Unable to locate subtitle

AWS Well-Architected Framework

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

概要とイントロダクション 1
はじめに 1
定義 2
アーキテクチャについて 4
一般的な設計原則 6
フレームワークの柱 7
運用上の優秀性 7
設計原則 8
定義 8
ベストプラクティス 9
リソース 17
セキュリティ 18
設計原則 18
定義 19
ベストプラクティス 19
リソース 26
信頼性 26
設計原則 27
定義 28
ベストプラクティス 28
リソース 33
パフォーマンス効率 34
設計原則 34
定義 35
ベストプラクティス 35
リソース 43
コスト最適化 43
設計原則 44
定義 45
ベストプラクティス 45
リソース 51
サステナビリティ 51
設計原則 52
定義 53

ベストプラクティス 53
レビュープロセス 61
まとめ 63
寄稿者 64
その他の資料 65
改訂履歴 66
付録: 質問とベストプラクティス 69
運用上の優秀性 69
組織 69
準備 104
運用 167
進化 208
セキュリティ 224
セキュリティの基礎 224
ID とアクセス管理 242
検知 285
インフラストラクチャ保護 295
データ保護 314
インシデント対応 336
アプリケーションのセキュリティ 352
信頼性 371
基盤 372
ワークロードアーキテクチャ 412
変更管理 452
障害管理 484
パフォーマンス効率 573
選択 573
レビュー 671
モニタリング 676
トレードオフ 686
コスト最適化 696
クラウド財務管理を実践する 696
経費支出と使用量の認識 718
費用対効果の高いリソース 756
需要を管理しリソースを供給する 786
継続的最適化 795

サステナビリティ 803
リージョンの選択 804
需要に合わせた調整 806
ソフトウェアとアーキテクチャ 819
データ 830
ハードウェアとサービス 850
プロセスと文化 860
注意 868

AWS Well-Architected Framework

発行日: 2023 年 4 月 10 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。このフレームワークを使用することによって、信頼性が高く、安全で、効率的で、費用対効果が高く、持続可能なシステムを設計して運用するための、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。

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 ホームページを参照してください。

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。AWS Well-Architected Tool (AWS WA Tool) は、AWS Well-Architected フレームワークを使ってアーキテクチャをレビューし、測定するための一貫したプロセスを提供する、クラウド内のサービスです。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サポートするため、AWS Well-Architected Labs を作成しました。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、AWS Well-Architected パートナープログラムのメンバーである 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。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 1.AWS Well-Architected フレームワークの柱

名前	説明
運用上の優秀性	The ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.
セキュリティ	The security pillar describes how to take advantage of cloud technologies to protect

名前	説明
	data, systems, and assets in a way that can improve your security posture.
信頼性	The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle. This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS.
パフォーマンス効率	The ability to use computing resources efficient ly to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
コスト最適化	The ability to run systems to deliver business value at the lowest price point.
持続可能性	The ability to continually improve sustainability impacts by reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from the provisioned resources and minimizing the total resources required.

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、ビジネス価値を提供する一連のコンポーネントを識別するために使用します。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。
- マイルストーンは、アーキテクチャにおける設計、実装、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、ビジネスの運営に必要なワークロードの集合体です。
- 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 または Zachman Framework 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 に、AWS には各チームにそ

の機能を持たせることに焦点を当てたプラクティス (方法、プロセス、標準、受け入れられている基準) があり、エキスパートを配置して、チームが満たすべき基準を引き上げます。2 番目は、メカニズムを実装し、自動チェックを実行して、基準が確満たされるようにします。

> 「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。この分散型アプローチは、Amazon のリーダーシップの原則によってサポートされており、顧客を起点として行動するすべての役割において文化を確立しています。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、またはキャパシティ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。
- 自動化でアーキテクチャ実験を容易にする: 自動化により、低コストでワークロードを作成およびレプリケートすると、手作業でかかるコストを回避できます。#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
- 準備
- 運用
- 進化

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 サポート 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 サポート センターから AWS サポート 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。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 Quick 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
- 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ローテーションなど、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
- 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
- ワークロードアーキテクチャ
- 変更管理
- 障害管理

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
- 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
- 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

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 Quick 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
- 定義
- ベストプラクティス

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

- 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
ホワイトペーパーの更新	規範ガイダンスを使用してベストプラクティスを更新、および新しいベストプラクティスを追加。質問 COST 11 を追加。	April 10, 2023
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	July 1, 2019

	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。	
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。	November 1, 2018
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

この付録は、AWS Well-Architected フレームワークにあるすべての質問とベストプラクティスをまとめたものです。

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。実装に関する規範的なガイダンスについては、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。

ベストプラクティス領域

- 組織
- 準備
- 運用
- 進化

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?
- 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?
- 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?

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

ガバナンスとは、企業がビジネス目標を達成するために使用する、ポリシー、ルール、フレームワーク一式です。ガバナンス要件は、組織内から生まれます。選択する技術の種類に影響する場合も、ワークロードを運用する方法に関連する場合もあります。組織のガバナンス要件を、ワークロードに組み込みます。コンフォーマンスとは、ガバナンス要件が組み込まれていることを示す能力のことです。

期待される成果:

- ガバナンス要件が、アーキテクチャの設計およびワークロードのオペレーションに組み込まれています。
- ガバナンス要件に従っている証拠を提供できます。
- ガバナンス要件は定期的に見直され更新されています。

一般的なアンチパターン:

- 組織が、ルートアカウントを多要素認証とすることを義務としている。この要件を実装できなかったため、ルートアカウントが侵害された。
- ワークロードの設計中に、IT 部門が承認していないインスタンスタイプを選択した。ワークロードを起動できず、再設計を行わなければならなくなった。
- ディザスタリカバリ計画を備えることが必須となっている。計画を作成しなかったため、ワークロードの停止が長引いた。
- チームは新しいインスタンスの使用を希望していたが、ガバナンス要件が更新されていないため、許可されなかった。

このベストプラクティスを活用するメリット:

- ガバナンス要件に従うと、ワークロードを組織のより大きなポリシーに合わせることができます。
- ガバナンス要件は、業界の標準と組織のベストプラクティスを反映しています。

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

関係者やガバナンス組織と協力して、ガバナンス要件を特定します。ガバナンス要件をワークロードに含めます。ガバナンス要件に従っている証拠を提供できるようにします。

お客様事例

AnyCompany Retail では、クラウドオペレーションチームが組織全体の関係者と協力して、ガバナンス要件を作成しました。例えば、Amazon EC2 インスタンスへの SSH アクセスを禁止しています。チームがシステムにアクセスする必要がある場合、AWS Systems Manager Session Manager を使用する必要があります。クラウドオペレーションチームは、新しいサービスを利用できるようになるたびに、ガバナンス要件を定期的に更新しています。

実装手順

1. 一元化されたチームがあればそれも含め、ワークロードの関係者を特定します。
2. 関係者と協力して、ガバナンス要件を特定します。
3. リストを作成したら、改善項目に優先順位を付け、ワークロードへの実装を開始します。
    a. AWS Config などのサービスを使用して、Governance-as-Code を作成し、ガバナンス要件に従っていることを検証します。

   b. AWS Organizations を使用する場合は、サービスコントロールポリシーを活用してガバナンス要件を実装できます。

4. 実装を検証するドキュメントを提供します。

実装計画に必要な工数レベル: 中。ガバナンス要件を満たさずに実装すると、ワークロードをやり直すことになる場合があります。

リソース

関連するベストプラクティス:

- OPS01-BP04 コンプライアンス要件を評価する - コンプライアンスはガバナンスに似ていますが、組織外に由来するものです。

関連するドキュメント:

- AWS Management and Governance Cloud Environment Guide (AWS の管理およびガバナンスに関するクラウド環境ガイド)
- 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)
- Governance in the AWS クラウド: The Right Balance Between Agility and Safety (AWS クラウドのガバナンス: 俊敏性と安全性のバランスを取る)
- ガバナンス、リスク、コンプライアンス (GRC) とは

関連動画:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks (AWS の管理とガバナンス: 設定、コンプライアンス、監査 - AWS Online Tech Talks)
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1) (AWS re:Inforce 2019: クラウド時代のガバナンス (DEM12-R1))
- AWS re:Invent 2020: Achieve compliance as code using AWS Config(AWS re:Invent 2020: AWS Config を使用してコードとしてのコンプライアンスを実現する)
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)(AWS re:Invent 2020: AWS GovCloud (US) における俊敏なガバナンス)

関連する例:

- AWS Config のコンフォーマンスパックの例

関連サービス:

- AWS Config
- AWS Organizations - サービスコントロールポリシー

OPS01-BP04 コンプライアンス要件を評価する

規制、業界、および社内のコンプライアンス要件は、組織の優先順位を定義するための重要な推進要素です。コンプライアンスフレームワークによって、特定の技術や地理的場所を使用できない場合もあります。外部コンプライアンスフレームワークが特定されない場合は、デューデリジェンスを適用します。コンプライアンスを検証する監査またはレポートを作成します。

自社製品が特定のコンプライアンス基準を満たしていることを宣伝する場合、継続的なコンプライアンスを確保するための内部プロセスが必要です。コンプライアンス標準の例としては、PCI DSS、FedRAMP、HIPAA があります。適用されるコンプライアンス標準は、ソリューションが保存または送信するデータの種類、ソリューションがサポートするリージョンなど、さまざまな要因によって決まります。

期待される成果:

- 規制、業界、および社内のコンプライアンス要件がアーキテクチャの選択に組み込まれています。
- コンプライアンスを検証して監査レポートを作成できます。

一般的なアンチパターン:

- ワークロードの一部が、クレジットカード業界のデータセキュリティ基準 (PCI DSS) フレームワークの対象となっているが、ワークロードはクレジットカードデータを暗号化せずに保存している。
- ソフトウェア開発者とアーキテクトが、組織が遵守すべきコンプライアンスフレームワークに気付いていない。
- 年次の Systems and Organizations Control (SOC2) Type II 監査が近く行われるが、コントロールが配置されていることを検証できない。

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。
- コンプライアンスフレームワークに合致する適切な場所や技術を選択します。
- 可監査性を持たせてワークロードを設計すると、コンプライアンスフレームワークを遵守していることを証明できます。

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

このベストプラクティスを実装することで、コンプライアンス要件をアーキテクチャ設計プロセスに組み込みます。チームメンバーは必要なコンプライアンスフレームワークを認識します。フレームワークに沿ってコンプライアンスを検証します。

お客様事例

AnyCompany Retail は、顧客のクレジットカード情報を保存しています。カードストレージチームの開発者は、PCI-DSS フレームワークに準拠する必要があることを理解しています。クレジットカード情報が PCI-DSS フレームワークに沿って安全に保存およびアクセスされていることを検証する手順を踏んできています。毎年、セキュリティチームと協力して、コンプライアンスを検証しています。

実装手順

1. セキュリティチームやガバナンスチームと協力して、ワークロードが準拠しなければならない業界、規制、組織内部のコンプライアンスフレームワークを精査します。コンプライアンスフレームワークをワークロードに組み込みます。
   a. AWS Compute Optimizer や AWS Security Hub CSPM などのサービスを使用して、AWS のリソースの継続的なコンプライアンスを検証します。
2. チームメンバーがコンプライアンス要件に沿ってワークロードを運用および進化できるように、コンプライアンス要件を教育します。コンプライアンス要件は、アーキテクチャや技術を選択する際に含める必要があります。
3. コンプライアンスフレームワークによっては、監査またはコンプライアンスレポートを作成する必要があります。組織と協力して、このプロセスをできるだけ自動化します。
   a. AWS Audit Manager のようなサービスを使用して、コンプライアンスを検証し、監査レポートを作成します。

b. AWS Artifact を使用して AWS のセキュリティとコンプライアンスに関するドキュメントをダウンロードできます。

実装計画に必要な工数レベル: 中。コンプライアンスフレームワークの実装は課題が多い場合があります。監査レポートやコンプライアンスドキュメントを作成するとさらに複雑になります。

リソース

関連するベストプラクティス:

- 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、コンプライアンス全体における重要な部分です。
- 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 - パイプラインの一部として、セキュリティ管理を検証します。新しい変更に関するコンプライアンスドキュメントを作成することもできます。
- SEC07-BP02 データ保護コントロールを定義する - 多くのコンプライアンスフレームワークは、データ処理およびストレージに関するポリシーベースです。
- SEC10-BP03 フォレンジック機能を備える - フォレンジック機能は、監査のコンプライアンスで使用できることがあります。

関連するドキュメント:

- AWS Compliance Center (AWS コンプライアンスセンター)
- AWS のコンプライアンスのリソース
- AWS Risk and Compliance Whitepaper (Amazon Web Services リスクとコンプライアンスホワイトペーパー)
- AWS 責任共有モデル
- コンプライアンスプログラムによる対象範囲内の AWS のサービス

関連動画:

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer(AWS re:Invent 2020: AWS Config を使用してコードとしてのコンプライアンスを実現する)
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing (AWS re:Invent 2021 - クラウドのコンプライアンス、保証、監査)

- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202) (AWS Summit ATL 2022 - AWS におけるコンプライアンス、保証、監査の実装 (COP202))

関連する例:

- AWS での PCI DSS および AWS Foundational Security Best Practices

関連サービス:

- AWS Artifact
- AWS Audit Manager
- AWS Compute Optimizer
- AWS Security Hub CSPM

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 サポート 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 サポート (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 サポート 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 サポート センターを利用して AWS サポート 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 サポート 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 サポート (AWS ナレッジセンター、AWS ディスカッションフォーラム、AWS サポート 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 サポート センターを利用して AWS サポート 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 サポート
- AWS サポート 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

ワークロードのリソースには、変更管理、トラブルシューティング、その他機能を受け持つ、特定できる所有者が必要です。所有者は、ワークロード、アカウント、インフラストラクチャ、プラットフォーム、アプリケーションに割り当てられます。所有権は、一元登録などのツール、またはリソースに添付されたメタデータを使用して記録されます。コンポーネントのビジネス価値で、それらに適用されるプロセスと手順が決まります。

期待される成果:

- リソースに、メタデータまたは一元登録を使用して特定できる所有者がいます。
- チームメンバーが、リソースを誰が所有しているかを特定できます。
- アカウントの所有者は、可能な限り 1 人です。

一般的なアンチパターン:

- AWS アカウントのその他の連絡先が入力されていない。
- リソースに、それを所有するチームを特定するタグがない。

- E メールマッピングのない ITSM キューがある。
- インフラストラクチャの重要な部分で 2 つのチームの所有権が重複している。

このベストプラクティスを活用するメリット:

- リソースの変更管理は、所有権が割り当てられていて、わかりやすくなっています。
- トラブルシューティングが発生した場合に、適切な所有者を関与させることができます。

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

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。所有権とは、リソースの変更を監督する人、トラブルシューティング中にリソースをサポートする人、または財務的な責任者を意味することもあります。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。

お客様事例

AnyCompany Retail は、所有権を、リソースの変更とサポートを担当するチームまたは個人と定義しています。AWS Organizations を活用して AWS アカウントを管理しています。予備のアカウント連絡先は、グループ受信箱を使用して設定されています。各 ITSM キューは、E メールエイリアスにマッピングされています。タグによって、誰が AWS リソースを所有しているかを特定できます。その他のプラットフォームやインフラストラクチャについては、所有権と連絡先情報を特定できる wiki ページがあります。

実装手順

1. 組織における所有権を定義することから始めます。所有権は、リソースのリスクに対して責任を持つ人、リソースの変更を担当する人、またはトラブルシューティング時にリソースをサポートする人などを意味します。また、リソースの財務的または管理的な所有権を意味することもあります。
2. AWS Organizations を使用してアカウントを管理します。アカウントのその他の連絡先を一元管理できます。
   a. 会社の E メールアドレスや電話番号を連絡先として使用することで、その E メールアドレスや電話番号の持ち主が組織からいなくなったとしても、連絡先にアクセスすることができます。例えば、請求、オペレーション、セキュリティ用に別々の E メール配信リストを作成し、アクティブな AWS アカウントごとに請求、セキュリティ、オペレーションの連絡先として設定し

ます。誰かが休暇を取っていたり、担当が変わったり、会社を辞めたりした場合でも、複数の人が AWS の通知を受け取り、対応できるようになります。

b. アカウントが AWS Organizations で管理されていない場合、必要に応じてその他の連絡先を利用して AWS が適切な担当者に連絡を取ることができます。アカウントのその他の連絡先は、個人よりもグループを指定して設定します。

3. タグを使用して AWS のリソースの所有者を特定します。所有者と連絡先情報の両方を、別々のタグで指定できます。
   a. AWS Config ルールを使用して、リソースに必須の所有権タグをつけるように強制できます。
   b. 組織においてタグ付け戦略を策定する方法に関する詳細なガイダンスについては、AWS Tagging Best Practices whitepaper (AWS のタグ付けに関するベストプラクティスホワイトペーパー) を参照してください。
4. その他のリソース、プラットフォーム、インフラストラクチャについては、所有権を特定するドキュメントを作成します。このドキュメントはチームメンバーが誰でも利用できるようにします。

実装計画に必要な工数レベル: 低。アカウントの連絡先情報およびタグを利用して、AWS リソースの所有権を割り当てます。その他のリソースについては、wiki の表などシンプルなものを使用して所有権と連絡先情報を記録するか、ITSM ツールを使用して所有権をマッピングします。

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 - リソースをサポートするプロセスと手順はリソースの所有権によって決まります。
- OPS02-BP04 チームメンバーが自らの責任範囲を把握する - チームメンバーは自分がどのリソースの所有者になっているかを理解する必要があります。
- 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 - 所有権はタグやアカウント連絡先などの仕組みを使用して確認できるようにする必要があります。

関連するドキュメント:

- AWS Account Management - Updating contact information (AWS アカウントの管理 - 連絡先情報の更新)
- AWS Config ルール - required-tags
- AWS Organizations - 組織の代替連絡先の更新

- AWS Tagging Best Practices whitepaper (AWS のタグ付けに関するベストプラクティスホワイトペーパー)

関連する例:

- AWS Config ルール - Amazon EC2 と必要なタグおよび有効な値

関連サービス:

- AWS Config
- AWS Organizations

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、特定のニ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、チームメンバーはニーズを認識し、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

- 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二ズムの存在: 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二ズムを提供します。これらのメカ二ズムにより、メンバーは、特定の二ーズについて、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二ズムが存在する

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。リクエストには、追加、変更、除外などがあります。このようなリクエストは変更管理プロセスを通ります。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。

期待される成果:

- 割り当てられた所有権に基づいて、プロセス、手順、リソースの変更をリクエストできます。
- 変更は、メリットとリスクを検討して、熟考の上で行われます。

一般的なアンチパターン:

- アプリケーションをデプロイする方法を更新しなければならないが、オペレーションチームからデプロイプロセスの変更をリクエストする方法がない。
- ディザスタリカバリ計画を更新しなければならないが、変更のリクエスト先になる特定できる所有者がいない。

このベストプラクティスを活用するメリット:

- プロセス、手順、リソースを、要件の変更に合わせて進化させることができます。
- 所有者は十分な情報に基づいて変更時期を決定できます。
- 変更は熟考の上で行われます。

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: 中

実装のガイダンス

このベストプラクティスを実装するには、プロセス、手順、リソースに対する変更のリクエストが可能である必要があります。変更管理プロセスは簡単なものでかまいません。変更管理プロセスを文書化します。

お客様事例

AnyCompany Retail は責任割り当て (RACI) マトリックスを使用して、プロセス、手順、リソースの変更を所有しているのが誰かを特定しています。文書化された変更管理プロセスは、簡単で従いやすいものです。RACI マトリックスとこのプロセスを使用して、誰でも変更リクエストを送信できます。

実装手順

1. ワークロードのプロセス、手順、リソースと、それぞれの所有者を特定します。ナレッジマネジメントシステムにそれらを記録します。
   a. 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、または 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 を実装していない場合は、先に実装します。
2. 組織の関係者と協力して、変更管理プロセスを作成します。このプロセスは、リソース、プロセス、手順の追加、変更、除外を対象とします。
   a. AWS Systems Manager Change Manager を、ワークロードリソースの変更管理プラットフォームとして使用できます。

3. ナレッジマネジメントシステムに、変更管理プロセスを記録します。

実装計画に必要な工数レベル: 中。変更管理プロセスの作成では、組織全体の複数の関係者と協調する必要があります。

リソース

関連する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 - 変更管理プロセスを構築する前に、リソースに特定できる所有者がいる必要があります。
- OPS02-BP02 プロセスと手順には特定の所有者が存在する - 変更管理プロセスを構築する前に、プロセスに特定できる所有者がいる必要があります。
- 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 - 変更管理プロセスを構築する前に、オペレーションアクティビティに特定できる所有者がいる必要があります。

関連するドキュメント:

- AWS Prescriptive Guidance - Foundation palybook for AWS large migrations: Creating RACI matrices (AWS の規範的ガイダンス - AWS の大規模移行における基礎プレイブック: RACI マトリックスの作成)
- Change Management in the Cloud Whitepaper (クラウドにおける変更管理ホワイトペーパー)

関連サービス:

- AWS Systems Manager Change Manager

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)。チーム間コミュニケーションチャネルが文書化されています。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。

期待される成果:

- チーム間の作業またはサポートに関する契約が、合意され文書化されています。
- 相互にサポートまたは協力するチームに、コミュニケーションチャネルおよび応答時間目標が定められています。

一般的なアンチパターン:

- 本稼働環境で問題が発生し、2 つの個別のチームが別個にトラブルシューティングを開始した。このサイロ化された作業のために停止時間が長くなった。
- オペレーションチームが開発チームの支援を必要としているが、応答時間の合意がない。リクエストが後回しにされる。

このベストプラクティスを活用するメリット:

- チームが相互にやり取りおよびサポートする方法を知っています。
- 応答性の目標が周知されています。
- コミュニケーションチャネルが明確に定義されています。

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: 低

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、チームが協力し合う方法についてあいまいさがなくなります。正式に合意を結ぶことで、チームの協力方法や互いにサポートする方法を体系化できます。チーム間コミュニケーションチャネルが文書化されます。

お客様事例

AnyCompany Retail の SRE チームは、開発チームとサービスレベルアグリーメントを結んでいます。開発チームがチケットシステムでリクエストを行う際に、15 分以内の応答を期待できます。サイトが停止した場合は、SRE チームが開発チームのサポートを受けながら調査を主導します。

実装手順

1. 組織全体の関係者と協力して、プロセスと手順に基づき、チーム間の契約を作成します。

   a. プロセスまたは手順が 2 チームで共有されている場合は、チームの協力方法に関するランブックを作成します。
   b. チーム間に依存関係がある場合は、リクエストについての応答 SLA を結びます。
2. 責任の所在をナレッジマネジメントシステムに記録します。

実装計画に必要な工数レベル: 中。チーム間に既存の契約がない場合、組織全体の関係者が合意に至るまでに工数がかかる場合があります。

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 - チーム間で契約を結ぶ前に、プロセスの所有権を特定する必要があります。
- 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 - チーム間で契約を結ぶ前に、オペレーションアクティビティの所有権を特定する必要があります。

関連するドキュメント:

- AWS Executive Insights - Empowering Innovation with the Two-Pizza Team (エグゼクティブのインサイト - 2 枚のピザチームでイノベーションを強化する)
- Introduction to DevOps on AWS - Two-Pizza Teams (AWS での DevOps 入門 - 2 枚のピザチーム)

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。二一

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。

期待される成果:

- コミュニケーションでコンテクスト、詳細、目標時間が提供されます。
- チームメンバーが、コミュニケーションに応答して、いつどのように行動すべきかを明確に理解できます。
- 変更カレンダーを活用して、想定される変更についての通知が提供されます。

一般的なアンチパターン:

- 毎週何度も誤検知でアラートが発生する。発生するたびに通知をミュートにしている。

- セキュリティグループの変更を依頼されたが、いつ行うべきかを指示されていない。
- システムがスケールアップしたがアクションは不要という通知を、何度もチャットで受け取る。チャットチャネルを避けていたため、重要な通知を見逃した。
- 本稼働環境に対する変更が、オペレーションチームに通知せずに行われた。変更によりアラートが発報され、チームに対するオンコールが発生した。

このベストプラクティスを活用するメリット:

- 組織がアラートによる疲弊を回避できます。
- チームメンバーは必要なコンテキストと目標を得て行動できます。
- 変更は変更ウィンドウ中に行われるため、リスクを軽減できます。

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

このベストプラクティスを実装するには、組織全体の関係者と協力して、コミュニケーション基準に関して合意を得る必要があります。この基準を、組織の誰もが確認できるようにします。誤検知や常にオンのアラートを特定して排除します。変更カレンダーを活用して、アクションが実行されるタイミングや保留中のアクティビティをチームメンバーが把握できるようにします。コミュニケーションによって、必要なコンテクストが提供され、アクションが明確になることを検証します。

お客様事例

AnyCompany Retail は、主なコミュニケーションメディアとしてチャットを使用しています。アラートやその他の情報は特定のチャネルに入力されます。誰かのアクションが必要な場合、期待される成果が明確に提示され、多くの場合、使用するランブックまたはプレイブックが指定されます。変更カレンダーを使用して本稼働システムに対する大きな変更をスケジュールしています。

実装手順

1. 誤検知のアラートや継続的に発報されるアラートを分析します。それらを排除するか、人による介入が必要なときに発報されるように変更します。アラートが発報される場合は、ランブックまたはプレイブックを指定します。
   a. アラート用のプレイブックやランブックの作成には、AWS Systems Manager ドキュメントを使用できます。

2. 2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。計画されたイベントに先立ち、Eメールリストまたはチャットチャネルを使用して、通知を送信します。
    a. Amazon Q Developer in chat applications を使用して、組織のメッセージプラットフォーム内でアラートを送信したりイベントに対応したりできます。
3. 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。
    a. AWS Systems Manager Change Calendar を使用して、変更を行う変更ウィンドウを作成できます。これにより、チームメンバーは安全に変更を行うことができるタイミングを知ることができます。
4. 脆弱性の通知とパッチ情報をモニターして、ワークロードコンポーネントに関連する予期できない潜在的なリスクの脆弱性を理解します。チームメンバーが対応できるように通知を送信します。
    a. AWS セキュリティ速報を購読して、AWS における脆弱性に関する通知を受信できます。

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 - 成果がわかっている場合は、ランブックを提供してコミュニケーションを実行可能なものにします。
- OPS07-BP04 プレイブックを使用して問題を調査する - 成果が不明の場合は、プレイブックによってコミュニケーションを実行可能なものにできます。

関連するドキュメント:

- AWS セキュリティ速報
- Open CVE

関連する例:

- Well-Architected Labs: Inventory and Patch Management (Level 100) (Well-Architected ラボ: インベントリおよびパッチ管理 (レベル 100))

関連サービス:

- Amazon Q Developer in chat applications
- AWS Systems Manager Change Calendar
- AWS Systems Manager ドキュメント

OPS03-BP05 実験の推奨

実験は、新しいアイデアを製品や機能に変える触媒となります。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。イノベーションを促進するために、チームメンバーは頻繁に実験することが奨励されます。結果が思わしくないものであっても、何をすべきでないかを知ることに価値があります。実験が成功したものの望ましくない結果が得られた場合、チームメンバーが罰せられることはありません。

期待される成果:

- イノベーションを育むために、組織では実験が奨励されます。
- 実験は学びの機会として使用されます。

一般的なアンチパターン:

- A/B テストを実行したいが、実験を実行する仕組みがない。テストを行うことができないまま UI の変更をデプロイした。その結果、カスタマーエクスペリエンスが低下した。
- 会社にはステージ環境と本稼働環境しかない。新機能や新製品を実験するサンドボックス環境がないため、本稼働環境で実験を行わなければならない。

このベストプラクティスを活用するメリット:

- 実験はイノベーションを促進します。
- 実験を通して、ユーザーからのフィードバックにより迅速に対応できます。
- 組織に学習の文化が築かれます。

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: 中

実装のガイダンス

実験は安全な方法で実行する必要があります。複数の環境を活用して、本稼働リソースを危険に晒すことなく、実験を行います。A/B テストや機能フラグを使用して、実験をテストします。チームメンバーがサンドボックス環境で実験を行えるようにします。

お客様事例

AnyCompany Retail では実験が奨励されています。チームメンバーは週の仕事の 20% を実験や新技術の学習に使用できます。サンドボックス環境があり、イノベーションを行うことができます。新機能には A/B テストが使用され、実際のユーザーのフィードバックを使用して機能を検証します。

実装手順

1. 組織全体でリーダーたちと協力して実験をサポートしてもらいます。チームメンバーは安全な方法で実験を行うことが奨励されます。
2. チームメンバーに、安全に実験できる環境を提供します。チームメンバーが本稼働環境に似た環境にアクセスできるようにする必要があります。
   a. 個別の AWS アカウントを使用して実験用のサンドボックス環境を作成できます。アカウントのプロビジョニングには、AWS Control Tower を使用できます。
3. 機能フラグや A/B テストを使用して安全に実験を行い、ユーザーからのフィードバックを収集します。
   a. AWS AppConfig 機能フラグを使用して、機能フラグを作成できます。
   b. 限定されたデプロイに対する A/B テストの実行には、Amazon CloudWatch Evidently を使用できます。
   c. AWS Lambda のバージョンを使用して、関数の新しいバージョンをデプロイし、ベータテストを実行できます。

実装計画に必要な工数レベル: 高。安全な方法で実験できる環境をチームメンバーに提供し実験を行うには、多額の投資が必要です。また、機能フラグを使用したり A/B テストをサポートしたりするために、アプリケーションコードの変更が必要になる場合があります。

リソース

関連するベストプラクティス:

- OPS11-BP02 インシデント後の分析を実行する - 実験に加えて、インシデントから学習することは、イノベーションの重要な推進要素です。
- 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、実験の重要な部分です。

関連するドキュメント:

- An Inside Look at the Amazon Culture: Experimentation, Failure, and Customer Obsession (Amazon 文化の裏側: 実験、失敗、顧客第一主義)
- Best practices for creating and managing sandbox accounts in AWS(AWS でのサンドボックスアカウントの作成と管理におけるベストプラクティス)
- Create a Culture of Experimentation Enabled by the Cloud (クラウドで可能になる実験文化の構築)
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros (SulAmérica Seguros におけるクラウドでの実験とイノベーションの実現)
- Experiment More, Fail Less (実験が多いほど失敗は少なくなる)
- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU (複数のアカウントを使用した AWS 環境の組織化 - サンドボックス OU)
- Using AWS AppConfig Feature Flags (AWS AppConfig の機能フラグを使用する)

関連動画:

- AWS On Air ft.Amazon CloudWatch Evidently | AWS Events (AWS On Air: Amazon CloudWatch Evidently 特集 | AWS イベント)
- AWS On Air San Fran Summit 2022 ft.AWS AppConfig Feature Flags integration with Jira (AWS On Air San Fran Summit 2022: Jira を使用した AWS AppConfig 機能フラグの統合)
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R) (AWS re:Invent 2022 - デプロイはリリースではない: 機能フラグで起動をコントロールする (BOA305-R))
- Programmatically Create an AWS アカウント with AWS Control Tower(AWS Control Tower を使用して AWS アカウントをプログラムで作成する)
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations(AWS Organizations のベストプラクティスを使用するマルチアカウント AWS 環境の設定)

関連する例:

- AWS Innovation Sandbox
- End-to-end Personalization 101 for E-Commerce (E コマース向けのエンドツーエンドのパーソナライゼーション 101)

関連サービス:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

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 サポート (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 サポート 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 サポートセンターを利用して AWS サポート 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 サポート
- 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:

- アプリケーションがテレメトリーを送出しない。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。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できない。

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。

お客様事例

AnyCompany Retail はマイクロサービスベースのアーキテクチャを獲得しました。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

1. 1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します。この場所は、テレメトリーの収集機能と分析機能の両方をサポートしていなくてはなりません。異常検出と自動インサイトが推奨機能です。
   a. 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。
2. 必要なテレメトリーを特定するには、次の質問から始めます。アプリケーションの状態はどうなっているか? アプリケーションは、この質問に対してまとめ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を使用してこ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関係者と協力して、テレメトリー要件の一覧を作成します。
   a. 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。
3. 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。

a. AWS Distro for Open Telemetry 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。

b. bAWS が提供する可観測性サービスを理解するには、One Observability Workshop を実施するか、担当の AWS アカウントチームにサポートをリクエストしてください。

c. 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 の運用の可視性を高めるために分散システムを装備するの記事をお読みください。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。

実装計画に必要な工数レベル: 高。アプリケーションの計測やテレメトリストレージの一元化には、多大な投資が必要な場合があります。

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 - 運用の可視性を高めるために分散システムを装備する
- 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 - Design Telemetry (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
- Implementing Logging and Monitoring with Amazon CloudWatch (Amazon CloudWatch を使用したログ記録とモニタリングの実装)
- Monitoring application health and performance with AWS Distro for OpenTelemetry (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)
- New – How to better monitor your custom application metrics using Amazon CloudWatch Agent (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 - Observability the open-source way (re:Invent 2021 - オープンソースの可観測性)
- Collect Metrics and Logs from Amazon EC2 instances with the CloudWatch Agent (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)
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks (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)
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks (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)

- Open Source Observability with AWS - AWS Virtual Workshop (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

関連サービス:

- Amazon CloudWatch

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。ユーザーアクティビティの例としては、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の情報があります。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。実際のユーザーアクティビティをキャプチャすることで、模擬アクティビティを構築して本稼働環境のワークロードのモニターやテストに使用できます。

期待される成果:

- ワークロードから、すべてのアプリケーションにまたがるユーザーアクティビティに関するテレメトリが送信されます。
- 模擬ユーザーアクティビティを活用して、オフピーク時のアプリケーションをモニターできます。

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た。顧客が当該機能を使用しているかどうかが、顧客に問い合わせない限りわからない。
- フロントエンドアプリケーションのデプロイ後、使用量が増加した。ユーザーアクティビティのテレメトリがないため、正確な問題を特定するのが難しい。
- オフピーク時にアプリケーションで問題が発生した。模擬ユーザーアクティビティを構成していなかったため、朝になってユーザーがオンラインになるまで問題に気付かなかった。

このベストプラクティスを活用するメリット:

- 一般的なユーザーパターンや意図しない動作を理解して、アプリケーションの機能をビジネス目標に合うように最適化できます。

- ユーザー視点でアプリケーションをモニターし、リンクの不具合やクリック応答速度の低下など、ユーザーエクスペリエンスに関する問題を検出できます。
- 影響を受けたユーザーが行った手順を追跡することで、問題の根本原因を特定できます。
- 模擬ユーザーアクティビティは、オフピーク時のパフォーマンス低下に対して早期に警告サインを出すため、実際のユーザーに影響が出る前に修正アクションを取ることができます。

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

ユーザーアクティビティ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。模擬ユーザーアクティビティを利用して、アプリケーションのオフピーク時のパフォーマンスに関するインサイトを取得します。

お客様事例

AnyCompany Retail は、アプリケーションの複数のレイヤーにユーザーアクティビティテレメトリを実装しています。フロントエンドテレメトリがマウスの挙動や移動に関するイベントを追跡し、バックエンドのマイクロサービスが、ユーザーの買い物カゴに商品を追加する、会計するなどのイベントを追跡したテレメトリを送信します。これら両方を合わせてユーザーエクスペリエンスに関する可観測性が提供されます。AnyCompany Retail はまた、模擬ユーザーテレメトリを使用して、ワークロードを利用するユーザーが少ないときに問題を捕捉します。

実装手順

1. ユーザーアクティビティに関するテレメトリ (メトリクス、イベント、ログ、トレース) を送信するようにアプリケーションをインストルメント化します。インストルメント化すると、ユーザーがユーザーインターフェイスを操作するたびに、フロントエンドコンポーネントが自動的にテレメトリを送信します。バックエンドアプリケーションは、ユーザーイベントとトランザクションのテレメトリを送信します。
    a. Amazon CloudWatch RUM を使用して、フロントエンドアプリケーションのエンドユーザーエクスペリエンスに関するインサイトを取得できます。
    b. AWS Distro for Open Telemetry を使用して、アプリケーションをインストルメント化し、テレメトリをキャプチャできます。
    c. Amazon Pinpoint を使用して、キャンペーンを通したユーザーの行動を分析し、ユーザーエンゲージメントに関するインサイトを取得できます。

    d. Enterprise Support を利用しているお客様は、テクニカルアカウントマネージャーから、モニタリング戦略の策定ワークショップをリクエストできます。このワークショップは、お客様がワークロードの可観測性戦略を策定するのを支援します。
2. 模擬ユーザーアクティビティを作成して、アプリケーションをモニターします。模擬ユーザーアクティビティはユーザーアクションをシミュレートし、アプリケーションが正常に動作しているかを検証します。
    a. Amazon CloudWatch Synthetics は、canary を使用してユーザーアクティビティをシミュレートできます。

実装計画に必要な工数レベル: 高。アプリケーションを完全にインストルメント化してユーザーアクティビティのテレメトリを収集するには、大規模な開発工数がかかる場合があります。

リソース

関連する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 - ユーザーアクティビティテレメトリを構築するには、アプリケーションテレメトリが必要です。
- OPS04-BP02 ワークロードテレメトリーを実装して設定する - 一部のユーザーアクティビティテレメトリは、ワークロードテレメトリとみなされます。

関連するドキュメント:

- アプリケーションを効率的にモニタリングする方法

関連動画:

- AWS re:Invent 2020: Monitoring production services at Amazon (AWS re:Invent 2020: Amazon における製造サービスのモニタリング)
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM (AWS re:Invent 2021 - Amazon CloudWatch RUM を使用してエンドユーザーインサイトを介しアプリケーションを最適化する)
- Testing and Monitoring APIs on AWS - AWS Online Tech Talks (AWS での API のテストとモニタリング - AWS Online Tech Talks)

関連する例:

- Amazon CloudWatch RUM Web Client
- AWS Distro for Open Telemetry
- Implementing Real User Monitoring of Amplify Application using Amazon CloudWatch RUM (Amazon CloudWatch RUM を使用して実際のユーザーのモニタリングを実装し、アプリケーションを増強する)
- 1 つの可観測性ワークショップ

