティング、ビジネス開発です。当該メンバーと協力して、業務のサイクルや、ワークロードの需要を変化させるイベントがあるかどうかを把握します。このデータを使用してワークロードの需要を予測します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS での Instance Scheduler
- Amazon SQS の開始方法
- AWS Cost Explorer
- Amazon QuickSight

COST09-BP01 需要を管理するためのバッファまたはスロットルを実装する

バッファリングとスロットリングは、ワークロードの需要を修正し、ピークを滑らかにします。クライアントが再試行を実行するときにスロットリングを実行します。バッファリングは、リクエストを保存し、後日まで処理を延期するために実装します。スロットルとバッファが、クライアントが要求された時間内にレスポンスを受け取るように設計されていることを確認します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

スロットリング: 需要元のソースに再試行機能がある場合は、スロットリングを実装できます。現在リクエストを処理できない場合は、後で再試行する必要があることがスロットリングによってソースに通知されます。ソースでは一定時間待機してから、リクエストが再試行されます。スロットリングの運用には、リソースの最大量およびワークロードのコストを制限できるという利点があります。AWS では、 Amazon API Gateway を使用してスロットリングを実装できます。スロットリングの実装の詳細については、 Well-Architected 信頼性の柱のホワイトペーパー を参照してください。

バッファベース: スロットリングと同様に、バッファはリクエスト処理を延期し、アプリケーションが異なる動作速度で実行されていても効果的に通信できるようにします。バッファベースのアプローチでは、キューを使用してプロデューサーからメッセージ (作業単位) を受信します。メッセージはコンシューマーによって読み取られ、処理されるため、コンシューマーのビジネス要件を満たせる動作速度でメッセージを実行できます。プロデューサーがデータの耐久性やバックプレッシャーなどのスロットルの問題に対処する必要があることを心配する必要はありません (コンシューマーの動作が遅いためにプロデューサーが遅くなります)。

AWS でバッファベースのアプローチを実装する際は、複数のサービスから選択できます。Amazon Simple Queue Service (Amazon SQS) は、1 人のコンシューマーが個別のメッセージを読むことができるキューを提供するマネージドサービスです。Amazon Kinesis は、多数のコンシューマーが同じメッセージを読み取ることができるストリームを提供します。

バッファベースのアプローチで設計する場合は、必要な時間内にリクエストを処理するようにワークロードを設計し、作業の重複リクエストを処理できるようにします。

実装手順

- クライアントの要件を分析する: クライアントリクエストを分析して、再試行を実行できるかどうかを判断します。再試行を実行できないクライアントの場合、バッファを実装する必要があります。全体的な需要、変化率、および要求される応答時間を分析して、必要なスロットルまたはバッファのサイズを決定します。
- バッファまたはスロットルを実装する: ワークロードにバッファまたはスロットルを実装します。Amazon Simple Queue Service (Amazon SQS) などのキューは、ワークロードコンポーネントにバッファを提供できます。Amazon API Gateway は、ワークロードコンポーネントのためにスロットリングを提供できます。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS での Instance Scheduler
- Amazon API Gateway
- Amazon Simple Queue Service
- Amazon SQS の開始方法
- Amazon Kinesis

COST09-BP03 リソースを動的に供給する

リソースを計画的なやり方でプロビジョニングします。これは、自動スケーリングなどの需要ベース、または需要が予測可能でリソースが時間に基づいて提供される時間ベースで行います。これらの手法を使用すると、過剰プロビジョニングやプロビジョニング不足を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

専用のインフラストラクチャで AWS Auto ScalingAWS API または SDK を使用して AWS API または SDK.これにより、環境を手動変更していた運用コストがなくなり、その結果、全体的なワークロードコストが削減され、実行速度が向上します。また、ワークロードリソースと需要を常に一致させることができます。

需要ベースの供給: クラウドの伸縮性を活用して、需要の変化に対応するリソースを提供します。API やサービス機能を活用すると、アーキテクチャ内のクラウドリソースの量をプログラムで動的に変更できます。これにより、アーキテクチャ内のコンポーネントの規模を変えたり、需要が急増したときにリソースの数を自動的に増加させてパフォーマンスを維持したり、需要が後退したときにキャパシティーを減少させてコストを節減させたりできます。

AWS Auto Scaling は、キャパシティーを調整し、安定した予測可能なパフォーマンスを可能な限り低いコストで維持するのに役立ちます。これは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、スポットフリート、Amazon Elastic Container Service (Amazon ECS)、Amazon DynamoDB、Amazon Aurora と統合されたフルマネージド型の無料サービスです。

Auto Scaling では、リソースの自動検出によってワークロード内の設定可能なリソースを検出できます。また、パフォーマンス、コスト、または両者のバランスを最適化するためのスケーリング戦略が組み込まれており、予測スケーリングによって定期的に発生する急増に対応することができます。

Auto Scaling では、手動スケーリング、スケジュールに基づくスケーリング、または需要ベースのスケーリングを実装できます。また、 Amazon CloudWatch のメトリクスとアラームを使用して、ワークロードのスケーリングイベントをトリガーできます。一般的なメトリクスは標準Amazon EC2メトリクスです。CPU 使用率、ネットワークスループット、 Elastic Load Balancing(ELB) で確認されたリクエストとレスポンスのレイテンシーなどがあります。可能な場合は、カスタマーエクスペリエンスの指標となるメトリクスを使用する必要があります。このメトリクスは一般には、ワークロード内のアプリケーションコードから生成されるカスタムメトリクスです。

需要ベースのアプローチで設計する場合、主に 2 つの点を考慮する必要があります。第 1 に、新しいリソースをどれだけ早くプロビジョニングする必要があるかを理解することです。第 2 に、需要と供給の差異が変動することを理解することです。需要の変動ペースに対処できるようにしておくだけでなく、リソースの不具合にも備えておく必要があります。

ELB は、複数のリソースに需要を分散することで、スケーリングを支援します。運用するリソースが増加したら、ロードバランサーにリソースを追加し需要を分散させます。Elastic Load Balancing では、Amazon EC2 インスタンス、コンテナ、IP アドレス、AWS Lambda 関数がサポートされています。

時間ベースの供給: 時間ベースのアプローチでは、リソースのキャパシティーを予測可能な需要、または時間ごとに明確に定義された需要に合わせます。このアプローチは、通常、リソースの使用率に依存せず、リソースが必要な特定の時間にそのリソースを確保します。また、起動手順、およびシステムや一貫性のチェックにより、遅延なくリソースを提供できます。時間ベースのアプローチでは、繁忙期に追加のリソースを投入したり、キャパシティーを拡大したりできます。

スケジュールされた Auto Scaling を使用して、時間ベースのアプローチを実装できます。営業開始時など、特定の時間にスケールアウトまたはスケールインするようにスケジュールできるため、ユーザーがアクセスしたときや需要が発生したときにリソースを利用可能にしておくことができます。

また、 AWS API や SDK および AWS CloudFormation を使用すると、必要に応じて自動的にプロビジョニングしたり、環境全体を削除したりできます。このアプローチは、所定の営業時間や一定期間にのみ実行される開発環境またはテスト環境に適しています。

API を使用した環境内のリソースサイズのスケーリング (垂直スケーリング) にも対応しています。たとえば、インスタンスのサイズやクラスを変更して、本番稼働ワークロードをスケールアップできます。これを行うには、インスタンスを停止・起動して、別のインスタンスのサイズやクラスを選択

します。この手法は、使用中にサイズの拡大、パフォーマンス (IOPS) の調整、ボリュームタイプの変更が可能な Amazon Elastic Block Store (Amazon EBS) Elastic Volumes などのリソースにも適用できます。