関連サービス:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Amazon Pinpoint

OPS04-BP04 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リソースの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。これらは、ワークロードの外にあるリソースです。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、対応が必要な場合を判断したり、ワークロードの状態に関する追加のコンテキストを提供したりします。

期待される成果:

- ワークロードは、外部依存のステータスに関するテレメトリを送信します。
- 依存関係に異常がある場合は通知されます。

一般的なアンチパターン:

- ユーザーがサイトにアクセスできない。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る。
- ショッピングカートアプリケーションがトランザクションを完了できない。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る。

このベストプラクティスを活用するメリット:

- 外部依存関係をモニタリングすると、問題について高度な通知が提供されます。
- 依存関係の正常性を認識できると、トラブルシューティングに役立ちます。

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: 中

実装のガイダンス

関係者と協力して、ワークロードが依存している外部依存関係を特定します。外部依存関係には、外部データベース、API、ワークロードと他の環境にあるリソース間のネットワーク接続性などがあります。モニタリング戦略を策定して、依存関係の正常性を把握し、ステータスが変化するとプロアクティブに警告します。

お客様事例

AnyCompany Retail の E コマースワークロードは、別の環境にあるデータベースに依存しています。毎晩、データがデータベースに入力され、E コマースプラットフォームで使用されます。ネットワーク接続とデータベースのサポートは他のチームが所有しています。E コマースチームは複数の canary アラームを構成し、ネットワーク接続性が低下したり、データベースにアクセスできなかったり、ジョブが完了できなかったりした場合はアラートを発報するようにしました。

実装手順

1. ワークロードが依存している外部依存関係を特定します。依存関係の正常性または到達可能性を追跡するテレメトリを実装します。
   a. AWS のお客様は、AWS Health Dashboard を使用して AWS のサービスの正常性をモニターし、正常性に関するイベントの通知を受け取ることができます。
   b. Amazon CloudWatch Synthetics を使用して、API、URL、ウェブサイトコンテンツをモニターできます。
2. 依存関係に異常があったりアクセスできない場合は組織に通知するアラートをセットアップします。
   a. Enterprise Support を利用しているお客様は、テクニカルアカウントマネージャーから、モニタリング戦略の策定ワークショップをリクエストできます。このワークショップは、お客様がワークロードの可観測性戦略を策定するのを支援します。
3. 依存関係に異常があった場合に備えて、依存関係の連絡先を特定します。依存関係の所有者への連絡方法、サービスアグリーメント、エスカレーションプロセスを文書化します。

実装計画に必要な工数レベル: 中。依存関係テレメトリの実装には、カスタムモニタリングソリューションの構築が必要になる場合があります。

リソース

関連する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 - 依存関係モニタリングをアプリケーションテレメトリに構築できます。

関連するドキュメント:

- Monitor your private internal endpoints 24x7 using CloudWatch Synthetics (CloudWatch Synthetics を使用して、プライベート内部エンドポイントを 24 時間 365 日モニターする)

関連動画:

- AWS re:Invent 2018: Monitor All Your Things: Amazon CloudWatch in Action with BBC (AWS re:Invent 2018: あらゆるものをモニターする: BBC による Amazon CloudWatch の実践)
- AWS re:Invent 2022 - Developing an observability strategy (AWS re:Invent 2022 - 可観測性戦略の策定)
- AWS re:Invent 2022 - Observability best practices at Amazon (AWS re:Invent 2022 - Amazon における可観測性のベストプラクティス)

関連する例:

- 1 つの可観測性ワークショップ
- Well-Architected Labs - Dependency Monitoring (Well-Architected ラボ - 依存関係のモニタリング)

関連サービス:

- Amazon CloudWatch Synthetics
- AWS Health

OPS04-BP05 トランザクショントレーサビリティを実装する

アプリケーションコードを実装し、イベントを発行するようにワークロードのコンポーネントを設定します。このイベントは、単一の論理演算の結果としてトリガーされ、ワークロードのさまざまな境界を越えて統合されます。マップを生成して、ワークロードとサービス全体にわたるトレースの流れを確認します。コンポーネント間の関係のインサイトを取得し、問題を特定して分析します。収集した情報を使用して、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。

期待される成果:

- ワークロード全体にわたるトランザクショントレースを収集して、コンポーネント間の関係のインサイトを取得できます。
- マップを生成して、ワークロードのトランザクションとイベントの流れをより明確に把握できます。

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。トランザクショントレーサビリティがないので、どの機能またはコンポーネントが原因であるかを検出できません。
- ワークロードでパフォーマンスのボトルネックが発生しています。トランザクショントレーサビリティがないため、アプリケーションコンポーネント間の関係を確認してボトルネックを特定することができません。
- トレースに使用される識別子はグローバルで一意ではないため、ワークロードの動作分析をすると、トレースの競合が発生します。

このベストプラクティスを活用するメリット:

- 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についてのインサイトが得られます。
- 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。
- トランスアクションを、生成された場所を問わず、一意に生成された識別子によって特定できます。

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: 低

実装のガイダンス

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。トランザクションに含めるべきデータは、グローバルに一意のトランザクション ID、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です。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。

お客様事例

AnyCompany Retail では、すべてのトランザクションでグローバルに一意な UUID が生成されています。この UUID は、トランザクション中にマイクロサービス間で渡されます。UUID は、ユーザーのワークロードとのインタラクション時にトランザクショントレース作成に使用されます。このトレースを使用してワークロードトポロジのマップが生成され、ワークロードの問題のトラブルシューティングとパフォーマンスの向上に使用されます。

実装手順

1. ワークロードのアプリケーションを計測して、トランザクショントレースを発行します。これを実行するには、トランザクションごとに一意の識別子を生成して、アプリケーション間で識別子を渡します。
   a. AWS Distro for OpenTelemetry の自動インストルメント化を使用して、アプリケーションコードを変更する必要なく、既存のアプリケーションにトレースを実装できます。
2. アプリケーショントポロジのマップを生成します。このマップを使用して、パフォーマンスを向上し、インサイトを取得し、トラブルシューティングに役立てます。
   a. AWS X-Ray でワークロード内のアプリケーションのマップを生成できます。

実装計画に必要な工数レベル: 中。トランザクショントレースの実装には、中程度の開発作業が必要になる場合があります。

リソース

関連する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 - アプリケーションテレメトリーはトランザクションのトレーサビリティと処理をカバーしており、最初に実装する必要があります。

関連するドキュメント:

- Discover application issues and get notifications with AWS X-Ray Insights (AWS X-Ray Insights を使用してアプリケーションの問題を検出し、通知を受け取る)
- How Wealthfront utilizes AWS X-Ray to analyze and debug distributed applications (Wealthfront が AWS X-Ray を使用して分散アプリケーションの分析とデバッグを行っている方法)
- AWS Distro for OpenTelemetry の新機能 – トレースのサポートが一般的に利用可能に

関連動画:

- AWS re:Invent 2018: Deep Dive into AWS X-Ray: Monitor Modern Applications (DEV324) (AWS re:Invent 2018: AWS X-Ray Deep Dive: モダンアプリケーションのモニタリング (DEV324))
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310) (AWS re:Invent 2022 - OpenTelemetry を使用したオブザーバブルなアプリケーションの構築 (BOA310))
- AWS re:Invent 2021 - Observability the open-source way (COP301-R) (AWS re:Invent 2021 - オープンソースのオブザーバビリティ)
- Capturing Trace Data with the AWS Distro for OpenTelemetry (AWS Distro for OpenTelemetry を使用してトレースデータをキャプチャする)
- Optimize Application Performance with AWS X-Ray(AWS X-Ray を使用してアプリケーションのパフォーマンスを最適化する)

関連する例:

- AWS X-Ray Multi API Gateway Tracing Example (AWS X-Ray の マルチ API ゲートウェイトレースの例)

関連サービス:

- AWS Distro for OpenTelemetry
- AWS X-Ray

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

デプロイされた変更はすべてテストし、本稼働でのエラーを回避する必要があります。このベストプラクティスは、バージョンコントロールからアーティファクトビルドへの変更をテストすることに重点を置いています。テストには、アプリケーションコードの変更に加えて、インフラストラクチャ、設定、セキュリティコントロール、運用手順も含める必要があります。テストは、単体テストからソフトウェアコンポーネント分析 (SCA) まで、さまざまな形態があります。ソフトウェアの統合および配信プロセスでテストをさらに早めると、アーティファクト品質の確実性が増します。

組織はすべてのソフトウェアアーティファクトにおいてテスト基準を作成する必要があります。テストを自動化すると、手間を軽減し、手動テストによるエラーを回避できます。手動テストが必要な場合もあります。開発者は自動テストの結果を確認して、ソフトウェアの品質を向上させるフィードバックループを作成する必要があります。

期待される成果:

- ソフトウェアの変更は、配信前にすべてテストされる。
- 開発者はテスト結果を利用できる。
- 組織に、すべてのソフトウェア変更に適用されるテスト基準がある。

一般的なアンチパターン:

- ソフトウェアの新しい変更を、テストせずにデプロイする。本稼働で実行に失敗し、その結果サービスが停止する。
- 新しいセキュリティグループが、本番前環境でのテストをせずに CloudFormation にデプロイされる。そのセキュリティグループによって、ユーザーがアプリにアクセスできなくなる。
- メソッドが変更されたが単体テストがなかった。本稼働にデプロイされた際にソフトウェアが失敗した。

このベストプラクティスを活用するメリット:

- ソフトウェアのデプロイにおける変更失敗率が軽減されます。
- ソフトウェアの品質が向上します。
- 開発者のコードの成立性に対する意識が上がります。
- セキュリティポリシーを確信を持ってロールアウトし、組織のコンプライアンスをサポートできます。
- 自動スケーリングポリシーの更新などインフラストラクチャの変更を事前にテストし、トラフィックのニーズ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

継続的統合の実践の一部として、アプリケーションコードからインフラストラクチャまで、すべての変更に対してテストを行います。テスト結果は、開発者が迅速にフィードバックを得られるように公開します。組織に、すべてのソフトウェア変更に適用されるテスト基準を備えます。

お客様事例

AnyCompany Retail は、継続的な統合パイプラインの一部として、すべてのソフトウェアアーティファクトに対して複数種類のテストを実行しています。テスト駆動開発を実践しているため、すべてのソフトウェアに単体テストがあります。アーティファクトがビルドされると、エンドツーエンドのテストが実行されます。1 ラウンド目のテストが完了すると、静的アプリケーションセキュリティスキャンを実行し、既知の脆弱性を探します。開発者は、各テストに合格するたびにメッセージを受け取ります。すべてのテストが完了すると、ソフトウェアアーティファクトはアーティファクトリポジトリに保存されます。

実装手順

1. 組織の関係者と協力して、ソフトウェアアーティファクトのテスト基準を作成します。すべてのアーティファクトが合格しなければならない基準のテストとは何でしょうか? テスト範囲に含める必要があるコンプライアンスやガバナンスの要件はありますか? コード品質テストを実施する必要がありますか? テストが完了した際に通知が必要なのは誰ですか?
    a. AWS デプロイパイプラインリファレンスアーキテクチャには、統合パイプラインの一部としてソフトウェアアーティファクトに対して実行できる、テストの種類の信頼できるリストが含まれています。
2. ソフトウェアテスト基準に基づいて必要なテストを行い、アプリケーションを計測します。テストの各セットは 10 分以内に完了する必要があります。テストは統合パイプラインの一部として実行する必要があります。
    a. Amazon CodeGuru Reviewer では、アプリケーションコードをテストして欠陥を検出できます。
    b. AWS CodeBuild を使用して、ソフトウェアアーティファクトに対しテストを実施できます。
    c. AWS CodePipeline は、ソフトウェアテストをパイプラインに組み込むことができます。

リソース

関連するベストプラクティス:

- OPS05-BP01 バージョン管理を使用する - すべてのソフトウェアアーティファクトはバージョン管理されたリポジトリにバックアップされる必要があります。
- OPS05-BP06 設計標準を共有する - 組織のソフトウェアテスト基準によって、設計基準がわかります。
- OPS05-BP10 統合とデプロイを完全自動化する - ソフトウェアテストは、より大きな統合およびデプロイパイプラインの一部として、自動で実行する必要があります。

関連するドキュメント:

- Adopt a test-driven development approach (テスト駆動の開発アプローチを導入する)
- Automated CloudFormation Testing Pipeline with TaskCat and CodePipeline (TaskCat と CodePipeline を使用した自動 CloudFormation テストパイプライン)
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools (オープンソースの SCA、SAST、DAST ツールを使用してエンドツーエンドの AWS DevSecOps CI/CD パイプラインを構築する)

- Getting started with testing serverless applications (サーバーレスアプリケーションのテストを開始する)
- My CI/CD pipeline is my release captain (CI/CD パイプラインが自分のリリースキャプテン)
- 「AWS における継続的インテグレーションと継続的デリバリーの実践」ホワイトペーパー

関連動画:

- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS(re:Invent 2020: テスト可能なインフラストラクチャ: AWS での統合テスト)
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development (Summit ANZ 2021 - CDK とテスト駆動開発でテストファースト戦略を獲得する)
- Testing Your Infrastructure as Code with AWS CDK (AWS CDK で Infrastructure as Code をテストする)

関連リソース:

- 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
- AWS Kubernetes DevSecOps パイプライン
- Policy as Code Workshop – Test Driven Development (Policy as Code ワークショップ - テスト駆動開発)
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild(AWS CodeBuild を使用して GitHub から Node.js アプリケーションの単体テストを実行する)
- Use Serverspec for test-driven development of infrastructure code (インフラストラクチャコードのデータ駆動開発に Serverspec を使用する)

関連サービス:

- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

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 マネジメントコンソール 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。標準を文書化し、アーキテクチャの進化に応じて最新の内容となるよう維持します。組織内で共有された標準が適用されている場合、標準の追加、変更、例外を申請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。

期待される成果:

- 設計標準が組織のチーム間で共有されています。

- 設計標準が文書化され、ベストプラクティスの進化に合わせて最新の内容に維持されています。

一般的なアンチパターン:

- 2 つの開発チームがそれぞれ独自のユーザー認証サービスを作成しました。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。
- 両チームは独自のインフラストラクチャを管理しています。新しいコンプライアンス要件により、インフラストラクチャの変更が必要になり、両チームは別々の方法で新たな要件を実装します。

このベストプラクティスを活用するメリット:

- 共有の標準を使用すると、ベストプラクティスの採用に役立ち、開発作業の利点の最大化につながります。
- 設計標準を文書化して更新することにより、組織はベストプラクティス、セキュリティ、コンプライアンス要件を最新の内容に維持することができます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

既存のベストプラクティス、設計標準、チェックリスト、業務手順、ガイダンス、ガバナンス要件をチーム間で共有します。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。公開されたコンテンツについてチームに周知させます。新しいベストプラクティスの登場に合わせて設計標準を最新の内容に維持するメカニズムを設けます。

お客様事例

AnyCompany Retail には、ソフトウェアアーキテクチャのパターンを作成する機能横断的なアーキテクチャチームがあります。このチームでは、コンプライアンスとガバナンスを組み込んだアーキテクチャを構築しています。この共有標準を採用するチームは、コンプライアンスとガバナンスが組み込み済みであるという利点が得られ、この設計標準を基盤に迅速に構築できます。このアーキテクチャチームは四半期ごとのミーティングでアーキテクチャのパターンを検討し、必要に応じて更新します。

実装手順

1. 設計標準の開発と更新を担当する機能横断的なチームを特定します。このチームは、組織全体にわたるステークホルダーと協力して、設計標準、チェックリスト、業務手順、ガイダンス、ガバナンス要件を開発し、設計標準を文書化して、組織内で共有します。
   a. AWS Service Catalog を使用すると、IaC (Infrastructure as Code) を使用して設計標準を提示するポートフォリオを作成でき、ポートフォリオをアカウント間で共有できます。
2. 新しいベストプラクティスが特定されると、設計標準を最新の内容に維持するメカニズムを施行します。
3. 設計標準が一元的に施行されていれば、変更、更新、例外を申請するプロセスを設けます。

実装計画に必要な工数レベル: 中。設計標準を作成して共有するプロセスを開発するには、組織全体のステークホルダーとの調整と協力が必要です。

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 - ガバナンス要件は設計標準に影響を及ぼします。
- OPS01-BP04 コンプライアンス要件を評価する - コンプライアンスは設計標準作成の際に重要な情報を提供します。
- OPS07-BP02 運用準備状況の継続的な確認を実現する - 運用準備状況チェックリストは、ワークロード設計時に設計標準を実装するメカニズムです。
- OPS11-BP01 継続的改善のプロセスを用意する - 設計標準の更新は継続的改善の一環です。
- OPS11-BP04 ナレッジ管理を実施する - ナレッジ管理プラクティスの一環として、設計標準を文書化して共有します。

関連するドキュメント:

- Automate AWS Backups with AWS Service Catalog(AWS Service Catalog を使用して AWS Backup を自動化する)
- AWS Service Catalog Account Factory-Enhanced (AWS Service Catalog Account Factory の機能を拡張)
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog (Expedia Group が AWS Service Catalog を使用してDatabase as a Service (DBaaS) サービスを構築した方法)

- Maintain visibility over the use of cloud architecture patterns (クラウドアーキテクチャパターンの使用に関する可視性を維持する)
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup (AWS Organizations を設定して AWS Service Catalog のポートフォリオの共有を簡素化する)

関連動画:

- AWS Service Catalog – Getting Started (AWS Service Catalog - 開始方法)
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert (AWS re:Invent 2020: エキスパートに学ぶ AWS Service Catalog ポートフォリオの管理)

関連する例:

- AWS Service Catalog Reference Architecture (AWS Service Catalog リファレンスアーキテクチャ)
- AWS Service Catalog ワークショップ

関連サービス:

- AWS Service Catalog

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。例としては、テスト駆動型デプロイ、コードレビュー、標準の導入、ペアプログラミングなどがあります。このようなプラクティスを継続的インテグレーションと継続的デリバリープロセスに組み込みます。

期待される成果:

- 組織はコードレビューやペアプログラミングなどのベストプラクティスを使用し、コード品質が向上します。
- デベロッパーとオペレーション担当者は、ソフトウェア開発ライフサイクルの一環として、コード品質のベストプラクティスを採用しています。

一般的なアンチパターン:

- コードレビューを行わずに、アプリケーションの主幹にコードをコミットしています。変更が自動的に本番環境にデプロイされ、アプリケーションの停止が発生します。

- 新しいアプリケーションの開発が、ユニットテスト、エンドツーエンドテスト、または統合テストなしで行われています。デプロイする前にアプリケーションをテストする方法がありません。
- エラーの対応には、本番環境でチームが手動の変更を加えています。テストやコードレビューを行わなわずに変更を加えており、継続的インテグレーションと継続的デリバリープロセスを介して変更がキャプチャされたりログに記録されたりしていません。

このベストプラクティスを活用するメリット:

- 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。
- ペアプログラミングやコードレビューなどのベストプラクティスを使用すると、コード品質が向上します。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

プラクティスを実装して、コード品質を向上し、デプロイする前にエラーを最低限に抑えます。テスト駆動開発、コードレビュー、ペアプログラミングなどのプラクティスを採用して、開発の質を向上します。

お客様事例

AnyCompany Retail では、コード品質の向上のためにいくつかのプラクティスを採用しており、アプリケーションのコーディング基準として、テスト駆動開発を採用しています。新しい機能には、スプリント中にデベロッパーが協力してペアプログラミングを行うことを予定しているものもあります。すべてのプルリクエストは、インテグレーションとデプロイ前に、シニアデベロッパーによるコードレビューを受けます。

実装手順

1. テスト駆動型開発、コードレビュー、ペアプログラミングなどのコード品質プラクティスを、継続的インテグレーションと継続的デリバリープロセスに採用します。このような手法を使用して、ソフトウェアの品質を向上させます。
   a. Amazon CodeGuru Reviewer は、機械学習を利用した Java と Python コードのプログラミングについてのレコメンデーションを提供します。
   b. AWS Cloud9 を使用して共有開発環境を作成すると、コードの共同開発ができます。

実装計画に必要な工数レベル: 中。ベストプラクティスを実施する方法は数多くありますが、組織全体での導入が難しい場合があります。

リソース

関連するベストプラクティス:

- OPS05-BP06 設計標準を共有する - コード品質プラクティスの一環として、設計標準を共有できます。

関連するドキュメント:

- Agile Software Guide (アジャイルソフトウェアガイド)
- My CI/CD pipeline is my release captain (CI/CD パイプラインが自分のリリースキャプテン)
- Automate code reviews with Amazon CodeGuru Reviewer (Amazon CodeGuru Reviewer を使用したコードレビューの自動化)
- Adopt a test-driven development approach (テスト駆動の開発アプローチを導入する)
- How DevFactory builds better applications with Amazon CodeGuru (DevFactory を使用してアプリケーション構築を向上する方法)
- On Pair Programming (ペアプログラミングについて)
- RENGA Inc. automates code reviews with Amazon CodeGuru (RENGA Inc. が Amazon CodeGuru を使用してコードレビューを自動化)
- The Art of Agile Development: Test-Driven Development (アジャイル開発の技術: テスト駆動開発)
- Why code reviews matter (and actually save time!) (コードレビューが重要である理由 (そして実際に時間の節約になる理由))

関連動画:

- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru (AWS re:Invent 2020: Amazon CodeGuru を使用したコード品質の継続的改善)
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development (Summit ANZ 2021 - CDK とテスト駆動開発でテストファースト戦略を獲得する)

関連サービス:

- Amazon CodeGuru Reviewer

- Amazon CodeGuru Profiler
- AWS Cloud9

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-BP06 本稼働ワークロード用のサポートプランを有効にする

OPS07-BP01 人材能力の確保

トレーニングを受けた、ワークロードをサポートするための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。担当者は、ワークロードを構成するプラットフォームとサービスについてのトレーニングを受けている必要があります。ワークロードのオペレーションに必要となるナレッジを提供します。ワークロードの通常の運用サポートと発生したインシデントのトラブルシューティングを行うために、十分な人数のトレーニングを受けた人材が必要です。人員の疲弊を避けるため、オンコール対応と休暇を考慮に入れたローテーションを組むうえで十分な人材を配置します。

期待される成果:

- ワークロードが利用可能な間、ワークロードのサポートを担当する、十分なトレーニングを受けた人材が確保されています。
- ワークロードを構成するソフトウェアとサービスについて、担当者にトレーニングを提供しています。

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運用するにあたって、トレーニングを受けたチームメンバーなしでワークロードをデプロイします。
- オンコール対応と人材の休暇を考慮したローテーションを行ううえで十分な人材が不足しています。

このベストプラクティスを活用するメリット:

- 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。
- チームメンバーが十分に配置されていれば、ワークロードをサポートでき、人員の疲弊を引き起こすリスクを軽減しつつ、オンコールローテーションを行うことができます。

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

ワークロードをサポートするために、十分にトレーニングを受けた担当者がいることを確認します。オンコール対応を含め、通常の運用アクティビティに対応するうえで十分なチームメンバーが配置されていることを確認します。

お客様事例

AnyCompany Retail では、ワークロードをサポートするチームが適切に配置され、トレーニングを受けていることを確認しており、オンコールローテーションをサポートするうえで十分な人数のエンジニアがいます。担当者は、ワークロード構築の基盤となっているソフトウェアとプラットフォームについてのトレーニングを受けており、認定資格の取得が奨励されています。十分な人材が配置されているため、ワークロードをサポートし、オンコールローテーションを組みつつ、担当者は休暇を取ることができます。

実装手順

1. オンコール業務を含め、ワークロードの運用とサポートに十分な人数の人材を割り当てます。
2. ワークロードを構成するソフトウェアとプラットフォームについてのトレーニングを人材に提供します。
   a. AWS トレーニングと認定 には、AWS についてのコースライブラリがあり、無料および有料のコース、オンラインコース、クラスルーム形式のコースが提供されています。
   b. AWS では、イベントやオンラインセミナーを開催しており、AWS のエキスパートから学ぶことができます。
3. 運用状況とワークロードの変化に応じて、チームの規模とスキルを定期的に評価します。運用要件に合わせてチームの規模とスキルを調整します。

実装計画に必要な工数レベル: 高。ワークロードをサポートするチームを雇用し、トレーニングするには、多大な労力が必要になる場合がありますが、長期的に多大な利点があります。

リソース

関連するベストプラクティス:

- OPS11-BP04 ナレッジ管理を実施する - チームメンバーは、ワークロードの運用とサポートを行ううえで必要となる情報を持っている必要があります。それを提供する鍵となるのが、ナレッジ管理です。

関連するドキュメント:

- AWS イベントとオンラインセミナー
- AWS トレーニングと認定

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 CSPM、 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 サポート 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 CSPM
- 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。問題発生時に工エラー処理と

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

ワークロードに対する変更が正常に行われた場合のプロセスと正常に行われなかった場合のプロセスを施行します。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。ワークロードに対する変更をデプロイする利点とリスクを評価します。すべての変更がガバナンスに準拠していることを確認します。

期待される成果:

- ワークロードに変更をデプロイする際に、情報に基づく意思決定を行います。
- 変更は、ガバナンスに準拠しています。

一般的なアンチパターン:

- デプロイが正常に行われなかった場合に対応するプロセスなしで、変更をワークロードにデプロイします。
- ガバナンス要件に準拠していない変更を本番環境に加えます。
- リソース使用率のベースラインを設定することなく、ワークロードの新しいバージョンをデプロイします。

このベストプラクティスを活用するメリット:

- ワークロードへの変更が正常に行われなかった場合の準備が整っています。
- ワークロードへの変更は、ガバナンスポリシーに準拠しています。

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: 低

実装のガイダンス

プレモータムを使用して、変更が正常に行われなかった場合のプロセスを開発します。変更が正常に行われなかった場合のプロセスを文書化します。すべての変更がガバナンスに準拠していることを確認します。ワークロードに対する変更をデプロイする利点とリスクを評価します。

お客様事例

AnyCompany Retail では、変更が正常に行われなかった場合のプロセスの検証のために、定期的にプレモータムを実施しています。このプロセスは文書化され、共有の Wiki で公開され、頻繁に更新されています。すべての変更がガバナンスに準拠しています。

実装手順

1. ワークロードに変更をデプロイする際に、情報に基づく意思決定を行います。デプロイの正常完了基準を設定し、レビューを行います。変更のロールバックをトリガーするシナリオまたは基準を作成します。変更をデプロイする利点と、変更が正常に実行されないリスクを比較検討します。
2. すべての変更がガバナンスポリシーに準拠していることを確認します。
3. 変更が正常に実行されない場合に備え、また軽減戦略を文書化するために、プレモータムを使用します。机上演習を行って、正常に完了しない変更をモデル化して、ロールバック手順を検証します。

実装計画に必要な工数レベル: 中。プレモータム演習の実施には、組織全体にわたるステークホルダーの調整と尽力が必要となります。

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、変更をデプロイするかを決定するうえでの重要な要素となります。
- OPS06-BP01 変更の失敗に備える - 障害が発生したデプロイの軽減策を設定し、プレモータムを使用して軽減策を検証します。
- OPS06-BP02 変更をテストし、検証する - 本番環境でのエラーの低減に向けて、すべてのソフトウェア変更について、デプロイ前に適切なテストを行う必要があります。
- OPS07-BP01 人材能力の確保 - システム変更をデプロイする際に情報に基づく決定を行うには、トレーニングを受けたワークロードサポート担当の人材が十分に配置されていることが不可欠です。

関連するドキュメント:

- Amazon Web Services: Risk and Compliance
- AWS 責任共有モデル
- Governance in the AWS クラウド: The Right Balance Between Agility and Safety (AWS クラウドのガバナンス: 俊敏性と安全性の適切なバランス)

OPS07-BP06 本稼働ワークロード用のサポートプランを有効にする

本稼働ワークロードが依存しているあらゆるソフトウェアやサービスのサポートを有効にします。本稼働のサービスレベルのニーズに合わせて、適切なサポートレベルを選択します。このような依存関係のためのサポートプランは、サービスの停止時やソフトウェアに問題が発生した場合に必要です。すべてのサービスおよびソフトウェアのベンダーについて、サポートプランやサービスのリクエスト方法を文書化します。サポートの連絡先が最新の状態に保たれていることを検証する仕組みを実装します。

期待される成果:

- 本稼働ワークロードが依存しているソフトウェアやサービスのサポートプランを実装します。
- サービスレベルのニーズに基づいて適切なサポートプランを選択します。

- サポートプラン、サポートレベル、サポートのリクエスト方法を文書化します。

一般的なアンチパターン:

- 重要なソフトウェアベンダーのサポートプランがない。ワークロードがその影響を受けたが、修正を急がせる手段もなければ、ベンダーからタイムリーに最新情報を得ることもできない。
- ソフトウェアベンダーの主要連絡先だった開発者が退社した。ベンダーのサポートに直接連絡できなくなった。時間をかけて汎用の問い合わせシステムを検索し移動しなければならず、必要なときに対応してもらうための時間が増えた。
- ソフトウェアベンダーに起因する本稼働の停止が発生した。サポートケースの記録方法に関するドキュメントがない。

このベストプラクティスを活用するメリット:

- 適切なサポートレベルを受けていると、サービスレベルのニーズを満たすのに必要な時間内で対応を得ることができます。
- サポートを受ける顧客として、本稼働で問題があればエスカレーションできます。
- インシデント発生時にソフトウェアやサービスのベンダーがトラブルシューティングを支援します。

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

本稼働ワークロードが依存しているあらゆるソフトウェアやサービスのベンダーのサポートプランを有効にします。サービスレベルのニーズに合わせた適切なサポートプランをセットアップします。AWS のお客様の場合は、本稼働ワークロードがある任意のアカウントで AWS Business Support 以上を有効にすることを意味します。サポートベンダーと定期的に打ち合わせ、サポートのオファー、プロセス、連絡先に関する最新情報を入手します。ソフトウェアやサービスのベンダーにサポートをリクエストする方法を、停止が発生した場合のエスカレーション方法を含めて文書化します。サポートの連絡先を最新の状態に保つ仕組みを実装します。

お客様事例

AnyCompany Retail では、すべての商用ソフトウェアおよびサービスの依存関係がサポートプランを備えています。例えば、本稼働ワークロードがあるすべてのアカウントで AWS Enterprise Support が有効になっています。問題が発生した場合は、開発者が誰でもサポートケースを作成でき

ます。サポートのリクエスト方法、通知を受ける担当者、ケースを迅速化するベストプラクティスに関する情報を掲載した wiki ページがあります。

実装手順

1. 組織の関係者と協力して、ワークロードが依存しているソフトウェアやサービスのベンダーを特定します。これらの依存関係を文書化します。
2. ワークロードに必要なサービスレベルを判断します。それらに合うサポートプランを選択します。
3. 商用のソフトウェアやサービスの場合は、ベンダーとサポートプランを締結します。
    a. すべての本稼働稼働用アカウントで AWS Business Support 以上を契約すると、AWS サポートからの応答時間が短縮されるため、これを強くお勧めします。プレミアムサポートがない場合は、問題に対処するアクションプランが必要となり、これには AWS サポート からの支援が必要です。AWS サポート が、さまざまなツール、テクノロジー、人、プログラムを組み合わせて提供します。これらは、パフォーマンスの最適化、コスト削減、より迅速なイノベーションの実現を積極的に支援するために設計されたものです。AWS Business Support には追加の利点があります。AWS Trusted Advisor や AWS Personal Health Dashboard へのアクセスや、応答時間の短縮などです。
4. ナレッジマネジメントツールにサポートプランを記録します。サポートのリクエスト方法、サポートケースが記録された場合の通知先、インシデント中のエスカレーション方法を含めます。wiki は、サポートプロセスや連絡先の変更に気付いた人が誰でも、ドキュメントに必要な更新を行うことができるため、良い仕組みです。

実装計画に必要な工数レベル: 低。ソフトウェアやサービスのほとんどのベンダーは、サポートプランの登録を提供しています。ナレッジマネジメントシステムにサポートのベストプラクティスを記録して共有すると、本稼働環境に問題が発生した場合にどうすべきかをチームが確実に把握できます。

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

関連するドキュメント:

- AWS サポート プラン

関連サービス:

- AWS Business Support
- AWS Enterprise Support

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

ワークロード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。ワークロードの正常性は、ビジネス成果 (KPI) の達成と、ワークロードコンポーネントおよびアプリケーションの状態によって測定されます。KPI の例には、放棄されたショッピングカート数、注文数、コスト、料金、配分されたワークロード費用があります。複数のコンポーネントからテレメトリーを収集することができるとはいえ、全体的なワークロードの正常性についてのインサイトを提供するサブセットを選択します。ビジネスニーズの変化に合わせ、ワークロードのメトリクスを時間と共に調整します。

期待される成果:

- ビジネス上の成果を反映する KPI の達成を検証するメトリクスが特定されています。
- ワークロードの正常性について一貫性あるビューを提供するメトリクスがあります。
- ワークロードのメトリクスは、ビジネスニーズの変化に合わせて定期的に評価されます。

一般的なアンチパターン:

- ワークロードのすべてのアプリケーションがモニタリングされていても、ワークロードがビジネス上の成果を達成しているかどうかを判断できません。
- ワークロードのメトリクスは定義されていても、ビジネス KPI に関連付けられていません。

このベストプラクティスを活用するメリット:

- ワークロードをビジネス上の成果の達成に対して測定することができます。
- ワークロードが正常な状態なのか、介入が必要なのかを把握できます。

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

このベストプラクティスの目標は、ワークロードの正常性を判断できることです。 ワークロードの正常性は、ビジネス成果の達成と、ワークロードのコンポーネントおよびアプリケーションの状態によって判断されます。ビジネス KPI からさかのぼって、メトリクスを特定します。コンポーネントとアプリケーションの主要なメトリクスを特定します。ビジネスニーズの変化に応じて、ワークロードのメトリクスを定期的に評価します。

お客様事例

AnyCompany Retail では、ワークロードの正常性をアプリケーションとコンポーネントのメトリクスのコレクションによって判断しています。ビジネス KPIから始まり、ビジネス上の成果を達成しているかを示す注文率などのメトリクスを特定しています。また、ページ応答率などの主要なアプリケーションのメトリクスと、データベースへのオープン接続数などのコンポーネントのメトリクスも使用しています。ワークロードのメトリクスは、ワークロードの正常性を決定するうえで有効であるかを確認するために、四半期ごとに再評価しています。

実装手順

1. ビジネス KPI から初めて、ビジネス上の成果を達成していることを示すメトリクスを特定します。メトリクスを伴わない KPI がある場合は、不足しているビジネス KPI すべてについて追加のメトリクスを使用して、ワークロードを測定します。
   a. カスタムメトリクスは、アプリケーションから Amazon CloudWatch にパブリッシュできます。
   b. AWS Distro for OpenTelemetry を使用すると、既存のアプリケーションのメトリクスを収集して、新しいメトリクスの追加に利用できます。
   c. Enterprise Support を利用している場合は、テクニカルアカウントマネージャーに、Building a Monitoring Strategy Workshop (モニタリング戦略策定ワークショップ) を申し込むことができます。このワークショップは、ワークロードのオブザーバビリティ戦略策定の支援となります。
2. ワークロードのアプリケーションとコンポーネントのメトリクスを特定します。個別のコンポーネントとアプリケーションの正常性を示す主要なメトリクスにはどのようなものがあるでしょう

か。 アプリケーションとコンポーネントはさまざまなメトリクスを生成する可能性があるとはいえ、全体的な正常性を示す主要なメトリクスを 1～3 種類選択します。

3. ワークロードのメトリクスを定期的に評価するメカニズムを導入します。ビジネス KPI が変更された場合、ステークホルダーと協力してワークロードのメトリクスを更新します。ワークロードのコンポーネントとアプリケーションの進化に合わせて、ワークロードのメトリクスを調整します。

実装計画に必要な工数レベル: 中。アプリケーションにビジネス KPI のメトリクスを追加するには、ある程度の労力が必要になる場合があります。

リソース

関連する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 - アプリケーションは、ビジネス上の成果をサポートするテレメトリーを生成する必要があります。
- OPS04-BP02 ワークロードテレメトリーを実装して設定する - ビジネス上の成果をサポートするワークロードのメトリクスを定義する前に、テレメトリーを送信するワークロードを計測する必要があります。
- OPS08-BP01 主要業績評価指標 (KPI) を特定する - ワークロードのメトリクスを選択する前に、主要業績評価指標 (KPI) を特定する必要があります。

関連するドキュメント:

- Adding metrics and traces to your application on Amazon EKS with AWS Distro for OpenTelemetry, AWS X-Ray, and Amazon CloudWatch (AWS Distro for OpenTelemetry、AWS X-Ray、Amazon CloudWatch を使用して、Amazon EKS のアプリケーションにメトリクスとトレースを追加する)
- Amazon Builder's Library の
- ヘルスチェックを実装する
- アプリケーションを効率的にモニタリングする方法
- How to better monitor your custom application metrics using Amazon CloudWatch Agent (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のモニタリングを改善する方法)

関連動画:

- AWS re:Invent 2020: Monitoring production services at Amazon (AWS re:Invent 2020: Amazon における本稼働サービスのモニタリング)
- AWS re:Invent 2022 - Building observable applications with OpenTelemetry (BOA310) (AWS re:Invent 2022 - OpenTelemetry を使用したオブザーバブルなアプリケーションの構築 (BOA310))
- How to Easily Setup Application Monitoring for Your AWS Workloads - AWS Online Tech Talks (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)
- Mastering Observability of Your Serverless Applications - AWS Online Tech Talks (サーバーレスアプリケーションのオブザーバビリティをマスターする - AWS オンラインテックトーク)

関連する例:

- 1 つの可観測性ワークショップ

関連サービス:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

OPS08-BP03 ワークロードメトリクスを収集および分析する

ワークロードのメトリクスを定期的にプロアクティブにレビューすることにより、傾向を特定し、対応が必要かどうかを判断し、ビジネス上の成果の達成を検証できます。ワークロードのアプリケーションとコンポーネントからのメトリクスを一元的に集約します。ダッシュボードと分析ツールを使用してテレメトリーを分析し、ワークロードの正常性を判断します。組織のステークホルダーと定期的にワークロードの正常性レビューを実施するメカニズムを導入します。