時間ベースのアプローチを設計する際は、主に 2 つの点を考慮する必要があります。1 つ目は使用パターンの一貫性についてであり、 第 2 に、パターンを変更した場合の影響です。 予測精度は、ワークロードをモニタリングし、ビジネスインテリジェンスを使用することで高めることができます。使用パターンに大幅な変更がある場合は、時間を調整して予測対象範囲に収まるようにします。

実装手順

- 時間ベースのスケジューリングを設定する: 需要の変化を予測できるため、時間ベースのスケーリングは適切な数のリソースを適時に提供できます。また、リソースの作成と設定が、需要の変化に対応するのに十分ではない場合にも役立ちます。ワークロード分析を活用して、AWS Auto Scaling を使用してスケジュールに基づくスケーリングを設定します。
- Auto Scaling を設定する: アクティブなワークロードメトリクスに基づいてスケーリングを設定するには、Amazon Auto Scaling を使用します。分析を使用して、正しいリソースレベルでトリガーするように Auto Scaling を設定し、ワークロードが要求された時間内にスケールすることを確認します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS での Instance Scheduler
- Getting Started With Amazon EC2 Auto Scaling
- Amazon SQS の開始方法
- Amazon EC2 Auto Scaling のスケジュールされたスケーリング

継続的最適化

質問

- COST 10 新しいサービスをどのように評価すればよいですか？

COST 10 新しいサービスをどのように評価すればよいですか？

AWS では新しいサービスと機能がリリースされるため、既存のアーキテクチャの決定をレビューし、現在でもコスト効率が最も優れているかどうかを確認することがベストプラクティスです。

ベストプラクティス

- COST10-BP01 ワークロードレビュープロセスを開発する
- COST10-BP02 このワークロードを定期的に見直し、分析する

COST10-BP01 ワークロードレビュープロセスを開発する

ワークロードレビューの基準とプロセスを定義するプロセスを開発します。レビューを行う際には、潜在的利益を織り込む必要があります。例えば、請求の 10% 以上の価値を持つコアワークロードは四半期ごとにレビューし、10% 未満のワークロードは年に 1 回レビューするなどです。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードの費用対効果が常に最大になるようにするには、ワークロードを定期的にレビューし、新しいサービス、機能、コンポーネントを実装する機会があるかどうかを把握する必要があります。全体的なコスト削減を達成するには、潜在的なコスト削減量に比例したプロセスを行う必要があります。たとえば、支出全体の 50% を占めるワークロードは、支出全体の 5% を占めるワークロードよりも定期的かつ徹底的にレビューする必要があります。外部要因または変動性を考慮します。ワークロードにより特定の地域、特定の市場セグメントにサービスが提供されていて、その領域での変化が予測される場合、レビュー頻度を高くすることでコスト削減につながる可能性があります。レビューで考慮すべきもう 1 つの要因は、変更を運用する労力です。変更のテストおよび検証に多大なコストがかかる場合は、レビューの頻度を下げる必要があります。

古くなったレガシーコンポーネントやリソースには維持するための長期的なコストがかかることや、新しい機能を実装できないことを考慮します。テストと検証にかかる現在のコストが、提案されている利益を上回っている場合があります。しかし、ワークロードと現在のテクノロジーとのギャップが時間の経過とともに大きくなるにつれて、変更にかかるコストが増加し、結果として巨額のコストになることがあります。たとえば、新しいプログラミング言語に移行するときの費用対効果は現時点で低いとします。しかし、5 年後には、その言語に精通した人材のコストが増加する可能性があります。ワークロードが増加すると、さらに大規模なシステムを新しい言語に移行することになり、結果的にこれまでよりもさらに多大な労力を要します。

ワークロードをコンポーネントに分割し、コンポーネントのコストを割り当て (コストの見積りで可)、各コンポーネントの横に要因 (労力や外部市場など) を一覧表示します。この指標を使用して、各ワークロードのレビュー頻度を決定します。たとえば、ウェブサーバーが高コストで、変更の労力が低く、外部要因が高い場合は、レビュー頻度が高くなります。中央データベースが中程度のコストで、変更の労力が高く、外部要因が低い場合は、レビューの頻度は中程度になります。

実装手順

- レビュー頻度を定義する: ワークロードとそのコンポーネントを確認する頻度を定義します。これは要因の組み合わせであり、組織内のワークロードによって異なる場合があります。また、ワークロード内のコンポーネントによって異なる場合もあります。一般的な要因には、収益またはブランドの観点から評価された組織にとっての重要性、ワークロードの実行にかかる総コスト (運用コストとリソースコストを含む)、ワークロードの複雑さ、変更の実装の容易性、ソフトウェアライセンス契約、ある変更がライセンス違反によるライセンス費用の重大な増加を生じさせるかどうかなどが含まれます。コンポーネントは、ウェブサーバーやデータベース、コンピューティングリソースやストレージリソースなど、機能的または技術的に定義できます。それに応じて要因のバランスをとり、ワークロードとそのコンポーネントのための期間を設定します。例えば、ワークロード全体は 18 か月ごとに、ウェブサーバーは 6 か月ごとに、データベースは 12 か月ごとに、コンピューティングおよび短期ストレージは 6 か月ごとに、長期ストレージは 12 か月ごとに、それぞれレビューすることとできます。
- レビューの十分性を定義する: ワークロードまたはワークロードコンポーネントのレビューに費やされる労力を定義します。レビュー頻度と同様に、これは複数の要因のバランスです。例えば、データベースコンポーネントの分析に 1 週間を、ストレージのレビューに 4 時間を、それぞれ費やすようにすることができます。

リソース

関連するドキュメント:

- AWS ニュースブログ
- クラウドコンピューティングのタイプ
- AWS の最新情報

COST10-BP02 このワークロードを定期的に見直し、分析する

既存のワークロードは、定義されたプロセスごとに定期的に見直されます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

新しい AWS のサービスと機能の利点を得るには、ワークロードでレビュープロセスを実行し、必要に応じて新しいサービスや機能を実装する必要があります。例えば、ワークロードを見直して、メッセージングコンポーネントを Amazon Simple Email Service (Amazon SES) に置き換えることができます。これにより、すべての機能を低コストで提供しながら、インスタンスのフリートの運用と維持にかかるコストを削減できます。

実装手順

- ワークロードを定期的に見直す: 定義したプロセスを使用して、指定した頻度でレビューを実行します。各コンポーネントに適正な労力を費やしていることを確認します。このプロセスは、コスト最適化のためにサービスを選択した最初の設計プロセスに似ています。サービスとこのサービスがもたらすメリットを分析します。今回は、長期的なメリットだけでなく、変更を行うコストも考慮します。
- 新しいサービスを実装する: 分析の結果、変更を実施する場合は、まずワークロードのベースラインを実行し、各アウトプットの現在のコストを把握します。変更を実施し、分析を実行して、各アウトプットの新しいコストを確認します。

リソース

関連するドキュメント:

- AWS ニュースブログ
- クラウドコンピューティングのタイプ
- AWS の最新情報

サステナビリティ

トピック

- リージョンの選択
- ユーザーの行動パターン
- ソフトウェアとアーキテクチャのパターン
- データパターン
- ハードウェアパターン
- 開発とデプロイのプロセス

リージョンの選択

質問

- SUS 1 リージョンをどのように選択して、持続可能性目標を目指しますか?

SUS 1 リージョンをどのように選択して、持続可能性目標を目指しますか?