期待される成果:

- ワークロードのメトリクスが、一元的な場所に収集されます。
- ワークロードの正常性の傾向を、ダッシュボードと分析ツールを使用して分析します。
- 組織で定期的なワークロードのメトリクスについてのレビューを実施します。

一般的なアンチパターン:

- 組織内で 2 つの別々のオブザーバビリティプラットフォームでワークロードからメトリクスを収集しています。これらのプラットフォームには互換性がないため、ワークロードの正常性は判断できません。
- ワークロードのコンポーネントのエラー率が徐々に増加しています。組織ではワークロードのメトリクスのレビューを定期的に実施していないため、このような傾向に気づきません。1 週間後、このコンポーネントでエラーが起こり、ワークロード障害が発生します。

このベストプラクティスを活用するメリット:

- ワークロードの正常性とビジネス上の成果の達成に対する意識が向上します。
- ワークロードの正常性の傾向が徐々に発展する可能性があります。

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

ワークロードのメトリクスを一元的な場所に収集します。ダッシュボードと分析ツールを使用して、ワークロード メトリクスを分析し、ワークロードの正常性に関するインサイトを獲得して、ワークロードの正常性の傾向を発展させ、ビジネス上の成果の達成を検証します。ワークロードのメトリクスの定期的なレビューを実施するメカニズムを導入します。

お客様事例

AnyCompany Retail では、毎週水曜日にワークロードメトリクスのレビューを実施しています。会社全体にわたるステークホルダーが集まり、前の週のメトリクスを確認します。このレビュー中には、分析ツールから収集した傾向とインサイトを中心に確認が行われます。主要なワークロードのメトリクスを提供する内部ダッシュボードは公開され、すべての従業員が確認し、検索できます。

実装手順

1. ワークロードの正常性に関連するワークロードのメトリクスを特定します。ビジネス KPI から始めて、ワークロードの正常性の全体像を提供するアプリケーション、コンポーネント、プラットフォームのメトリクスを特定します。
   a. カスタムメトリクスは、Amazon CloudWatch に公開できます。Amazon CloudWatch エージェントを活用して 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できます。
   b. AWS Distro for OpenTelemetry を使用すると、既存のアプリケーションのメトリクスを収集して、新しいメトリクスの追加に利用できます。

   c. Enterprise Support を利用している場合は、テクニカルアカウントマネージャーに、Building a Monitoring Strategy Workshop (モニタリング戦略策定ワークショップ) を申し込むことができます。このワークショップは、ワークロードのオブザーバビリティ戦略策定の支援となります。

2. ワークロードのメトリクスを一元的なプラットフォームに収集します。ワークロードのメトリクスが別のプラットフォームに分割されている場合は、傾向の分析と構築が困難となる場合があります。プラットフォームには、ダッシュボードと分析機能が必要です。

   a. Amazon CloudWatch を使用して、ワークロードのメトリクスを収集して保存できます。マルチアカウントトポロジの場合、ログアーカイブアカウントと呼ばれる一元的なロギングとモニタリングアカウントの使用をお勧めします。

3. ワークロードメトリクスの統合ダッシュボードを構築します。このビューをメトリクスのレビューと傾向の分析に使用します。

   a. カスタム CloudWatch ダッシュボードを作成して、ワークロードのメトリクスを統合ビューで収集できます。

4. ワークロードメトリクスのレビュープロセスを導入します。技術担当者および非技術担当者を含むステークホルダーと共に、毎週、隔週、または毎月の頻度でワークロードのメトリクスを確認します。このようなレビューセッションを使用すると、傾向を特定し、ワークロードの正常性に関するインサイトを獲得できます。

実装計画に必要な工数レベル: 高。ワークロードのメトリクスが一元的に収集されていない場合、単一のプラットフォームに統合するには多額の投資が必要になる場合があります。

リソース

関連する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 - ワークロードのメトリクスを選択する前に、主要業績評価指標 (KPI) を特定する必要があります。
- OPS08-BP02 ワークロードのメトリクスを定義する - 収集と分析を行う前に、ワークロードのメトリクスを定義する必要があります。

関連するドキュメント:

- Power operational insights with Amazon Quick (Amazon QuickSight を使用して運用上のインサイトを活用する)

- Using Amazon CloudWatch dashboards custom widgets (Amazon CloudWatch ダッシュボードでのカスタムウィジェットの使用)

関連動画:

- Create Cross Account & Cross Region CloudWatch Dashboards (クロスアカウントとクロスリージョンの CloudWatch ダッシュボードを作成する)
- Monitor AWS Resources Using Amazon CloudWatch Dashboards (Amazon CloudWatch ダッシュボードを使用して AWS リソースをモニタリングする)

関連する例:

- AWS 管理とガバナンスツールワークショップ - CloudWatch ダッシュボード
- Well-Architected Labs - Level 100: Monitoring with CloudWatch Dashboards (Well-Architected ラボ - レベル 100: Amazon CloudWatch ダッシュボードを使用したモニタリング)

関連サービス:

- Amazon CloudWatch
- AWS Distro for OpenTelemetry

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

ワークロードメトリクスのベースラインを設定すると、ワークロードの正常性とパフォーマンスの把握につながります。ベースラインを使用すると、パフォーマンスが低下しているか過剰になっているアプリケーションとコンポーネントを特定できます。ワークロードのベースラインを設定することにより、問題がインシデントになる前に軽減できるようになります。ベースラインは、アクティビティのパターンを開発し、メトリクスが期待値から逸脱した場合の異常検出を実装するための基盤となります。

期待される成果:

- 通常の状況におけるワークロードのベースラインレベルのメトリクスが設定されており、
- ワークロードが正常に機能しているかどうかを判断できます。

一般的なアンチパターン:

- 新しい機能のデプロイ後、リクエストのレイテンシーの急激なドロップが発生しました。処理済みのリクエストと全体的なレイテンシーの複合メトリクスのベースラインは設定されておらず、変更により改善がみられたのか、エラーが発生したのかを判断することができません。
- ユーザーアクティビティの突然なスパイクが発生しましたが、メトリクスのベースラインは設定されていません。このアクティビティのスパイクは、徐々にアプリケーションでのメモリーリークにつながり、最終的にワークロードがオフラインになってしまいます。

このベストプラクティスを活用するメリット:

- 主要なコンポーネントとアプリケーションのメトリクスを使用して、ワークロードのアクティビティの通常のパターンを把握できます。
- ワークロード、ワークロードのアプリケーションとコンポーネントが正常に動作しているか、介入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確立しない場合のリスクレベル: 中

実装のガイダンス

履歴データを使用して、ワークロードのアプリケーションとコンポーネントのワークロードメトリクスのベースラインを設定します。メトリクスのベースラインをメトリクスのレビューミーティングとトラブルシューティングで活用します。ワークロードのパフォーマンスを定期的に確認し、ベースラインをアーキテクチャの進化に合わせて調整します。

お客様事例

AnyCompany Retail では、すべてのコンポーネントとアプリケーションのベースラインが設定されています。AnyCompany Retail は、履歴データを使用して、2 か月のメトリクス期間にわたるワークロードメトリクスのベースラインを作成しました。AnyCompany Retail では、2 か月ごとにベースラインを再評価して、実際のデータに基づいて調整をおこなっています。

実装手順

1. 履歴データを使用して、ワークロードのメトリクスからさかのぼって、主要なコンポーネントとアプリケーションのメトリクスのベースラインを設定します。コンポーネントまたはアプリケーションごとのメトリクス数を制限して、モニタリング疲れを回避します。
    a. Amazon CloudWatch Metrics Insights を使用して、大規模なメトリクスをクエリして、傾向とパターンを特定します。

   b. Amazon CloudWatch 異常検出は、機械学習アルゴリズムを使用して、メトリクスの動作パターンを特定し、ベースラインを決定し、異常を検出します。
   c. Amazon DevOps Guru は、機械学習を使用して、ワークロードの運用上の問題を検出する機能を提供します。
   d. Enterprise Support を利用している場合は、テクニカルアカウントマネージャーに、Building a Monitoring Strategy Workshop (モニタリング戦略策定ワークショップ) を申し込むことができます。このワークショップは、ワークロードのオブザーバビリティ戦略策定の支援となります。
2. 特に重要なビジネスイベントの前に、ワークロードのメトリクスのベースラインを定期的に確認するメカニズムを導入します。少なくとも四半期に 1 回、履歴データを使用してワークロードのメトリクスのベースラインを評価します。メトリクスレビューミーティングでは、このベースラインを使用します。

実装計画に必要な工数レベル: 低。ワークロードのメトリクスが定まったら、ベースラインを設定するうえで、通常の動作パターンを特定するために十分なデータを収集する必要がある場合があります。

リソース

関連するベストプラクティス:

- OPS08-BP02 ワークロードのメトリクスを定義する - ベースラインを設定する前に、まずワークロードのメトリクスを決定する必要があります。
- OPS08-BP03 ワークロードメトリクスを収集および分析する - メトリクスのベースラインを設定する前に、ワークロードのメトリクスを収集して分析する必要があります。
- 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 - このベストプラクティスは、ベースラインを基盤に構築され、使用傾向を明らかにします。
- 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 - しきい値の特定とアラートの作成には、メトリクスのベースラインが必要となります。
- OPS08-BP07 ワークロードの異常が検出された場合に警告する - 異常検出には、メトリクスのベースラインが設定されている必要があります。

関連するドキュメント:

- AWS Observability Best Practices - Alarms (AWS のオブザーバビリティのベストプラクティス - アラーム)

- アプリケーションを効率的にモニタリングする方法
- How to set up CloudWatch Anomaly Detection to set dynamic alarms, automate actions, and drive online sales (CloudWatch 異常検出をセットアップして、動的アラームを設定し、アクションを自動化し、オンライン販売を促進する方法)
- Operationalizing CloudWatch Anomaly Detection (CloudWatch 異常検出を運用化する)

関連動画:

- AWS re:Invent 2020: Monitoring production services at Amazon (AWS re:Invent 2020: Amazon における本稼働サービスのモニタリング)
- AWS re:Invent 2021- Get insights from operational metrics at scale with CloudWatch Metrics Insights (AWS re:Invent 2021- CloudWatch Metrics Insights を使用して、大規模に運用メトリクスからインサイトを取得する)
- AWS re:Invent 2022 - Developing an observability strategy (COP302) (AWS re:Invent 2022 - オブザーバビリティ戦略を策定する (COP302))
- AWS Summit DC 2022 - Monitoring and observability for modern applications (AWS Summit DC 2022 - モダンアプリケーションのモニタリングとオブザーバビリティ)
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310) (AWS Summit DC 2022 - AWS を使用したフルスタックオブザーバビリティとアプリケーションモニタリング (COP310))

関連する例:

- AWS CloudTrail and Amazon CloudWatch Integration Workshop (AWS CloudTrail と Amazon CloudWatch の統合ワークショップ)

関連サービス:

- Amazon CloudWatch
- Amazon DevOps Guru

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。Quick 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口グの収集

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
- Quick
- 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口グの収集

- 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タ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 サポート 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
- サポート 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 システム停止時の顧客コミュニケーション計画を定義する
- 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。

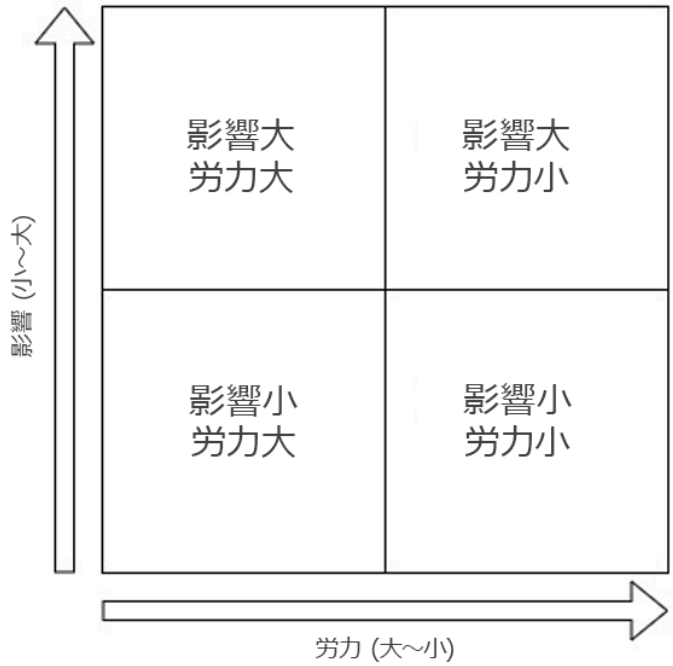
実装手順

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、サポート 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 システム停止時の顧客コミュニケーション計画を定義する

システム停止時に顧客やステークホルダーに継続して情報を提供するうえで、信頼性の高いシステム停止時向けのコミュニケーション計画を定義し、テストします。ユーザーが利用するサービスが影響を受けたタイミングと、サービスが正常に戻ったタイミングの両方で直接ユーザーにコミュニケーションを行います。

期待される成果:

- 予定されたメンテナンスから、ディザスタリカバリ計画が発動されるような予期せぬ大規模な障害まで、さまざまな状況に対応するコミュニケーション計画が準備されています。
- コミュニケーションでは、システムの問題に関して、明確かつ透明性の高い情報を提供し、システムのパフォーマンスについての顧客の憶測を回避します。
- カスタムのエラーメッセージとステータスページを使用して、ヘルプデスクへのリクエストのスパイクを低減し、ユーザーに継続的に情報提供を行います。
- コミュニケーション計画は定期的にテストし、実際にシステム停止が発生した際に、意図したとおりに機能することを確認します。

一般的なアンチパターン:

- ワークロードの停止が発生しても、コミュニケーション計画がありません。ユーザーには停止に関する情報が提供されていないため、トラブルチケットシステムにリクエストが殺到します。
- システム停止中にユーザーに E メール通知を送信しますが、サービス復旧のタイムラインが記載されていないため、ユーザーは停止を回避する計画を立てることができません。
- システム停止の際のコミュニケーション計画はあっても、テストしたことがなく、テストしていれば明らかになっていた可能性のある重要なステップを見逃していたため、システム停止が発生すると、コミュニケーション計画は失敗に終わります。
- システム停止中にユーザーに送信する通知には、過剰な技術的な詳細と AWS の NDA に基づく情報が記載されています。

このベストプラクティスを活用するメリット:

- システム停止中にコミュニケーションを継続すると、問題に関する進捗状況と解決までの推定時間を顧客は確実に把握できます。
- 明確に定義されたコミュニケーション計画を策定すると、顧客とエンドユーザーは、確実に十分な情報を入手でき、停止の影響を軽減するために必要となる追加の手順を実行できます。
- 適切なコミュニケーションを行うと、計画されたシステム停止と計画外の停止に関する意識が向上するため、顧客満足度が向上し、意図せぬ反応を制限して、顧客維持を促進できます。
- システム停止に関して、タイムリーかつ透明性が高いコミュニケーションを行うことにより、信頼性が高まり、顧客との関係を維持するうえで必要となる信頼が築かれることになります。
- システム停止の際や危機的状況下で実証済みのコミュニケーション戦略を採用することで、復旧能力の妨害につながる憶測やゴシップを低減できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

システム停止中に顧客に情報を提供するコミュニケーション計画は包括的なものであり、顧客に表示されるエラーページ、カスタム API エラーメッセージ、システムステータスバナー、正常性ステータスページなど、複数のインターフェイスをカバーしています。システムに登録ユーザーが含まれている場合は、E メール、SMS、プッシュ通知などのメッセージチャネル経由でコミュニケーションを行い、パーソナライズしたメッセージコンテンツを顧客に送信することができます。

顧客コミュニケーションツール

防御の最前線として、ウェブアプリケーションとモバイルアプリケーションは、システム停止中にわかりやすく情報豊富なエラーメッセージを提供し、トラフィックをステータスページにリダイレクトする機能を備えている必要があります。Amazon CloudFront は、カスタムのエラーコンテンツを定義して提供する機能を含む、フルマネージドコンテンツ配信ネットワークです。CloudFront のカスタムエラーページは、コンポーネントレベルの停止に関する顧客メッセージの最初のレイヤーとして適しています。CloudFront は、ステータスページの管理とアクティベーションを簡素化し、計画された停止や計画外の停止中にすべてのリクエストをインターセプトすることもできます。

カスタム API エラーメッセージを使用すると、個別のサービスに分離されている場合に停止が起こった際に、影響を検出して軽減するうえで役立ちます。Amazon API Gateway を使用すると、REST API のカスタムレスポンスを設定できます。これにより、API Gateway がバックエンドサービスに到達できない場合に、API の利用者に明確かつ意味あるメッセージを提供できます。カス

タムメッセージは、サービスレイヤーの停止により特定のシステム機能が低下した場合に、停止バナーコンテンツと通知のサポートにも使用できます。

ダイレクトメッセージは、最もパーソナライズされたタイプの顧客メッセージです。Amazon Pinpoint は、スケーラブルなマルチチャネルコミュニケーション向けのマネージドサービスです。Amazon Pinpoint を使用すると、SMS、E メール、音声、プッシュ通知、またはユーザー定義のカスタムチャネルを介して、影響を受ける顧客ベースに広範囲にわたりメッセージをブロードキャストするキャンペーンを構築できます。Amazon Pinpoint を使用してメッセージングを管理すると、メッセージキャンペーンを明確に定義し、テストでき、ターゲットを絞った顧客セグメントにインテリジェントに適用できます。設定したキャンペーンは、スケジュールしたり、イベントでトリガーしたりすることができ、簡単にテストを行うことができます。

お客様事例

AnyCompany Retail では、ワークロードで障害が発生すると、ユーザーに E メール通知を送信します。この E メールには、どのビジネス機能で障害が発生しているを説明し、サービスが復旧するタイミングについての現実的な見積りを提供します。さらに、ワークロードの正常性に関するリアルタイムの情報を表示するステータスページも提供しています。このコミュニケーション計画は、開発環境において年に 2 度テストし、効果的であることを確認しています。

実装手順

1. メッセージング戦略のコミュニケーションチャネルを決定します。アプリケーションのアーキテクチャの側面を考慮に入れて、顧客にフィードバックを提供するための最適な戦略を決定します。これには、エラーページとステータスページ、カスタム API エラーレスポンス、ダイレクトメッセージなど、概説したガイダンス戦略の 1 つ以上の戦略が含まれる場合があります。
2. アプリケーションのステータスページを設計します。ステータスページまたはカスタムエラーページが顧客に適していると判断した場合は、それらのページのコンテンツとメッセージを設計する必要があります。エラーページでは、アプリケーションが利用できない理由、再び利用できるようになるタイミング、その間にできることをユーザーに説明します。アプリケーションが Amazon CloudFront を利用している場合、カスタムエラーレスポンスを提供するか、Lambda at Edge を使用してエラーを翻訳したり、ページのコンテンツを書き換えたりすることができます。CloudFront を使用すると、宛先をアプリケーションコンテンツからメンテナンスや停止のステータスページを含む静的な Amazon S3 コンテンツのオリジンに置換することもできます。
3. サービスの適正な API エラーステータスセットを設計します。バックエンドサービスにアクセスできない場合に API Gateway が生成するエラーメッセージやサービスティアの例外には、エンドユーザーへの表示に適したユーザーフレンドリーなメッセージが含まれていない場合があります。バックエンドサービスのコードを変更する必要なく、API Gateway カスタムエラーレスポン

スを設定して、キュレートした API エラーメッセージにHTTP レスポンスコードをマップできます。

4. システムのエンドユーザーに対して関連性が高い内容とし、過度に技術的な詳細が含まれないよう、ビジネスの観点からメッセージを設計します。対象ユーザーを考慮してメッセージを調整します。例えば、社内ユーザーに対しては、代替システムを活用する回避策や手動のプロセスに誘導することができます。外部ユーザーに対しては、システム復旧まで待つか、システム復旧時に通知を受け取るためのアップデートにサブスクライブするようお願いすることができます。予期せぬシステム停止、計画されたメンテナンス、特定機能の障害や利用できなくなるといった部分的なシステム障害など、複数のシナリオについて、事前承認されたメッセージングを定義します。
5. 顧客メッセージをテンプレート化して自動化します。メッセージのコンテンツを設定したら、Amazon Pinpoint またはその他のツールを使用して、メッセージングキャンペーンを自動化することができます。Amazon Pinpoint を使用すると、影響を受けた特定のユーザー向けに顧客ターゲットセグメントを作成し、メッセージをテンプレートに変換できます。メッセージングキャンペーンの設定方法については、「Amazon Pinpoint チュートリアル」を確認してください。
6. メッセージング機能を顧客が使用するシステムに密結合することは避けてください。停止発生時にメッセージが正常に送信できることを確認するためには、メッセージング戦略がシステムデータストアやサービスに対して強制依存関係を持つべきではありません。メッセージングの可用性のために、複数のアベイラビリティーゾーンまたはリージョンからメッセージを送信する機能を構築することを検討してください。AWS サービスを使用してメッセージを送信している場合は、コントロールプレーンのオペレーションではなくデータプレーンのオペレーションを活用して、メッセージを呼び出します。

実装計画に必要な工数レベル: 高。コミュニケーション計画とコミュニケーションを送信するメカニズムの開発には、かなりの労力が必要になる場合があります。

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 - 担当者が対応方法を把握しておけるように、コミュニケーション計画にはランブックが関連付けられている必要があります。
- OPS11-BP02 インシデント後の分析を実行する - 停止後にはインシデント後分析を実施し、今後の停止を防ぐメカニズムを特定します。

関連するドキュメント:

- Error Handling Patterns in Amazon API Gateway and AWS Lambda (Amazon API Gateway と AWS Lambda のエラー処理パターン)
- Amazon API Gateway responses (Amazon API Gateway のレスポンス)

関連する例:

- AWS Health Dashboard
- Summary of the AWS Service Event in the Northern Virginia (US-EAST-1) Region (バージニア北部 (US-EAST-1) リージョンにおける AWS サービスイベントの概要)

関連サービス:

- AWS サポート
- AWS カスタマーアグリーメント
- Amazon CloudFront
- Amazon API Gateway
- Amazon Pinpoint
- Amazon S3

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。Quick 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:

- Quick
- 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

ワークロードを社内外のアーキテクチャのベストプラクティスに対して評価します。1 年に 1 回以上ワークロードのレビューを実施します。ソフトウェア開発サイクルの中で改善の機会を優先事項にします。

期待される成果:

- 年 1 回以上、アーキテクチャのベストプラクティスに対してワークロードを分析します。
- ソフトウェア開発プロセスにおいて改善の機会が等しく優先されます。

一般的なアンチパターン:

- ワークロードを数年前にデプロイして以来、アーキテクチャレビューを実施していない。
- 改善の機会の優先度が低く、ずっと後回しにされている。
- 組織のベストプラクティスに対する変更の実装について基準がない。

このベストプラクティスを活用するメリット:

- ワークロードがアーキテクチャのベストプラクティスに準拠した最新の状態に保たれます。
- ワークロードの進化を熟考した方法で実現できます。
- 組織のベストプラクティスを活用して、すべてのワークロードを改善できます。

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 年に 1 回以上ベースで、ワークロードの構造的レビューを実施します。社内外のベストプラクティスを使用してワークロードを評価し、改善の機会を特定します。ソフトウェア開発サイクルの中で改善の機会を優先事項にします。

お客様事例

AnyCompany Retail のすべてのワークロードは、毎年のアーキテクチャレビュープロセスを経由します。ベストプラクティスについての独自のチェックリストを作成し、すべてのワークロードに適用しています。AWS Well-Architected Tool のカスタムレンズ機能を活用し、ツールやベストプラクティスのカスタムレンズを使ってレビューを実施しています。レビューで見い出された改善の機会は、ソフトウェア開発サイクルの中で優先事項になっています。

実装手順

1. 1 年間に 1 回以上、本稼働ワークロードの定期的なアーキテクチャレビューを実施します。AWS 固有のベストプラクティスを含む文書化された構造基準を使用します。
   a. これらのレビューには、社内で独自に定義した基準を使用することをお勧めします。社内基準がない場合は、AWS Well-Architected フレームワークを使用することをお勧めします。
   b. AWS Well-Architected Tool を使用して社内ベストプラクティスのカスタムレンズを作成し、アーキテクチャレビューを実施できます。
   c. お客様は AWS ソリューションアーキテクトに連絡して、ワークロードのガイド付き Well-Architected フレームワークレビューを実施できます。
2. レビュー中に特定された改善機会を、ソフトウェア開発プロセスの中で優先事項に設定します。

実装計画に必要な工数レベル: 低。AWS Well-Architected フレームワークを使用して年次のアーキテクチャレビューを実施できます。

リソース

関連するベストプラクティス:

- OPS11-BP02 インシデント後の分析を実行する - インシデント後の分析は、改善項目を洗い出すもう一つの機会です。学んだ教訓で、アーキテクチャのベストプラクティスの社内リストを豊かにできます。
- OPS11-BP08 教訓を文書化して共有する - 独自のアーキテクチャのベストプラクティスを作成したら、組織全体で共有します。

関連するドキュメント:

- AWS Well-Architected Tool - カスタムレンズ
- AWS Well-Architected ホワイトペーパー - レビュープロセス
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool(Well-Architected レビューを、カスタムレンズと AWS Well-Architected ツールを使用してカスタマイズする)
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization (AWS Well-Architected カスタムレンズのライフサイクルを組織に実装する)

関連動画:

- Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool(レベル 100: AWS WELL-ARCHITECTED ツールのカスタムレンズ)

関連する例:

- AWS Well-Architected Tool

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

ナレッジ管理は、チームメンバーが業務を遂行するために情報を検索する際に役立ちます。従業員の学びが促進される組織では、個人を支援する情報が自由に共有されています。情報は探索したり検索したりできます。情報は正確かつ最新の内容です。新しい情報を作成し、既存の情報を更新し、古い情報をアーカイブするメカニズムが存在します。ナレッジ管理プラットフォームの最も一般的な例は、wiki などのコンテンツ管理システムです。

期待される成果:

- チームメンバーはタイムリーで正確な情報にアクセスできます。
- 情報は検索できます。
- 情報を追加、更新、アーカイブするメカニズムが導入されています。

一般的なアンチパターン:

- 一元化されたナレッジストレージがありません。チームメンバーは、個人のローカルマシンで自分用のメモを管理しています。
- 組織でホストする Wiki はあっても、情報を管理するメカニズムがないため、情報が古くなっています。
- 不足する情報が特定されても、チームの wiki にその情報の追加を要請するプロセスがありません。チームが独自に情報を追加しても、重要なステップを見逃してしまい、使用停止につながります。

このベストプラクティスを活用するメリット:

- 情報が自由に共有されるため、チームメンバーに支援が行き届きます。
- ドキュメントは最新の内容で検索可能であるため、新しいチームメンバーのオンボーディングがより迅速になります。
- 情報はタイムリーな内容で正確かつ実用的です。

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

ナレッジ管理は、従業員の学びが促進される組織の重要な側面です。まず、ナレッジを保存する中央リポジトリが必要です (一般的な例には、自己ホスト型の wiki があります)。ナレッジを追加、更新、アーカイブするためのプロセスを開発する必要があります。文書化すべき対象の基準を策定して、全チームメンバーが貢献できるプロセスを導入します。

お客様事例

AnyCompany Retail では、社内 Wiki をホストして、すべてのナレッジを保存しています。チームメンバーには、日常業務を遂行する際にナレッジベースに情報を追加することが推奨されています。四半期ごとに、部門横断的なチームが、更新が最も少ないページを評価し、アーカイブまたは更新する必要があるかを判断しています。

実装手順

1. まず、ナレッジを保存するコンテンツ管理システムを特定します。組織全体にわたるステークホルダーからの賛同を得ます。
   a. 既存のコンテンツ管理システムがない場合は、自己ホスト型の wiki を導入するか、バージョン管理リポジトリの導入から始めるかを検討します。
2. 情報を追加、更新、アーカイブするためのランブックを作成します。チームにこのプロセスについての教育を提供します。
3. コンテンツ管理システムに保存すべきナレッジを特定します。チームメンバーが実行する日常業務のアクティビティ (ランブックとプレイブック) から始めます。ステークホルダーと協力して、追加するナレッジに優先順位を付けます。
4. ステークホルダーと協力し、定期的に古い情報を特定し、アーカイブするか、最新の状態に更新します。

実装計画に必要な工数レベル: 中。既存のコンテンツ管理システムがない場合は、自己ホスト型の wiki またはバージョン管理されたドキュメントリポジトリを設定することができます。

リソース

関連するベストプラクティス:

- OPS11-BP08 教訓を文書化して共有する - ナレッジ管理を行うと、学んだ教訓の情報共有が容易になります。

関連するドキュメント:

- Atlassian - ナレッジマネジメント

関連する例:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

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 マネジメントコンソール、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。Quick 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
- Quick
- 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

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。実装に関する規範的なガイダンスについては、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。

ベストプラクティス領域

- セキュリティの基礎
- 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
- アプリケーションのセキュリティ

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?

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:
- SEC01-BP02 セキュアアカウントのルートユーザーおよびプロパティ
- SEC01-BP03 管理目標を特定および検証する:
- SEC01-BP04 セキュリティ脅威に関する最新情報を入手する:
- SEC01-BP05 セキュリティに関する推奨事項を常に把握する
- 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
- SEC01-BP07 脅威モデルを使用して脅威を特定し、緩和策の優先順位を付ける
- 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

SEC01-BP01 アカウントを使用してワークロードを分ける:

マルチアカウント戦略を取り、環境 (本番稼働、開発、テストなど) とワークロードの間に共通ガードレールを構成し、分離を確立します。アカウントレベルの分類は、セキュリティ、請求、アクセスのために強力な分離境界を提供するため、強く推奨されます。

期待される成果: クラウドオペレーション、無関係のワークロード、環境を別々のアカウントに分類し、クラウドインフラストラクチャ全体のセキュリティを向上させるアカウント構造。

一般的なアンチパターン:

- データ重要度レベルの異なる複数の無関係のワークロードを同一アカウントに配置する。
- きちんと定義されていない組織単位 (OU) 構造。

このベストプラクティスを活用するメリット:

- 誤ってワークロードにアクセスした場合の影響範囲が抑えられます。
- AWS サービス、リソース、およびリージョンへのアクセスの一元的ガバナンス。
- ポリシーとセキュリティサービスの一元管理により、クラウドインフラストラクチャのセキュリティを維持する。

- アカウント作成とメンテナンスプロセスの自動化。
- コンプライアンスや規制要件に対応した、インフラストラクチャの集中監査。

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 アカウント は、さまざまなデータ重要度レベルで稼働するワークロードまたはリソース間にセキュリティ分離境界を提供します。AWS は、マルチアカウント戦略を通して大規模にクラウドワークロードを管理し、この分離境界を活用するためのツールを提供します。AWS でのマルチアカウント戦略のコンセプト、パターン、および実装に関するガイダンスについては、「Organizing Your AWS Environment Using Multiple Accounts」(複数のアカウントを使用した AWS 環境の組織化) を参照してください。

一元管理下に複数の AWS アカウント がある場合、アカウントを組織単位 (OU) の層によって定義された階層に組織化する必要があります。次に、OU とメンバーアカウントに対してセキュリティ管理を組織化して適用することにより、組織内のメンバーアカウントに対して一貫性のある予防的制御を確立できます。セキュリティ管理は継承されるため、OU 階層の下位レベルにあるメンバーアカウントに対するアクセス許可をフィルタリングすることができます。優れた設計によりこの継承を利用して、各メンバーアカウントに対して望ましいセキュリティ管理を達成するのに必要なセキュリティポリシーの件数と複雑性を軽減します。

AWS Organizations および AWS Control Tower は、AWS 環境でこのマルチアカウント構造を実装および管理するのに使用できる 2 つのサービスです。AWS Organizations では、1 つまたは複数の OU 層 (それぞれに複数のメンバーアカウントを含む) で定義された階層にアカウントを組織化できます。サービス管理ポリシー (SCP) により、組織管理者がメンバーアカウントできめ細やかな予防的コントロールを確立し、AWS Config はメンバーアカウントに関して積極的かつ検出的コントロールを確立するのに使用できます。多くの AWS サービス が AWS Organizations と統合し、組織内のすべてのメンバーアカウントで、委任管理制御とサービス固有のタスク実行を提供します。

AWS Organizations 上の層にある AWS Control Tower は、ランディングゾーンにマルチアカウント AWS 環境向けの ワンクリックベストプラクティスセットアップを提供します。ランディングゾーンは、Control Tower によって確立されるマルチアカウント環境への入口です。Control Tower には、AWS Organizations と比較していくつかの利点があります。アカウントガバナンスを改善する 3 つの利点には次のようなものがあります。

- 組織に対して承認されたアカウントに自動適用される、統合された必須のセキュリティガードレール。

- 所定の OU セットに対してオン/オフと切り替えられるオプションのガードレール。
- AWS Control Tower Account Factory では、事前承認されたベースラインと組織内の構成オプションを含むアカウントを自動的にデプロイできます。

実装手順

1. 組織単位構造を設計する: 組織単位を適切に設計することにより、サービスコントロールポリシーやその他のセキュリティコントロールの作成と保守に必要な管理負担を軽減できます。組織単位構造は、貴社のビジネスニーズ、データ重要度、およびワークロード構造に合致したものである必要があります。
2. マルチアカウント環境向けのランディングゾーンを作成する: ランディングゾーンは一貫したセキュリティとインフラストラクチャ基盤を提供します。そこから組織はワークロードを迅速に開発、立ち上げ、デプロイできます。カスタムビルドのランディングゾーンまたは AWS Control Tower を使用して、環境のオーケストレーションを実行できます。
3. ガードレールを確立する: ランディングゾーンを通して環境に一貫性のあるセキュリティガードレールを実装します。AWS Control Tower は、必須とオプションのコントロールのリストを提供します。必須コントロールは、Control Tower 実装時に自動的にデプロイされます。強く推奨されたコントロールとオプションのコントロールのリストを確認し、ニーズに適したコントロールを実装します。
4. 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、指定したリージョンのみに伝播されます。このアクションは、Control Tower 使用時はコンソール経由で、または AWS Organizations で IAM アクセス許可を調整することにより実行できます。
5. AWSCloudFormation StackSets を検討する: 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しやすくなります。

リソース

関連する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

関連するドキュメント:

- AWS Control Tower

- AWS セキュリティ監査のガイドライン
- IAM ベストプラクティス
- CloudFormation StackSets を使用して、複数の AWS アカウント とリージョン全体にリソースをプロビジョニングする
- 組織関連の FAQ
- AWS Organizations 用語およびコンセプト
- AWS Organizations マルチアカウント環境の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
- AWS アカウント管理リファレンスガイド
- 複数のアカウントを使用した AWS 環境の組織化

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
- AWS Control Tower を使って複数のアカウントをビルドおよび統制する
- 既存の組織に対して Control Tower を有効化する

関連ワークショップ:

- Control Tower Immersion Day

SEC01-BP02 セキュアアカウントのルートユーザーおよびプロパティ

ルートユーザーは AWS アカウント で最も権限が高いユーザーであり、アカウント内の全リソースに対する完全な管理者アクセスがあるだけでなく、場合によってはセキュリティポリシーによる制限の対象外となります。ルートユーザーへのプログラムによるアクセスを無効化し、ルートユーザーに対する適切なコントロールを確立し、さらにルートユーザーの定期的使用を避けることにより、ルート認証情報を不用意に曝露するリスク、それによるクラウド環境の侵害を軽減することができます。

期待される成果: ルートユーザーをセキュリティ保護することにより、ルートユーザー認証情報を不正使用した場合の偶発的または意図的な損害が生じる可能性が低減されます。検出コントロールを確立することによっても、ルートユーザーを使ったアクションが取られると適切な担当者にアラートを送信できます。

一般的なアンチパターン:

- ルートユーザー認証情報を必要とする少数以外のタスクに対してもルートユーザーを使用する。
- 緊急時に重要なインフラストラクチャ、プロセス、担当者が正常に機能するかどうかを検証するために、定期的な緊急時対応計画のテストを怠っている。
- 典型的なアカウントログインフローのみを考慮し、代替アカウント回復方法を考慮することも、テストすることもしていない。
- DNS、E メールサーバー、および携帯電話会社がアカウント復旧フローで使用されるにもかかわらず、重要なセキュリティ境界の一部として対処していない。

このベストプラクティスを活用するメリット: ルートユーザーへのアクセスを確保することにより、アカウントでアクションをコントロールおよび監査できるという安心感が向上する。

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

AWS は、アカウントを保護するのに役立つ多くのツールを提供しています。ただし、これらの対策の一部は既定では有効になっていないため、実装するには直接的な措置を講じる必要があります。これらの推奨事項を、AWS アカウント をセキュリティ保護するための基本的なステップと考えてください。これらのステップを実装する際、セキュリティ管理を継続的に評価およびモニタリングすることが重要となります。

AWS アカウント を初めて作成する際は、アカウント内のすべての AWS のサービスとリソースに完全なアクセス許可を持つ 1 つの ID から始めます。この ID は、AWS アカウント のルートユーザーと呼ばれます。アカウント作成に使用した E メールアドレスとパスワードを使用すれば、ルートユーザーとしてログインできます。AWS ルートユーザーに付与されるアクセス許可が昇格したため、特にそれを必要とするタスクを実行する AWS ルートユーザーの使用は制限する必要があります。ルートユーザーのログイン認証情報は注意して保護し、AWS アカウント ルートユーザーに対しては多要素認証 (MFA) を必ず有効にしておく必要があります。

ユーザー名、パスワード、多要素認証 (MFA) デバイスを使用してルートユーザーにログインする通常の認証フローに加えて、アカウントに関連付けられた E メールアドレスと電話番号にアクセスし、AWS アカウント ルートユーザーにログインするためのアカウント復旧フローもあります。そのため、復旧メールを送信するルートユーザーの E メールアカウントと、そのアカウントに関連する電話番号をセキュリティ保護することも同程度に重要となります。また、ルートユーザーに関連付けられた E メールアドレスが、同じ AWS アカウント の E メールサーバーやドメインネームサービ

ス (DNS) リソースでホストされている場合、潜在的な循環依存性についても考慮する必要があります。

AWS Organizations を使用する場合、それぞれにルートユーザーが含まれる AWS アカウント が複数あります。1 つのアカウントを管理アカウントに指定し、その管理アカウントの下に何層ものメンバーアカウントを追加することができます。管理アカウントのルートユーザーのセキュリティ保護を優先してから、メンバーアカウントのルートユーザーに対処してください。管理アカウントのルートユーザーをセキュリティ保護する戦略は、メンバーアカウントのルートユーザーとは異なり、メンバーアカウントのルートユーザーに対しては予防的なセキュリティコントロールを講じることができます。

実装手順

ルートユーザーのコントロールを確立するには、次の実装ステップが推奨されます。該当する場合、推奨事項は CIS AWS Foundations ベンチマークバージョン 1.4.0 に相互参照されます。AWS アカウント およびリソースのセキュリティ保護については、これらのステップに加え、AWS ベストプラクティスガイドラインも参照してください。

予防的コントロール

1. アカウントに対して、正確な 連絡先情報を設定します。
   a. この情報は、紛失したパスワードの復旧フロー、紛失した MFA デバイスアカウントの復旧フロー、およびチームとの重要なセキュリティ関連のコミュニケーションに使用されます。
   b. 企業ドメインによってホストされた E メールアドレスを使用します (ルートユーザーの E メールアドレスとしては、できれば配布リストのほうが望ましい)。個人の E メールアカウントではなく配布リストを使うことにより、長期的にはルートアカウントへのアクセスに対して冗長性と継続性を追加することになります。
   c. 連絡先情報に記載された電話番号は、この目的専用の安全なものである必要があります。この電話番号をどこかに記載したり、誰かと共有したりしないでください。
2. ルートユーザーにはアクセスキーを作成しないでください。アクセスキーが存在する場合は、それを削除します (CIS 1.4)。
   a. ルートユーザーに対する長期保存可能なプログラム認証情報 (アクセスキーとシークレットキー) は排除します。
   b. ルートユーザーのアクセスキーがすでにある場合、それらのキーを使うプロセスを、AWS Identity and Access Management (IAM) ロールからの臨時アクセスキーを使い、次に ルートユーザーのアクセスキーを削除することにより、移行させる必要があります。
3. ルートユーザーの認証情報を保管する必要があるかどうかを決定します。

   a. AWS Organizations を使用して新しいアカウントを作成している場合、新規メンバーアカウントのルートユーザーの初期パスワードはランダムな値に設定され、決して公開されることはありません。必要に応じ、AWS Organization 管理アカウントからのパスワードリセットフローを使って、メンバーアカウントへのアクセスを獲得することを検討してください。

   b. スタンドアロン AWS アカウント または管理 AWS Organization アカウントに対しては、ルートユーザーの認証情報を作成して安全に保管することを検討してください。ルートユーザーの MFA を有効にする

4. AWS マルチアカウント環境のメンバーアカウントのルートユーザーに対しては、予防的コントロールを有効にします。

   a. メンバーアカウントに対して、ルートユーザー向けのルートアクセスキーの作成を許可しない予防的ガードレールの有効化を検討してください。

   b. メンバーアカウントに対して、ルートユーザーとしてのアクションを許可しない予防的ガードレールの有効化を検討してください。

5. ルートユーザーの認証情報が必要な場合:

   a. 複雑なパスワードを使用します。

   b. ルートユーザー、特に AWS Organizations 管理 (支払者) アカウント (CIS 1.5) に対しては多要素認証 (MFA) を有効化します。

   c. 回復力とセキュリティのために、ハードウェア MFA デバイスを検討してください。これは、単回使用デバイスを使用することにより、MFA コードを含むデバイスが他の目的に再使用される可能性が少なくなるためです。電池式のハードウェア MFA デバイスが定期的に交換されていることを検証してください。(CIS 1.6)

      - ルートユーザーに対して MFA を設定するには、仮想 MFA または ハードウェア MFA デバイスのいずれかを有効化する手順に従ってください。

   d. バックアップ用に複数の MFA デバイスを登録することを検討してください。 アカウントごとに最大 8 台の MFA デバイスを登録できます。

      - ルートユーザーに対して複数の MFA デバイスを登録すると、MFA デバイス紛失時にアカウントを復旧するフローが無効になることに注意してください。

   e. パスワードは安全に保管し、電子的にパスワードを保管する際は循環依存関係を検討してください。入手するために同じ AWS アカウント へのアクセスが必要となる方法でパスワードを保管しないでください。

6. オプション: ルートユーザーに対して定期的なパスワードローテーションスケジュールを設定することを検討します。

- 認証情報管理のベストプラクティスは、規制およびポリシー要件によって異なります。MFA によって保護されるルートユーザーは、認証の単一要素としてパスワードに依存しません。
- 定期的にルートユーザーパスワードを変更することにより、誤って露出したパスワードが不正使用されるリスクを低減します。

検出コントロール

- ルート認証情報の使用を検出するアラームを作成します (CIS 1.7)。Amazon GuardDuty を有効にすることにより、RootCredentialUsage 所見を使ってルートユーザー API 認証情報の使用をモニタリングおよびアラートを発行します。
- AWS Config 用の AWS Well-Architected セキュリティの柱コンフォーマンスパックに含まれる検出コントロール、またはAWS Control Tower を使用している場合は、Control Tower 内にある強く推奨されるコントロールを評価および実装します。

運用ガイダンス

- 組織で、ルートユーザー認証情報へのアクセスが必要な担当者を決定します。
  - 1 人の担当者がすべての必要な認証情報とルートユーザーアクセスを取得するために MFA にアクセスするのを回避するため、2 人制を採用します。
  - アカウントに関連付けられた電話番号と E メールエイリアス (パスワードリセットと MFA リセットフローに使用される) は、個人ではなく、組織が管理するよう徹底してください。
- ルートユーザーは例外的にのみ使用します (CIS 1.7)。
  - AWS のルートユーザーを、たとえ運営業務であっても日常的なタスクに使用してはなりません。ルートユーザーを必要とする AWS タスクを実行するには、ルートユーザーとしてのみログインしてください。その他すべてのアクションは、適切なロールを持つ他のユーザーが実行しなければなりません。
- ルートユーザーにアクセスできることを定期的にチェックし、ルートユーザー認証情報を使用する必要がある緊急事態の前に手順をテストしておきます。
- アカウントに関連付けられた E メールアドレスと、その他の連絡先に記載された E メールアドレスが有効であることを定期的にチェックします。これらの E メールの受信箱に、<abuse@amazon.com>から受信したセキュリティ通知が届いていないかどうかモニタリングしてください。また、アカウントに関連付けられた電話番号があれば、それが通じることも確認してください。

- ルートアカウントの不正使用に対処するインシデント対応手順を準備しておきます。AWS アカウント に対するインシデント対応戦略の策定に関する詳細については、AWS セキュリティインシデント対応ガイドと、セキュリティの柱のホワイトペーパーの「インシデント対応」セクションに記載されたベストプラクティスを参照してください。

リソース

関連する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:
- SEC02-BP01 強力なサインインメカニズムを使用する
- 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
- SEC10-BP05 アクセスを事前プロビジョニングする

関連するドキュメント:

- AWS Control Tower
- AWS セキュリティ監査のガイドライン
- IAM ベストプラクティス
- Amazon GuardDuty – ルート認証情報使用アラート
- CloudTrail によるルート認証情報使用モニタリングに関するステップバイステップガイダンス
- AWS での使用が認可された MFA トークン
- AWS に break glass アクセスを実装する
- AWS アカウント を改善するためのトップ 10 セキュリティアイテム
- AWS アカウント の不正なアクティビティに気付いた場合はどうすればよいですか?

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
- Security Best Practices the Well-Architected Way (Well-Architected の手法によるセキュリティのベストプラクティス)
- AWS re:inforce 2022 – Security best practices with AWS IAM からの「Limiting use of AWS root credentials (AWS ルート認証情報の使用を制限する)」

関連する例とラボ:

- ラボ: AWS アカウント and root user (AWS アカウントのセットアップとルートユーザー)

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 CSPM: 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脅威を特定し、緩和策の優先順位を付ける

脅威のモデル化を実行し、ワークロードの潜在的脅威と関連付けられた緩和策を特定し、最新の状態を維持します。脅威に優先順位を付け、セキュリティコントロール緩和策を調整して防止、検出、対応を行います。ワークロードの内容、および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保持および維持します。

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

脅威のモデル化とは何ですか?

定義は、「脅威のモデル化は、価値のある対象を保護する文脈、脅威と緩和策を特定、伝達、理解するためのもの」となっています。 – The Open Web Application Security Project (OWASP) Application Threat Modeling (オープン Web アプリケーションセキュリティプロジェクト (OWASP) アプリケーション脅威モデル化)

脅威をモデル化すべきなのはなぜですか?

システムは複雑であり、時代とともに次第に複雑かつ高性能となり、提供するビジネス価値は向上し、顧客満足度とエンゲージメントは強化されています。つまり、IT 設計を決定する際は、増え続けるユースケースの件数を考慮する必要があるということです。このような複雑で数が多いユースケースの組み合わせは、非構造化アプローチでは一般に脅威の検出と緩和に効果がありません。代わりに必要となるのは、システムに対する潜在的な脅威を列挙し、緩和策を考案し、その緩和策に優先順位をつけて、組織の限定的なリソースがシステム全体のセキュリティ体制の改善に最大の効果を発揮できるような体系的アプローチです。

脅威のモデル化は、このような体系的アプローチを提供する設計となっており、その狙いは、ライフサイクルの後半と比較すると相対的にコストと労力が低い設計プロセスの早い段階で問題を発見し、対処することです。このアプローチは、「シフトレフト (前倒し)」セキュリティの業界原則と一致しています。最終的に脅威のモデル化は組織のリスク管理プロセスと統合し、脅威駆動型アプローチを使用して、実装するコントロールの決定を促します。

脅威のモデル化は、いつ実行すべきですか?

ワークロードのライフサイクルにおけるできるだけ早い段階で脅威のモデル化を開始することにより、より柔軟に特定した脅威への対策を実施できるようになります。ソフトウェアのバグと同様、脅威を特定するのが早いほど、その対策のコスト効率が向上します。脅威モデルはライブドキュメントであり、ワークロードの変化に応じて進化し続ける必要があります。大きな変化、脅威の状況における変化が生じた場合や、新たな機能またはサービスを採用した場合などを含む、経時的な脅威モデルを保持します。

実装手順

脅威のモデル化の実行方法を教えてください

脅威のモデル化にはさまざまな実行方法があります。プログラミング言語と同様、それぞれに長所と短所があり、自分に最も適した方法を選択する必要があります。1 つのアプローチは、Shostack's 4

Question Frame for Threat Modeling (脅威のモデル化のための Shostack の 4 つの質問フレーム) から始めるやり方です。これは、脅威のモデル化の演習に構造を与える自由形式の質問です。

1. うまくいっているものは何か?

   この質問の目的は、構築しているシステム、さらにはセキュリティに関連するシステムに関する詳細を理解してそれに合意するのを支援することです。構築している対象を視覚化できるため、モデルや図を作成するのが、この質問に対する回答として最も良くある方法です。たとえば、データフロー図などです。システムに関する推測と重要な詳細を書き留めることも、対象範囲を定義するのに役立ちます。これにより、脅威モデルに貢献する担当者全員が同一の作業に集中でき、範囲外のトピック (システムの古いバージョンを含む) に時間を取られて回り道をすることを回避できます。たとえば、ウェブアプリケーションを構築している場合、ブラウザクライアントのオペレーティングシステムの信頼できるブートシーケンスをモデル化する脅威については、あまり時間をかける価値があるとは思えません。

2. どんな問題が起きる可能性があるでしょうか?

   ここで、システムに対する脅威を特定します。脅威とは、望ましくない影響を生じさせ、システムのセキュリティに悪影響を及ぼす恐れのある、偶発的または意図的なアクションや事象を指します。どのような問題が起きるかをはっきりと理解していなければ、何も対策は打てません。

   何が問題になるのかに関して、定型的なリストは存在しません。このリストを作成するには、チーム内の個人全員と脅威のモデル化に関与する関係担当者間のブレインストーミングとコラボレーションが必要となります。ブレインストーミングは、STRIDE などの脅威を特定するモデルを使用すると実施しやすくなります。これは、評価するためのさまざまなカテゴリ (スプーフィング、改ざん、否認、情報漏洩、サービス拒否、権限昇格) を提案するものです。さらに、既存のリストを見直し、OWASP トップ 10、HiTrust 脅威カタログ、そして組織独自の脅威カタログなどのインスピレーションを調査することもブレインストーミングに役立ちます。

3. それをどうするのですか?

   前の質問と同様、考えられる緩和策について定型的なリストはありません。このステップに対する入力項目は、特定された脅威、アクター、および前のステップからの改善点です。

   セキュリティとコンプライアンスは、AWS とお客様との間で共有される責任です。「それをどうするのですか?」という質問を行うときは、「誰がその責任者なのか?」ということも尋ねていると理解することが重要です。お客様と AWS 間の責任のバランスを理解することにより、お客様のコントロール下にある脅威のモデル化演習の範囲を理解するのに役立ちます。これは通常、AWS サービス設定オプションとお客様独自のシステムごとの緩和策を組み合わせたものです。

共有責任の AWS 担当部分については、AWS サービスが多くのコンプライアンスプログラムの範囲内であることに気づくと思います。これらのプログラムは、セキュリティとクラウドのコンプライアンスを維持するためにAWS に配置された堅牢なコントロールを理解するのに役立ちます。これらのプログラムからの監査レポートは、AWS 顧客向けに AWS Artifact からダウンロードできます。

どの AWS サービスを使用していても、必ずお客様の責任となる要素が存在し、これらの責任に合わせた緩和策を脅威モデルに組み込む必要があります。AWS サービス自体のセキュリティコントロール緩和のためには、たとえば、AWS Identity and Access Management (認証と承認)、データ保護 (静止時と転送時)、インフラストラクチャセキュリティ、ログ、モニタリングなどのドメインを含む、さまざまなドメイン全体にセキュリティコントロールの実装を検討することが推奨されます。各 AWS サービスのドキュメントには、専用のセキュリティに関する章が入っており、緩和策とみなされるセキュリティコントロールに関するガイダンスを提供します。重要ですので、記述しているコードとコード依存関係を考慮し、それらの脅威に対応するために設定できるコントロールについて考えてください。これらのコントロールは、入力の検証、セッションの取扱い、および範囲の取り扱いなどが考えられます。多くの場合、脆弱性の大部分はカスタムコードで発生するため、この領域を注視してください。

4. うまくいきましたか?

狙いは、チームと組織が脅威モデルの質と、脅威のモデル化を行う際の時間的な速さを改善することです。これらの改善は、練習、学習、指導、レビューを組み合わせることで実現します。深く掘り下げて実践的な学習を行うため、お客様とチームが「Threat modeling the right way for builders training course (ビルダー向けの正しい脅威モデル化トレーニングコース)」またはワークショップを終了することが推奨されます。さらに、組織のアプリケーション開発ライフサイクルに脅威モデル化を統合する方法についてガイダンスを求めている場合、AWS セキュリティブログの「How to approach threat modeling (脅威のモデル化にアプローチする方法)」を参照してください。

リソース

関連するベストプラクティス:

- SEC01-BP03 管理目標を特定および検証する:
- SEC01-BP04 セキュリティ脅威に関する最新情報を入手する:
- SEC01-BP05 セキュリティに関する推奨事項を常に把握する
- 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

関連するドキュメント:

- How to approach threat modeling (脅威のモデル化にアプローチする方法) (AWS セキュリティブログ)
- 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)

関連動画:

- AWS Summit ANZ 2021 - How to approach threat modelling (脅威モデル化へのアプローチ方法)
- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery (セキュリティのスケーリング - 迅速かつ安全な配信のための最適化)

関連トレーニング:

- Threat modeling the right way for builders (ビルダー向けの正しい脅威モデル化) – AWS Skill Builder virtual self-paced training （ AWS スキルビルダーの仮想セルフペーストレーニング)
- Threat modeling the right way for builders (ビルダー向けの正しい脅威モデル化) - AWS ワークショップ

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 アイデンティティ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

サインイン (サインイン認証情報を使った認証) は、多要素認証 (MFA) などのメカニズムを使わない場合、特にサインイン認証情報が不用意に開示されたり、容易に推測されたりする場合に、リスクが発生する恐れがあります。MFA や強力なパスワードポリシーを要求することで、これらのリスクを軽減する強力なサインインのメカニズムを使用します。

期待される成果: AWS Identity and Access Management (IAM) ユーザー、AWS アカウント ルートユーザー、AWS IAM アイデンティティセンター (AWS シングルサインオンの後継サービス)、およびサードパーティー ID プロバイダー向けに強力なサインインメカニズムを使用することにより、AWS の認証情報に対する意図しないアクセスのリスクを軽減します。これは、MFA が必須となり、強力なパスワードポリシーが適用され、異常なログイン動作が検出されることを意味します。

一般的なアンチパターン:

- 複雑なパスワードや MFA など、自分のアイデンティティに対して強力なパスワードポリシーを適用しない。
- 複数のユーザー間で同一の認証情報を共有する。
- 疑わしいサインインに対して検出コントロールを使用しない。

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: 高

実装のガイダンス

人的 ID が AWS にサインインする方法は多数あります。AWS ベストプラクティスは、AWS に認証する際にフェデレーション (直接フェデレーションまたは AWS IAM アイデンティティセンター を使用) を使って、一元化された ID プロバイダーに依存する方法です。この場合、ID プロバイダーまたは Microsoft Active Directory を使って、セキュアなサインインプロセスを確立する必要があります。

最初に AWS アカウント を開いたとき、AWS アカウント ルートユーザーから始めます。ユーザー (およびルートユーザーを必要とする タスク) へのアクセスを設定するには、アカウントのルートユーザーのみを使用する必要があります。.AWS アカウント を開いた直後にアカウントのルートユーザーに対して MFA を有効化し、AWS ベストプラクティスガイドを使用してルートユーザーをセキュリティ保護することが重要です。

AWS IAM アイデンティティセンター でユーザーを作成する場合、そのサービスでサインインプロセスをセキュリティ保護します。消費者アイデンティティについては、Amazon Cognito user pools を使用して、そのサービスで、またはAmazon Cognito user pools がサポートする ID プロバイダーの 1 つを使ってサインインプロセスをセキュリティ保護します。

AWS Identity and Access Management (IAM) ユーザーを使用している場合、IAM を使ってサインインプロセスをセキュリティ保護することになります。

サインイン方法に関係なく、強力なサインインポリシーを適用することが不可欠です。

実装手順

一般的な強力なサインインに関する推奨事項は次の通りです。実際に行う設定は、貴社のポリシーによって設定するか、または NIST 800-63 のような標準を使います。

- MFA が必要です。人的 ID とワークロードに対しては、MFA を義務付けることがIAM のベストプラクティスです。MFAを有効にすることで、追加のセキュリティ層が提供されます。この層では、ユーザーがサインイン認証情報、ワンタイムパスワード (OTP)、またはハードウェアデバイスから暗号的に検証および生成された文字列を提供することが求められます。
- 最小パスワード文字数を適用します。これは、パスワードの強さにおける主な要素です。
- パスワードの複雑性を適用すると、パスワードを推測しにくくなります。
- ユーザー自身によるパスワードの変更を許可します。
- 共有認証情報ではなく、個別の ID を作成します。個別の ID を作成することで、各ユーザーに固有のセキュリティ認証情報を付与することができます。個別のユーザーを作成することで、各ユーザーのアクティビティを監査する機能が利用できます。

IAM Identity Center レコメンデーション

- IAM Identity Center は、デフォルトディレクトリを使用する際、パスワードの文字数、複雑性、および再使用要件を確立する、事前定義された パスワードポリシーを提供します。
- MFA を有効にし、アイデンティティソースがデフォルトディレクトリ、AWS Managed Microsoft AD、または AD Connector の場合、MFA に対してコンテキストアウェアまたは常時オン設定を行います。
- ユーザーが、自分の MFA デバイスを登録できるようにします。

Amazon Cognito user pools ディレクトリのレコメンデーション:

- パスワードの強さ設定を行います。
- ユーザーに対してMFA を義務付けます。
- 疑わしいサインインをブロックできる適応型認証などの機能に対して、Amazon Cognito user pools上級セキュリティ設定を使用します。

IAM ユーザーのレコメンデーション:

- IAM Identity Center または直接フェデレーションを使用することが理想的です。しかし、IAM ユーザー向けのニーズもあるでしょう。その場合は、IAM ユーザー向けにパスワードポリシーを設定します。パスワードポリシーを使用して、最小文字数、またはアルファベット以外の文字が必要かどうかなどの要件を定義できます。
- IAM ポリシーを作成して、MFA サインインを適用し、ユーザーが自分のパスワードと MFA デバイスを管理できるようにします。

リソース

関連するベストプラクティス:

- 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
- SEC03-BP08 組織内でリソースを安全に共有する

関連するドキュメント:

- AWS IAM アイデンティティセンター (AWS シングルサインオンの後継サービス) パスワードポリシー
- IAM ユーザーのパスワードポリシー
- AWS アカウント のルートユーザーのパスワードの設定
- Amazon Cognito パスワードポリシー
- AWS 認証情報
- IAM セキュリティのベストプラクティス

関連動画:

- Managing user permissions at scale with AWS IAM アイデンティティセンター (AWS SSO を使用した大規模なユーザー権限の管理)
- Mastering identity at every layer of the cake (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

何らかの認証を行う際、認証情報が誤って開示、共有、盗難されたりなどのリスクを軽減または排除するには、長期的認証情報ではなく一時的な認証情報を使うことが推奨されます。

期待される成果: 長期的認証情報のリスクを軽減するには、人的および機械両方の ID にできるだけ一時的な認証情報を使用するようにします。長期的認証情報を使用すると、多くのリスクが生じます。たとえば、パブリックな GitHub リポジトリにコードでアップロードすることができます。一時的な認証情報を使うことにより、認証情報が侵害されるリスクが大幅に減少します。

一般的なアンチパターン:

- 開発者が、フェデレーションを使って CLI から一時的な認証情報を取得するのではなく、IAM users からの長期的なアクセスキーを使用する。
- 開発者がコードに長期的アクセスキーを埋め込んで、そのコードをパブリック Git リポジトリにアップロードする。
- 開発者が、モバイルアプリに長期的アクセスキーを埋め込んで、アプリストアで公開する。
- ユーザーが長期的アクセスキーを他のユーザー、または従業員と共有し、長期的アクセスキーを所有したまま離職する。
- 一時的認証情報を使用できるのに、マシン ID に対して長期的なアクセスキーを使用する。

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: 高

実装のガイダンス

すべての AWS API と CLI リクエストに対して、長期的認証情報ではなく一時的なセキュリティ認証情報を使用します。AWS サービスに対する API および CLI リクエストは、ほとんどの場合、AWS アクセスキーを使って署名する必要があります。これらのリクエストの署名に使用する認証情報は、一時的でも長期的でもかまいません。長期的認証情報 (長期的アクセスキー) を使用すべき唯一の状況は、IAM ユーザーまたは AWS アカウント ルートユーザーを使用している場合です。AWS に対してフェデレーションを行うか、または他の方法により IAM ロールを担う場合、一時的認証情報が生成されます。サインイン認証情報を使って AWS マネジメントコンソール にアクセスしても、AWS サービスへのコールを行うために一時的な認証情報が生成されます。長期的認証情報が必要な状況はほとんどなく、一時的な認証情報でほとんどのタスクを遂行できます。

一時的な認証情報を優先して長期的な認証情報の使用を回避することは、フェデレーションと IAM ロールを優先して IAM ユーザーの使用を減少させる戦略と一致していなければなりません。IAM ユーザーは過去に人的とマシン ID 両方に対して使用されましたが、長期的アクセスキー使用におけるリスクを回避するため、それを使用しないよう推奨しています。

実装手順

従業員、管理者、開発者、オペレーター、および顧客などの人的 ID の場合:

- 一元化された ID プロバイダーに依存して、人間ユーザーが一時的な認証情報を使って AWS にアクセスするには、ID プロバイダーにフェデレーションを使用することを義務付ける必要があります。ユーザーに対するフェデレーションは、各 AWS アカウント の直接フェデレーションで、または AWS IAM Identity Center (AWS IAM アイデンティティセンター の後継サービス) および好みの ID プロバイダーを使って行うことができます。フェデレーションは、長期的な認証情報を排除するだけでなく、IAM ユーザーを使用する場合と比較して多数の利点があります。ユーザーは 直接フェデレーション用のコマンド行から、または IAM Identity Center を使用して、一時的な認証情報をリクエストすることができます。つまり、IAM ユーザーまたは、ユーザー向けの長期的認証情報を必要なケースはほとんどないということです。
- Software as a Service (SaaS) などのサードパーティーに、AWS アカウント のリソースへのアクセスを付与する際、クロスアカウントロールおよびリソースベースポリシーを使用できます。
- 消費者や顧客向けのアプリケーションに AWS リソースへのアクセスを許可する必要がある場合、Amazon Cognito アイデンティティ プールまたはAmazon Cognito user pools を使用して、一時的な認証情報を提供できます。認証情報のアクセス許可は、IAM ロールによって設定されます。 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 の場合、長期的認証情報を使用しなければならない場合があります。これらの場合、IAM ロールで AWS にアクセスする際に、ワークロードが一時的な認証情報を使用するよう義務付ける必要があります。

- Amazon Elastic Compute Cloud (Amazon EC2) の場合、Amazon EC2 に対して ロールを使用できます。

- AWS Lambda では、一時的な認証情報を使って AWS アクションを実行するためのサービス権限を付与する Lambda 実行ロールを設定できます。AWS サービスが、IAM ロールを使って一時的な認証情報を付与する類似モデルは多数あります。
- IoT デバイスの場合、AWS IoT Core 認証情報プロバイダーを使って、一時的な認証情報をリクエストできます。
- オンプレミスのシステム、または AWS 外で実行され、AWS リソースへアクセスする必要があるシステムの場合、IAM Roles Anywhere を使用できます。

一時的な認証情報が選択肢として使えず、長期的認証情報を使う必要があるシナリオがあります。これらの状況では、が定期的に認証情報を監査してローテーションし、さらに 長期的認証情報が必要なユースケースに対して定期的にアクセスキーをローテーションします。長期的認証情報が必要となるかもしれない例には、WordPress プラグインやサードパーティーの AWS クライアントなどが考えられます。長期的認証情報を使用すべき状況、またはデータベースログインなどの AWS アクセスキー以外の認証情報については、AWS Secrets Manager など、シークレット管理を処理するために設計されたサービスを使用できます。Secrets Manager 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を簡単に管理、ローテーション、安全に保存できます。長期的認証情報のローテーションについては、「アクセスキーのローテーション」を参照してください。

リソース

関連するベストプラクティス:

- 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
- SEC03-BP08 組織内でリソースを安全に共有する

関連するドキュメント:

- 一時的なセキュリティ認証情報

- AWS 認証情報
- IAM セキュリティのベストプラクティス
- IAM ロール
- IAM Identity Center
- ID プロバイダーとフェデレーション
- アクセスキーのローテーション
- 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)
- AWS アカウントのルートユーザー

関連動画:

- Managing user permissions at scale with AWS IAM Identity Center (AWS SSO を使用した大規模なユーザー権限の管理) (AWS IAM アイデンティティセンター の後継サービス)
- Mastering identity at every layer of the cake (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

ワークロードには、データベース、リソース、およびサードパーティーサービスにアイデンティティを証明するための自動機能が必要となります。これは、API アクセスキー、パスワード、および OAuth トークンなどの、シークレットアクセス認証情報を使って実現されます。これらの認証情報を保存、管理、ローテーションする専用のサービスを使用することで、認証情報が侵害される可能性を低減することができます。

期待される成果: 次の目標を達成するアプリケーションの認証情報を安全に管理するメカニズムを実装する:

- ワークロードに必要なシークレットを特定する。
- 長期的認証情報を短期的認証情報と置き換える (可能な場合) ことによりその数を減らす。
- 安全なストレージと、残りの長期的認証情報の自動化されたローテーションを確立する。
- ワークロードに存在するシークレットへのアクセスを監査する。
- 開発プロセス中、ソースコードに組み込まれたシークレットがないことを継続的に監視する。
- 認証情報が誤って開示される可能性を減らす。

一般的なアンチパターン:

- 認証情報をローテーションしない。
- ソースコードまたは設定ファイルに長期的認証情報を保管する。
- 認証情報を暗号化せずに保管する。

このベストプラクティスを活用するメリット:

- シークレットが、保管時と転送時に暗号化される。
- 認証情報へのアクセスが、API (認証情報の自動販売機と考える) 経由でゲート化される。
- 認証情報へのアクセス (読み出しと書き込み) が監査およびログ記録される。
- 懸念事項の分離: 認証情報のローテーションは、アーキテクチャの他の部分から分離できる別のコンポーネントによって実行されます。
- シークレットは、ソフトウェアコンポーネントに対してオンデマンドで配布され、中央ロケーションでローテーションが発生する。
- 認証情報へのアクセスは、非常にきめ細やかに制御できます。

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

従来、データベースやサードパーティーの API、トークンなどの認証に使用する認証情報は、ソースコードや環境ファイルに埋め込まれている場合がありました。AWS は、これらの認証情報を安全に保管し、自動的にローテーションし、その使用を監査するメカニズムを複数提供しています。

シークレット管理に対する最善のアプローチは、削除、置換、ローテーションのガイダンスに従うことです。最も安全な認証情報は、保管、管理、処理が不要なものです。認証情報によっては、ワークロードの機能にとって不要となった、安全に削除できるものもあります。

ワークロードの正常な機能に依然として必要な認証情報については、長期的認証情報を一時的または短期的な認証情報と置換する機会があるかもしれません。たとえば、AWS シークレットアクセスキーをハードコーディングする代わりに、IAM ロールを使って長期的認証情報を一時的認証情報と置換することを検討してみてください。

存続期間の長いシークレットによっては、削除も置換もできないものがあります。これらのシークレットは、AWS Secrets Manager などのサービスに保管して、一元的に保管、管理したり、定期的にローテーションしたりすることができます。

ワークロードのソースコードと設定ファイルの監査を行うと、さまざまなタイプの認証情報が明らかになる可能性があります。次の表は、一般的なタイプの認証情報を取り扱うための戦略をまとめたものです。

Credential type	Description	Suggested strategy
IAM access keys	AWS IAM access and secret keys used to assume IAM roles inside of a workload	Replace: Use IAM ロール assigned to the compute instances (such as Amazon EC2 or AWS Lambda) instead. For interoperability with third parties that require access to resources in your AWS アカウント, ask if they support AWS クロスアカウントアクセス. For mobile apps, consider using temporary credentials through Amazon Cognito ID プール (フェデレーティッドアイデンティティ). For workloads running outside of AWS, consider IAM Roles Anywhere or AWS Systems Manager ハイブリッドアクティベーション.
SSH keys	Secure Shell private keys used to log into Linux EC2 instances, manually or as part of an automated process	Replace: Use AWS Systems Manager or EC2 Instance Connect to provide programmatic and human access to EC2 instances using IAM roles.
Application and database credentials	Passwords – plain text string	Rotate: Store credentials in AWS Secrets Manager and establish automated rotation if possible.

Credential type	Description	Suggested strategy
Amazon RDS and Aurora Admin Database credentials	Passwords – plain text string	Replace: Use the Amazon RDS との Secrets Manager 統合 or Amazon Aurora. In addition, some RDS database types can use IAM roles instead of passwords for some use cases (for more detail, see IAM データベース認証).
OAuth tokens	Secret tokens – plain text string	Rotate: Store tokens in AWS Secrets Manager and configure automated rotation.
API tokens and keys	Secret tokens – plain text string	Rotate: Store in AWS Secrets Manager and establish automated rotation if possible.

一般的なアンチパターンは、ソースコード、設定ファイル、またはモバイルアプリ内に IAM アクセスキーを埋め込むことです。IAM アクセスキーが AWS サービスと通信する必要がある場合、一時的 (短期的) セキュリティ認証情報を使用します。これらの短期的な認証情報は、EC2 インスタンス用の IAM ロール、Lambda 関数の実行ロール、モバイルユーザーアクセスのための Cognito IAM ロール、および IoT デバイス用の IoT Core ポリシーを通して提供できます。サードパーティー向けの場合は、IAM ユーザーをサーバーして、サードパーティーにそのユーザー向けのシークレットアクセスキーを送信するよりも、アカウントのリソースへの必要なアクセス権を持つ IAM ロールにアクセスを委譲する方法を優先します。

ワークロードに、他のサービスやリソースとの相互運用に必要なシークレットの保管が必要となるケースが多数あります。AWS Secrets Manager は、これらの認証情報の安全な管理、さらには API トークン、パスワード、およびその他の認証情報の保管、使用、ローテーション専用です。

AWS Secrets Manager は、機密性の高い認証情報を確実かつ安全に保管して取扱うための主な機能を 5 つ提供しています: 保管時の暗号化、転送中の暗号化、総合的な監査、きめ細やかなアクセスコントロール、および拡張可能な認証情報のローテーション。AWS パートナーによるその他のシークレット管理サービス、または類似の機能や保証を提供するローカルで開発されたソリューションも使用できます。

実装手順

1. Amazon CodeGuru などの自動化ツールを使用して、ハードコード化された認証情報を含むコードパスを特定します。
   - Amazon CodeGuru を使って、コードリポジトリをスキャンします。レビューが完了したら、CodeGuru で `Type=Secrets` をフィルターして、問題のあるコードの行を突き留めます。
2. 削除または置換できる認証情報を特定します。
   a. すでに不要な認証情報を特定して、削除用にマークします。
   b. ソースコードに埋め込まれた AWS シークレットキーについては、必要なリソースに関連付けられた IAM ロールと置換します。ワークロードの一部が AWS 外であるにもかかわらず AWS リソースにアクセスする IAM 認証情報が必要な場合、IAM Roles Anywhere または AWS Systems Manager ハイブリッドアクティベーションを検討してください。
3. ローテーション戦略を使用すべきその他のサードパーティー、存続期間の長いシークレットについては、Secrets Manager をコードに統合して、ランタイムにサードパーティーのシークレットを取得します。
   a. CodeGuru コンソールは、検出された認証情報を使って Secrets Manager を作成できます。
   b. Secrets Manager から取得したシークレットをアプリケーションコードに統合します。
      - サーバーレス Lambda 関数では、言語に依存しない Lambda 拡張子を使用できます。
      - EC2 インスタンスまたはコンテナに対しては、AWS が複数のよく使用されるプログラミング言語で、Secrets Manager からシークレットを取得するためのクライアント側コードの例を提供しています。
4. 定期的にコードベースをレビューして再スキャンすることで、コードに新たなシークレットが追加されていないことを確認します。
   - git-secrets などのツールを使って、ソースコードリポジトリに新しいシークレットがコミットされるのを防止することを検討してください。
5. 予想外の使用、不適切なシークレットへのアクセス、またはシークレットの削除試行がないかどうか、Secrets Manager アクティビティをモニタリングします。
6. 認証情報に対する人的曝露を減少させます。この目的に特化した IAM ロールに対する認証情報を読み出し、書き込み、および変更するためのアクセスを制限し、一部の運用ユーザーにのみ、その役割を担うためのアクセスを提供します。

リソース

関連するベストプラクティス:

- SEC02-BP02 一時的な認証情報を使用する
- 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

関連するドキュメント:

- Getting started with AWS Secrets Manager (AWS シークレットマネージャーの開始方法)
- ID プロバイダーとフェデレーション
- Amazon CodeGuru Introduces Secrets Detector (Amazon CodeGuru がシークレットディテクターを提供)
- How AWS Secrets Manager uses AWS Key Management Service (AWS Secrets Manager が AWS Key Management Service を使用する方法について)
- Secret encryption and decryption in Secrets Manager (Secrets Manager におけるシークレット暗号化と復号化)
- Secrets Manager ブログエントリ
- Amazon RDSと AWS Secrets Manager の統合を発表

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)
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector (Amazon CodeGuru Reviewer Secrets Detector を使ってハードコード化されたシークレットを見つける)
- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager (AWS re:Inforce 2022 - AWS Secrets Manager を使用したハイブリッドワークロードのシークレットの保護)

関連ワークショップ:

- Store, retrieve, and manage sensitive credentials in AWS Secrets Manager (AWS Secrets Manager で機密性の高い認証情報を保存、取得、管理する)
- AWS Systems Manager ハイブリッドアクティベーション

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 マネジメントコンソール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 アイデンティティセンター (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 マネジメントコンソール、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 アイデンティティセンター (AWS IAM アイデンティティセンター 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

認証情報を定期的に監査およびローテーションして、リソースへのアクセスに認証情報を使用できる期間を制限します。長期的認証情報を使用すると多くのリスクが生じ、これらのリスクは長期的認証情報を定期的にローテーションすることにより軽減できます。

期待される成果: 認証情報のローテーションを実装することにより、長期的認証情報の使用に関連するリスクを軽減します。認証情報ローテーションポリシーの不遵守を定期的に監査して、是正します。

一般的なアンチパターン:

- 認証情報の使用を監査しない。
- 必要がないのに、長期的認証情報を使う。
- 長期的認証情報を使用して、定期的にローテーションしない。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

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、多要素認証 (MFA) などの定義された管理方法が実施され、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廃止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 ID に移行すると、認証情報レポートを生成してユーザーを監査できます。

また、ID プロバイダーで MFA を実施およびモニタリングすることをお勧めします。AWS Config ルール を設定するか、またはAWS Security Hub CSPM セキュリティスタンダードを使って、ユーザーの MFA が有効になっているかどうかをモニタリングできます。IAM Roles Anywhere を使って、マシン ID の一時的な認証情報を提供することを検討してください。IAM ロールと一時的な認証情報の使用が不可能なときは、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。

実装手順

- 認証情報を定期的に監査する: ID プロバイダーと IAM で設定されている ID を監査することで、承認された ID のみがワークロードにアクセスできるようになります。こういった ID には、IAM