ビジネス要件と持続可能性目標の両方に基づいて、ワークロードを実装するリージョンを選択します。

ベストプラクティス:

SUS01-BP01 Amazon の再生可能エネルギープロジェクトに近いリージョンと、グリッドの炭素強度が他の場所 (またはリージョン) よりも低く公表されているリージョンを選択する

Amazon の再生可能エネルギープロジェクトに近いリージョンであり、グリッドの公開されている炭素集約度が他の場所 (またはリージョン) よりも低いリージョンを選択します。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

Amazon の再生可能エネルギープロジェクトに近いリージョンであり、グリッドの公開されている炭素集約度が他の場所 (またはリージョン) よりも低いリージョンを選択します。

リソース

関連するドキュメント:

- Amazon Around the Globe (世界中の Amazon)
- 再生可能エネルギーの方法論
- What to Consider when Selecting a Region for your Workloads (ワークロードに応じたリージョンを選択する際の注意点)

ユーザーの行動パターン

質問

- SUS 2 ユーザーの行動パターンをどのように利用して、持続可能性目標を目指しますか?

SUS 2 ユーザーの行動パターンをどのように利用して、持続可能性目標を目指しますか?

ユーザーがワークロードやその他のリソースを使用する方法によって、持続可能性の目標を達成するための改善点を特定できます。継続的にユーザーの負荷に合うようにインフラストラクチャをスケールし、ユーザーをサポートするために必要な最小リソースのみがデプロイされているようにします。サービスレベルをお客様のニーズと整合させます。ユーザーがリソースを消費するために必要なネットワークを制限できるようにリソースを配置します。未使用の既存アセットを削除します。作成されたが未使用のアセットを特定し、作成を止めます。チームメンバーには、必要な機能をサポートし持続可能性への影響を最小限にするデバイスを提供します。

ベストプラクティス:

SUS02-BP01 ユーザーの負荷に合わせてインフラストラクチャをスケールする

使用率が低い、または使用されていない期間を特定し、リソースをスケールダウンして余分な容量を排除し効率性を改善します。

一般的なアンチパターン:

- ユーザーの負荷に合わせてインフラストラクチャをスケールしない。
- 常に手動でインフラストラクチャをスケールする。
- スケーリングイベントの後、スケールダウンして元に戻すのではなく、キャパシティーを増加させたままにする。

このベストプラクティスを活用するメリット: ワークロードの伸縮性を設定してテストすることで、ワークロードの環境への影響を削減し、費用を節減し、パフォーマンスベンチマークを維持できます。クラウドの伸縮性を活用して、ユーザーの負荷の急増時や急増後にキャパシティを自動的にスケールして、お客様のニーズを満たすために必要なリソースのみを正確に使用できるようにすることができます。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 伸縮性は、持っているリソースの供給を、それらのリソースに対する需要と一致させます。インスタンス、コンテナ、機能には、伸縮性のためのメカニズムがあり、自動スケーリングと組み合わせて、またはサービスの機能として提供されます。アーキテクチャの伸縮性を使用して、ユーザーの負荷が低い時間帯にワークロードを迅速かつ容易にスケールダウンできるようにします。

  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Amazon EC2 Auto Scaling を使用して、アプリケーションのユーザー負荷を処理するための適切な数の Amazon EC2 インスタンスがあることを確認できます。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Application Auto Scaling を使用して、個別の AWS サービス (Lambda 関数や Amazon Elastic Container Service (Amazon ECS) サービスなど) のリソースを、Amazon EC2 を超えて自動的にスケーリングします。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Kubernetes Cluster Autoscaler を使用して、AWS の Kubernetes クラスターを自動的にスケーリングします。
- スケールアップまたはスケールダウンのメトリクスが、デプロイされているワークロードのタイプに対して検証されていることを確認します。動画トランスコーディングアプリケーションをデプロイしようとする場合、100% の CPU 使用率が想定されるため、プライマリメトリクスにするべきではありません。必要な場合は、スケーリングポリシーとして カスタマイズされたメトリクス (メモリ使用率など) を使用できます。適切なメトリクスを選ぶには、Amazon EC2 の以下のガイダンスを考慮してください。
  - メトリクスは有効な利用率メトリクスでなければならず、インスタンスのどの程度ビジーかを記述する必要があります。
  - メトリクス値は、Auto Scaling グループ内のインスタンス数に比例して増減する必要があります。
- 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 手動スケーリング ではなく 動的スケーリング を Auto Scaling グループに使用します。また、動的スケーリングでは、 ターゲット追跡スケーリングポリシー を使用することをお勧めします。
- ワークロードのデプロイがスケールアップとスケールダウンの両方のイベントに対処できることを確認します。スケールダウンイベントのテストシナリオを作成して、ワークロードが期待どおりに動作することを確認します。専用のインフラストラクチャで アクティビティ履歴 を使用して、Auto Scaling グループのスケーリングアクティビティをテストし、確認することができます。
- ワークロードを評価して予測可能なパターンを見つけ、あらかじめわかっていた、および計画的な需要の変化を予測してプロアクティブにスケールします。予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Amazon EC2 Auto Scaling で予測スケーリング を使用して、キャパシティを誇張する必要性をなくします。

リソース

関連するドキュメント:

- Getting Started With Amazon EC2 Auto Scaling
- 機械学習を利用した EC2 の予測スケーリング
- Analyze user behavior using Amazon OpenSearch Service, Amazon Data Firehose and Kibana
- Amazon CloudWatch とは?
- 「AWS X-Ray とは何ですか。」
- VPC フローログ
- Amazon RDS での Performance Insights を使用した DB 負荷のモニタリング
- Amazon EC2 Auto Scaling での予測スケーリングのネイティブサポートの概要
- メモリ利用率メトリクスに基づいて Amazon EC2 Auto Scaling ポリシーを作成する方法 (Linux)
- Karpenter の概要 - オープンソースの高性能 Kubermetes Cluster Autoscaler

関連動画:

- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

関連する例:

- ラボ: Amazon EC2 Auto Scaling グループの例
- ラボ: Karpenter による自動スケーリングの実装

SUS02-BP02 SLA を持続可能性の目標に合わせる

可用性やデータ保持期間などに関するサービスレベルアグリーメント (SLA) を定義、更新し、ビジネス要件を満たしながら、ワークロードをサポートするために必要なリソース数を最小化します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- ビジネス要件を満たしながら持続可能性の目標をサポートする SLA を定義します。
- ビジネス要件を超えるのではなく、要件に合わせて SLA を再定義します。
- 持続可能性への影響を大幅に削減できる事項と、サービスレベルの許容できる範囲での低下をトレードオフします。
- ビジネスクリティカルな機能を優先し、クリティカルでない機能にはサービスレベル (応答時間や回復時間目標など) を引き下げる設計パターンを使用します。

リソース

関連するドキュメント:

- AWS サービスレベルアグリーメント (SLA)
- Importance of Service Level Agreement for SaaS Providers

関連動画:

- Building Sustainably on AWS (AWS でサステナブルに構築する)

SUS02-BP03 未使用アセットの創出と維持の停止

アプリケーションアセット (事前コンパイル済みのレポート、データセット、静的イメージなど) とアセットのアクセスパターンを分析し、冗長性、低使用率、および廃止できそうなターゲットを特定します。生成されたアセットを冗長性コンテンツ (重複または共通のデータセットと出力が含まれる月次レポートなど) と統合し、出力が重複する際に消費されるリソースをなくします。使用されていないアセット (既に販売していない製品の画像など) を廃止し、消費されていたリソースを解放して、ワークロードをサポートするために使用されるリソース数を削減します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 静的アセットを管理し、不要になったアセットは削除します。
- 生成されたアセットを管理し、不要になったアセットは生成を止めて削除します。
- 重複して生成されるアセットは統合し、冗長プロセスを排除します。
- 不要になったアセットをお客様に代わって管理しているサードパーティーに、アセットの生成と保存を止めるように指示します。
- サードパーティーに、お客様の代わりに生成されている冗長アセットを統合するように指示します。