ユーザー、AWS IAM アイデンティティセンター ユーザー、Active Directory ユーザー、またはさまざまなアップストリーム ID プロバイダーのユーザーを含みますが、これらに限定されません。たとえば、組織を離れた人を削除したり、不要になったクロスアカウントのロールを削除したりします。IAM エンティティがアクセスするサービスへのアクセス許可を定期的に監査するプロセスを用意します。これにより、未使用のアクセス許可を削除するために変更する必要があるポリシーを特定できます。認証情報レポートと AWS Identity and Access Management 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。Amazon CloudWatch を使って、AWS 環境内で呼び出される特定の API コールのアラームを設定します。Amazon GuardDuty は、想定外のアクティビティがあるとアラートを発動します。これは、IAM 認証情報に対する過度に寛容なアクセスまたは意図しないアクセスを示している可能性があります。

- 認証情報を定期的にローテーションする: 一時的な認証情報を使用できない場合、長期的 IAM アクセスキーを定期的にローテーションしてください (最大 90 日ごと)。知らない間にアクセスキーが開始された場合でも、これによりその認証情報を使ってリソースにアクセスされる期間を制限できます。IAM ユーザーのアクセスキーのローテーションについては、「アクセスキーのローテーション」を参照してください。
- IAM アクセス許可を確認する: AWS アカウント のセキュリティを改善するには、各 IAM ポリシーを定期的に確認してモニタリングします。ポリシーが最小特権の原則に準拠していることを確認します。
- IAM リソース作成および更新の自動化を検討する: IAM Identity Center は、ロールやポリシー管理など多くの IAM タスクを自動化します。または、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、人為的ミスが生じる可能性を減らすことができます。
- IAM Roles Anywhere を使用して、マシン ID の IAM ユーザーを置換する: IAM Roles Anywhere を使用すると、オンプレミスサーバーなど、従来は不可能であった領域でロールを使用できるようになります。IAM Roles Anywhere は、信頼された X.509 証明書を使って AWS を認証し、一時的な認証情報を受け取ります。IAM Roles Anywhere を使用することにより、長期的認証情報がオンプレミス環境に保管されなくなるため、これらの認証情報をローテーションする必要がなくなります。X.509 証明書の有効期限が近づいたら、モニタリングとローテーションが必要となることに注意してください。

リソース

関連するベストプラクティス:

- 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

関連するドキュメント:

- Getting started with AWS Secrets Manager (Amazon SQS の開始方法)
- IAM 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
- AWS アカウント アカウントの認証情報レポートの取得

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 アイデンティティセンター (AWS SSO を使用した大規模なユーザー権限の管理)
- Mastering identity at every layer of the cake (すべての層での ID の把握)

関連する例:

- Well-Architected ラボ - IAM ユーザーの自動クリーンアップ
- Well-Architected ラボ - IAM グループおよびロールの自動デプロイ

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 アイデンティティセンター (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 アイデンティティセンター (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 アイデンティティセンター (AWS IAM アイデンティティセンター 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 組織内でリソースを安全に共有する
- SEC03-BP09 サードパーティーと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 アイデンティティセンター
- 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リソースに対する特定のアクションを実行するために ID が必要とするアクセス許可のみを付与するのがベストプラクティスです。グループと ID 属性を使用して、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。例えば、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、開発者がプロジェクトから離れると、基盤となるアクセスポリシーに変更を加えることなく、その開発者のアクセスは自動的に取り消されます。

期待される成果: ユーザーは、ジョブの実行に必要なアクセス許可のみを持つ必要があります。ユーザーには、限られた時間内に特定のタスクを実行するためだけに本番環境へのアクセスが与えられ、タスクが完了したらアクセスを取り消す必要があります。アクセス許可は、ユーザーが別のプロジェクトまたは職務に移った場合を含め、不要になったときに取り消す必要があります。管理者権限は、信頼できる管理者の少数のグループのみに付与する必要があります。アクセス許可の変化を避けるため、アクセス許可は定期的にレビューする必要があります。マシンまたはシステムアカウントには、タスクを完了するために必要な最小セットのアクセス許可を付与する必要があります。

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
- ルートユーザーを日常業務に使用する
- 過度に寛容でありながら、完全な管理者権限がないポリシーを作成する。
- アクセス許可をレビューして、最小特権アクセスを許可するかどうかを把握しない。

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: 高

実装のガイダンス

最小特権の原則には、特定のタスクの遂行に必要な最小セットのアクションを実行する許可のみを ID に付与する必要があると記載されています。これは、ユーザビリティ、効率性、セキュリティのバランスを取ります。この原則の下で運用すると、意図しないアクセスを制限し、誰がどのリソースにアクセスできるかを追跡するのに役立ちます。デフォルトでは、IAM ユーザーとロールにはアクセス許可はありません。ルートユーザーにはデフォルトでフルアクセスがあり、厳格に制御、監視する必要があり、ルートアクセスが必要なタスクにのみ使用する必要があります。

IAM ポリシーは、IAM ロールまたは特定のリソースに明示的にアクセス許可を付与するために使用されます。例えば、アイデンティティベースのポリシーは、IAM グループにアタッチでき、S3 バケットはリソースベースのポリシーで制御できます。

IAM ポリシーの作成では、AWS がアクセスを許可または拒否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、依頼者が AWS 組織に属していない場合、`PrincipalOrgID` 条件キーを使用して、アクションを拒否できます。

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。異なるポリシータイプを層にして、深層防御を確立し、ユーザーの全体的なアクセス許可を制限する必要があります。どのアクセス許可がどのような条件の下で付与できるかも制限できます。例えば、アプリケーションチームが独自の IAM ポリシーを作成することは許可できますが、アクセス許可の境界 を適用して、チームが受け取る最大のアクセス許可を制限する必要もあります。

実装手順

- 最小特権ポリシーを実装する: IAM グループおよびロールに最小特権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。

  - API 使用状況に関するベースポリシー: 必要なアクセス許可を判断する 1 つの方法は、AWS CloudTrail ログをレビューすることです。このレビューでは、ユーザーが AWS 内で実際に実行するアクションに合わせてカスタマイズされたアクセス許可を作成できます。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。IAM Access Advisor を組織またはアカウントレベルで使用して、特定のポリシーの最終アクセスの情報を追跡できます。
- 職務に応じた AWS マネージドポリシーの使用を検討してください。 きめ細かいアクセス許可ポリシーの作成を開始するとき、どこから始めればよいかわからない場合があります。AWS には、例えば請求、データベース管理者、データサイエンティストなど、一般的な職種に対するマネージドポリシーがあります。これらのポリシーは、最小特権ポリシーの実装方法を判断している間、ユーザーの持つアクセスを絞り込むことができます。
- 必要でないアクセス許可を削除する: 必要でないアクセス許可を削除し、過度に寛容なポリシーを削ります。IAM Access Analyzer ポリシー生成は、アクセス許可ポリシーの微調整に役立ちます。
- ユーザーの本番環境へのアクセスが制限されるようにする: ユーザーは、有効なユースケースのある本番環境にのみアクセスできるようにする必要があります。ユーザーが、本番稼働アクセスが必要な特定のタスクを実行した後は、アクセスを取り消す必要があります。本番環境へのアクセスを制限することは、本番に影響する意図しないイベントを回避するのに役立ち、意図しないアクセスの影響範囲を狭めます。
- 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。
- アクセス許可のリソースタグを検討する: リソースタグを使用する属性ベースのアクセスコントロールモデルでは、リソースの目的、所有者、環境、またはその他の基準に基づいてアクセスを付与できます。例えば、リソースタグを使用して、開発と本番環境を区別することができます。これらのタグを使用して、開発者を開発環境に制限することができます。タグ付けとアクセス許可ポリシーを組み合わせることで、きめ細かいリソースアクセスを達成でき、すべての職務に複雑な、カスタムポリシーを定義する必要がなくなります。
- AWS Organizations のサービスコントロールポリシーを使用します。 サービスコントロールポリシーは、組織のメンバーアカウントで利用できる最大のアクセス許可を一元管理します。重要なのは、サービスコントロールポリシーでは、メンバーアカウントでルートユーザーのアクセス許可を制限できることです。AWS Organizations を強化する規範的マネージドコントロールを提供する、AWS Control Tower の使用も検討してください。Control Tower 内で独自のコントロールを定義できます。

- 組織のユーザーライフサイクルポリシーを確立する: ユーザーライフサイクルポリシーは、ユーザーが AWS にオンボードされたとき、ジョブロールまたはスコープを変更したとき、または AWS へのアクセスが不要になったときに実行するタスクを定義します。アクセス許可レビューは、ユーザーのライフサイクルの各ステップで実行し、アクセス許可が適切に制限されていることを検証して、アクセス許可の変化を回避します。
- アクセス許可をレビューする定期的スケジュールを確立し、不要なアクセス許可を削除する: ユーザーアクセスを定期的にレビューして、過度に寛容なアクセスがないことを検証する必要があります。AWS Config および IAM Access Analyzer は、ユーザーアクセス許可を監査するときに役立ちます。
- 職種マトリックスを確立する: 職種マトリックスは、AWS フットプリント内で必要なさまざまなロールとアクセスレベルを可視化します。職種マトリックスを使用して、組織内でのユーザーの責任に基づいてアクセス許可を定義し、分離できます。個々のユーザーまたはロールにアクセス許可を直接適用する代わりに、グループを使用します。

リソース

関連するドキュメント:

- 最小特権を付与する
- 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)
- IAM アクセス許可の境界を使用して開発者にアクセス許可管理を委任する
- 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 ポリシーシミュレーターを使用した 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)
- Attribute-based access control (ABAC)

- 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
- タグ付けを使用して環境を整理しアカウンタビリティを促進
- AWS のタグ付け戦略
- AWS リソースのタグ付け

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 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?)

関連する例:

- ラボ: ロールの作成を委任する IAM アクセス許可の境界
- ラボ: EC2 の IAM タグベースのアクセスコントロール

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、最小特権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。人間とマシンアクセス両方について使用しないアイデンティティとアクセス許可を継続的にモニタリングして削除します。

期待される成果: アクセス許可ポリシーは、最小特権原則に準拠する必要があります。職務やロールの定義がはっきりしてくるにつれ、アクセス許可ポリシーを見直し、必要でないアクセス許可を削除する必要があります。このアプローチにより、不注意による認証情報漏洩や不正アクセスによる影響を軽減することができます。

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
- 過度に寛容でありながら、完全な管理者権限がないポリシーを作成する。
- 不要になった後もアクセス許可ポリシーを保持する。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

チームやプロジェクトが始まったばかりの場合、革新とアジリティを刺激するために、寛容な許可ポリシーが使われる可能性があります。たとえば、開発またはテスト環境であれば、開発者にはさまざまな AWS サービスへのアクセスを付与できます。継続的にアクセスを評価し、アクセスを、現在のジョブを完了するために必要なサービスおよびサービスアクションのみに制限することが推奨されます。この評価は、人的およびマシン ID 両方にお薦めします。マシン ID は、システムまたはサービスアカウントと呼ばれることもありますが、AWS にアプリケーションまたはサービスへのアクセスを付与するアイデンティティです。このアクセスは、本稼働環境で特に重要です。ここでは、過剰に寛容なアクセス許可を使うと影響が大きく、顧客データを開示してしまう可能性があるためです。

AWS は、使用されていないユーザー、ロール、アクセス許可、および認証情報を特定するための方法を複数提供しています。AWS は、Amazon S3 バケットのオブジェクトなど AWS リソースへの関連付けられたアクセスキー、およびアクセスを含む、IAM ユーザーとロールのアクセス活動を分析するのにも役立ちます。AWS Identity and Access Management Access Analyzer ポリシー生成により、プリンシパルが実際にやりとりするサービスやアクションに基づいて、限定的な許可ポリシーを作成することができます。Attribute-based access control (ABAC) は、各ユーザーに直接権限ポリシーをアタッチするのではなく、ユーザーの属性を利用してアクセス許可を与えることができるため、アクセス権限管理の簡素化に役立ちます。

実装手順

- AWS Identity and Access Management Access Analyzerを使用する: IAM Access Analyzer は、組織内のリソースや、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しているアカウントを特定するのに役立ちます。
- IAM Access Analyzer ポリシー生成を使用する: IAM Access Analyzer ポリシー生成は、IAM ユーザーまたはロールのアクセスアクティビティに基づいて、きめ細やかなアクセス許可ポリシーを作成するのに役立ちます。
- IAM ユーザーとロールに対して許容可能な期間と使用ポリシーを決定する: 最終アクセスタイムスタンプを使って、使用されていないユーザーとロールを特定し、それを削除します。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特定してスコープを決定できます。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 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。最終アクセス時間情報は、AWS マネジメントコンソール 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。

- AWS CloudTrail にデータイベントを口グ記録することを検討する: デフォルトで、CloudTrail は Amazon S3 オブジェクトレベルアクティビティ (たとえば、`GetObject` および `DeleteObject`) または Amazon DynamoDB テーブルアクティビティ (たとえば、`PutItem` および `DeleteItem`) などのデータイベントを口グ記録しません。これらのイベントの口グ記録を有効にして、特定の Amazon S3 オブジェクトまたは DynamoDB テーブルアイテムにアクティビティする必要があるユーザーとロールを決定します。

リソース

関連するドキュメント:

- 最小特権を付与する
- 必要でない認証情報を削除する
- AWS CloudTrailとは?
- IAM ポリシーを管理する
- DynamoDB の口グ記録とモニタリング
- Amazon S3 バケットとオブジェクトの CloudTrail イベント口ギングの有効化
- AWS アカウント アカウントの認証情報レポートの取得

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)
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive (ディープダイブ)

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、このアクセスを必要とする特定のリソースのみへのアクセスに限定します。

期待される成果: AWS リソースのうちどれが、誰と共有されるのかを把握します。共有されたリソースを継続的にモニタリングおよび監査し、認証されたプリンシパルとのみ共有されていることを確認します。

一般的なアンチパターン:

- 共有されたリソースのインベントリを保持しない。
- リソースへのクロスアカウントまたはパブリックアクセスの承認の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

アカウントが AWS Organizations にある場合、リソースへのアクセスを、組織全体、特定の組織単位、または個別のアカウントに付与することができます。アカウントが組織のメンバーでない場合、個別のアカウントとリソースを共有することができます。直接クロスアカウントアクセスを付与するには、Amazon Simple Storage Service (Amazon S3) バケットポリシーなどのリソースベースのポリシーを使用するか、別のアカウントのプリンシパルがアカウントの 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している場合、アクセスが認証済みのプリンシパルにのみ付与されていることを確認してください。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。

AWS Identity and Access Management 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単純化します。IAM Access Analyzer を AWS Organizations で設定して、すべてのアカウントが表示されることを確認します。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、検出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を設計する際、信頼ポリシーを使用して、ロールを引き受けるケースを制御できます。たとえば、`PrincipalOrgId` 条件キーを使用して、AWS Organizations 外からのロールを引き受けようとするのを拒否できます。

AWS Config は、設定が誤っているリソースを報告し、AWS Config ポリシーチェックを通して、パブリックアクセスが設定されたリソースを検出できます。AWS Control TowerやAWS Security Hub CSPM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、公開されたリソースを特定および修復します。たとえば、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 スナップショットのうち AWS アカウント によって復元できるものがあるかどうかが検出されます。

実装手順

- AWS Organizations に対して AWS Config を有効化することを検討する: AWS Config では、AWS Organizations 内の複数アカウントからの検出結果を、委任された管理者アカウントに集計することができます。これにより、全体像が把握でき、アカウント全体に AWS Config ルール をデプロイして、パブリックにアクセス可能なリソースを特定できます。
- AWS Identity and Access Management Access AnalyzerIAM Access Analyzer を設定すると、外部エンティティと共有している、組織やアカウント内のリソース (Amazon S3 バケットや IAM ロールなど) を特定するのに役立ちます。

- AWS Config で自動修復を使用し、Amazon S3 バケットのパブリックアクセス設定の変更に対応します: Amazon S3 バケットに対して、パブリックアクセスのブロック設定を自動的に再有効化することができます。
- モニタリングを実装して、Amazon S3 バケットがパブリックになった場合はアラートを発動する: Amazon S3 パグリックアクセスブロックが無効な場合と、Amazon S3 バケットがパブリックになったかどうかを特定するために、モニタリングとアラートを設定しておく必要があります。さらに、AWS Organizations を使用している場合、Amazon S3 パブリックアクセスポリシーへの変更を防ぐサービスコントロールポリシーを作成する必要があります。AWS Trusted Advisor は、オープンアクセス権限がある Amazon S3 バケットをチェックします。誰にでもアクセスを付与、アップロード、削除するバケット権限は、バケットのアイテムを誰でも追加、変更、または削除できるようにすることで、セキュリティ関連の問題の原因となることがあります。Trusted Advisor のチェックは、バケットの明示的なアクセス許可を検証します。また、バケットに関連付けられたポリシーで、バケットのアクセス許可を上書きする可能性があるものについても検証します。また、AWS Config を使って、Amazon S3 バケットにパブリックアクセスがないかモニタリングできます。詳細については、「AWS Config を使って、パブリックアクセスを許可する Amazon S3 バケットをモニタリングおよび対応する」を参照してください。アクセスをレビューする間、どのようなタイプのデータが Amazon S3 バケットに含まれているかを考慮することが重要です。Amazon Macie は、PII、PHI などの機密性の高いデータ、およびプライベートまたは AWS キーなどの認証情報を検出して保護します。

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 コントロールライブラリ
- 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
- Amazon EventBridge を使って AWS Trusted Advisor チェック結果をモニタリングする
- 組織内のすべてのアカウントで AWS Config ルールを管理する
- AWS Config および AWS Organizations

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 組織内でリソースを安全に共有する

ワークロードの数が増えるにつれて、それらのワークロードのリソースへのアクセスを共有したり、複数のアカウントでリソースを複数回プロビジョニングしたりする必要が生じます。開発環境、テスト環境、本番環境などの環境を区分けするための構造があるかもしれません。ただし、分離構造があっても、安全に共有する能力は制限できません。重複するコンポーネントを共有することにより、運用諸経費を削減し、同一リソースを複数回作成する間に見逃したものを推測しなくても、一貫したエクスペリエンスを実現できます。

期待される成果: 安全な方法を使用して組織内のリソースを共有することにより、意図しないアクセスを最小限に抑え、データ損失防止イニシアチブに役立てます。個々のコンポーネントを管理するのと比較して、運用諸経費を削減し、同じコンポーネントを何度も手動で作成することによるエラーを減らし、ワークロードのスケーラビリティを向上させることができます。削減できた時間を活用して、マルチポイント障害シナリオを解決し、自信を持ってコンポーネントが不要になる時を判断できるようになります。外部共有リソースの分析に関する規範的ガイダンスについては、「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」を参照してください。

一般的なアンチパターン:

- 継続的にモニタリングして、予定外の外部共有が生じたときに自動的にアラートを発動するプロセスがない。
- 共有すべき/すべきでない内容に関する基準がない。
- 必要な時点で明示的に共有するのではなく、広く開かれたポリシーをデフォルトとしている。
- 必要に応じて重複する基本的リソースを手動で作成する。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

アクセスコントロールとパターンを構築し、信頼できるエンティティとのみ共有リソースの消費を安全に管理します。共有リソースをモニタリングして、継続的に共有リソースアクセスをレビューし、不適切なまたは予想外の共有があればアラートを発動します。パブリックおよびクロスアカウントアクセスを分析するをレビューして、外部からのアクセスを必要なリソースのみに限定し、継続的にモ

ニタリングし、自動的にアラートを出すプロセスを確立するためのガバナンスを確立するのに役立ちます。

AWS Organizations 内のクロスアカウント共有は、多数の AWS サービス (AWS Security Hub CSPM、Amazon GuardDuty、および AWS Backup) によってサポートされています。これらのサービスを使用すると、中央アカウントでデータを共有し、中央アカウントからアクセス可能、あるいは中央アカウントからリソースとデータを管理できます。例えば、AWS Security Hub CSPM は個別アカウントから中央アカウントに検出結果を送信するため、すべての検出結果を確認することができます。AWS Backup は、リソースのバックアップを取り、アカウント全体で共有します。AWS Resource Access Manager (AWS RAM) を使用して、VPC サブネットおよび Transit Gateway 添付ファイル、AWS Network Firewall、または Amazon SageMaker AI パイプラインなど他の一般的なリソースを共有できます。

リソースを組織内のリソースのみと共有するよう制限するには、サービスコントロールポリシー (SCP) を使って、外部プリンシパルへのアクセスを防止します。リソースを共有する際、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と、意図しないアクセスから保護するのに役立ちます。データ境界とは、信頼できるアイデンティティのみが、期待されるネットワークから信頼できるリソースにアクセスするよう徹底するのに役立つ予防的な一連のガードレールです。これらのコントロールは、どのリソースが共有可能かについて適切な制限を設け、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` 条件を使用することで、Amazon S3 バケットにアクセスするアイデンティティが組織に確実に属するようにできます。SCP は、サービスにリンクされたロール (LSR) または AWS サービスプリンシパルに適用されないことに注意してください。

Amazon S3 を使用する際、 は Amazon S3 バケットに対して ACL を無効化し、IAM ポリシーをし応してアクセスコントロールを定義します。Amazon CloudFront から Amazon S3 オリジンに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。これは、AWS Key Management Service のサーバー側暗号化を含む追加機能をサポートします。

場合によっては、組織外のリソースを共有したり、リソースにサードパーティーのアクセスを付与したりするかもしれません。リソースを外部で共有する権限管理に関する規定的なガイダンスについては、「Permissions management」を参照してください。

実装手順

1. AWS Organizations を使用します。

AWS Organizations は、組織を作成して一元管理するときに、複数の AWS アカウント を統合できるアカウント管理サービスです。アカウントを組織単位 (OU) にグループ化し、OU ごとに異なるポリシーをアタッチすることにより、予算、セキュリティ、コンプライアンスのニーズに対応できます。また、AWS 人工知能 (AI) と機械学習 (ML) サービスがどのようにデータを収集して保管するかをコントロールし、Organizations と統合された AWS サービスのマルチアカウント管理を使用できます。

2. AWS Organizations を AWS サービスと統合します。

   組織のメンバーアカウントを代理してタスクを実行する AWS サービスを有効にすると、AWS Organizations が各メンバーアカウントでそのサービスに対して IAM サービスがリンクされたロールを作成します。AWS マネジメントコンソール、AWS API、または AWS CLI を使用して、信頼できるアクセスを管理する必要があります。信頼されたアクセスを可能にするための規範的ガイダンスについては、「AWS Organizations を他の AWS サービスおよび Organizations と併用できる AWS サービスの使用」を参照してください。

3. データ境界を確立します。

   AWS 境界は、AWS Organizations によって管理される組織として表現されるのが普通です。オンプレミスネットワークとシステムとともに、AWS リソースへのアクセスは、My AWS の境界としてみなしているものです。この境界の目標は、アイデンティティが信頼され、リソースが信頼され、そしてネットワークが予想されている場合にそのアクセスが許可されていることを検証することにあります。

   a. 境界を定義および実装します。

      各認証条件について、AWS ホワイトペーパーの「境界の構築」の境界の実装に記載されたステップに従います。ネットワーク層に関する規範的ガイダンスについては、「ネットワークの保護」を参照してください。

   b. 継続的にモニタリングとアラートを行います。

      AWS Identity and Access Management Access Analyzer は、組織内のリソースや、外部エンティティと共有しているアカウントを特定するのに役立ちます。IAM Access Analyzer を AWS Security Hub CSPM と統合して、IAM Access Analyzer から Security Hub CSPM へリソースの検出結果を送信および集計し、環境のセキュリティ体制を分析するのに役立てます。統合を有効にするには、各アカウントの各リージョンでIAM Access Analyzer と Security Hub CSPM の両方を有効にします。また、AWS Config ルール を使用して、設定を監査し、Amazon Q Developer in chat applications と AWS Security Hub CSPM を併用する適切な当事者にアラー

トを出すことができます。次に、AWS Systems Manager の自動化ドキュメントを使用して、非準拠のリソースを修復できます。

c. 外部で共有するリソースを継続的にモニタリングおよびアラート通知するための規範的ガイダンスについては、「パブリックおよびクロスアカウントアクセスを分析する」を参照してください。

4. AWS サービスのリソース共有を使用し、それに従って制限します。

   多くの AWS サービスを使用すると、別のアカウントとリソースを共有したり、別アカウントにある Amazon マシンイメージ (AMI) および AWS Resource Access Manager (AWS RAM) などのリソースをターゲットとすることができます。`ModifyImageAttribute` API を制限して、AMI を共有する信頼できるアカウントを指定します。AWS RAM を使用している場合、`ram:RequestedAllowsExternalPrincipals` 条件を指定し、共有を自組織にのみ限定して、信頼されていないアイデンティティからのアクセスを防止します。規範的ガイダンスおよび注意事項については、「リソース共有および外部ターゲット」を参照してください。

5. AWS RAM を使用して、自分のアカウント内または他の AWS アカウント と安全に共有します。

   AWS RAM は、自分のアカウントおよび他の AWS アカウント アカウントのロールやユーザーで作成したリソースを安全に共有するのに役立ちます。マルチアカウント環境の場合、AWS RAM ではリソースを作成したら、それを他のアカウントと共有できます。このアプローチにより、運用諸経費を削減し、Amazon CloudWatch および AWS CloudTrail との統合を通じて、一貫性、可視性、監査可能性を提供することができます。これは、クロスアカウントアクセスを使用している場合は享受できません。

   リソースベースポリシーを使って過去に共有したリソースがある場合、`PromoteResourceShareCreatedFromPolicy` API または同等のコマンドを使って、リソース共有を完全な AWS RAM リソース共有に昇格させることができます。

   場合によっては、リソースを共有するための追加ステップが必要かもしれません。たとえば、暗号化されたスナップショットを共有するには、AWS KMS キーを共有する必要があります。

リソース

関連するベストプラクティス:

- 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-BP09 サードパーティーとリソースを安全に共有する
- SEC05-BP01 ネットワークレイヤーを作成する

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
- AWS Organizations と使用できる AWS サービス
- AWS でデータ境界を確立する: 信頼できるアイデンティティのみが会社データにアクセスできるようにします

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)

関連ツール:

- Data Perimeter Policy Examples (データ境界ポリシーの例)

SEC03-BP09 サードパーティーとリソースを安全に共有する

クラウド環境のセキュリティは、組織内にとどまりません。組織が、データの一部を管理するのにサードパーティーに依存することもあります。サードパーティー管理システムの権限管理は、一時的な認証情報を使用する最小特権の原則を用いたジャストインタイムアクセスの実践に従う必要があります。サードパーティーと密に連携することにより、意図しないアクセスの影響が及ぶ範囲とリスクをともに縮小することができます。

期待される成果: ユーザーと関連付けられた長期的AWS Identity and Access Management (IAM) 認証情報、IAM アクセスキー、およびシークレットキーは、認証情報が有効かつアクティブである限り、誰でも使用できます。IAM ロールと一時的な認証情報を使うと、そういった機密性の高い詳細の管理と運用間接費など、長期的認証情報を維持するための業務を減らすことにより、総合的なセキュリティスタンスが改善されます。IAM 信頼ポリシーの外部 ID に対して汎用一意識別子 (UUID) を使用し、IAM ロールにアタッチされた IAM ポリシーを制御かに置くことにより、サード

パーティーに付与されたアクセスを監査して、過度に寛容でないことを確認できます。外部共有リソースの分析に関する規範的ガイダンスについては、「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」を参照してください。

一般的なアンチパターン:

- 条件なしでデフォルトの IAM 信頼ポリシーを使用する。
- 長期的 IAM 認証情報とアクセスキーを使う。
- 外部 ID を再使用する。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

AWS Organizations 外のリソースを共有したり、アカウントにサードパーティーのアクセスを付与したりする場合があります。たとえば、サードパーティ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、サードパーティー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します。さらに、外部 ID 条件を使って信頼ポリシーを定義します。外部 ID を使用すると、自分またはサードパーティーが各顧客、サードパーティー、またはテナンシーに対して一意の ID を生成できます。一意の ID を作成後は、自分以外の人物によってコントロールできなくなります。サードパーティーは、外部 ID を安全に、監査可能かつ再現可能な方法で顧客に関連付けるプロセスを実装する必要があります。

また、IAM Roles Anywhere を使用すると、AWS を用いる AWS 外のアプリケーションに対して IAM ロールを管理できます。

サードパーティーが貴社の環境にアクセスする必要がなくなった場合は、ロールを削除します。サードパーティーに長期的な認証情報を提供することは避けてください。共有をサポートする他の AWS サービスを継続的に把握しておきます。たとえば、AWS Well-Architected Tool では ワークロードを他の AWS アカウント と共有でき、AWS Resource Access Manager は所有する AWS リソースを安全に他のアカウントと共有するのに役立ちます。

実装手順

1. クロスアカウントロールを使って、外部アカウントへのアクセスを提供します。

   クロスアカウントロールを使うと、顧客にサービスを提供するために外部アカウントとサードパーティーによって保存されている機密情報の量が減ります。クロスアカウントロールがあると、アクセスを管理および監査する能力を維持しながら、AWS Partner または組織内の他のアカ

ウントなど、アカウントの AWS リソースへのアクセスをサードパーティーに安全に付与できます。

サードパーティーが、ハイブリッドインフラストラクチャからサービスを提供したり、またはオフサイトロケーションにデータをプルする場合があります。IAM Roles Anywhere を使用すると、サードパーティーのワークロードが AWS ワークロードと安全に対話し、長期的認証情報の二ーズをさらに軽減することができます。

外部アカウントアクセスを提供するために、ユーザーと関連付けられた長期的認証情報、またはアクセスキーを使用しないでください。かわりに、クロスアカウントロールを使ってクロスアカウントアクセスを提供します。

2. サードパーティーに外部 ID を使用します。

外部 ID を使用することにより、IAM 信頼ポリシーで誰がロールを担うかを指定できます。信頼ポリシーでは、ロールを引き受けるユーザーが、操作を行う条件とターゲットを実施する必要があります。またこの方法により、アカウントの所有者が特定の状況下でのみロールを引き受けることを許可する方法も提供できます。外部 ID の主な機能は、代理人の混乱問題に対処して防止することにあります。

外部 ID は、AWS アカウント 所有者であり、自分のものに加えて他の AWS アカウント にアクセスするサードパーティーに対してロールを設定した場合、または異なる顧客に代わってロールを引き受けるポジションにある場合に使用します。サードパーティーまたは AWS Partner と連携して、IAM 信頼ポリシーに含める外部 ID 条件を確立します。

3. 汎用一意識別子を使用します。

汎用一意識別子 (UUID) など、外部 ID に対してランダムな一意の値を生成するプロセスを実装します。サードパーティーが異なる顧客間で外部 ID を再使用しても、「代理人の混乱」問題に対処できません。これは、顧客 A が、顧客 B のロール ARN と重複した外部 ID を使用することで、顧客 B のデータを表示できる可能性があるためです。マルチテナント環境では、サードパーティーが異なる AWS アカウント で複数の顧客をサポートするため、サードパーティーが各 AWS アカウント に対する外部 ID として異なる一意の ID を使用する必要があります。サードパーティーは、重複した外部 ID を検出して、各顧客をそれぞれの外部 ID に安全にマッピングする責任があります。サードパーティーは、外部 ID を指定する際にのみロールを引き受けることができることをテストして検証する必要があります。サードパーティーは、外部 ID が必要となるまで、顧客ロール ARN と外部 ID を保存することを控える必要があります。

外部 ID はシークレットとして取り扱われませんが、外部 ID は電話番号、氏名、アカウント ID など推測しやすい値であってはなりません。外部 ID を読み取り専用にすることで、設定のなりすましを目的として外部 ID が変更されないようにします。

ご自身またはサードパーティーが外部 ID を生成できます。ID 生成に責任がある担当者を決定するプロセスを定義します。外部 ID を作成するエンティティにかかわらず、サードパーティーは顧客間で一貫した一意性とフォーマットを適用します。

4. 顧客が提供する長期的認証情報を廃止します。

   長期的認証情報の使用を廃止して、クロスアカウントロールまたは IAM Roles Anywhere を使用します。長期的認証情報を使用する必要がある場合、ロールベースのアクセスに移行する計画を立ててください。キーの管理に関する詳細については、「ID 管理」を参照してください。また、AWS アカウント チームおよびサードパーティーと連携して、リスク軽減ランブックを確立します。セキュリティインシデントの潜在的影響に対する反応と軽減の規範的ガイダンスについては、「インシデント対応」を参照してください。

5. セットアップに規範的ガイダンスがあるか、または自動化されていることを確認します。

   アカウントのクロスアカウントアクセス向けに作成されたポリシーは、least-privilege principle に従う必要があります。サードパーティーは、ロールポリシードキュメント、または AWS CloudFormation テンプレートまたは同等のものを使用する自動化されたセットアップメカニズムを提供する必要があります。これにより、手動ポリシー作成に関連してエラーが発生する可能性が減り、監査可能証跡を提供します。CloudFormation テンプレートを使用したクロスアカウントロール作成の詳細については、「クロスアカウントロール」を参照してください。

   サードパーティーは、自動化され、監査可能なセットアップメカニズムを提供する必要があります。ただし、必要なアクセスを概説したロールポリシードキュメントを使用することにより、ロールのセットアップを自動化する必要があります。CloudFormation テンプレートまたは同等のものを使用して、監査業務の一環として、ドリフト検出で変化をモニタリングする必要があります。

6. 変更するアカウント。

   アカウント構成、サードパーティーのニーズ、または提供されるサービスオファリングは変わる可能性があります。変更と障害を予想し、それに応じて適切な人材、プロセス、テクノロジーを計画する必要があります。定期的に提供するアクセスのレベルを監査し、予想外の変更があった場合にアラートを出す検出方法を実装します。外部 ID のロールとデータストアをモニタリングおよび監査します。予想外の変更やアクセスパターンの結果、サードパーティーのアクセスを一時

的または恒久的に取り消す準備をしておく必要があります。また、実行にかかる時間、関与する人材、コスト、および他のリソースの影響など、取り消し操作の影響を測定します。

検出方法に関する規範的ガイダンスについては、「検出のベストプラクティス」を参照してください。

リソース

関連するベストプラクティス:

- SEC02-BP02 一時的な認証情報を使用する
- 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
- SEC04 の検出

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)
- AWS アカウント 間の IAM ロールを使用したアクセスの委任
- IAM を使用して別の AWS アカウントのリソースにアクセスするにはどうすればよいですか?
- IAM でのセキュリティのベストプラクティス
- クロスアカウントポリシーの評価論理
- 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
- Collecting Information from AWS CloudFormation Resources Created in External Accounts with Custom Resources (カスタムリソースで外部アカウントに作成された AWS CloudFormation のリソースから情報を収集する)
- Securely Using External ID for Accessing AWS Accounts Owned by Others (他人が所有する AWS アカウントへのアクセスに外部 ID を安全に使用する)
- Extend IAM roles to workloads outside of IAM with IAM Roles Anywhere (IAM Roles Anywhere で AWS IAM ロールを AWS 外のワークロードに拡張する)

関連動画:

- How do I allow users or roles in a separate AWS アカウント access to my AWS アカウント? (別の AWS アカウントのユーザーやロールに、私の AWS アカウントへのアクセスを許可するにはどうすればよいですか?)
- AWSre:Invent 2018: 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)
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions (ベストプラクティスと設計の決定)

関連する例:

- Well-Architected Lab - Lambda cross account IAM role assumption (Lambda クロスアカウント IAM ロール引き受け) (レベル 300)
- Configure cross-account access to Amazon DynamoDB (Amazon DynamoDB へのクロスアカウントアクセスを設定する)
- AWS STS Network Query Tool (AWS STS ネットワーククエリツール)

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

サービスとアプリケーションからセキュリティイベントログを保持します。これは、監査、調査、運用のユースケースにおけるセキュリティの基本原則であり、ガバナンス、リスク、コンプライアンス (GRC) の標準、ポリシー、手順によって推進される共通のセキュリティ要件です。

期待される成果: 組織は、セキュリティインシデント対応など、内部プロセスまたは義務を遂行する必要がある場合、AWS サービスおよびアプリケーションからのセキュリティイベントログを、確実かつ一貫して迅速に取得できるようにする必要があります。運用側の成果を改善するためにログの一元化を検討してください。

一般的なアンチパターン:

- ログが永久に保存される、またはすぐに削除される。
- 誰でもログにアクセスできる。
- ログガバナンスと使用について、手動プロセスのみに依存する。
- 必要な場合に備えて、あらゆるタイプのログを保存する。
- 必要な場合にのみログ整合性をチェックする。

このベストプラクティスを活用するメリット: セキュリティインシデントの根本原因分析 (RCA) メカニズムを導入し、ガバナンス、リスク、コンプライアンス義務のための証拠資料とします。

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

セキュリティ調査または要件に基づいた他のユースケース中、インシデントの全容とタイムラインを記録して理解するために関連ログをレビューできる必要があります。ログはまた、関心のある特定のアクションが発生したことを示すアラート生成にも必須です。クエリと取得メカニズムとアラートを選択、有効化、保存、設定することが非常に重要となります。

実装手順

- ログのソースを選択して有効にします。 セキュリティ調査の前に、関連するログを取得し、過去にさかのぼって AWS アカウント でアクティビティを再構築する必要があります。ワークロードに関連するログソースを選択して、有効化します。

  ログソース選択条件は、ビジネスで必要なユースケースに基づいたものである必要があります。AWS CloudTrail または AWS Organizations 証跡を使って各 AWS アカウント に証跡を確立し、そのための Amazon S3 バケットを設定します。

AWS CloudTrail は、AWS のサービスアクティビティをキャプチャする AWS アカウント に対して API コールをトラッキングするログサービスです。これは、デフォルトで有効になっており、管理イベントは 90 日間保持され、AWS マネジメントコンソール、AWS CLI、または AWS を使用して CloudTrail イベント履歴から検索することが可能です。データイベントをより長く保持および確認するには、CloudTrail 証跡を作成して、Amazon S3 バケットと、そしてオプションで Amazon CloudWatch ロググループと関連付けます。または、CloudTrail Lake を作成できます。これは、CloudTrail ログを最長 7 年間保持し、SQL ベースのクエリ施設を提供します。

AWS は、VPC を使用している顧客は、VPC フローログとAmazon Route 53 リゾルバーのクエリログをそれぞれ使用して、ネットワークトラフィックと DNS ログを有効化し、それらを Amazon S3 バケットまたは CloudWatch ロググループにストリーミングすることを推奨しています。VPC、サブネット、またはネットワークインターフェイス向けに VPC フローログを作成できます。VPC フローログについては、コストを削減するためにどこでどのようにフローログを使用するかを選択できます。

AWS CloudTrail ログ、VPC フローログ、および Route 53 リゾルバーのクエリログは、AWS でセキュリティ調査をサポートするための基本的なログ記録ソースです。また、Amazon Security Lake を使用して、このログデータを収集、正規化、そして Apache Parquet フォーマットと Open Cybersecurity Schema Framework (OCSF) に保存することもできます。Security Lake は、他の AWS ログ、およびサードパーティーソースからのログもサポートします。

AWS のサービスは、Elastic Load Balancing ログ、AWS WAF ログ、AWS Config レコーダーログ、Amazon GuardDuty 検出結果、Amazon Elastic Kubernetes Service (Amazon EKS) 監査ログ、および Amazon EC2 インスタンスのオペレーティングシステムおよびアプリケーションログなど、基本ログソースによってキャプチャされないログを生成できます。ログ記録とモニタリングオプションの詳しいリストについては、『AWS セキュリティインシデント対応ガイド』の「付録 A: クラウド機能の定義 – ログとイベント」を参照してください。

- 各 AWS サービスとアプリケーションの調査ログ機能: 各 AWS サービスとアプリケーションは、ログストレージのオプションを提供し、それぞれが独自の保持とライフサイクル機能が備わっています。2 つの最も良く使用するログストレージサービスは、Amazon Simple Storage Service (Amazon S3) と Amazon CloudWatch です。保持期間が長い場合、費用対効果と柔軟なライフサイクル機能のために Amazon S3 を使用することが推奨されます。プライマリログ記録オプションが Amazon CloudWatch ログの場合、オプションとして、アクセス頻度の低いログを Amazon S3 にアーカイブすることを検討する必要があります。

- ログストレージを選択する: ログストレージの選択肢は、通常、使用するクエリツール、保持機能、精通度、そしてコストに関連しています。ログストレージの主なオプションは、Amazon S3 バケットまたは CloudWatch ロググループです。

  Amazon S3 バケットは、ライフサイクルポリシーがオプションで備わっている、費用対効果に優れ、耐久性の高いストレージを提供します。Amazon S3 バケットに保存されているログは、Amazon Athena などのサービスを使ってクエリすることができます。

  CloudWatch ロググループは、CloudWatch Logs Insights により、耐久性の高いストレージとビルトインクエリ施設を提供します。

- 適切なログ保持を識別する: Amazon S3 バケットまたは CloudWatch ロググループを使ってログを保存する場合、各ログソースに対して適切なライフサイクルを確立して、ストレージと取得コストを最適化する必要があります。顧客のログは通常 3 ヶ月 ~ 1 年間で、すぐにクエリでき、最長 7 年間保持されます。可用性と保持の選択は、セキュリティ要件と、法令、規制、およびビジネス上の義務の組み合わせに合わせるべきです。

- 適切な保持とライフサイクルポリシーを使って各 AWS サービスとアプリケーションのログを有効にする: 組織の各 AWS サービスまたはアプリケーションについて、特定のログ設定ガイダンスを確認してください。

  - AWS CloudTrail 証跡を設定する
  - VPC フローログを設定する
  - Amazon GuardDuty 検出結果エクスポートを設定する
  - AWS Config 記録を設定する
  - AWS WAF Web ACL トラフィックを設定する
  - AWS Network Firewall ネットワークトラフィックログを設定する
  - Elastic Load Balancing アクセスログを設定する
  - Amazon Route 53 リゾルバーのクエリログを設定する
  - Amazon RDS ログを設定する
  - Amazon EKS コントロールプレーンログを設定する
  - Amazon EC2 インスタンスとオンプレミスサーバーに対して Amazon CloudWatch エージェントを設定する

- ログのクエリメカニズムを選択して実装する: ログのクエリについては、CloudWatch ログ分析情報 を CloudWatch ロググループに保存されたデータに、Amazon Athena と Amazon OpenSearch Service を Amazon S3 に保存されたデータに使用できます。また、セキュリティ情報とイベント管理 (SIEM) サービスなど、サードパーティーのクエリツールを使用することもできます。

ログクエリツールを選択するためのプロセスは、セキュリティオペレーションの人材、プロセス、およびテクノロジー側面を考慮する必要があります。オペレーション、ビジネス、セキュリティの要件を満たし、長期的にアクセスとメンテナンスが可能なツールを選択します。ログクエリツールは、スキャンするログの数がツールの制限内に収まっている場合、動作が最適であることに注意してください。コストや技術的な制約から、複数のクエリツールを所有することも珍しくありません。

たとえば、過去 90 日間のデータにはサードパーティーのセキュリティ情報とイベント管理 (SIEM) ツールを使用しながらも、SIEM のログインジェストコストが原因で 90 日以前のデータをクエリする際は Athena を使用するとった場合です。どのような実装であっても、必要なツールの数を最小限に抑えることで、特にセキュリティイベントの調査時に、運用効率が最大となるアプローチであることを確認してください。

- アラートにログを使用する: AWS は、複数のセキュリティサービスを使ってアラートを提供しています:
  - 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 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 は、AWS CloudTrail 管理およびデータイベント、DNS ログ、VPC フローログ、および Amazon EKS 監査ログなどのソースからの情報をインジェスト、集計、および分析します。GuardDuty は、CloudTrail、VPC フローログ、DNS クエリログ、および Amazon EKS から直接独立したデータストリームをプルします。Amazon S3 バケットポリシーを管理したり、ログを収集および保存する方法を変更したりする必要はありません。独自の調査やコンプライアンス目的で、これらのログを保持することは引き続き推奨されています。
  - AWS Security Hub CSPM 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。

  また、これらのサービスの対象外となるセキュリティアラートや、自分の環境に関連する特定なアラートについては、カスタムアラート生成エンジンを使用することもできます。これらのアラートや指示を構築するたの情報については、『AWS セキュリティインシデント対応ガイド』の「検出」を参照してください。

リソース

関連するベストプラクティス:

- SEC04-BP02 ログ、結果、メトリクスを一元的に分析する
- SEC07-BP04 データのライフサイクル管理を定義する
- SEC10-BP06 ツールを事前デプロイする

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)
- Amazon Security Lake の開始方法
- 開始方法: Amazon CloudWatch Logs
- 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:

- AWS re:Invent 2022 - Introducing Amazon Security Lake (Amazon Security Lake の紹介)

関連する例:

- Assisted Log Enabler for AWS (AWS 向けの Assisted Log Enabler)
- AWS Security Hub CSPM Findings Historical Export (AWS セキュリティハブ検出結果履歴レポート)

関連ツール:

- Snowflake for Cybersecurity

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 CSPM 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 CSPM 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 CSPM 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 CSPM 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 CSPM
- 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 CSPM 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 CSPM

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 CSPM
- 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 CSPM 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 CSPM

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 CSPM Findings (Amazon GuardDuty および AWS Security Hub CSPM の調査結果の修復)
- Threat management in the cloud: Amazon GuardDuty and AWS Security Hub CSPM (クラウドにおける変更管理: Amazon GuardDuty および AWS Security Hub CSPM)

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

機密度要件を共有するコンポーネントを階層化し、不正アクセスによる潜在的な影響範囲を最小化します。たとえば、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。トラフィックは、隣接する次に最も機密度が低いリソースからのみ流れる必要があります。ロードバランサーの背後にある Web アプリケーションを考慮します。ロードバランサーからデータベースに直接アクセスできてはいけません。データベースに直接アクセスすべきなのは、ビジネスロジックまたは Web サーバーのみです。

期待される成果: 階層型ネットワークを作成する。階層型ネットワークを使用すると、類似のネットワーキングコンポーネントを論理的にグループ化できます。また、不正ネットワークアクセスの影響の潜在的範囲が縮小されます。適切に階層化されたネットワークでは、不正なユーザーが AWS 環境内で追加リソースをピボットするのが困難になります。内部ネットワークパスをセキュリティ保護するだけでなく、ウェブアプリケーションや API エンドポイントなどのネットワークエッジも保護する必要があります。

一般的なアンチパターン:

- 単一 VPC またはサブネットですべてのリソースを作成する。
- 過度に寛容なセキュリティグループを使用する。
- サブネットを使用しない。
- データベースなどのデータストアに直接アクセスを許可する。

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

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。Lambda 関数などのサーバーレスワークロードを、VPC 内または Amazon API Gateway の背後にデプロイすることを検討してください。インターネットアクセスが不要な AWS Fargate タスクは、ルートがインターネットとの経路がないサブネットに配置する必要があります。この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。AWS 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。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 (Amazon VPC) と Transit Gateway 間のトラフィックは AWS プライベートネットワークに留まりますが、これにより不正ユーザーへの外的露出やセキュリティ上の問題を軽減することができます。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。

実装手順

- Reachability Analyzer を使用して、設定に基づく送信元と送信先間のパスを分析する: Reachability Analyzer では、VPC に接続されたリソースに対する接続の検証を自動化できます。この分析は設定を確認することで行われます (分析を行う際にネットワークパケットは送信されません)。
- Amazon VPC Network Access Analyzer を使用して、リソースへの意図しないネットワークアクセスを特定する: Amazon VPC Network Access Analyzer を使用すると、ネットワークアクセス要件を指定して、潜在的なネットワークパスを特定できます。
- リソースがパブリックサブネットにあるべきかどうか考慮する: VPC のパブリックサブネットには、パブリックソースからのインバウンドネットワークトラフィックを絶対に受信しなければならない場合を除き、リソースを配置しないでください。
- 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) 各ネットワークレイヤーのサブネットを作成し、マイクロセグメンテーションを強化します。正しい ルートテーブルをサブネットと関連付けて、ルーティングとインターネット接続を制御できていることも確認します。
- AWS Firewall Manager を使用して、VPC セキュリティグループを管理する: AWS Firewall Manager は、複数のセキュリティグループを使用する管理上の負担を軽減します。
- AWS WAF を使って一般的な Web 脆弱性を保護する: AWS WAF は、SQL インジェクションなど一般的な Web 脆弱性がトラフィックにないか点検することにより、エッジセキュリティを強化するのに役立ちます。また、特定の国や地域から発信される IP アドレスからのトラフィックを制限することもできます。
- Amazon CloudFront をコンテンツ配信ネットワーク (CDN) として使用する: Amazon CloudFront は、データをユーザーの近くに保管することにより、Web アプリケーションのスピードを向上させるのに役立ちます。また、HTTPS の適用、地域へのアクセス制限、ネットワークトラフィックが CloudFront を経由した場合にのみリソースへのアクセスを許可することで、エッジセキュリティを改善することもできます。
- アプリケーションプログラミングインターフェイス作成時に Amazon API Gateway を使用する (API): Amazon API Gateway は、セキュアな REST、HTTPS、および WebSocket APIs を公開、モニタリング、およびセキュリティ保護するのに役立ちます。

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

- Reachability Analyzer
- Amazon VPC Network Access Analyzer

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高速化と保護)
- AWS re:Inforce 2022 - Validate effective network access controls on AWS(AWS で効果的なネットワークアクセスコントロールを検証する)
- AWS re:Inforce 2022 - Advanced protections against bots using AWS WAF (AWS WAF を使用したボットからの高度な保護)

関連する例:

- Well-Architected ラボ - Automated Deployment of VPC (VPC の自動デプロイ)
- ワークショップ: Amazon VPC Network Access Analyzer

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。

期待される成果: 脆弱性管理プログラムを作成して維持する。Amazon EC2 インスタンス、Amazon Elastic Container Service (Amazon ECS) コンテナ、および Amazon Elastic Kubernetes Service (Amazon EKS) ワークロードなどのリソースを定期的にスキャンしてパッチを適用する。Amazon Relational Database Service (Amazon RDS) データベースなど、AWS マネージドリソースのメンテナンスウィンドウを設定する。静的コードスキャンを使って、アプリケーションソースコードに一般的な問題がないかどうか検査する。組織に必要なスキルがあるかどうか、または外部のアシスタンスを雇用できるかどうか調べるために、Web アプリケーションペネトレーションテストを検討します。

一般的なアンチパターン:

- 脆弱性管理プログラムがない。
- 重大度またはリスク回避を考慮せずに、システムパッチ適用を実施する。

- ベンダーが提供する耐用年数 (EOL) を過ぎたソフトウェアを使用する。
- セキュリティの問題を分析する前に、本番環境にコードをデプロイする。

このベストプラクティスを活用するメリット:

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

脆弱性管理プログラムには、セキュリティ評価、問題の特定、優先順位付け、問題解決の一環としてのパッチ適用の実施などが含まれます。オートメーションは、ワークロードの問題や意図しないネットワークへの露出を継続的にスキャンし、修復を実行するための鍵となります。リソースの作成と更新を自動化することにより時間の節約となり、それ以上の問題を生じさせる設定エラーのリスクを低減します。優れた設計の脆弱性管理プログラムでは、ソフトウェアライフサイクルの開発およびデプロイ段階における脆弱性テストも考慮する必要があります。開発とデプロイ中に脆弱性管理を実装することにより、脆弱性が本番環境に入り込む可能性を低減させます。

脆弱性管理プログラムを実装するには、AWS 責任共有モデルと、それが特定のワークロードにどのように関連するかを理解する必要があります。責任共有モデルでは、AWS に AWS クラウド のインフラストラクチャを保護する責任があります。このインフラストラクチャは、AWS クラウド クラウドサービスを実行するハードウェア、ソフトウェア、ネットワーク、および施設で構成されています。ユーザーには、実績データ、セキュリティ設定、Amazon EC2 インスタンスの管理タスクなどクラウド内のセキュリティ、さらにはAmazon S3 オブジェクトが適切に分類・設定されていることを確認する責任があります。脆弱性管理へのアプローチは、利用するサービスによっても異なります。たとえば、AWS はマネージド型のリレーショナルデータベースサービス Amazon RDS に対するパッチ適用を管理しますが、自己ホスト型データベースのパッチ適用はユーザーの責任となります。

AWS には、脆弱性管理プログラムに役立つ様々なサービスがあります。Amazon Inspector は、ソフトウェアの問題と意図しないネットワークアクセスを検出するために、継続的に AWS ワークロードをスキャンします。AWS Systems Manager Patch Manager を使うと、Amazon EC2 インスタンス全体のパッチ適用を管理できます。Amazon Inspector と Systems Manager は、AWS Security Hub CSPM で表示できます。これは、AWS セキュリティチェックを自動化して、セキュリティアラートを一元化するのに役立つクラウドセキュリティ体制管理サービスです。

Amazon CodeGuru を使うと、静的コード分析を使って、Java および Python アプリケーションの潜在的問題を特定できます。

実装手順

- Amazon Inspector を設定する: Amazon Inspector は新たに起動された Amazon EC2 インスタンス、Lambda 関数、および Amazon ECR にプッシュされた適格なコンテナイメージを自動的に検出し、ソフトウェア問題、潜在的な欠陥、および意図しないネットワーク露出がないかスキャンします。
- ソースコードをスキャンする: ライブラリと依存関係をスキャンして、問題と欠陥がないか調べます。Amazon CodeGuru は、Java と Python アプリケーションの両方について 一般的なセキュリティ問題を修復するための推奨事項を伝えます。The OWASP Foundation は、Source Code Analysis Tools (SAST ツール) を公開しています。
- 既存環境をスキャンしてパッチを適用するメカニズム、さらには CI/CD パイプラインのビルドプロセスの一環としてスキャンするメカニズムを導入する: 新しい脅威からの保護を強化するため、依存関係や OS の問題をスキャンしてパッチを適用するメカニズムを実装します。そのメカニズムを定期的に実行します。ソフトウェア脆弱性管理は、パッチを適用したりソフトウェア問題に対処したりする状況を理解するのに不可欠です。継続的インテグレーション/継続的デリバリー (CI/CD) パイプラインの早期に脆弱性評価を組み込むことで、潜在的なセキュリティ脆弱性の問題修復の優先度を決定します。アプローチは、利用する AWS サービスによっても異なります。Amazon EC2 インスタンスで実行するソフトウェアの潜在的問題をチェックするには、Amazon Inspector をパイプラインに追加して、問題や潜在的欠陥が検出されたらアラートを発動して、ビルドプロセスを停止します。Amazon Inspector は継続的にリソースをモニタリングします。脆弱性管理には、OWASP Dependency-Check、Snyk、OpenVAS、パッケージマネージャー、および AWS Partner ツールを使うこともできます。
- AWS Systems Manager を使用する: 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。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 は、Microsoft アプリケーション、Windows sellable サービスパック、およびLinux ベースインスタンスのマイナーアップグレードなど、オペレーティングシステムとアプリケーション両方の Amazon EC2 インスタンスに対するパッチ適用に使用できます。Amazon EC2 に加え、Patch Manager はオンプレミスサーバーへのパッチ適用にも使用できます。

  サポート対象であるオペレーティングシステムの一覧については、Systems Manager ユーザーガイドの「Supported operating systems (サポートされるオペレーティングシステム) 」で確認してください。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。
- AWS Security Hub CSPM を使用する: Security Hub CSPM AWS の総合的なセキュリティ状態を把握できます。複数の AWS サービス全体のセキュリティデータを収集して、標準化されたフォー

マットでそれらの検出結果を提供し、AWS サービス全体のセキュリティ検出結果の優先順位を決定できます。

- AWS CloudFormation を使用する: AWS CloudFormation は、複数のアカウントや環境にまたがるリソースデプロイの自動化やリソースアーキテクチャの標準化により、脆弱性管理を支援する Infrastructure as code (IaC) サービスです。

リソース

関連するドキュメント:

- AWS Systems Manager
- Security Overview of AWS Lambda (AWS Lambda のセキュリティ概要)
- Amazon CodeGuru
- Improved, Automated Vulnerability Management for Cloud Workloads with a New Amazon Inspector (改善、自動化された新しい Amazon Inspector のクラウドワークロードの脆弱性管理)
- Automate vulnerability management and remediation in AWS using Amazon Inspector and AWS Systems Manager – Part 1 (Amazon Inspector と AWS Systems Manager を使って AWS の脆弱性管理と修復を自動化する - パート 1)

関連動画:

- Securing Serverless and Container Services (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 (Amazon EC2 インスタンスメタデータサービスにおけるセキュリティベストプラクティス)

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 マネジメントコンソール 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

ワークロードが処理するデータの型と分類、関連するビジネスプロセス、データの保管場所、データ所有者を理解することは非常に重要です。また、ワークロードに適用する法的要件やコンプライアン

ス要件、そしてどのようなデータコントロールを適用すべきなのかを理解する必要があります。データの特定は、データ分類作業の最初のステップです。

このベストプラクティスを活用するメリット:

データ分類により、ワークロード所有者は、機密データを保存する場所を特定し、そのデータへのアクセス方法や共有方法を決定できます。

データ分類は、次の質問への回答となります:

- どのようなタイプのデータを持っていますか?

  次のようなデータが考えられます:

  - 企業秘密、特許、または契約合意などの知的財産 (IP)。
  - 個人と結びついた医療履歴情報を含む医療記録などの、保護対象医療情報 (PHI)。
  - 氏名、住所、生年月日、国民 ID または登録番号などの個人を特定できる情報 (PII)。
  - 会員番号 (PAN)、カード会員名、有効期限、サービスコード番号などクレジットカードのデータ。
  - 機密性の高いデータはどこに保存しますか?
  - データにアクセス、変更、削除できる人は誰ですか?
  - データの誤った取扱いを防ぐためには、ユーザーのアクセス許可を把握することが不可欠です。
- 作成、読み取り、更新、削除 (CRUD) 操作を実行できるのは誰ですか?
  - 誰がデータに対するアクセス許可を管理できるかを理解することにより、アクセス許可が昇格する可能性を考慮します。
- データが意図せずに開示されたり、変更または削除された場合、ビジネスに対してどのような影響が生じますか?
  - データが変更、削除、または誤って開示された場合のリスク結果を理解します。

これらの質問に対する回答を把握することにより、次のようなアクションを取ることができます。

- 機密データの範囲 (機密データの場所の数など) を縮小し、機密データへのアクセスを承認済みユーザーのみに限定します。
- 暗号化、データ紛失防止、およびアイデンティティやアクセス管理など、適切なデータ保護メカニズムとテクニックを実装できるよう、さまざまなデータ型について理解を深めます。
- データの正しいコントロール目的を提供することにより、コストを最適化します。

- データの型や量、機密度の異なるデータをどのように隔離しているかなど、規制当局や監査人からの質問に自信を持って答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: 高

実装のガイダンス

データ分類は、データの機密性を識別する行為です。タグ付けを行って、データを簡単に検索し、追跡できるようにすることもあります。また、データ分類を行うと、データの重複を減らし、ストレージやバックアップのコストを削減すると同時に、検索プロセスを高速化できます。

Amazon Macie などのサービスを使用して、機密データの検出と分類両方を大規模に自動化します。Amazon EventBridge や AWS Config など他のサービスは、暗号化されていない Amazon Simple Storage Service (Amazon S3) バケットや Amazon EC2 EBS ボリュームまたはタグが付いていないデータリソースなどのデータセキュリティの問題に対する修復を自動化するために使用できます。AWS サービス統合の完全なリストについては、「EventBridge ドキュメント」を参照してください。

顧客のメール、サポートチケット、製品レビュー、およびソーシャルメディアなどの構造化されていないデータで PII を検出することは、Amazon Comprehend を使うことにより実現します。これは、機械学習 (ML)を用いて構造化されていないテキストで人、場所、感情、話題などのインサイトや関係性を見つけ出す自然言語処理 (NLP) サービスです。データ識別に役立つ AWS サービスのリストについては、「AWS サービスを使って PHI と PII データを検出する一般的テクニック」を参照してください。

データ分類と保護をサポートするもう 1 つの方法は、AWS リソースのタグ付けです。タグ付けを行うと、リソースの管理、特定、整理、検索、およびフィルタリングに使用できる AWS リソースにメタデータを割り当てることができます。

場合によっては、特定のワークロードやサービスが既知のデータ分類のプロセスまたは伝送を保存することが期待されている場合など、リソース全体 (S3 バケットなど) にタグを付けるよう選択することがあります。

適切な場合、管理業務とセキュリティメンテナンスを行いやすくするため、個別のオブジェクトではなく S3 バケットにタグ付けをすることもできます。

実装手順

Amazon S3 内の機密データを検出する:

1. 開始する前に、Amazon Macie コンソールと API オペレーションにアクセスするための適切なアクセス権限があることを確認してください。さらに詳しい情報については、「Amazon Macie の開始方法」を参照してください。
2. Amazon Macie を使用して、機密データが Amazon S3 内にある場合は自動化されたデータ検出を実行します。
   - Amazon Macie の開始方法ガイドを使って、機密データ検出結果のリポジトリを設定し、機密データの検出ジョブを作成します。
   - How to use Amazon Macie to preview sensitive data in S3 buckets (Amazon Macie を使って、S3 バケットで機密データをプレビューする方法)

     デフォルトでは、Macie が、自動化された機密データの検出に推奨されるマネージドデータ識別子のセットを使用して、オブジェクトを分析します。分析は、アカウントまたは組織に対して自動化された機密データ検出を実行する際に Macie が特定のマネージドデータ識別子、カスタムデータ識別子、許可リストを使うように設定することにより、カスタマイズすることができます。特定のバケット (たとえば、通常 AWS ログデータを保存する S3 バケットなど) を除外することにより、分析の対象範囲を調整できます。
3. 自動化された機密データ検出を設定するには、「Performing automated sensitive data discovery with Amazon Macie」 (Amazon Macie を使って自動化された機密データ検出を実行する) を参照してください。
4. また、Automated Data Discovery for Amazon Macie (Amazon Macie の自動化されたデータ検出) も検討してください。

Amazon RDS 内の機密データを検出する:

Amazon Relational Database Service (Amazon RDS) データベース内のデータ検出の詳細については、「Enabling data classification for Amazon RDS database with Macie」(Macie で Amazon RDS データベースのデータ分類を有効化する) を参照してください。

DynamoDB 内の機密データを検出する:

- Detecting sensitive data in DynamoDB with Macie (Macie を使って DynamoDB 内で機密データを検出する) では、Amazon Macie を使って、Amazon S3 にエクスポートしてスキャンすることで、Amazon DynamoDB テーブル内で機密データを検出する方法を説明しています。

AWS パートナーのソリューション

- 広範囲の AWS Partner Network の使用を検討してください。AWS パートナーには、AWS サービスと直接統合する広範囲のツールとコンプライアンスフレームワークがあります。パートナーは、組織のニーズを満たすのに役立つカスタマイズされたガバナンスとコンプライアンスのソリューションを提供します。
- データ分類におけるカスタマイズされたソリューションについては、「Data governance in the age of regulation and compliance requirements」(規制およびコンプライアンス要件時代のデータガバナンス) を参照してください。

AWS Organizations を使用してポリシーを作成およびデプロイすることにより、組織が採用するタグ付け標準を自動的に適用できます。タグポリシーを使用すると、有効なキー名と各キーに対して有効な値を指定できます。モニタリングのみの選択も可能で、既存のタグを評価し、クリーンアップする機会を提供します。タグが選択した標準を満たすと、タグポリシーで適用をオンにして、非準拠のタグが作成されるのを防ぐことができます。詳細については、Securing resource tags used for authorization using a service control policy in AWS Organizations (AWS Organizations のサービスコントロールポリシーを使用して、認可に使用するリソースタグを保護する) および preventing tags from being modified except by authorized principals (タグを許可されたプリンシパル以外が変更できないようにする) のサンプルポリシーを参照してください。

- AWS Organizations でタグポリシーの使用を開始するには、より高度なタグポリシーに移行する前に、「Getting started with tag policies」(タグポリシーの開始方法) を参照してください。組織単位 (OU) または組織全体に拡大する前に、単純なタグポリシーを単一のアカウントにアタッチした場合の効果を理解することにより、タグポリシーの遵守を適用する前にタグポリシーの効果を確認することができます。Getting started with tag policies (タグポリシーの開始方法) には、より高度なポリシー関連タスクの実行方法へのリンクが記載されています。
- データ分類をサポートする他の AWS サービスおよび機能を評価することを検討してください。これらは、データ分類ホワイトペーパーに列挙されています。

リソース

関連するドキュメント:

- Getting started with Amazon Macie (Amazon SQS の開始方法)
- Automated data discovery with Amazon Macie (Amazon Macie を使用した自動データ検出)
- Getting started with tag policies (タブポリシーの開始方法)
- Detecting PII entities (PI エンティティの検出)

関連ブログ:

- How to use Amazon Macie to preview sensitive data in S3 buckets (Amazon Macie を使って、S3 バケットで機密データをプレビューする方法)
- Performing automated sensitive data discovery with Amazon Macie (Amazon Macie を使って自動化された機密データ検出を実行する)
- Common techniques to detect PHI and PII data using AWS Services (AWS のサービスを使って PHI と PII データを検出する一般的なテクニック)
- Detecting and redacting PII using Amazon Comprehend (Amazon Comprehend を使用した PII の検出と再編集)
- Securing resource tags used for authorization using a service control policy in AWS Organizations (AWS Organizations のサービスコントロールポリシーを使用して、認可に使用するリソースタグを保護する)
- Enabling data classification for Amazon RDS database with Macie (Macie で Amazon RDS データベースのデータ分類を可能にする)
- Detecting sensitive data in DynamoDB with Macie (Macie を使った DynamoDB の機密データの検出)
-

関連動画:

- Event-driven data security using Amazon Macie (Amazon Macie を使用したイベント駆動型データセキュリティ)
- Amazon Macie for data protection and governance (データ保護とガバナンスのための Amazon Macie)
- Fine-tune sensitive data findings with allow lists (許可リストで機密データ検出を微調整する)

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

保管中のデータには暗号化の使用を適用する必要があります。暗号化は、不正なアクセスや偶発的な開示が発生した場合、機密性の高いデータの機密を保持します。

期待される成果: プライベートデータが保管中にデフォルトで暗号化される。暗号化を行うと、データの機密性を維持し、意図的または不注意によるデータの開示や流出に対する保護層を追加して強化できます。暗号化されたデータは、まずそれを解除しないと読み出すこともアクセスすることもできません。暗号化されずに保管されたデータは、インベントリに入れて制御する必要があります。

一般的なアンチパターン:

- デフォルトで暗号化する設定を使用しない。
- 複合キーに過度に寛容なアクセスを提供する。
- 暗号化および復号化キーの使用をモニタリングしない。
- データを暗号化せずに保管する。
- データの使用、タイプ、分類に関係なく、すべてのデータに同じ暗号化キーを使う。

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

暗号化キーとワークロード内のデータ分類をマッピングします。このアプローチは、データに単一の暗号化キーまたは非常にわずかな暗号化キーを使用する場合、過度に寛容なアクセスから保護するのに役立ちます (「(ワークロード内のデータを特定する) を参照してください)。

AWS Key Management Service (AWS KMS) は、多くの AWS サービスと統合し、保管中のデータを暗号化しやすくします。例えば、Amazon Simple Storage Service (Amazon S3) では、新しいオブジェクトが自動的に暗号化されるように、バケットの暗号化をデフォルトで設定できます。AWS KMS を使用する際は、どの程度厳格にデータを制限すべきかを検討してください。デフォルトでサービス制御型の AWS KMS キーは、AWS がユーザーに変わって管理および使用します。基盤と

なる暗号化キーにへのアクセスを細かく管理すべき機密データの場合、カスタマーマネージドキー (CMK) を検討してください。キーポリシーを使用することで、ローテーションやアクセス管理など、CMK を完全に制御できます。

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。AWS Config ルール を使用して、Amazon Elastic Block Store (Amazon EBS) ボリューム、Amazon Relational Database Service (Amazon RDS) インスタンス、および Amazon S3 バケットなどに対して暗号化を使用していることを自動的に確認します。

AWS はまた、クライアント側の暗号化も提供するため、クラウドにアップロードする前にデータを暗号化できます。AWS Encryption SDK は、エンベロープ暗号化を使ってデータを暗号化する方法を提供します。ラッピングキーを提供すると、AWS Encryption SDK が暗号化する各データオブジェクトに対して固有のデータキーを生成します。マネージド単一テナントハードウェアセキュリティモジュール (HSM) が必要な場合は、AWS CloudHSM を検討します。AWS CloudHSM では、FIPS 140-2 レベル 3 検証済み HSM で暗号化キーを生成、インポート、管理できます。AWS CloudHSM のユースケースには、認証局 (CA) 発行用プライベートキーの保護、Oracle データベースに対する Transparent Database Encryption (TDE) の有効化などが挙げられます。AWS CloudHSM Client SDK は、データを AWS にアップロードする前に、AWS CloudHSM 内に保管されたキーを使って、クライアント側でデータを暗号化できるソフトウェアを提供します。Amazon DynamoDB Encryption Client ではまた、DynamoDB テーブルにアップロードする前のアイテムを暗号化および署名することもできます。

実装手順

- 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。

  新しい Amazon 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 Amazon 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。

  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。

  Amazon RDS 暗号化を設定する: 暗号化オプションを使用して、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。

  各データ分類に対する適切なプリンシパルへのアクセスを制限するポリシーを使って AWS KMS キーを作成および設定する: 例えば、本番環境データの暗号化のために AWS KMS キーを 1 つ、

開発またはテストデータの暗号化のためにもう 1 つ作成します。他の AWS アカウント に対してキーアクセスを提供することもできます。開発環境と本番環境のアカウントは別にすることを検討してください。本番環境で開発アカウントのアーティファクトを復号化する必要がある場合、開発アーティファクトを暗号化するのに使用する CMK ポリシーを編集し、本番アカウントにアーティファクトを復号化する機能を付与できます。次に、本番環境が本番で使用するために復号化されたデータをインジェストできます。

追加の AWS サービスで暗号化を設定する: 他の AWS サービスを使用する場合は、サービスの暗号化オプションを決定するために、そのサービスの「セキュリティドキュメント」を参照してください。

リソース

関連するドキュメント:

- AWS Crypto Tools
- AWS ドキュメント
- AWS Encryption SDK
- AWS KMS Cryptographic Details Whitepaper (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 cryptographic services and tools (AWS 暗号化サービスとツール)
- Amazon EBS Encryption (Amazon EBS 暗号化)
- Default encryption for Amazon EBS volumes (Amazon EBS ボリュームのデフォルトの暗号化)
- Encrypting Amazon RDS Resources (Amazon RDS リソースの暗号化)
- Amazon 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:

- How Encryption Works in AWS (AWS の暗号化の仕組み)
- Securing Your Block Storage on AWS (AWS でブロックストレージをセキュリティ保護する)

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 CSPM 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

保管中のデータを保護するには、分離やバージョニングなどのメカニズムを使ってアクセス制御を実施し、最小特権の原則を適用してください。データへパブリックアクセスが付与されるのを防止します。

期待される成果: 「知る必要」に基づき、認証されたユーザーのみがデータへアクセスできるようにします。定期的なバックアップとバージョニングでデータを保護し、意図しない、または不注意によるデータの改ざんや削除を防止します。重要なデータを他のデータから分離して、機密性とデータ整合性を保護します。

一般的なアンチパターン:

- 機密度要件と分類の異なるデータを一緒に保管する。
- 復号化キーに、過度に寛容なアクセス許可を使用する。
- データを不適切に分類する。
- 重要なデータの詳細なバックアップを保持しない。
- 本番データへの永続的なアクセスを提供する。
- データアクセスを監査することも、定期的にアクセス許可を審査することもしていない。

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

アクセス (最小特権を使用)、分離、バージョニングなど、複数のコントロールによって保管中のデータを保護できます。データへのアクセスは、AWS CloudTrail などの探査メカニズムと、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パブリックで利用可能なデータ量の削減します。

Amazon Glacier のボールトロックと Amazon S3 オブジェクトロックは、Amazon S3 のオブジェクトに対して必須のアクセス制御を提供します。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。

実装手順

- 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、最小特権を用いたアクセスコントロールを適用します。

- さまざまな分類レベルに基づいてデータを分離する: データ分類レベルには異なる AWS アカウント を使用し、それらのアカウントの管理には AWS Organizations を使用します。
- AWS Key Management Service (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。
- 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: 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ったり書き込んだりできないようにすることです。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。パブリックアクセスを許可してはならないバケットが、パブリックアクセスを防ぐように正しく構成されていることを確認します。デフォルトでは、すべての S3 バケットはプライベートであり、明示的にアクセスが許可されたユーザーのみがアクセスできます。
- AWS IAM Access Analyzer を有効にする: IAM Access Analyzer は、Amazon S3 バケットを分析して、S3 ポリシーが外部エンティティにアクセスを付与した時点で検出結果を生成します。
- 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 (該当する場合)。
- Amazon S3 インベントリを使用する: Amazon S3 インベントリは、S3 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ます。
- 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。
- AWS Resource Access Manager Shares を定期的にレビューして、リソースを共有し続けるかどうかを決定します。 Resource Access Manager では、AWS Network Firewall ポリシー、Amazon Route 53 リゾルバールール、およびサブネットなど、Amazon VPC 内のリソースを共有できます。定期的に共有リソースを監査し、共有が不要になったリソースは共有を停止します。

リソース

関連する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

関連するドキュメント:

- AWS KMS Cryptographic Details Whitepaper (AWS KMS 暗号化の詳細についてのホワイトペーパー)

- Introduction to Managing Access Permissions to Your Amazon S3 Resources (Amazon S3 リソースへのアクセス許可の管理の導入)
- Overview of managing access to your AWS KMS resources (AWS KMS リソースへのアクセス管理の概要)
- AWS Config ルール
- Amazon S3 + Amazon CloudFront: A Match Made in the Cloud (理想的な組み合わせ)
- Using versioning (バージョニングの使用)
- Locking Objects Using Amazon S3 Object Lock (Amazon S3 Object Lock を使ってオブジェクトをロックする)
- Sharing an Amazon EBS Snapshot (Amazon EBS スナップショットの共有)
- 共有 AMI
- Hosting a single-page application on Amazon S3 (Amazon S3 でのシングルページアプリケーションのホスティング)

関連動画:

- Securing Your Block Storage on AWS (AWS でブロックストレージをセキュリティ保護する)

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、Quick 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。

  - Quick
- 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

組織的、法的、コンプライアンス要件を満たすための組織のポリシー、法的義務と標準に基づいて、定義された暗号化要件を適用します。機密データを仮想プライベートクラウド (VPC) の外部に送信する場合は、暗号化されたプロトコルのみを使用します。暗号化を行うと、データが信頼できないネットワークを伝送中も、データの機密性を保持できます。

期待される成果: すべてのデータは、安全な TLS プロトコルと暗号スイートを使用して伝送中に暗号化する必要があります。データへの不正なアクセスを軽減するためには、リソースとインターネット間のネットワークトラフィックを暗号化する必要があります。内部 AWS 環境内にのみあるネットワークトラフィックは、可能な場合に TLS を使って暗号化する必要があります。AWS 内部ネット

ワークはデフォルトで暗号化され、VPC 内のネットワークトラフィックは、トラフィック (Amazon EC2 インスタンス、Amazon ECS コンテナなど) を生成しているリソースに権限のない人がアクセスしない限り、なりすましや盗聴を行うことはできませんIPsec 仮想プライベートネットワーク (VPN) を使ってネットワーク間のトラフィックを保護することを検討してください。

一般的なアンチパターン:

- 廃止されたバージョンの SSL、TLS、および暗号スイートコンポーネント (SSL v3.0、1024-bit RSA キー、および RC4 暗号) を使用する。
- パブリック向けリソースとの間で暗号化されていない (HTTP) トラフィックを許可する。
- X.509 証明書をモニタリングし、期限が切れる前に交換しない。
- TLS に自己署名 X.509 証明書を使用する。

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

AWS のサービスには、通信に TLS を使用し、AWS API との通信の際に伝送中データの暗号化を利用できる、HTTPS エンドポイントが用意されています。HTTP など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、Amazon CloudFront または Application Load Balancer でHTTPS に自動的にリダイレクトすることもできます。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、外部ネットワークまたは AWS Direct Connect 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。クライアントが AWS API に電話かける際に、最低でも TLS 1.2 を使用していることを確認してください。AWS は、2023 年 6 月に TLS 1.0 と 1.1 の使用を廃止予定です。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。