リソース

関連するドキュメント:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage (サステナビリティのための AWS インフラストラクチャの最適化、パートII : ストレージ)

関連動画:

- Building Sustainably on AWS (AWS でサステナブルに構築する)

SUS02-BP04 ワークロードの地理的な配置を、ユーザーのロケーションに合わせて最適化する

ネットワークのアクセスパターンを分析し、顧客が地理的にどこから接続しているかを特定します。ネットワークトラフィックが経由しなければならない距離を削減できるリージョンとサービスを選択し、ワークロードをサポートするために必要なネットワークリソースの総量を減らします。

一般的なアンチパターン:

- 自分の場所に基づいてワークロードのリージョンを選択する。

このベストプラクティスを活用するメリット: ワークロードを顧客の近くに配置することで、ネットワーク上のデータ移動を減らし、環境負荷を低減しながら、最小限のレイテンシーを実現します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 以下の主な要素に基づいて、ワークロードのデプロイに適切なリージョンを選択します。
  - 持続可能性目標: 以下で説明されています リージョンの選択.
  - データの場所: 大量のデータを使用するアプリケーション (ビッグデータや機械学習など) では、アプリケーションコードをできるだけデータの近くで実行してください。
  - ユーザーの場所: ユーザー向けアプリケーションの場合は、ワークロードの顧客ベースに近いリージョンを選びます。
  - その他の制約: 以下で説明されているように、セキュリティやコンプライアンスなどの制約を考慮します。 What to Consider when Selecting a Region for your Workloads (ワークロードに応じたリージョンを選択する際の注意点).
- 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 AWS ローカルゾーン を使用して、動画レンダリングやグラフィックス集約的な仮想デスクトップアプリケーションなどのワークロードを実行します。ローカルゾーンでは、エンドユーザーの近くにコンピューティングリソースやストレージリソースがあるというメリットが得られます。
- ローカルキャッシュまたは AWS キャッシュソリューション を、頻繁に使用するリソースに使用すると、パフォーマンスを向上させ、データ移動を削減し、環境への影響を低減できます。

  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、Amazon CloudFront を使用すると、画像、スクリプト、動画などの静的コンテンツだけでなく、API 応答やウェブアプリケーションなどの動的コンテンツをキャッシュできます。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、Amazon ElastiCache を使用して、ウェブアプリケーションのコンテンツをキャッシュします。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、DynamoDB Accelerator を使用して、DynamoDB テーブルにインメモリアクセラレーションを追加します。
- ワークロードのユーザーの近くでコードを実行できるサービスを使用します。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、Lambda@Edge は、オブジェクトがキャッシュにないときに実行される、コンピューティング負荷の高いオペレーションに使用します。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、Amazon CloudFront 関数 は、HTTP リクエストまたはレスポンス操作など、短時間実行の関数で実行できるシンプルなユースケースに使用します。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、AWS IoT Greengrass を使用して、接続されたデバイスのローカルコンピューティング、メッセージング、データキャッシュを実行します。
- コネクションプーリングを使用して、接続の再利用を可能にし、必要なリソースを削減します。
- 永続的な接続や同期更新に依存しない分散されたデータストアを使用して、リージョンのユーザーに一貫性のあるサービスを提供します。
- 事前にプロビジョンされた静的ネットワーク容量を、共有の動的容量に置き換え、持続可能性に対するネットワーク容量の影響を他のサブスクライバーと共有します。

リソース

関連するドキュメント:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- Amazon ElastiCache のドキュメント
- 「Amazon CloudFront とは」
- Amazon CloudFront の主な特徴
- Lambda@Edge
- CloudFront 関数
- AWS IoT Greengrass

関連動画:

- Building Sustainably on AWS (AWS でサステナブルに構築する)

関連する例:

- AWS Networking Workshops

SUS02-BP05 実行されるアクティビティに応じてチームメンバーのリソースを最適化する

チームメンバーに提供されるリソースを最適化することで、ニーズをサポートしながら持続可能性への影響を最小限に抑えます。例えば、レンダリングやコンパイルなどの複雑なオペレーションを、使用率が低く高性能な単一ユーザーのシステムで行うのではなく、使用率の高い共有クラウドデスクトップで行います。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 使用方法に合わせてワークステーションや他のデバイスをプロビジョンします。
- 仮想デスクトップとアプリケーションストリーミングを使用して、アップグレードとデバイス要件を制限します。
- プロセッサーやメモリの負荷が大きいタスクをクラウドに移動します。
- デバイスのライフサイクルにおけるプロセスやシステムの影響を評価し、ビジネス要件を満たしながらデバイスを交換する必要性を最小限にするソリューションを選択します。
- デバイスのリモート管理を実装して出張を少なくします。

リソース

関連するドキュメント:

- Amazon WorkSpaces とは?
- Amazon AppStream 2.0 のドキュメント
- NICE DCV
- AWS Systems Manager Fleet Manager

関連動画:

- Building Sustainably on AWS (AWS でサステナブルに構築する)

ソフトウェアとアーキテクチャのパターン

質問

- SUS 3 ソフトウェアとアーキテクチャのパターンをどのように利用して、持続可能性目標を目指しますか?

SUS 3 ソフトウェアとアーキテクチャのパターンをどのように利用して、持続可能性目標を目指しますか?

負荷平滑化を実行しデプロイされたリソースが一貫して高使用率で維持されるパターンを実装し、リソースの消費を最小化します。時間の経過とともにユーザーの行動が変化したため、コンポーネントが使用されずアイドル状態になることがあります。パターンとアーキテクチャを改定して、使用率の低いコンポーネントを統合し、全体の使用率を上げます。不要になったコンポーネントは使用停止にします。ワークロードコンポーネントのパフォーマンスを理解し、リソースの消費が最も大きいコンポーネントを最適化します。顧客がお客さまのサービスにアクセスするために使用するデバイスを把握し、デバイスをアップグレードする必要性を最小化するパターンを実装します。

ベストプラクティス:

SUS03-BP01 非同期のジョブおよびスケジュールされたジョブ向けにソフトウェアとアーキテクチャを最適化する

ソフトウェアの効率的な設計とアーキテクチャを使用し、作業単位で必要な平均リソースを最小化します。コンポーネントの使用率が平均化されるメカニズムを実装し、タスクの合間でアイドルになるリソースを削減して、負荷のスパイクの影響を最小化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 即時処理を必要としないリクエストをキューに入れます。
- 直列化を増やしてパイプライン全体の使用率を平均化します。
- 個別のコンポーネントの容量を変更し、リソースが入力を待ってアイドル状態になるのを防ぎます。
- バッファを作成し、レート制限を設定して外部サービスの消費を円滑化します。
- 使用できる中で最も効率的なハードウェアを使用し、ソフトウェアを最適化します。

- キュー駆動型アーキテクチャ、パイプライン管理、オンデマンドインスタンスワーカーを使用して、バッチ処理の使用率を最大化します。
- タスクをスケジュールして、同時実行による負荷のスパイクとリソースの競合を避けます。
- 電力の炭素強度が最も低い時間帯にジョブを処理します。

リソース

関連するドキュメント:

- Amazon Simple Queue Service とは?
- Amazon MQ とは?
- Amazon SQS に基づいたスケーリング
- AWS Step Functions とは?
- AWS Lambda とは?
- AWS Lambda を Amazon SQS に使用する
- Amazon EventBridge とは?