実装手順

- 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 EC2 インスタンスに対してのみ HTTPS プロトコルを許可するよう、セキュリティグループを設定します。
- エッジサービスで安全なプロトコルを設定する: HTTPS を Amazon CloudFront と設定して、自分のセキュリティ体制やユースケースに適した セキュリティプロファイルを使用します。
- 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 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。

- ロードバランサーで安全なプロトコルを設定する: リスナーに接続し、クライアントがサポートするなかで最強の暗号スイートを提供するセキュリティポリシーを選択します。Application Load Balancer に HTTPS リスナーを作成します。
- 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。
- 安全なプロトコルを設定する: AWS サービスのドキュメントをレビューして、転送時の暗号化機能を決定します。
- Amazon S3 バケットにアップロードする際、安全なアクセスを設定する: Amazon S3 バケットポリシーコントロールを使用して、データに対して安全なアクセスを適用します。
- AWS Certificate Managerの使用を検討する: ACM では、AWS サービスで使用するためのパブリック TLS 証明書をプロビジョニング、管理、およびデプロイできます。
- プライベート PKI ニーズに対してAWS Private Certificate Authority の使用を検討する: AWS Private CA では、プライベート認証局 (CA) 階層を作成し、暗号化された TLS チャネルの作成に使用できるエンドエンティティ X.509 証明書を発行することができます。

リソース

関連するドキュメント:

- AWS ドキュメント
- Using HTTPS with CloudFront (CloudFront で HTTPS を使う)
- Connect your VPC to remote networks using AWS Virtual Private Network (AWS Virtual Private Network を使用して VPC をリモートネットワークに接続する)
- Create an HTTPS listener for your Application Load Balancer (アプリケーションロードバランサーの HTTPS リスナーを作成する)
- チュートリアル: Amazon Linux 2 で SSL/TLS を設定する
- Using SSL/TLS to encrypt a connection to a DB instance (SSL/TLS を使用した DB インスタンスへの接続の暗号化)
- Configuring security options for connections (接続のセキュリティオプションを設定する)

SEC09-BP03 意図しないデータアクセスの検出を自動化する

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 (Site-to-Site 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

インシデント対応のために最初に作成する文書は、インシデント対応計画です。インシデント対応計画は、インシデント対応プログラムと戦略の基礎とな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するメリット: インシデント対応のプロセスを熟考し、明確に定義することは、インシデント対応プログラムを成功させ、拡張性を持たせるための鍵となります。セキュリティイベントが発生した場合、明確な手順とワークフローがあれば、タイムリーに対応できます。

既にインシデント対応プロセスがある場合もあります。現在の状態にかかわらず、インシデント対応プロセスを定期的に更新、反復、テストすることが重要です。

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

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、ヨーロッパの個人を特定できる情報 (PII) データを含むワークロードを運用している場合は、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。

実装手順

役割と責任の定義

セキュリティイベントに対処するためには、組織横断的な規律と行動力が必要です。組織内には、人事 (HR)、経営陣、法務部など、インシデント発生時に責任、説明責任、相談、情報提供の役割を持つ担当者が多くいるはずです。これらの役割と責任、および第三者が関与する必要があるかどうかを検討してください。多くの地域には、義務や禁止事項を規定する現地の法律があることに注意してください。セキュリティ対応計画のために責任、説明責任、相談、情報提供 (RACI) チャートを作成す

るのはマニュアル的に思われるかもしれませんが、作成することで、迅速かつ直接的なコミュニケーションを促進し、イベントのさまざまな段階のリーダーシップを明確に説明できます。

インシデントが発生した場合、影響の測定に役立つ情報や背景を提供できる対象分野のエキスパート (SME) である、影響を受けるアプリケーションやリソースの所有者と開発者を巻き込むことが重要です。インシデント対応について開発者やアプリケーション所有者の専門知識に頼る際は、事前にやり取りを行い、関係を構築してください。アプリケーション所有者や SME (クラウド管理者やエンジニアなど) は、不慣れまたは複雑な環境、対応者がアクセスできない状況下で対応することが必要な場合もあります。

最後に、信頼できるパートナーは、さらなる専門知識や価値のある調査を提供できるため、調査や対応に関与する可能性があります。自分のチームにこれらのスキルがない場合は、外部の人材に支援を依頼するということも検討できます。

AWS 対応チームとサポートを理解する

- AWS サポート
  - サポート には、AWS ソリューションの成功と運用上の健全性をサポートするツールや専門知識を利用できるさまざまなプランが用意されています。AWS 環境の計画、導入、最適化に役立つテクニカルサポートや、より多くのリソースが必要な場合は、AWS ユースケースに最適なサポートプランを選択できます。
  - お客様 の AWS リソースに影響を与える問題についてのサポートを受けるには、AWS マネジメントコンソール (サインインが必要) のサポートセンターが中心的な窓口になります。サポートへのアクセスは AWS Identity and Access Management によって制御されます。サポート 機能へのアクセスの詳細については、 Getting started with サポートを参照してください。
- AWS カスタマーインシデント対応チーム (CIRT)
  - AWS カスタマーインシデント対応チーム (CIRT) は、24 時間 365 日の体制で専門のグローバル AWS チームであり、 AWS 責任共有モデルのカスタマーサイドでのアクティブなセキュリティイベント中にカスタマーをサポートします。
  - AWS CIRT がお客様をサポートすると、AWS で発生しているセキュリティイベントの優先順位付けと復旧を支援します。AWS サービスログを使用して根本原因の分析を支援し、復旧のための推奨事項を提示します。また、将来のセキュリティイベントを回避するのに役立つセキュリティに関する推奨事項やベストプラクティスを提供することもできます。
  - AWS のお客様は、AWS CIRT と サポート ケースで連携できます。
- DDoS 対応のサポート
  - AWS オファー AWS Shieldは、AWS で実行中のウェブアプリケーションを保護するマネージドで分散型のサービス拒否 (DDoS) 保護サービスを提供します。Shield は、常時検出と自動インラ

イン緩和により、アプリケーションのダウンタイムとレイテンシーを最小限に抑えることができるため、DDoS 保護のメリットを得るために サポート と連携する必要はありません。Shield のティアには、AWS Shield Standard および AWS Shield Advanced の 2 つのティアがあります。これら 2 つのティアの違いについては、 Shield 機能のドキュメントを参照してください。

- AWS Managed Services (AMS)
  - AWS Managed Services (AMS) は AWS インフラストラクチャ管理を継続的に提供するため、お客様はアプリケーションに集中できます。インフラストラクチャを維持するためのベストプラクティスを実行することで、AMS によって運用上のオーバーヘッドとリスクの軽減を支援します。AMS は、変更リクエスト、モニタリング、パッチ管理、セキュリティ、バックアップサービスなどの一般的なアクティビティを自動化し、インフラストラクチャをプロビジョニング、実行、サポートするためにライフサイクル全体にわたるサービスを利用できます。
  - AMS は、一連のセキュリティ検出コントロールの展開に責任を持ち、24 時間 365 日、第一線でアラートに対応します。アラートが発生すると、AMS は標準的な自動プレイブックと手動プレイブックに従って、一貫した対応が行われていることを確認します。これらのプレイブックは、オンボーディング中に AMS のお客様と共有され、AMS との対応策を練り、調整することができます。

インシデント対応計画の策定

インシデント対応計画は、インシデント対応プログラムと戦略の基礎となるように設計されています。インシデント対応計画は正式な文書にする必要があります。インシデント対応計画には通常、次のセクションが含まれます。

- インシデント対応チームの概要: インシデント対応チームの目標と機能の概要を説明します。
- ロールと責任: インシデントに対応する利害関係者を一覧で表示し、インシデントが発生時のそれぞれの役割を詳しく説明します。
- コミュニケーション計画: 連絡先情報とインシデント発生時の連絡方法を詳しく説明します。
- バックアップの通信方法: インシデント時の通信のバックアップとして帯域外通信を行うことがベストプラクティスです。安全な帯域外通信チャネルを提供するアプリケーションの例は AWS Wickr です。
- インシデント対応の段階と取るべき措置: インシデント対応のフェーズ (検出、分析、根絶、封じ込め、復旧など) を列挙し、それらのフェーズで取るべき大まかなアクションも含めます。
- インシデントの重要度と優先順位の定義: インシデントの重大度を分類する方法、インシデントの優先順位付け方法、および重要度の定義がエスカレーション手順にどのように影響するかを詳しく説明します。

これらのセクションは、さまざまな規模や業界の企業で共通していますが、各組織のインシデント対応計画は異なります。組織に最適なインシデント対応計画を立てる必要があります。

リソース

関連するベストプラクティス:

- 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?)

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

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。

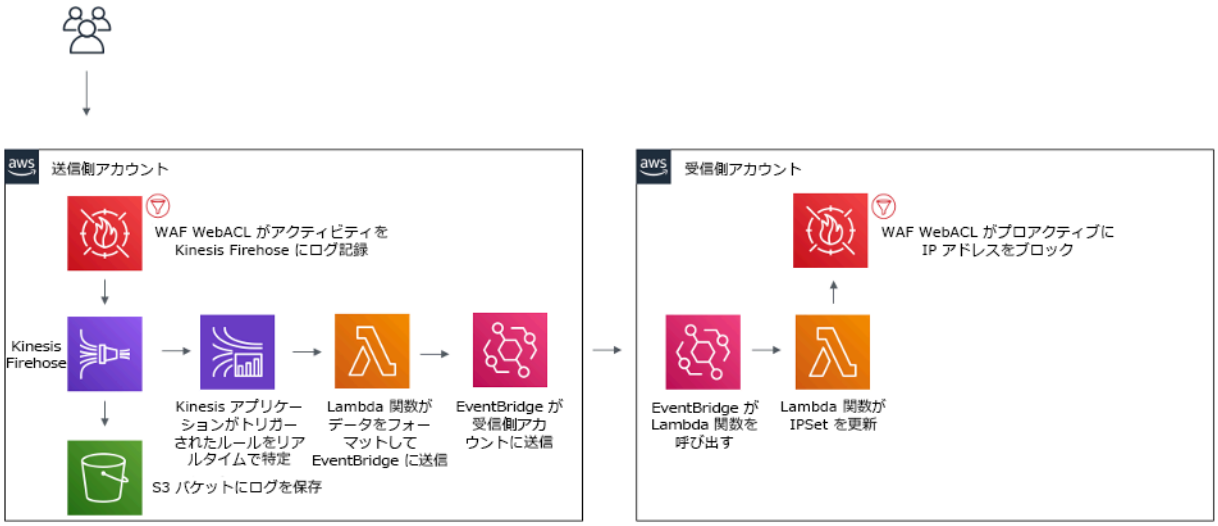


図 3: AWS WAF が既知の悪意ある IP アドレスのブロックを自動化します。

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 マネジメントコンソール 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 アクティビティをログに記録します。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 マネジメントコンソール 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

組織が成長し進化するにつれて、脅威の状況も変化するため、インシデント対応能力を継続的に見直すことが重要になります。この評価を行う方法の 1 つとして、ゲームデーやシミュレーションの実施があります。シミュレーションでは、脅威アクターの戦術、手法、手順 (TTP) を模倣するように設計された現実のセキュリティイベントシナリオを使用します。これにより、組織は実際に発生する

可能性のある模擬サイバーイベントに対応することで、インシデント対応能力を訓練し、評価できます。

このベストプラクティスを活用するメリット: シミュレーションには次のようにさまざまな利点があります。

- サイバー脅威への準備状況を検証し、インシデント対応者の信頼度を高めます。
- ツールとワークフローの精度と効率性をテストします。
- インシデント対応計画に沿うように、コミュニケーションとエスカレーションの方法を改良します。
- あまり一般的でないベクトルに対応する機会を提供します。

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: 中

実装のガイダンス

シミュレーションには主に 3 つのタイプがあります。

- 机上演習: 机上でのシミュレーションは、インシデントに対応するさまざまな利害関係者が参加して役割や責任を実践し、確立されたコミュニケーションツールやプレイブックを活用するディスカッションベースのセッションです。演習は、通常はバーチャル会場、実際の施設、またはそれらの組み合わせが可能で、丸 1 日かけて進行します。ディスカッションベースのため、机上演習ではプロセス、人材、コラボレーションに焦点を当てます。テクノロジーは議論に不可欠ですが、インシデント対応ツールやスクリプトを実際に使用することは、一般的に机上演習には含まれません。
- パープルチーム演習: パープルチーム演習は、インシデント対応者 (ブルーチーム) と模擬の脅威アクター (レッドチーム) のコラボレーションレベルを高めるものです。ブルーチームはセキュリティオペレーションセンター (SOC) のメンバーで構成されますが、実際のサイバーイベントに関与する他の利害関係者が参加することもあります。レッドチームは、攻撃的なセキュリティのトレーニングを受けたペネトレーションテストチームまたは主な利害関係者で構成されています。レッドチームは、シナリオが正確で実現可能なものになるように、演習のファシリテーターと協力して作業します。パープルチーム演習では、インシデント対応の取り組みを支援する検出メカニズム、ツール、標準運用手順 (SOP) に重点が置かれます。
- レッドチーム演習: レッドチーム演習では、攻撃側 (レッドチーム) は、あらかじめ決められた範囲から、ある目的または一連の目的を達成するためのシミュレーションを行います。防御側 (ブルーチーム) は、演習の範囲と期間について必ずしも知識を持っているとは限らないため、実際のインシデントにどのように対応するか、より現実的に評価できます。レッドチーム演習では侵入テスト

になる可能性があります。そのため慎重に行い、コントロールを実施して、演習によって環境に実害を与えないことを確認してください。

定期的にサイバーシミュレーションを実施することを検討してください。演習は、タイプごとにそれぞれのメリットを参加者と組織全体にもたらします。それほど複雑ではないタイプのシミュレーション (机上演習など) から始めて、より複雑なシミュレーションタイプ (レッドチーム演習) に進むこともできます。セキュリティの成熟度、リソース、目標とする成果に基づいてシミュレーションタイプを選択する必要があります。お客様によっては、複雑さやコスト面から、レッドチーム演習を選択しない場合があります。

実装手順

選択したシミュレーションタイプにかかわらず、シミュレーションは通常、以下のような実施手順に従います。

1. 演習の中核となる要素の定義: シミュレーションのシナリオと目的を定義します。いずれも、リーダーの承認が必要です。
2. 主な利害関係者の特定: 少なくとも、演習には演習のファシリテーターと参加者が必要です。シナリオによっては、追加で法務、コミュニケーション、経営幹部などの利害関係者が関与する場合があります。
3. シナリオの構築とテスト: 特定の要素が実現不可能な場合は、シナリオの構築中に再定義が必要なこともあります。このステージのアウトプットとして、シナリオの最終版が完成することが期待されます。
4. シミュレーションの進行: シミュレーションのタイプによって、使用する進行内容 (紙ベースのシナリオと技術的に高度なシミュレーションシナリオの比較) が決まります。ファシリテーターは、演習進行の戦略を目的に合わせて調整し、最大の効果が得られるように、できるだけすべての参加者に演習に参加してもらう必要があります。
5. アフターアクションレビュー (AAR) の作成: うまくいった部分、改善の余地がある部分、潜在的なギャップを特定します。AAR では、シミュレーションの有効性だけでなく、シミュレートされたイベントに対するチームの反応も測定して、今後のシミュレーションの進捗を経時的に追跡できるようにする必要があります。

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)

関連動画:

- AWS GameDay - セキュリティエディション

アプリケーションのセキュリティ

質問

- SEC 11 設計、開発、デプロイのライフサイクル全体を通じて、アプリケーションのセキュリティ特性をどのように組み込み、検証すればよいのでしょうか?

SEC 11 設計、開発、デプロイのライフサイクル全体を通じて、アプリケーションのセキュリティ特性をどのように組み込み、検証すればよいのでしょうか?

スタッフのトレーニング、自動化によるテスト、依存関係の理解、ツールやアプリケーションのセキュリティ特性の検証は、本稼働ワークロードにおいてセキュリティの問題が発生する可能性を減らすうえで役立ちます。

ベストプラクティス

- SEC11-BP01 アプリケーションのセキュリティに関するトレーニングを実施する
- SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する
- SEC11-BP03 定期的にペネトレーションテストを実施する
- SEC11-BP04 手動のコードレビュー
- SEC11-BP05 パッケージと依存関係のサービスを一元化する
- SEC11-BP06 ソフトウェアをプログラムでデプロイする
- SEC11-BP07 パイプラインのセキュリティ特性を定期的に評価する
- SEC11-BP08 ワークロードチームにセキュリティのオーナーシップを根付かせるプログラムを構築する

SEC11-BP01 アプリケーションのセキュリティに関するトレーニングを実施する

組織のビルダーに対して、アプリケーションのセキュアな開発と運用のための一般的な手法に関するトレーニングを実施します。セキュリティを重視した開発手法を導入することは、セキュリティのレビューステージでしか検知されない問題が発生する可能性を減らすうえで役立ちます。

期待される成果: セキュリティを考慮したソフトウェアの設計と構築脅威モデルを起点とするセキュアな開発プラクティスについて組織のビルダーをトレーニングすることで、開発されるソフトウェアの全体的な品質とセキュリティを向上させることができます。このアプローチによって、セキュリティレビューステージ後に必要な再作業を削減でき、ソフトウェアや機能をリリースするまでの時間を短縮できます。

このベストプラクティスでは、セキュアな開発は、開発されるソフトウェア、およびソフトウェア開発ライフサイクル (SDLC) をサポートするツールまたはシステムを指します。

一般的なアンチパターン:

- セキュリティレビューを待ち、その後、システムのセキュリティ特性を考慮する。
- セキュリティに関するすべての意思決定をセキュリティチームに委ねる。
- SDLC での意思決定方法に関するコミュニケーションの欠如が、全体的なセキュリティの期待や組織のポリシーに影響を与える。
- セキュリティレビュープロセスが遅延する。

このベストプラクティスを活用するメリット:

- 開発サイクルの早い段階で、組織のセキュリティ要件に関するより良い理解を得る。
- 潜在的なセキュリティの問題をすばやく識別および修正し、機能リリースまでの時間を短縮する。
- ソフトウェアとシステムの品質の向上。

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

組織のビルダーに対してトレーニングを実施します。セキュリティに関するトレーニングについては、脅威モデリングのコースから始めることを推奨します。理想的には、ビルダーは、ワークロードに関する情報に自分たちでアクセスできることが望まれます。このアクセスによって、ビルダーは他のチームに尋ねることなく、開発するシステムのセキュリティ特性に関する十分な情報に基づいた意思決定を行えます。レビューにおけるセキュリティチームの関与プロセスは、明確に定義され、容易に実行できる必要があります。レビュープロセスの各ステップは、セキュリティトレーニングに含める必要があります。既存の実装パターンやテンプレートを利用できる場合、それらは容易に見つけることができ、全体的なセキュリティ要件にリンクされている必要があります。AWS CloudFormation、AWS Cloud Development Kit (AWS CDK) Constructs、Service Catalog、または他のテンプレートツールの使用を検討し、カスタム構成に必要な時間を短縮します。

実装手順

- 良い基盤を築くため、ビルダーのトレーニングを脅威モデリングのコースから始め、セキュリティをどのように考慮すべきかについてのトレーニングを実施します。
- AWS トレーニング と認定、業種、または AWS パートナートレーニングへのアクセスを提供します。
- セキュリティチーム、ワークロードチーム、および他のステークホルダー間の責任分担を明確にするために、組織のセキュリティレビュープロセスについてのトレーニングを実施します。
- 利用可能な場合、コードの例やテンプレートを含め、セキュリティ要件を満たすためのセルフサービス型のガイダンスを提供します。
- セキュリティレビュープロセスとトレーニングについて、ビルダーチームからフィードバックを定期的に取得し、得られたフィードバックをもとに改善を行います。
- ゲームデーやバグバッシュキャンペーンを活用して、問題数の低減やビルダーのスキル向上に役立てます。

リソース

関連するベストプラクティス:

- SEC11-BP08 ワークロードチームにセキュリティのオーナーシップを根付かせるプログラムを構築する

関連するドキュメント:

- AWS トレーニング と認定
- How to think about cloud security governance (クラウドのセキュリティガバナンスをどのように考えるか)
- How to approach threat modeling (脅威モデリングにアプローチする方法)
- Accelerating training – The AWS Skills Guild (トレーニングの加速化 – AWS Skills Guild)

関連動画:

- Proactive security: Considerations and approaches (プロアクティブなセキュリティ: 考慮事項とアプローチ)

関連する例:

- Workshop on threat modeling (脅威モデリングについてのワークショップ)
- Industry awareness for developers (開発者向けの業界認識)

関連サービス:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK) (AWS CDK) Constructs
- Service Catalog
- AWS BugBust

SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する

開発およびリリースライフサイクル全体を通じて、セキュリティ特性のテストを自動化します。自動化により、リリース前にソフトウェアの潜在的な問題を一貫して繰り返し確認することが容易になります。これにより、提供されるソフトウェアにおけるセキュリティ問題のリスクが減ります。

期待される成果: 自動化テストの目標は、開発の初期段階に、また多くの場合開発ライフサイクルをとおして、潜在的な問題を検知するプログラムを使用した手段を提供することです。リグレッションテストを自動化すると、すでにテスト済みのソフトウェアに変更を加えた後も、そのソフトウェアが期待どおりに動作することを確認するための機能テストおよび非機能テストを実行できます。機能していない認証、または不足している認証など、一般的な設定ミスをチェックするためのセキュリティユニットテストを定義すると、開発プロセスの初期にこれらの問題を識別し修正できます。

テストの自動化では、アプリケーションの要件と期待される機能性に基づいた、アプリケーション検証の目的別テストケースを使用します。自動化テストの結果は、生成されたテスト結果とそれに対応する期待される結果の比較に基づき、全体的なテストライフサイクルを促進します。リグレッションテストやユニットテストスイートなどのテスト手法は、自動化に最も適しています。セキュリティ特性のテストを自動化することで、ビルダーはセキュリティレビューを待つことなく、自動化されたフィードバックを得ることができます。静的または動的なコード分析の形式の自動化テストは、コード品質を改善し、開発ライフサイクルの初期での潜在的なソフトウェアの問題の検知に役立ちます。

一般的なアンチパターン:

- テストケースおよび自動化テストの結果のコミュニケーションの欠如。
- リリース直前のみでの自動化テストの実施。

- 頻繁に変更される要件に関する自動化テストケース。
- セキュリティテストの結果への対処方法に関するガイダンスの欠如。

このベストプラクティスを活用するメリット:

- システムのセキュリティ特性を評価するチームへの依存の低減。
- 複数のワークストリームにわたる一貫した検出結果による一貫性の向上。
- 本稼働ソフトウェアでのセキュリティ問題の低減。
- ソフトウェアの問題を早期に検知することにより、検知から修正までの時間を短縮。
- システム的な動作、または複数のワークストリームにわたって繰り返される動作の可視性の向上により、組織全体での改善を促進。

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

ソフトウェアを構築する際は、アプリケーションのビジネスロジックに基づいた機能性要件と、アプリケーションの信頼性、パフォーマンス、セキュリティにフォーカスした非機能性要件の両方をテストする、ソフトウェアテストのさまざまなメカニズムを採用します。

静的アプリケーションセキュリティテスト (SAST) は、ソースコードの異常なセキュリティパターンを分析し、脆弱性のあるコードを検知します。SAST はさまざまな既知のセキュリティ問題のテストにおいて、ドキュメント (要件仕様、設計文書、設計仕様) やアプリケーションのソースコードなどの、静的なインプットに依存します。静的コードアナライザーは、大規模なコードの迅速な分析に役立ちます。NIST の品質グループは、バイトコードスキャナーやバイナリコードスキャナーのオープンソースツールを含むソースコードセキュリティアナライザーの比較結果を提供しています。

潜在的な予期していない動作を識別するため、実行中のアプリケーションでテストを実施する、動的分析セキュリティテスト (DAST) によって静的テストを補完します。動的テストは、静的分析では検知できない潜在的な問題の検知に使用できます。コードリポジトリ、ビルド、パイプラインステージでテストを実施することで、コード内にあるさまざまなタイプの潜在的な問題をチェックすることができます。Amazon CodeWhisperer は、ビルダーの IDE 内で、セキュリティスキャンを含むコードのレコメンデーションを提供します。Amazon CodeGuru Reviewer は、アプリケーション開発中の重大な問題、セキュリティの問題、検知が難しいバグを識別し、コード品質の改善のためのレコメンデーションを提供します。

開発者のためのセキュリティワークショップでは、SAST と DAST のテスト手法を含むリリースパイプライン自動化のための AWS CodeBuild、AWS CodeCommit、AWS CodePipeline などの AWS 開発者ツールを使用します。

SDLC を進める中で、セキュリティチームと一緒に定期的なアプリケーションレビューを含む反復プロセスを確立します。リリース準備レビューの一部として、これらのセキュリティレビューで得たフィードバックに対処し検証します。これらのレビューにより、アプリケーションの堅固なセキュリティを確立でき、潜在的な問題に対処するための実行可能なフィードバックをビルダーに提供できます。

実装手順

- セキュリティテストを含む、一貫した IDE、コードレビュー、CI/CD ツールを実装します。
- 単に修正が必要な問題をビルダーに伝えるのではなく、SDLC のどこでパイプラインをブロックするのが適切かを考慮します。
- 開発者のためのセキュリティワークショップは、リリースパイプラインでの静的および動的テストの統合の例を提供します。
- 開発者の IDE と統合された Amazon CodeWhisperer、コミットのコードスキャン用の Amazon CodeGuru Reviewer などの自動化ツールを使用したテストまたはコード分析の実施は、適切なタイミングでビルダーにフィードバックを提供するのに役立ちます。
- AWS Lambda を使用した構築では、Amazon Inspector を使用して機能内のアプリケーションコードをスキャンできます。
- AWS CI/CD ワークショップは、AWS での CI/CD パイプライン構築の出発点を提供します。
- 自動化テストを CI/CD パイプラインに含める際は、ソフトウェアの問題の検知と修正を追跡するチケットシステムを使用します。
- 検出結果を生成するセキュリティテストでは、修正のガイダンスをリンクすることで、ビルダーのコード品質の改善を支援します。
- 自動化ツールからの結果を定期的に分析し、次の自動化、ビルダートレーニング、啓発活動の優先順位付けに役立てます。

リソース

関連するドキュメント:

- 継続的デリバリーと継続的なデプロイ

- AWS DevOps コンピテンシーパートナー
- アプリケーションセキュリティの AWS セキュリティコンピテンシーパートナー
- Choosing a Well-Architected CI/CD approach (Well-Architected CI/CD アプローチの選択)
- Monitoring CodeCommit events in Amazon EventBridge and Amazon CloudWatch Events (Amazon EventBridge と Amazon CloudWatch Events での CodeCommit イベントの監視)
- Secrets detection in Amazon CodeGuru Review (Amazon CodeGuru Review でのシークレット検知)
- Accelerate deployments on AWS with effective governance (効果的なガバナンスによる AWS でのデプロイの加速)
- AWS での安全なハンズオフデプロイメントの自動化

関連動画:

- Hands-off: Automating continuous delivery pipelines at Amazon (ハンズオフ: Amazon での継続的デリバリーパイプラインの自動化)
- Automating cross-account CI/CD pipelines (クロスアカウント CI/CD パイプラインの自動化)

関連する例:

- Industry awareness for developers (開発者向けの業界認識)
- AWS CodePipeline Governance (AWS CodePipeline ガバナンス) (GitHub)
- Security for Developers workshop (開発者のためのセキュリティワークショップ)
- AWS CI/CD Workshop (AWS CI/CD ワークショップ)

SEC11-BP03 定期的にペネトレーションテストを実施する

定期的にソフトウェアのペネトレーションテストを実施します。このメカニズムは、自動化されたテストや手動のコードレビューでは検知できない、ソフトウェアの潜在的な問題を識別するうえで役立ちます。また、発見的コントロールの効率について把握するうえでも有効です。ペネトレーションテストでは、保護する必要があるデータを公開する、または予期したよりも広範なアクセス許可を付与するなど、ソフトウェアを予期しない方法で実行できるかどうかの確認を試みます。

期待される成果: ペネトレーションテストは、アプリケーションのセキュリティ特性の検知、修正、検証に使用されます。ソフトウェア開発ライフサイクル (SDLC) の一部として、定期的かつ計画的にペネトレーションテストを実施します。ペネトレーションテストでの検出結果は、ソフトウェアのリ

リース前に対処する必要があります。ペネトレーションテストでの検出結果を分析し、自動化によって検知できる問題があるかどうかを識別します。アクティブなフィードバックメカニズムを含む、定期的で反復可能なペネトレーションテストを実施することで、ビルダーへのガイダンスの提供やソフトウェア品質の向上に役立ちます。

一般的なアンチパターン:

- 既知のセキュリティの問題、または広く発生しているセキュリティの問題に対してのみペネトレーションテストを実施する。
- 依存するサードパーティツールやライブラリを除いてアプリケーションのペネトレーションテストを実施する。
- 実装されたビジネスロジックを評価せずに、パッケージセキュリティの問題のみについてペネトレーションテストを実施する。

このベストプラクティスを活用するメリット:

- リリース前のソフトウェアのセキュリティ特性についての信頼性の向上。
- 好ましいアプリケーションパターンを識別する機会を創出することによる、ソフトウェアのさらなる品質の向上。
- 開発サイクルの早期に、ソフトウェアのセキュリティ特性を改善するための自動化や追加のトレーニングを識別するフィードバックループの確立。

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

ペネトレーションテストは、計画されたセキュリティ侵入シナリオを実行して、セキュリティコントロールを検知、修正、検証する、構造化されたセキュリティテストです。ペネトレーションテストは、アプリケーションの現在の設計とその依存関係に基づいてデータを収集する調査から始まります。セキュリティに特化した厳選されたテストシナリオの一覧が作成され実施されます。これらのテストの主な目的は、環境への意図しないアクセスやデータへの不正アクセスに悪用される可能性のある、アプリケーションのセキュリティの問題を発見することです。新しい機能をリリースしたり、機能や技術的な実装の大きな変更をアプリケーションに加えたりする際は、必ずペネトレーションテストを実施する必要があります。

ペネトレーションテストを実施するのに最適な開発ライフサイクルのステージを特定します。このテストは、システムの機能がリリースに十分に近く、さらに問題の修正に十分な時間を確保できる時期に行う必要があります。

実装手順

- コンテキストを維持するために、脅威モデルに基づいて、ペネトレーションテストのスコープを決定する構造化されたプロセスを確立します。
- ペネトレーションテストの実施に最適な開発ライフサイクルの時期を特定します。これは、アプリケーションで大きな変更が予定されておらず、問題の修正に十分な時間を確保できる時期である必要があります。
- ペネトレーションテストから期待される検出結果、および問題の修正に関する情報の取得について、ビルダーへのトレーニングを実施します。
- 一般的、または反復的なテストを自動化するためのツールを使用して、ペネトレーションテストプロセスの時間を短縮します。
- ペネトレーションテストでの検出結果を分析してシステム的なセキュリティの問題を識別し、このデータを使用して追加の自動化テストと継続的なビルダー教育に役立てます。

リソース

関連するベストプラクティス:

- SEC11-BP01 アプリケーションのセキュリティに関するトレーニングを実施する
- SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する

関連するドキュメント:

- AWS ペネトレーションテストでは、AWS でのペネトレーションテストの詳細なガイダンスを提供しています。
- Accelerate deployments on AWS with effective governance (効果的なガバナンスによる AWS でのデプロイの加速)
- AWS セキュリティコンピテンシーパートナー
- Modernize your penetration testing architecture on AWS Fargate (AWS Fargate でのペネトレーションテストアーキテクチャのモダナイゼーション)
- AWS Fault injection Simulator

関連する例:

- Automate API testing with AWS CodePipeline (AWS CodePipeline での API テストの自動化) (GitHub)
- Automated security helper (セキュリティヘルパーの自動化) (GitHub)

SEC11-BP04 手動のコードレビュー

作成するソフトウェアについて、手動のコードレビューを実施します。このプロセスは、コードを記述した人物が、コードの品質を確認する唯一のユーザーでないことを検証するうえで役立ちます。

期待される成果: 開発に手動のコードレビューステップを含めることで、開発中のソフトウェアの品質を改善したり、経験の浅いチームメンバーのスキルアップを支援したり、自動化を使用できる領域を識別したりすることができます。手動のコードレビューは、自動化ツールやテストによってサポートすることができます。

一般的なアンチパターン:

- デプロイ前にコードレビューを実施していない。
- コードの作成とレビューを同じ担当者が行っている。
- コードレビューの支援と調整に自動化を使用していない。
- コードレビューの前に、アプリケーションセキュリティについてビルダーをトレーニングしていない。

このベストプラクティスを活用するメリット:

- コード品質の向上。
- 共通のアプローチの再利用によるコード開発の一貫性の向上。
- ペネトレーションテストや後工程において検知される問題数の低減。
- チーム内での知識の移転の改善。

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

全体的なコード管理フローの一部として、レビューステップを実装します。詳細は、分岐、プルリクエスト、マージで使用するアプローチによって異なります。それらのアプローチでは AWS CodeCommit、または GitHub、GitLab、Bitbucket のようなサードパーティソリューションを使用す

る場合があります。使用する手法にかかわらず、プロセスを本稼働環境にデプロイする前に、プロセスのレビューが必要であることを認識することが重要です。Amazon CodeGuru Reviewer などのツールを使用すると、コードレビュープロセスを簡単に調整することができます。

実装手順

- コード管理フローの一部として手動のレビューステップを実装し、次に進む前にこのレビューを実施します。
- コードレビューの管理と支援のために Amazon CodeGuru Reviewer の使用を検討します。
- コードを次のステージに進める前にコードレビューの完了を必須とする承認フローを実装します。
- 手動のコードレビューで発見された問題を自動的に検知するプロセスがないか確認します。
- コード開発プラクティスに沿って手動のコードレビューを統合します。

リソース

関連するベストプラクティス:

- SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する

関連するドキュメント:

- Working with pull requests in AWS CodeCommit repositories (AWS CodeCommit でのプルリクエストの使用)
- Working with approval rule templates in AWS CodeCommit (AWS CodeCommit での承認ルールテンプレートの使用)
- About pull requests in GitHub (GitHub でのプルリクエストについて)
- Automate code reviews with Amazon CodeGuru Reviewer (Amazon CodeGuru Reviewer を使用したコードレビューの自動化)
- Automating detection of security vulnerabilities and bugs in CI/CD pipelines using Amazon CodeGuru Reviewer CLI (Amazon CodeGuru Reviewer CLI を使用した CI/CD パイプラインでのセキュリティ脆弱性とバグの検知の自動化)

関連動画:

- Continuous improvement of code quality with Amazon CodeGuru (Amazon CodeGuru を使用したコード品質の継続的な改善)

関連する例:

- Security for Developers workshop (開発者のためのセキュリティワークショップ)

SEC11-BP05 パッケージと依存関係のサービスを一元化する

ビルダーチームに対して、ソフトウェアパッケージとその他の依存関係を取得するための一元化されたサービスを提供します。これにより、記述するソフトウェアに含まれる前に、パッケージの検証が可能になります。また、これは組織で使用中のソフトウェアを分析するためのデータソースとなります。

期待される成果: ソフトウェアは、作成されたコードと、一連の他のソフトウェアパッケージによって構成されます。このため、JSON パーサーや暗号化ライブラリなどの繰り返し使用される機能を容易に実装することができます。これらのパッケージのソースと依存関係を論理的に一元化することで、パッケージが使用される前に、そのパッケージのプロパティを検証するためのメカニズムをセキュリティチームに提供することができます。またこのアプローチによって、既存のパッケージの変更による予期しない問題のリスクや、インターネット上の任意のパッケージをビルダーチームが使用することによるリスクを低減できます。このアプローチを手動および自動化されたテストフローと組み合わせて使用することで、開発中のソフトウェアの品質についての信頼性を高めることができます。

一般的なアンチパターン:

- インターネット上の任意のリポジトリからパッケージを取得する。
- ビルダーに提供する前に、新しいパッケージをテストしていない。

このベストプラクティスを活用するメリット:

- 構築中のソフトウェアで使用されるパッケージのより良い理解。
- 誰が、どのようなパッケージを使用しているかを把握することで、パッケージの更新が必要な場合に、ワークロードチームに通知することができる。
- 問題のあるパッケージがソフトウェアで使用されるリスクの低減。

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

ビルダーが簡単に使用できるように、パッケージと依存関係の一元化されたサービスを提供します。一元化されたサービスは、実装されるモノリシックシステムとしてではなく、論理的に一元化します。このアプローチを使用することで、ビルダーのニーズに合ったサービスを提供できます。更新が必要な場合や新しい要件が発生した際に新しいパッケージをリポジトリに追加する効率的な方法を実装します。AWS CodeArtifact または類似する AWS パートナーソリューションなどの AWS サービスは、このような機能を提供します。

実装手順:

- ソフトウェアが開発されているすべての環境で利用可能な、論理的に一元化されたリポジトリサービスを実装します。
- AWS アカウント のベンディングプロセスの一部として、リポジトリへのアクセスを含めます。
- パッケージをリポジトリに公開する前に、パッケージの自動化テストを構築します。
- 最も大きな変更が発生する頻繁に使用されるパッケージ、言語、チームに関するメトリクスを維持します。
- 新しいパッケージのリクエストとフィードバックの提供を行うための自動化されたメカニズムをビルダーチームに提供します。
- リポジトリ内のパッケージを定期的にスキャンして、新しく発見された問題の潜在的な影響を識別します。

リソース

関連するベストプラクティス:

- SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する

関連するドキュメント:

- Accelerate deployments on AWS with effective governance (効果的なガバナンスによる AWS でのデプロイの加速)
- Tighten your package security with CodeArtifact Package Origin Control toolkit (CodeArtifact Package Origin Control ツールキットを使用してパッケージのセキュリティを高める)
- Detecting security issues in logging with Amazon CodeGuru Reviewer (Amazon CodeGuru Reviewer を使用してセキュリティの問題を口グで検知する)