関連動画:

- Building Sustainably on AWS
- Moving to event-driven architectures

SUS03-BP02 使用率が低い、またはまったく使用しないワークロードのコンポーネントを削除またはリファクタリングする

ワークロードアクティビティをモニターして、個別のコンポーネントの時間経過による使用率の変化を特定します。未使用のコンポーネントや不要になったコンポーネントを削除し、使用率の低いコンポーネントはリファクタリングして、無駄なリソースを制限します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 機能コンポーネントの負荷を (トランザクションフローや API コールなどのインジケーターを使用して) 分析し、未使用および使用率の低いコンポーネントを特定します。
- 不要になったコンポーネントは使用停止にします。

- 使用率の低いコンポーネントはリファクタリングします。
- 使用率の低いコンポーネントを他のリソースと統合して、使用効率を改善します。

リソース

関連するドキュメント:

- 「AWS X-Ray とは何ですか。」
- Amazon CloudWatch とは?
- ServiceLens を使用してアプリケーションのヘルスをモニターする
- Automated Cleanup of Unused Images in Amazon ECR (Amazon ECR における未使用画像の自動クリーンアップ)

関連動画:

- Building Sustainably on AWS (AWS でサステナブルに構築する)

SUS03-BP03 時間やリソースを最も多く消費するコード領域を最適化する

ワークロードアクティビティをモニターして、リソースの消費が最も大きいアプリケーションコンポーネントを特定します。それらのコンポーネント内で実行されているコードを最適化して、パフォーマンスを最大化しながらリソースの使用量を最小化します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- リソース使用率が作用するパフォーマンスをモニターして、作業単位でリソースを多く必要とするコンポーネント特定し、最適化の対象とします。
- コードプロファイラーを使用して、時間またはリソースを最も多く使用するコードの領域を特定し、最適化の対象とします。
- アルゴリズムを同じ結果が生成される、より効率的なバージョンに置き換えます。
- ハードウェアアクセラレーションを使用して、実行時間が長いコードブロックの効率を改善します。
- 最も効率的なオペレーティングシステムとプログラム言語をワークロードに使用します。
- 不要なソートと書式設定を削除します。

- データの変更頻度と消費方法に基づいて使用されるリソースを最小化するデータ転送パターンを使用します。例えば、状態変更情報は、クライアントがリソースを消費してポーリングし価値のない「変更なし」のメッセージを受信するのではなく、クライアントにプッシュします。

リソース

関連するドキュメント:

- Amazon CloudWatch とは?
- Amazon CodeGuru Profiler とは?
- FPGA インスタンス
- The AWS SDKs on Tools to Build on AWS (AWS の構築ツールの AWS SDK)

関連動画:

- Building Sustainably on AWS (AWS でサステナブルに構築する)

SUS03-BP04 お客様のデバイスや機器への影響を最適化する

お客さまのサービスを利用するために顧客が使用しているデバイスや機器、その予想ライフサイクル、およびそれらのコンポーネントを交換することによる金融および持続可能性に対する影響を理解します。顧客のデバイスの交換や機器のアップグレードの必要性を最小化するソフトウェアパターンとアーキテクチャを実装します。例えば、新機能の実装には、より古いハードウェアやオペレーティングシステムのバージョンと後方互換性のあるコードを使用します。また、ペイロードのサイズを管理して、対象デバイスのストレージ容量を超えないようにします。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 顧客が使用するデバイスのインベントリを作成します。
- マネージド型 Device Farm を代表的なハードウェアセットと共に使用してテストを行い、変更の影響を理解して、サポート対象のデバイスを最大化する開発を繰り返します。
- ペイロードを構築する際にネットワーク帯域幅とレイテンシーを考慮し、低帯域幅、高レイテンシーのリンクでもアプリケーションが問題なく動作できる能力を実装します。
- データペイロードを事前に処理して、ローカルでの処理要件を削減し、データ転送要件を制限します。

- コンピューティングの負荷が高いアクティビティはサーバー側 (画像のレンダリングなど) で実行するか、アプリケーションストリーミングを使用して、古い型のデバイスでのユーザーエクスペリエンスを改善します。
- 特にインタラクティブセッションの場合は、出力を分割してページ番号を付け、ペイロードを管理しローカルストレージの要件を制限します。

リソース

関連するドキュメント:

- AWS Device Farm とは?
- Amazon AppStream 2.0 のドキュメント
- NICE DCV
- Amazon Elastic Transcoder のドキュメント

関連動画:

- Building Sustainably on AWS

SUS03-BP04 データアクセスとストレージパターンのサポートが最も優れたソフトウェアパターンとアーキテクチャを使用する

データがどのようにワークロード内で使用されているか、ユーザーに消費されているか、転送されているか、保存されているかを理解します。データの処理と保存の要件を最小化するテクノロジーを選択します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- データアクセスとストレージパターンを分析します。
- 不要な処理を (分析の実行時などに) 回避するため、Parquet などの効率的なファイル形式でデータファイルを保存し、プロビジョンする合計ストレージを削減します。
- 圧縮データをネイティブに操作するテクノロジーを使用します。
- 主要なクエリパターンに対して最も優れたサポートをするデータベースエンジンを使用します。
- データベースインデックスを管理してインデックスの設計が効率的なクエリ実行を確実にサポートするようにします。

- 消費されるネットワーク容量が削減できるネットワークプロトコルを選択します。

リソース

関連するドキュメント:

- Athena でサポートされる圧縮ファイル形式
- Amazon Redshift を使用した列指向のデータ形式からの COPY
- Firehose で入力レコード形式を変換する
- AWS Glue の ETL 入出力の形式オプション
- 列指向形式に変換して Amazon Athena でのクエリパフォーマンスを改善する
- Amazon Redshift を使用して圧縮されたデータファイルを Amazon S3 からロードする
- Amazon Aurora での Performance Insights を使用した DB 負荷のモニタリング
- Amazon RDS での Performance Insights を使用した DB 負荷のモニタリング
- AWS IoT FleetWise

関連動画:

- Building Sustainably on AWS

データパターン

質問

- SUS 4 データのアクセスパターンおよび使用パターンをどのように利用して、持続可能性目標を目指しますか?

SUS 4 データのアクセスパターンおよび使用パターンをどのように利用して、持続可能性目標を目指しますか?

データ管理プラクティスを実装して、ワークロードのサポートに必要なプロビジョンされたストレージと、それを使用するために必要なリソースを削減します。データを理解し、データのビジネス価値とデータの使用方法を最もよくサポートするストレージテクノロジーと設定を使用します。必要性が小さくなった場合はより効率的で性能を落としたストレージにデータをライフサイクルし、データが不要になった場合は削除します。

ベストプラクティス:

SUS04-BP01 データ分類ポリシーを実装する

データを分類して、ビジネス成果にとっての重要性を理解します。この情報を使用して、データをより エネルギー効率が高いストレージに移動するタイミングや、安全に削除するタイミングを検討します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- データのディストリビューション、保持、削除の要件を検討します。
- ボリュームやオブジェクトへのタグ付けを使用してメタデータを記録し、それを使用してデータ分類を含む管理方法を検討します。
- 環境を定期的に監査してタグ付けおよび分類されていないデータを探し、そのデータを適切に分類してタグ付けします。

リソース

関連するドキュメント:

- Data Classification Process
- Leveraging AWS クラウド to Support Data Classification
- AWS Organizations のタグポリシー

SUS04-BP02 データのアクセスパターンとストレージパターンをサポートするテクノロジーを使用する

データへのアクセス方法や保存方法をもっともよくサポートするストレージを使用し、ワークロードをサポートしながらプロビジョニングされるリソースを最小化します。例えば、ソリッドステートドライブ (SSD) は磁気式ドライブよりもエネルギー消費が大きいため、アクティブなデータのユースケースのみに使用するべきです。アクセスの少ないデータに対して、エネルギー効率の高いアーカイブクラスのストレージを使用します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- データのアクセスパターンをモニタリングします。

- アクセスパターンに基づいて適切なテクノロジーにデータを移行します。
- アーカイブデータを目的に合わせて設計されたストレージに移行します。

リソース

関連するドキュメント:

- Amazon EBS ボリュームタイプ
- Amazon EC2 インスタンスストア
- Amazon S3 Intelligent-Tiering
- Amazon S3 のストレージクラスを使用する
- Amazon CloudWatch とは?
- Amazon S3 Glacier とは?

関連動画:

- Architectural Patterns for Data Lakes on AWS

SUS04-BP03 ライフサイクルポリシーを使用して不要なデータを削除する

データすべてのライフサイクルを管理し、削除のタイムラインを自動的に適用して、ワークロードに必要な合計ストレージを最小化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- すべてのデータ分類タイプのライフサイクルポリシーを定義します。
- ライフサイクルルールを適用するための自動ライフサイクルポリシーを設定します。
- 未使用のボリュームとスナップショットを削除します。
- ライフサイクルルールに基づいて、該当する場合はデータを集約します。

リソース

関連するドキュメント:

- Amazon ECR ライフサイクルポリシー

- Amazon EFS ライフサイクル管理
- Amazon S3 Intelligent-Tiering
- AWS Config ルールを使用してリソースを評価する
- Amazon S3 でストレージのライフサイクルを管理する
- AWS Elemental MediaStore のオブジェクトライフサイクルポリシー

関連動画:

- Amazon S3 Lifecycle

SUS04-BP04 ブロックストレージの過剰プロビジョニングを最小化する

プロビジョニングされる合計ストレージを最小化するには、ワークロードに適したサイズ割り当てのブロックストレージを作成します。伸縮自在なボリュームを使用し、データの増加に合わせて、コンピューティングリソースに添付されたストレージをサイズ変更することなく拡張します。伸縮自在なボリュームを定期的に確認し、現在のデータサイズに合わせてプロビジョンされたボリュームを縮小します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- データボリュームの使用率をモニターします。
- 伸縮自在なボリュームとマネージド型のブロックデータサービスを使用して、永続的データの増加に応じて追加のストレージの割り当てを自動化します。
- データボリュームの使用率の目標レベルを設定し、予想される範囲外のボリュームはサイズ変更します。
- データに合わせて読み取り専用ボリュームのサイズを設定します。
- データをオブジェクトストアに移行して、ブロックストレージの固定ボリュームサイズを超える容量をプロビジョンするのを回避します。

リソース

関連するドキュメント:

- Amazon EBS Elastic Volumes
- Amazon FSx のドキュメント

- Amazon CloudWatch とは?
- Amazon Elastic File System とは?

SUS04-BP04 不要なデータや重複するデータを削除する

データの複製は必要なときにのみ行い、消費される合計ストレージを最小化します。ファイルおよびブロックレベルでデータの重複を排除するバックアップテクノロジーを使用します。サービスレベルアグリーメント (SLA) の要件を満たすために必要な場合を除き、RAID (Redundant Array of Independent Drives) 設定の使用を制限します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- ブロックレベルとオブジェクトレベルでデータを重複排除できる仕組みを使用します。
- ブロック、ファイル、オブジェクトレベルでデータの増分のバックアップおよび重複排除を実行できるバックアップテクノロジーを使用します。
- RAID は SLA を満たすために必要な場合にのみ使用します。
- ログおよび追跡データを一元化し、同一のログエントリの重複を排除して、必要に応じて冗長性を調整するメカニズムを確立します。
- キャッシュを事前入力は、正当な場合にのみ行います。
- キャッシュのモニタリングとオートメーションを確立し、それに従ってキャッシュをサイズ変更します。
- ワークロードの新しいバージョンをプッシュする際に、オブジェクトストアとエッジキャッシュから古いデプロイとアセットを削除します。

リソース

関連するドキュメント:

- Amazon EBS スナップショット
- CloudWatch Logs のログデータ保持期間を変更する
- Amazon FSx for Windows File Server でのデータの重複排除
- データの重複排除を含む Amazon FSx for ONTAP の機能
- Amazon CloudFront でのファイルの無効化
- AWS Backup を使用してバックアップを行い、Amazon EFS ファイルシステムを復元する

- Amazon CloudWatch Logs とは?
- Amazon RDS でのバックアップの操作

関連する例:

- ラボ: Amazon Redshift データ共有を使用したデータパターンの最適化

SUS04-BP06 共有ファイルシステムまたはオブジェクトストレージを使用して共通データにアクセスする

共有ストレージと単一の信頼できるソースを採用し、データの複製を避けてワークロードに必要な合計ストレージを削減します。必要に応じて共有ストレージからデータを取得します。未使用のボリュームをデタッチして利用可能なリソースを増やします。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- データに複数のコンシューマーが存在する場合は、データを共有ストレージに移行します。
- 必要に応じて共有ストレージからデータを取得します。
- 使用パターンに合わせてデータを削除し、有効期限 (TTL) 機能を導入してキャッシュされたデータを管理します。
- クライアントがアクティブに使用していないボリュームをクライアントからデタッチします。

リソース

関連するドキュメント:

- Amazon FSx
- キャッシング戦略
- Amazon Elastic File System とは?
- Amazon S3 とは?

SUS04-BP07 ネットワーク間でのデータ移動を最小限に抑える

共有ストレージを使用し、その地域のデータストアからデータにアクセスして、ワークロードにおけるデータ移動をサポートするために必要なネットワークリソースの総量を最小化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 可能な限りコンシューマーの近くにデータを保存します。
- リージョン固有のデータは、消費するリージョン内に保存されるので、パーティションは、リージョンでサービスを消費します。
- ネットワーク全体で変更をコピーする場合、ファイルまたはオブジェクトレベルの重複ではなくブロックレベルの重複を使用します。
- データを圧縮してからネットワーク経由で移動します。

リソース

関連するドキュメント:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- AWS グローバルインフラストラクチャ
- Amazon CloudFront の主な特徴 (CloudFront グローバルエッジネットワークなど)
- Amazon OpenSearch Service での HTTP リクエストの圧縮
- Amazon EMR を使用して中間データを圧縮する
- 圧縮されたデータファイルを Amazon S3 から Amazon Redshift にロードする
- Amazon CloudFront を使用して圧縮ファイルを提供する

SUS04-BP08 データは再作成が難しい場合にのみバックアップする

ストレージの消費を最小化するには、ビジネス価値のあるデータまたはコンプライアンス要件を満たすために必要なデータのみをバックアップします。バックアップポリシーを精査し、リカバリーシナリオでは価値のないエフェメラルストレージを除外します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- データ分類を使用して、バックアップが必要なデータを設定します。
- 簡単に再作成できるデータを除外します。
- バックアップから一時データを除外します。

- 共通の場所からデータを復元するために必要な時間がサービスレベルアグリーメント (SLA) を超える場合を除き、データのローカルコピーを除外します。

リソース

関連するドキュメント:

- AWS Backup を使用してバックアップを行い、Amazon EFS ファイルシステムを復元する
- Amazon EBS スナップショット
- Amazon Relational Database Service でのバックアップの操作

ハードウェアパターン

質問

- SUS 5 ハードウェアの管理および使用のプラクティスは、持続可能性目標を目指すうえでどのように役立ちますか?

SUS 5 ハードウェアの管理および使用のプラクティスは、持続可能性目標を目指すうえでどのように役立ちますか?

ハードウェア管理のプラクティスを変更することで、ワークロードの持続可能性に対する影響を軽減する機会を探します。プロビジョンおよびデプロイする必要があるハードウェア数を最小化し、個別のワークロードにおいて最も効率のいいハードウェアを選択します。

ベストプラクティス:

SUS05-BP01 ニーズに合わせて最小限のハードウェアを使用する

クラウドの能力を使用して、ワークロードの実装を頻繁に変更できます。ニーズの変化に応じて、デプロイされたコンポーネントを更新します。

このベストプラクティスを活用しない場合のリスクレベル: ミディアム

実装のガイダンス

- 水平スケーリングを有効にし、オートメーションを使用して、負荷の増加に応じてスケールアウトし、負荷の減少に応じてスケールインします。

- ワークロードの変動に合わせて少しずつスケールします。
- スケーリングを周期的な使用パターンに合わせます (例えば、ペイロードシステムを隔週の負荷の高いプロセッシングアクティビティに合わせます)。負荷は日、週、月、年によって異なるためです。
- 容量の一時的な削減ができるようにサービスレベルアグリーメント (SLA) を交渉すると同時に、オートメーションを使用して代替リソースをデプロイします。

リソース

関連するドキュメント:

- AWS Compute Optimizer のドキュメント
- Lambda のオペレーション: パフォーマンスの最適化
- Auto Scaling のドキュメント

SUS05-BP02 影響が最も少ないインスタンスタイプを使用する

新しいインスタンスタイプのリリースを継続的にモニタリングし、機械学習のトレーニング、推論、ビデオのトランスコーディングなどの特定のワークロードをサポートするように設計されたインスタンスタイプを含む、エネルギー効率の改善を活用します。

一般的なアンチパターン:

- インスタンスの 1 つのファミリーのみを使用する。
- x86 インスタンスのみを使用する。
- Amazon EC2 Auto Scaling 設定で 1 つのインスタンスタイプを指定する。
- AWS インスタンスが設計されていない方法で使用されている (たとえば、メモリ集中型のワークロードに計算用に最適化されたインスタンスを使用した場合)。
- 新しいインスタンスタイプを定期的に評価しない。
- 次のような AWS サイズ最適化ツールからのレコメンデーションを確認しない : AWS Compute Optimizer。

このベストプラクティスを活用するメリット: エネルギー効率が高く、適切なサイズのインスタンスを使用することで、環境への影響とワークロードのコストを大幅に削減できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- ワークロード環境への影響を削減できるインスタンスタイプを学習し、試します。
  - 例えば、 AWS の最新情報 を購読して、最新の AWS テクノロジーとインスタンスの情報を入手します。
  - さまざまな AWS インスタンスタイプについて学びます。
  - Amazon EC2 のエネルギー使用量 1 ワットあたりで最高のパフォーマンスを発揮する AWS Graviton ベースのインスタンスについて、以下のビデオをご覧ください。 re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances (AWS Graviton2 プロセッサ搭載 Amazon EC2 インスタンスの詳細) と Deep dive into AWS Graviton3 and Amazon EC2 C7g instances.
- ワークロードを計画し、最も影響の少ないインスタンスタイプに移行することができます。
  - ワークロードの新機能やインスタンスを評価するプロセスを定義します。クラウドの俊敏性を利用して、新しいインスタンスタイプがワークロード環境の持続可能性をどのように改善するかをすばやくテストします。プロキシメトリクスを使用して、1 つの作業単位を完了するのに必要なリソース数を測定します。
  - 可能な場合は、異なる数の vCPU と異なる量のメモリで動作するようにワークロードを変更して、インスタンスタイプの選択肢を最大化します。
  - ワークロードのパフォーマンス効率を向上させるために、Graviton ベースのインスタンスへの移行を検討します (AWS Graviton Fast Start と AWS Graviton2 for ISVs を参照) を指定する必要があります。ワークロードを AWS Graviton ベースの Amazon Elastic Compute Cloud インスタンスに移行する際の考慮事項を覚えておいてください。
  - 以下の利用において、AWS Graviton オプションの選択を検討します : AWS マネージドサービス。
  - 持続可能性に対する影響が最も少なく、かつビジネス要件を満たすインスタンスを提供するリージョンにワークロードを移行します。
  - 機械学習ワークロードの場合は、次のようなカスタム Amazon Machine Learning チップに基づく Amazon EC2 インスタンスを使用します : AWS Trainiumio1 AWS Inferentia、および Amazon EC2 DL1。
  - 予想されるコストと使用状況に合わせたカスタムの予算を設定するには、 Amazon SageMaker Inference Recommender を使用して、機械学習推論エンドポイントのサイズを適切に設定します。
  - リアルタイムの動画トランスコーディングを伴うワークロードについては、 Amazon EC2 VT1 インスタンスを使用します。

  - スパイキーなワークロード (追加のキャパシティが必要になることがあまりないワークロード) の場合は、 バーストパフォーマンスインスタンスを使用します。
  - ステートレスで耐障害性の高いワークロードについては、 Amazon EC2 スポットインスタンス を使用して、クラウドの全体使用率を高め、未使用のリソースの持続可能性に対する影響を軽減します。
- ワークロードインスタンスを操作して、最適化します。
  - 一次的なワークロードについては、 インスタンス Amazon CloudWatch メトリクス (`CPUUtilization` など) を評価して、インスタンスがアイドルか使用率が低いかを識別します。
  - 安定したワークロードの場合は、AWS サイズ適正化ツール (AWS Compute Optimizer など) を定期的にチェックし、インスタンスの最適化とサイズ適正化の機会を識別します。

リソース

関連するドキュメント:

- 持続可能性のために AWS インフラストラクチャを最適化する、パート I: コンピューティング
- AWS Graviton プロセッサ
- AWS Inferentia
- AWS Trainium
- Amazon EC2 DL1
- Amazon EC2 バーストパフォーマンスインスタンス
- Amazon EC2 キャパシティ予約フリート
- Amazon EC2 スポットフリート
- Amazon EC2 スポットインスタンス
- Amazon EC2 VT1 インスタンス
- Amazon EC2 インスタンスタイプ
- AWS Compute Optimizer
- 関数: Lambda 関数の設定

関連動画:

- Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances (AWS Graviton2 プロセッサ搭載 Amazon EC2 インスタンスの詳細)

- Deep dive into AWS Graviton3 and Amazon EC2 C7g instances

関連する例:

- ラボ: 適切なサイズのレコメンデーション
- ラボ: Compute Optimizer によるサイズ適正化
- ラボ: ハードウェアパターンの最適化と持続可能性 KPI の観察

SUS05-BP03 マネージドサービスを使用する

マネージドサービスは、平均使用率を高く保つ責任と、デプロイされたハードウェアの持続可能性に対する最適化の責任を AWS に移します。マネージドサービスを使用して、サービスの持続可能性に対する影響を、そのサービスのすべてのテナント間に分散し、お客様単体の関与を軽減します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- 自己ホスト型サービスからマネージドサービスに移行します。例えば、マネージド型の Amazon Relational Database Service (Amazon RDS) インスタンスを使用するか (独自の Amazon RDS インスタンスを Amazon Elastic Compute Cloud (Amazon EC2) で維持する代わりに)、あるいは AWS Fargateのようなマネージドコンテナサービスを (独自のコンテナインフラストラクチャを実装する代わりに) 使用します。

リソース

関連するドキュメント:

- AWS Fargate
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Amazon Redshift
- Amazon Relational Database Service (RDS)

SUS05-BP04 GPU の使用を最適化する

グラフィック処理ユニット (GPU) は高電力消費のソースになることがあります。GPU ワークロードの種類は多く、レンダリング、トランスコーディング、機械学習トレーニング、モデリングなどさまざまです。GPU インスタンスは必要な時間だけ実行し、必要がないときはオートメーションで廃棄して、消費されるリソースを最小化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- GPU は、CPU ベースの代替手段より効率的な場合にのみ、タスクに使用します。
- 使用しないときは、オートメーションを使用して GPU インスタンスを解放します。
- 専用の GPU インスタンスではなく、柔軟なグラフィックスアクセラレーションを使用します。
- ワークロードに特化したカスタムの専用ハードウェアを活用します。

リソース

関連するドキュメント:

- 高速コンピューティング
- AWS Inferentia
- AWS Trainium
- EC2 インスタンスの高速コンピューティング
- Amazon EC2 VT1 インスタンス
- Amazon Elastic Graphics

開発とデプロイのプロセス

質問

- SUS 6 開発およびデプロイのプロセスは、持続可能性目標を目指すうえでどのように役立ちますか?

SUS 6 開発およびデプロイのプロセスは、持続可能性目標を目指すうえでどのように役立ちますか?

開発、テスト、デプロイのプラクティスを変更することで、持続可能性に対する影響を減らす機会を探します。

ベストプラクティス:

SUS06-BP01 持続可能性の改善を迅速に導入できる方法を採用する

本稼働環境にデプロイする前に、潜在的な改善をテストして検証します。改善に際して将来的に起こりうる利点を計算する際のテストにかかるコストを考慮します。低コストのテスト方法を開発し、小規模な改善を実施します。

このベストプラクティスが確立されていない場合のリスクレベル: ミディアム

実装のガイダンス

- 持続可能性の要件を開発プロセスに追加します。
- 持続可能性の改善策を開発、テスト、デプロイするために、リソースを並行して働かせることができるようにします。
- 本稼働環境にデプロイする前に、持続可能性に対する影響をもたらしうる改善をテストして検証します。
- 最小限に実行可能である代表的なコンポーネントを使用して、潜在的な改善をテストします。
- テスト済みの持続可能性の改善が利用可能になったら本番環境にデプロイします。

リソース

関連するドキュメント:

- AWS enables sustainability solutions (AWS が実現するサステナビリティソリューション)

関連する例:

- ラボ: コストと使用状況レポートを効率性レポートに変える

SUS06-BP02 ワークロードを最新に保つ

最新のオペレーティングシステム、ライブラリ、およびアプリケーションを使用すると、ワークロードの効率が上がり、さらに効率的なテクノロジーを簡単に導入できます。最新のソフトウェアにはまた、ワークロードの持続可能性に対する影響をより正確に測定する機能が含まれている場合があります。これは、ベンダーが独自の持続可能性の目標を満たすための機能でもあります。

一般的なアンチパターン:

- 現在のアーキテクチャは、時間が経つと更新されずに静的なものになると想定している。
- 更新されたソフトウェアおよびパッケージがワークロードと互換性があるかどうかを評価するためのシステムまたは定期的なミーティングがない。
- あなたは、理由なしで、時間の経過とともにアーキテクチャの変更を導入します。

このベストプラクティスを活用するメリット: ワークロードを最新に保つプロセスを確立することで、新しい機能と能力を採用し、問題を解決し、ワークロードの効率性を高めることができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

- ワークロードに応じた新しい機能やインスタンスを評価するプロセスとスケジュールを定義します。クラウドの俊敏性を利用して、新しい機能がワークロードをどのように改善するかをすばやくテストします。
  - 持続可能性への影響を削減する。
  - パフォーマンスの効率を高める。
  - 計画した改善にとっての障壁を取り除く。
  - 持続可能性に対する影響の測定能力と管理能力を高める。
- ワークロードソフトウェアおよびアーキテクチャをインベントリに登録して、更新する必要があるコンポーネントを特定する。専用のインフラストラクチャで AWS Systems Manager インベントリ を使用して、Amazon EC2 インスタンスからオペレーティングシステム (OS)、アプリケーション、インスタンスのメタデータを収集し、どのインスタンスがソフトウェアポリシーで要求されるソフトウェアと設定を実行しているか、どのインスタンスがアップデートする必要があるかを迅速に把握することが可能です。
- ワークロードのコンポーネントを更新する方法を理解します。
  - Linux または Windows サーバーイメージ向けの Amazon Machine Images (AMI) の更新を管理するには、以下を使用します EC2 Image Builder.

  - 既存のパイプラインで Amazon Elastic Container Registry (Amazon ECR) を使用して、 Amazon Elastic Container Service (Amazon ECS) イメージの管理と と Amazon Elastic Kubernetes Service イメージの管理ができます。
  - AWS Lambda には、 バージョン管理機能が含まれています。
- 更新プロセスにオートメーションを使用して、新しい機能をデプロイする労力のレベルを軽減し、手動プロセスに起因するエラーを抑制します。例えば AWS Systems Manager パッチマネージャー などのツールを使用して、システム更新のプロセスを自動化し、 AWS Systems Manager メンテナンスウィンドウを使用してアクティビティをスケジューリングします.

リソース

関連するドキュメント:

- AWS アーキテクチャセンター
- AWS の最新情報
- AWS デベロッパーツール
- AWS Systems Manager パッチマネージャー

関連する例:

- Well-Architected ラボ: インベントリおよびパッチ管理
- ラボ: AWS Systems Manager

SUS06-BP03 ビルド環境の利用率を高める

オートメーションと Infrastructure as Code を使用して、必要に応じて本番稼働前の環境を起動し、使用しないときは停止します。一般的なパターンとしては、開発チームのメンバーの勤務時間と重なるように可用性期間のスケジュールを設定することがあります。休止は、状態を維持し、必要なときにのみインスタンスを迅速にオンラインにする便利な手段です。バーストキャパシティ、スポットインスタンス、伸縮自在なデータベースサービス、コンテナ、その他のテクノロジーを備えたインスタンスタイプ使用して、開発およびテストのキャパシティを使用に合わせて調整します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- オートメーションを使用して、開発環境とテスト環境を最大限に活用します。

- オートメーションを使用して開発環境とテスト環境のライフサイクルを管理します。
- 最小限に実行可能である代表的な環境を使用して、潜在的な改善を開発およびテストします。
- オンデマンドインスタンスを使用してデベロッパーのデバイスを支給します。
- オートメーションを使用して、ビルドリソースの効率を最大化します。
- バーストキャパシティ、スポットインスタンス、その他のテクノロジーを備えたインスタンスタイプを使用して、ビルドキャパシティを使用状況に合わせて調整します。
- 要塞ホストのフリートをデプロイするのではなく、ネイティブなクラウドサービスを採用して、インスタンスシェルのアクセスを保護します。

リソース

関連するドキュメント:

- AWS Systems Manager Session Manager
- Amazon EC2 バーストパフォーマンスインスタンス
- AWS CloudFormation とは?

SUS06-BP04 マネージド型 Device Farm を使用してテストする

マネージド型の Device Farm は、ハードウェアの製造やリソースの使用の持続可能性に対する影響を複数のテナントに分散させます。マネージド型 Device Farm は、さまざまなデバイスタイプを提供するため、あまり使われない古いハードウェアをサポートすることで、不要なデバイスのアップグレードによるお客様の持続可能性に対する影響を回避できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

マネージド型 Device Farm を代表的なハードウェアセットと共に使用してテストを行い、変更の影響を理解して、サポート対象のデバイスを最大化する開発を繰り返します。

リソース

関連するドキュメント:

- AWS Device Farm とは?

注意

お客様は、この文書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤーまたはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または暗示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。