- Supply chain Levels for Software Artifacts (SLSA) (ソフトウェアアーティファクトのサプライチェーンレベル)

関連動画:

- Proactive security: Considerations and approaches (プロアクティブなセキュリティ: 考慮事項とアプローチ)
- The AWS Philosophy of Security (re:Invent 2017) (AWS のセキュリティ哲学、re:Invent 2017)
- When security, safety, and urgency all matter: Handling Log4Shell (セキュリティ、安全性、緊急性のすべてが問題になるとき: Log4Shell への対応)

関連する例:

- Multi Region Package Publishing Pipeline (マルチリージョンパッケージ公開パイプライン) (GitHub)
- Publishing Node.js Modules on AWS CodeArtifact using AWS CodePipeline (AWS CodePipeline を使用した Node.js モジュールの AWS CodeArtifact への公開) (GitHub)
- AWS CDK Java CodeArtifact Pipeline Sample (AWS CDK Java CodeArtifact パイプラインの例) (GitHub)
- Distribute private .NET NuGet packages with AWS CodeArtifact (AWS CodeArtifact を使用したプライベート .NET NuGet パッケージの配布) (GitHub)

SEC11-BP06 ソフトウェアをプログラムでデプロイする

可能な場合は、ソフトウェアのデプロイをプログラムで行います。この手法により、デプロイに失敗したり、人的エラーにより予期しない問題が発生したりする可能性を低減できます。

期待される成果: AWS クラウド でセキュアに構築するためには、原則としてデータへの人的関与を排除します。この原則には、ソフトウェアのデプロイ方法も含まれます。

ソフトウェアのデプロイにおいて人的な依存を排除することで、テスト済みのソフトウェアがデプロイされ、デプロイが常に一貫して実行されるという信頼性を大幅に高めることができます。異なる環境で動作させるために、ソフトウェアを変更する必要はありません。12 要素のアプリケーション開発の原則、具体的には構成の外部化の原則を用いることで、変更を必要とすることなく、複数の環境に同じコードをデプロイすることができます。暗号化を用いたソフトウェアパッケージの署名は、環境間で変更がないことを証明するための便利な手法です。このアプローチによって、変更プロセスでのリスクを低減し、ソフトウェアリリースの一貫性を向上させることができます。

一般的なアンチパターン:

- 本稼働環境にソフトウェアを手動でデプロイする。
- 環境に合わせて手動でソフトウェアに変更を加える。

このベストプラクティスを活用するメリット:

- ソフトウェアリリースプロセスの信頼性の向上。
- 変更の失敗がビジネスの機能性に与えるリスクの低減。
- 変更リスクの低減によるリリース頻度の向上。
- デプロイ中に予期しないイベントが発生した場合の自動ロールバック機能。
- テスト済みのソフトウェアとデプロイされたソフトウェアが同じであることを暗号化によって証明する能力。

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

持続的な人的アクセスを環境から排除するために AWS アカウント 構造を構築し、CI/CD ツールを使用してデプロイを実施します。環境固有のデータを AWS Systems Manager Parameter Store などの外部ソースから取得するようにアプリケーションを設計します。テスト後にパッケージに署名し、デプロイ中にこれらの署名を検証します。アプリケーションコードをプッシュするように CI/CD パイプラインを設定し、Canary を使用してデプロイの成功を確認します。AWS CloudFormation または AWS CDK などのツールを使用してインフラストラクチャを定義し、AWS CodeBuild と AWS CodePipeline を使用して CI/CD のオペレーションを実行します。

実装手順

- 明確に定義された CI/CD パイプラインを構築して、デプロイプロセスを合理化します。
- AWS CodeBuild と AWS Code Pipeline を使用して、パイプラインへのセキュリティテストの統合を容易にする CI/CD 機能を提供します。
- ホワイトペーパー「Organizing Your AWS Environment Using Multiple Accounts (複数のアカウントを使用した AWS 環境の組織化)」で説明している環境の分離に関するガイダンスに従います。
- 本稼働ワークロードが実行されている環境への持続的な人的アクセスがないことを確認します。
- 構成データの外部化をサポートするようアプリケーションを設計します。

- ブルー/グリーンデプロイモデルを使用したデプロイを検討します。
- Canary を実装してソフトウェアのデプロイの成功を検証します。
- AWS Signer または AWS Key Management Service (AWS KMS) などの暗号化ツールを使用して、デプロイするソフトウェアパッケージの署名と検証を行います。

リソース

関連するベストプラクティス:

- SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する

関連するドキュメント:

- AWS CI/CD Workshop (AWS CI/CD ワークショップ)
- Accelerate deployments on AWS with effective governance (効果的なガバナンスによる AWS でのデプロイの加速)
- 安全なハンズオフデプロイメントの自動化
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys (AWS Certificate Manager Private CA および AWS Key Management Service 非対称キーを使用したコードの署名)
- Code Signing, a Trust and Integrity Control for AWS Lambda (コード署名、AWS Lambda の信頼性および整合性のコントロール)

関連動画:

- Hands-off: Automating continuous delivery pipelines at Amazon (ハンズオフ: Amazon での継続的デリバリーパイプラインの自動化)

関連する例:

- Blue/Green deployments with AWS Fargate (AWS Fargate を使用したブルー/グリーンデプロイ)

SEC11-BP07 パイプラインのセキュリティ特性を定期的に評価する

アクセス許可の分離に特に注意しながら、Well-Architected セキュリティの柱の原則をパイプラインに適用します。パイプラインインフラストラクチャのセキュリティ特性を定期的に評価します。パイ

プラインのセキュリティを効率的に管理することにより、パイプラインを通過するソフトウェアのセキュリティを確保することができます。

期待される成果: ソフトウェアの構築とデプロイに使用するパイプラインは、環境内の他のワークロードと同じ推奨プラクティスに従う必要があります。パイプラインに実装されるテストは、テストを使用するビルダーによって編集できないようにする必要があります。パイプラインへのアクセス許可は、実施するデプロイに必要なものだけに制限し、誤った環境へのデプロイを防ぐための安全対策を実装する必要があります。パイプラインは長期的な認証情報に依存せず、構築環境の整合性を検証できるようにステータスを送信する必要があります。

一般的なアンチパターン:

- ビルダーが回避可能なセキュリティテスト。
- デプロイパイプラインへの広範すぎるアクセス許可。
- 入力を検証するように設定されていないパイプライン。
- CI/CD インフラストラクチャに関連付けられているアクセス許可を定期的にレビューしていない。
- 長期的な認証情報、またはハードコード化された認証情報の使用。

このベストプラクティスを活用するメリット:

- パイプラインをとおして構築およびデプロイされるソフトウェアの整合性についての信頼性の向上。
- 不審なアクティビティが存在するデプロイを停止する能力。

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

IAM ロールをサポートするマネージド CI/CD サービスから始めることで、認証情報の流出リスクを低減することができます。セキュリティの柱の原則を CI/CD パイプラインインフラストラクチャに適用すると、セキュリティの改善が可能な領域を判断するのに役立ちます。AWS デプロイパイプラインリファレンスアーキテクチャに沿うことは、CI/CD 環境の構築の良い起点となります。パイプラインの実装を定期的にレビューし、予期しない動作のログを分析すると、ソフトウェアのデプロイで使用されているパイプラインの使用パターンの理解に役立ちます。

実装手順

- AWS デプロイパイプラインリファレンスアーキテクチャを起点とします。

- AWSIAM Access Analyzer を使用して、プログラム的にパイプラインの最小特権の IAM ポリシーを生成することを検討します。
- 予期しないアクティビティや異常なアクティビティを検知するために、監視と警告をパイプラインに実装します。AWS のマネージドサービスである Amazon EventBridge を使用すると、AWS Lambda や Amazon Simple Notification Service (Amazon SNS) などのターゲットにデータをルーティングすることができます。

リソース

関連するドキュメント:

- AWS Deployment Pipelines Reference Architecture (AWS デプロイパイプラインリファレンスアーキテクチャ)
- Monitoring AWS CodePipeline (AWS CodePipeline のモニタリング)
- Security best practices for AWS CodePipeline (AWS CodePipeline のセキュリティのベストプラクティス)

関連する例:

- DevOps monitoring dashboard (DevOps モニタリングダッシュボード) (GitHub)

SEC11-BP08 ワークロードチームにセキュリティのオーナーシップを根付かせるプログラムを構築する

ビルダーチームが、作成するソフトウェアに関するセキュリティの決定を行えるようにするプログラムやメカニズムを構築します。セキュリティチームは、引き続きレビュー中にこれらの決定を検証する必要がありますが、ビルダーチームにセキュリティのオーナーシップを根付かせることで、より迅速でセキュアなワークロードの構築が可能になります。また、このメカニズムは構築するシステムの運用に良い影響を与える、オーナーシップのカルチャーを育みます。

期待される成果: セキュリティのオーナーシップと意思決定をビルダーチームに根付かせるには、セキュリティについての考え方についてビルダーにトレーニングを実施したり、ビルダーチームにセキュリティスタッフを配置または連携させてトレーニングを補強したりします。いずれのアプローチも適切で、チームは開発サイクルの初期にセキュリティに関する質の高い意思決定を行えるようになります。このオーナーシップモデルは、アプリケーションセキュリティのトレーニングを前提にしています。特定のワークロードの脅威モデルから始めると、適切なコンテキストでの設計思考にフォー

カスするのに役立ちます。セキュリティにフォーカスしたビルダーコミュニティの構築、またはセキュリティエンジニアグループとビルダーチームを連携させることの別の利点として、ソフトウェアの作成についてより深い理解を得られることが挙げられます。この理解は自動化に関する次の改善領域の特定に役立ちます。

一般的なアンチパターン:

- セキュリティ設計に関するすべての意思決定をセキュリティチームに委ねる。
- 開発プロセスの十分に早い段階でセキュリティ要件を策定していない。
- プログラムの運用に関して、ビルダーとセキュリティスタッフからのフィードバックを入手していない。

このベストプラクティスを活用するメリット:

- セキュリティレビューを完了するまでの時間の短縮。
- セキュリティレビューのステージでようやく検知されるセキュリティ問題の削減。
- 作成されるソフトウェアの全体的な品質の向上。
- システム的な問題、または価値の高い改善領域を識別し、理解する機会の創出。
- セキュリティレビューでの結果に基づく再作業量の削減。
- セキュリティ機能に関する認識の改善。

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

SEC11-BP01 アプリケーションのセキュリティに関するトレーニングを実施する のガイダンスから始めます。その後、組織に最も適切と思われるプログラムの運用モデルを識別します。主な 2 つのパターンは、ビルダーのトレーニングの実施、またはビルダーチームへのセキュリティスタッフの配置です。初期アプローチの決定後、組織に適したモデルであるかどうかを検証するために、単一または小規模のワークロードチームに対してテストを実施します。プログラムの実行と成功には、組織のビルダーおよびセキュリティ部門のリーダーシップによるサポートが役立ちます。このプログラムを構築する際は、プログラムの価値を計測できるメトリクスを選択することが重要です。AWS がこの課題にどのようにアプローチしたかを学ぶことは有益です。ベストプラクティスは、主に組織の変化と文化にフォーカスしています。ビルダーとセキュリティのコミュニティ間のコラボレーションをサポートするツールを使用します。

実装手順

- アプリケーションのセキュリティに関するビルダーのトレーニングから始めます。
- ビルダーの教育のためのコミュニティとオンボーディングプログラムを作成します。
- プログラムの名称を決定します。よく使用される名称は、ガーディアン、チャンピオン、アドボケイトなどです。
- ビルダートレーニング、セキュリティエンジニアの配置、セキュリティスタッフとの連携、という三択から、使用するモデルを選択します。
- セキュリティ部門、ビルダー部門、および他の潜在的な関連部門からプロジェクトスポンサーを特定します。
- プログラムへの参加人数、レビューに要した時間、ビルダーやセキュリティスタッフからのフィードバックを計測するメトリクスを追跡します。これらのメトリクスを使用して、改善を行います。

リソース

関連するベストプラクティス:

- SEC11-BP01 アプリケーションのセキュリティに関するトレーニングを実施する
- SEC11-BP02 開発およびリリースライフサイクル全体を通じてテストを自動化する

関連するドキュメント:

- How to approach threat modeling (脅威モデリングにアプローチする方法)
- How to think about cloud security governance (クラウドのセキュリティガバナンスをどのように考えるか)

関連動画:

- Proactive security: Considerations and approaches (プロアクティブなセキュリティ: 考慮事項とアプローチ)

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。実装に関する規範的なガイダンスについては、信頼性の柱のホワイトペーパーを参照してください。

ベストプラクティス領域

- 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

デフォルトのクォータに注意して、ワークロードアーキテクチャに対するクォータ引き上げリクエストを管理しましょう。ディスクやネットワークなど、どのクラウド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。

期待される成果: 主要メトリクスのモニタリング、インフラストラクチャのレビュー、自動修復手順に適切なガイドラインを設けて、サービスの機能低下や停止につながるサービスのクォーターや制約に達していないことを検証することで、AWS アカウントのサービスの機能低下や停止を防止できます。

一般的なアンチパターン:

- 使用しているサービスのハードまたはソフト上のクォータや制限を理解せずにワークロードをデプロイする。
- 必要なクォータの分析と再設定、またはサポートへの事前連絡をせずに、代替ワークロードをデプロイする。
- クラウドサービスには制限がなく、料金、制限、回数、数量を気にせずにサービスを使用できると考えている。
- クォータは自動的に増加すると考えている。
- クォータリクエストのプロセスやスケジュールを知らない。
- クラウドサービスのデフォルトのクォータが、リージョン間で比較する各サービスで同一だと考えている。
- サービスの制約は破ることが可能で、システムによりリソースの制約を超えて自動スケールまたは制限の増加が追加されると考えている。
- リソースの使用率にストレスをかけるためにピーク時トラフィックでアプリケーションをテストしていない。
- 必要なリソースサイズを分析せずにリソースをプロビジョニングする。
- 実際に必要な分または予想されるピークを遥かに超えるリソースタイプを選択することでキャパシティを過剰プロビジョニングする。
- 新規顧客イベントや新技術のデプロイに先駆けて、新しいレベルのトラフィックのキャパシティ要件を評価していない。

このベストプラクティスを活用するメリット: サービスクォータとリソースの制約をモニタリングおよび自動管理すると、エラーを予防できます。ベストプラクティスに従っていないと、顧客のサービスにおけるトラフィックパターンの変化により、停止または機能低下が起こる可能性があります。これらの値をすべてのリージョンとすべてのアカウントでモニタリングし管理することで、有害なイベントや計画外のイベントにおける、アプリケーションの回復力が向上します。

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: 高

実装のガイダンス

Service Quotas は、250 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 を使用して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の AWS ドキュメントにも記載されています (例えば、Amazon VPC クォータを参照してください)。

スロットルされた API のレート制限など、一部のサービス上の制限は、Amazon 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一部の制限には、プロビジョンド IOPS、割り当てられた Amazon RDS ストレージ、Amazon EBS ボリューム割り当てなどがあります。Amazon Elastic Compute Cloud には、インスタンス、Amazon Elastic Block Store、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、サポート に連絡し、緩和策の有無についてお問い合わせください。

サービスクォータはその性質上、リージョン固有である場合も、グローバルである場合もあります。クォータに達している AWS サービスを使用すると、通常の使用で予想どおりに動作しないことや、サービスの停止や機能低下を招くことがあります。例えば、あるサービスクォータではリージョンで使用する DL Amazon EC2 数を制限しており、Auto Scaling グループ (ASG) を使用したトラフィックスケーリングイベント中にその制限に達する可能性があります。

各アカウントのサービスクォータについて、使用量を定期的に評価し、そのアカウントにおける適切なサービス制限を判断する必要があります。このようなサービスクォータは、意図せず必要以上のリソースをプロビジョニングすることを防止する運用上のガードレールとして存在しています。また、API オペレーションにおけるリクエスト率を制限し、不正使用からサービスを保護する役目も持っています。

サービスの制約は、サービスクォータとは異なります。サービスの制約は、リソースタイプごとに決まっている特定のリソースの制限を表します。ストレージキャパシティ (例: gp2 のサイズ制限は 1 GB ~ 16 TB) だったりディスクスループット (10,0000 iops) だったりします。制限に達する可能性のある使用量について、リソースタイプの制約を監督し定期的に評価することが不可欠です。予期せず制約に達した場合、アカウントのアプリケーションまたはサービスが機能低下または停止する恐れがあります。

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ある場合は、サポート に連絡し、緩和策の有無についてお問い合わせください。

修正されたクォータの調整について詳しくは、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 を参照してください。

Service Quotas をモニタリングおよび管理できる AWS のサービスやツールが多数用意されています。これらのサービスやツールは、クォータレベルの自動または手動チェックの提供に活用するものです。

- AWS Trusted Advisor は、いくつかのサービスの一部の側面に対する利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を提供しています。クォータに迫っているサービスの特定に役立ちます。
- AWS マネジメントコンソールでは、サービスのクォータ値の表示、管理、新しいクォータのリクエスト、クォータリクエストのステータスのモニタリング、クォータの履歴の表示を行う手段を提供しています。
- AWS CLI および CDK は、サービスクォータのレベルと使用量を自動で管理およびモニタリングする、プログラムによる手段を提供します。

実装手順

Service Quotas の場合:

- AWS Service Quotas をレビューします。
- 既存のサービスクォータを把握し、使用されているサービス (IAM Access Analyzer など) を判断します。サービスクォータで制御されている AWS のサービスは約 250 あります。次に、各アカウントとリージョンで使用されている可能性のある特定のサービスクォータ名を判断します。リージョンごとに約 3,000 のサービスクォータ名があります。
- このクォータ分析を AWS Config で強化して、AWS アカウントで使用されているすべての AWS リソースを見つけます。
- AWS CloudFormation データを使用して、使用されている AWS リソースを判断します。AWS マネジメントコンソールまたは `list-stack-resources` AWS CLI コマンドを使用して作成されたリソースを見つけます。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。
- 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。
- 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 を使用してプログラムでアクセスできる情報を使用します。

- サービスクォータが制限に近づいたか達した場合にアラートを発して知らせる自動モニタリング手段を作成します (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 および REL01-BP04 クォータをモニタリングおよび管理する を参照)。
- 同一アカウント内のあるリージョンでサービスクォータが変更されたが他のリージョンでは変更されていない場合を確認する、自動またはプログラムによる手段を作成します (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 および REL01-BP04 クォータをモニタリングおよび管理する を参照)。
- アプリケーションログやメトリクスのスキャンを自動化して、クォータまたはサービス制約のエラーが発生しているか判断します。エラーが発生している場合は、モニタリングシステムにアラートを送信します。
- 特定のサービスでクォータの引き上げが必要であると判断された場合に、クォータで必要な変更を計算するエンジニアリング手順を作成します (REL01-BP05 クォータ管理を自動化する を参照)。
- サービスクォータの変更をリクエストするプロビジョニングおよび承認ワークフローを作成します。これには、リクエストが拒否または一部承認された場合の例外ワークフローを含めます。
- 本稼働環境またはロード済み環境にロールアウトする前に、新しい AWS サービスをプロビジョニングして使用するにあたって、サービスクォータをレビューするエンジニアリング手段を作成します (例: ロードテストアカウント)。

サービス制約の場合:

- 読み取りがリソースの制約に近づいているリソースについてアラートを発報するモニタリングおよびメトリクス手段を作成します。必要に応じて CloudWatch を活用して、メトリクスまたはログをモニタリングします。
- 制約のある各リソースについて、アプリケーションまたはシステムにとって有意なアラートのしきい値を設定します。
- 制約が使用量に近い場合、リソースタイプを変更するワークフローまたはインフラストラクチャ管理手順を作成します。このワークフローには、ベストプラクティスとして負荷テストを含め、新しいタイプが新しい制約のある適切なリソースタイプであることを検証します。
- 既存の手順やプロセスを使用して、特定したリソースを推奨の新しいリソースタイプに移行します。

リソース

関連するベストプラクティス:

- 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
- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL10-BP01 複数の場所にワークロードをデプロイする
- 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
- REL11-BP03 すべてのレイヤーの修復を自動化する
- 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

関連するドキュメント:

- AWS Well Architected フレームワークの信頼性の柱: 可用性
- 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
- クォータの引き上げのリクエスト方法
- サービスエンドポイントとクォータ
- Service Quotas ユーザーガイド
- AWS のクォータモニター
- AWS Fault Isolation Boundaries (AWS の障害分離境界)
- Availability with redundancy (冗長性を備えた可用性)
- データのための AWS
- 継続的インテグレーションとは
- 継続的デリバリーとは
- APN パートナー: 設定管理を支援できるパートナー

- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS のテナント別アカウント SaaS 環境でアカウントのライフサイクルを管理する)
- Managing and monitoring API throttling in your workloads (ワークロードの API スロットリングの管理とモニタリング)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations で AWS Trusted Advisor の推奨事項を大規模に表示する)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower でサービス制限の緩和とエンタープライズサポートを自動化する)

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Service Quotas を使用して AWS のサービスのクォータを表示および管理する)
- AWS IAM Quotas Demo (AWS IAM のクォータのデモ)

関連ツール:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

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

複数のアカウントまたはリージョンをご利用の場合は、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。

期待される成果: 複数のアカウントまたはリージョンにまたがる設定、またはゾーン、リージョン、アカウントのフェイルオーバーを使用したレジリエンス設計になっている設定において、サービスクォータの枯渇によってサービスやアプリケーションが影響を受けないようにします。

一般的なアンチパターン:

- 1 つの分離リージョンでのリソース使用量の増加を許可するが、他のリージョンではキャパシティーを維持するメカニズムがない。
- 分離リージョン内のすべてのクォータを手動で設定する。
- 主要ではないリージョンの能力が低下している際に、将来のクォータの必要性について、回復力のあるアーキテクチャ (アクティブやパッシブなど) の影響を考慮していない。
- ワークロードが実行されているすべてのリージョンやアカウントにおいて、クォータを定期的に評価し、必要な変更を行っていない。
- 複数のリージョンやアカウントにまたがって緩和をリクエストする際に、クォータリクエストテンプレートを活用していない。
- クォータの緩和は、コンピューティング予約リクエストのように、コストに影響があるという誤った考えのもと、サービスクォータを更新していない。

このベストプラクティスを活用するメリット: リージョンでのサービスが利用不可になった際に、セカンダリリージョンやアカウントで現在の負荷を処理できることを検証します。これにより、リージョンを利用できない場合に発生するエラー数や機能低下のレベルを削減できます。

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

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。高レベルの回復性を維持するには、サービスクォータを継続的に評価する必要があります (自動または手動で)。

アクティブ/アクティブ、アクティブ/パッシブ - ホット、アクティブ/パッシブ - コールド、アクティブ/パッシブ - パイロットライトの各アプローチを使用した設計の実装により、複数のリージョンにまたがるワークロードが増えると、すべてのリージョンおよびアカウントのクォータレベルを把握することが不可欠になってきます。サービスクォータが正しく設定されている場合、過去のトラフィックパターンが常に適した指標になるとは限りません。

同じくらい重要なのが、サービスクォータ名の制限が各リージョンで常に同じとは限らないことです。あるリージョンでは値が 5 であり、別のリージョンでは値が 10 であることもありえます。負荷時の一貫した回復力を提要するために、このようなクォータの管理は、すべての同じサービス、アカウント、リージョンを網羅する必要があります。

異なるリージョン (アクティブリージョンまたはパッシブリージョン) 間の全てのサービスクォータの差異を調整し、これらの差異を継続的に調整するプロセスを作成します。パッシブリージョンのフェイルオーバーのテスト計画は、ピーク時のアクティブキャパシティまでスケールすることはめったにありません。つまり、ゲームデーや机上演習では、リージョン間のサービスクォータの差異を見つけられない場合があり、したがって適切な制限を維持できない場合があります。

サービスクォータのドリフトとは、特定の名前のクォータにおけるサービスクォータの制限が、すべてのリージョンではなく、1 つのリージョンで変更される状況であり、追跡と評価が非常に重要になります。トラフィックを伴う、またはトラフィックを伴う可能性があるリージョンでのクォータの変更を、考慮する必要があります。

- 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。これらの値を比較して差異を把握する必要があります。

実装手順

- 使用量のリスクレベルを超えている可能性がある Service Quotas の値をレビューします。AWS Trusted Advisor は 80% および 90% のしきい値で違反を警告します。
- パッシブリージョンのサービスクォータの値をレビューします (アクティブ/パッシブ設計の場合)。プライマリリージョンで障害があった場合に、負荷が正常にセカンダリリージョンで実行されることを検証します。
- サービスクォータのドリフトが同一アカウント内のリージョン間で発生し、それに伴って制限が変更された場合の評価を自動化します。
- お客様の組織単位 (OU) がサポートされている方法で構成されている場合、サービスクォータテンプレートを更新して、クォータの変更を反映し、複数のリージョンやアカウントに適用する必要があります。
  - テンプレートを作成して、リージョンをクォータの変更に関連付けます。
  - 既存のすべてのサービスクォータテンプレートをレビューして、変更が必要かどうかを確認します (リージョン、制限、アカウント)。

リソース

関連する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-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
- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL10-BP01 複数の場所にワークロードをデプロイする
- 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
- REL11-BP03 すべてのレイヤーの修復を自動化する
- 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

関連するドキュメント:

- AWS Well Architected フレームワークの信頼性の柱: 可用性
- 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
- クォータの引き上げのリクエスト方法
- サービスエンドポイントとクォータ
- Service Quotas ユーザーガイド
- AWS のクォータモニター
- AWS Fault Isolation Boundaries (AWS の障害分離境界)
- Availability with redundancy (冗長性を備えた可用性)
- データのための AWS

- 継続的インテグレーションとは
- 継続的デリバリーとは
- APN パートナー: 設定管理を支援できるパートナー
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS のテナント別アカウント SaaS 環境でアカウントのライフサイクルを管理する)
- Managing and monitoring API throttling in your workloads (ワークロードの API スロットリングの管理とモニタリング)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations で AWS Trusted Advisor の推奨事項を大規模に表示する)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower でサービス制限の緩和とエンタープライズサポートを自動化する)

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Service Quotas を使用して AWS のサービスのクォータを表示および管理する)
- AWS IAM Quotas Demo (AWS IAM のクォータのデモ)

関連サービス:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

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、サービスの制約、物理リソースの制限には変更できないものもあることに注意します。このような制限が信頼性に影響を及ぼさないように、アプリケーションとサービスのアーキテクチャを設計します。

例えば、ネットワーク帯域幅、サーバーレス関数呼び出しのペイロードサイズ、API Gateway のスロットルバーストレート、データベースへの同時ユーザー接続数などは変更できません。

期待される成果: アプリケーションやサービスは、通常のトラフィック状況および高トラフィック状況で期待されるとおりに動作します。アプリケーションやサービスは、リソースの固定制約またはサービスクォータの制限内で機能するように設計されています。

一般的なアンチパターン:

- 単一のサービスリソースを使用する設計を選択し、スケーリング時に障害が発生する原因となる設計上の制約があることを認識していない。
- テスト中にサービスの固定クォータに到達してしまう非現実的なベンチマークを採用している。(例: バースト制限を利用しながら長時間テストを実行する)
- 固定サービスクォータを超えてもスケーリングしたり変更したりできない設計を選択する。(例: 256KB のペイロードサイズの SQS)
- 高トラフィックイベント中に危険にさらされる可能性があるサービスクォータのしきい値をモニタリングしたり警告したりするために、オブザーバビリティが設計および実装されていない。

このベストプラクティスを活用するメリット: 予測されるあらゆるサービス負荷レベルで、アプリケーションが中断や低下することなく実行されることを確認できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

ソフトサービスクォータ、つまりより高い容量ユニットに置き換えられるリソースとは異なり、AWS サービスの固定クォータは変更できません。つまり、アプリケーションの設計で使用する場合、このようなタイプのすべての AWS サービスの容量のハード制限を評価する必要があります。

ハード制限は Service Quotas コンソールに表示されます。列に `ADJUSTABLE = No`と表示されていれば、そのサービスにはハード制限があります。ハード制限は、一部のリソース設定ページにも表示されます。例えば、Lambda には調整できない特定のハード制限があります。

例としては、Lambda 関数で実行する Python アプリケーションを設計する場合、Lambda が 15 分以上実行される可能性があるかを判断するためにアプリケーションを評価する必要があります。コードがこのサービスクォータ制限を超過して実行される可能性がある場合は、代替テクノロジーや設計を検討する必要があります。本番環境へのデプロイ後にこの制限に達すると、修正されるまでアプリケーションの機能が低下し、中断することになります。ソフトクォータとは異なり、このような制限は、重大度 1 の緊急事態が発生した場合でも、変更できません。

アプリケーションがテスト環境にデプロイされたら、ハード制限に到達できるかどうかを確認する戦略を行使する必要があります。ストレステスト、負荷テスト、カオステストは、導入テスト計画の一環とする必要があります。

実装手順

- 使用できる AWS サービスの完全なリストをアプリケーションの設計段階で確認します。
- これらすべてのサービスについて、ソフトクォータ制限とハードクォータ制限を確認します。すべての制限が Service Quotas コンソールに表示されているとは限りません。サービスによっては、このような制限について、別の場所で説明されています。
- アプリケーションを設計する際は、ビジネス上の成果、ユースケース、依存するシステム、可用性目標、ディザスタリカバリオブジェクトなど、ワークロードのビジネスとテクノロジーの推進要因を確認します。ビジネスとテクノロジーの推進要因に従って、ワークロードに適した分散システムを特定するプロセスを進めます。
- リージョン全体とアカウント全体にわたるサービス負荷を分析します。ハード制限の多くは、サービスのリージョンに基づいていますが、アカウントに基づく制限もあります。
- ゾーン障害時とリージョン障害時のリソース使用状況について、回復力あるアーキテクチャを分析します。「アクティブ/アクティブ」、「アクティブ/パッシブ – ホット」、「アクティブ/パッシブ – コールド」、「アクティブ/パッシブ – パイロットライト」のアプローチを使用するマルチリージョン設計を進めると、このような障害ケースはより高い使用状況につながり、ハード制限に達するユースケースを生み出す可能性が出てきます。

リソース

関連する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-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
- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL10-BP01 複数の場所にワークロードをデプロイする
- 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
- REL11-BP03 すべてのレイヤーの修復を自動化する
- 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

関連するドキュメント:

- AWS Well Architected フレームワークの信頼性の柱: 可用性
- 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
- クォータの引き上げのリクエスト方法
- サービスエンドポイントとクォータ
- Service Quotas ユーザーガイド
- AWS のクォータモニター
- AWS Fault Isolation Boundaries (AWS の障害分離境界)
- Availability with redundancy (冗長性を備えた可用性)
- データのための AWS
- 継続的インテグレーションとは
- 継続的デリバリーとは
- APN パートナー: 設定管理を支援できるパートナー
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS のテナント別アカウント SaaS 環境でアカウントのライフサイクルを管理する)
- Managing and monitoring API throttling in your workloads (ワークロードの API スロットリングの管理とモニタリング)

- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations で AWS Trusted Advisor の推奨事項を大規模に表示する)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower でサービス制限の緩和とエンタープライズサポートを自動化する)
- Actions, Resources, and Condition Keys for Service Quotas Services (AWS サービスのアクション、リソース、および条件キー)

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Service Quotas を使用して AWS のサービスのクォータを表示および管理する)
- AWS IAM Quotas Demo (AWS IAM のクォータのデモ)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: ループをクローズして柔軟に対応: サイズを問わず、システムを制御する方法)

関連ツール:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

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。

期待される成果: 管理およびモニタリングするアクティブで自動化されたシステムが導入されます。このような運用ソリューションを採用すると、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確認することができます。クォータの変更が要請されると、これらは積極的に修正されます。

一般的なアンチパターン:

- サービスクォータのしきい値をチェックするようにモニタリングを設定していない。
- ハード制限の値は変更できないにもかかわらず、モニタリングを設定していない。
- ソフトクォータの変更を要請して確保するのに必要な時間は即時または短期間であると想定している。
- サービスクォータに近づいた際のアラームは設定していても、アラートの対応方法に関するプロセスがない。
- AWS 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、その他の AWS サービスのモニタリングを行っていない。
- 「アクティブ/アクティブ」、「アクティブ/パッシブ – ホット」、「アクティブ/パッシブ – コールド」、「アクティブ/パッシブ – パイロットライト」のアプローチなど、複数リージョンの回復力ある設計のクォータ管理を考慮していない。
- リージョン間のクォータの違いを評価していない。
- 特定のクォータ引き上げ要請について、すべてのリージョンのニーズを評価していない。
- マルチリージョンクォータ管理向けのテンプレートを活用していない。

このベストプラクティスを活用するメリット: AWS Service Quotas を自動的に追跡し、これらのクォータに対する使用状況をモニタリングすることで、クォータ制限に近づいていることを確認できます。このモニタリングデータを使用して、クォータの枯渇による低下を制限することもできます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

サポートされているサービスについては、評価してアラートまたはアラームを送信できるさまざまなサービスを設定することで、クォータをモニタリングできます。これにより、使用状況のモニタリングがサポートされ、クォータに近づいていることのアラートが送信されます。このアラームは、AWS Config、Lambda 関数、Amazon CloudWatch、または AWS Trusted Advisor からトリガーできます。また、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して抽出し、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。

実装手順

モニタリング:

- 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。Amazon EC2 `DescribeInstances` などの API オペレーションを使用して、現在のリソース消費の情報を収集します。
- 以下を使用して、サービスに不可欠で適用できる現在のクォータをキャプチャします。
  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS ドキュメントを
  - AWS サービス固有のページ
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- AWS Service Quotas を使用します。これは、250 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。
- さまざまなしきい値で現在のサービス制限をモニタリングするには、Trusted Advisor サービス制限を使用します。
- リージョンの使用状況の増加をチェックするには、サービスクォータ履歴 (コンソールまたは AWS CLI) を使用します。
- 各リージョンと各アカウントのサービスクォータの変化を比較して、必要に応じて同等性を確立します。

管理:

- 自動化: AWS Config カスタムルールを設定し、リージョン間でサービスクォータをスキャンして、違いを比較します。
- 自動化: リージョン全体にわたるサービスクォータをスキャンするスケジュールされた Lambda 関数を設定して、違いを比較します。
- 手動: リージョン全体にわたるサービスクォータをスキャンするために、AWS CLI、API、または AWS コンソールを介してサービスクォータをスキャンして、違いを比較します。違いについてのレポートを作成します。
- リージョン間でクォータの違いが確認された場合は、必要に応じてクォータの変更を要請します。
- すべての変更の結果を確認します。

リソース

関連する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-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
- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL10-BP01 複数の場所にワークロードをデプロイする
- 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
- REL11-BP03 すべてのレイヤーの修復を自動化する
- 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

関連するドキュメント:

- AWS Well Architected フレームワークの信頼性の柱: 可用性
- 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
- クォータの引き上げのリクエスト方法
- サービスエンドポイントとクォータ
- Service Quotas ユーザーガイド
- AWS のクォータモニター
- AWS Fault Isolation Boundaries (AWS の障害分離境界)
- Availability with redundancy (冗長性を備えた可用性)
- データのための AWS

- 継続的インテグレーションとは
- 継続的デリバリーとは
- APN パートナー: 設定管理を支援できるパートナー
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS のテナント別アカウント SaaS 環境でアカウントのライフサイクルを管理する)
- Managing and monitoring API throttling in your workloads (ワークロードの API スロットリングの管理とモニタリング)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations で AWS Trusted Advisor の推奨事項を大規模に表示する)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower でサービス制限の緩和とエンタープライズサポートを自動化する)
- Actions, Resources, and Condition Keys for Service Quotas Services (AWS サービスのアクション、リソース、および条件キー)

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Service Quotas を使用して AWS のサービスのクォータを表示および管理する)
- AWS IAM Quotas Demo (AWS IAM のクォータのデモ)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: ループをクローズして柔軟に対応: サイズを問わず、システムを制御する方法)

関連ツール:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor

- AWS CDK
- AWS Systems Manager
- AWS Marketplace

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 サポート 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するか、アクセスできない場合、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。障害が発生したリソースまたはアクセスできないリソースと、代替のリソースの合計リソース数がクォータ内に収まることを確認します。このギャップを算出する際は、ネットワーク障害、アベイラビリティーゾーンの不具合、またはリージョン規模の不具合などのユースケースを考慮する必要があります。

期待される成果: 現在のサービスのしきい値を使用して、規模を問わず、リソースやリソースへのアクセスで障害に対応できます。リソースプランニングでは、ゾーン障害、ネットワーク障害、さらにはリージョン規模の不具合が考慮されています。

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。
- ピーク時のサービスクォータの計算時に静的安定性の原則を考慮しない。
- 各リージョンに求められる合計クォータを計算する際に、アクセスできないリソースの可能性を考慮しない。
- AWS サービス障害分離境界と、予測される異常な使用パターンを考慮していないサービスがある。

このベストプラクティスを活用するメリット: サービス中断イベントがアプリケーションの可用性に影響を与えるような場合、クラウドを使用すると、このようなイベントを軽減または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、またはアクセスできない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。このようなフェイルオーバーの状況に対応し、サービス制限の枯渇による追加の低下を発生させないようなクォータ戦略を策定します。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

クォータ制限を評価する際は、何らかの劣化が原因で発生する可能性のあるフェイルオーバーのケースを検討します。以下のタイプのフェイルオーバーのケースを検討する必要があります。

- 中断したり、アクセスできない VPC。
- アクセスできないサブネット。
- 多くのリソースへのアクセスに影響を及ぼすレベルまで低下したアベイラビリティーゾーン。
- まざまなネットワークルートまたは受信ポイントと送信ポイントがブロックされたり、変更されたりしている。
- 多くのリソースへのアクセスに影響を及ぼすレベルまで低下したリージョン。
- 複数のリソースがあるが、すべてのリソースがリージョンやアベイラビリティーゾーンの不具合の影響を受けているわけではない。

上記のリストのような障害は、フェイルオーバーイベントを開始するトリガーになる可能性があります。ビジネスへの影響は大きく異なる可能性があり、フェイルオーバーの決定は状況や顧客ごとに異なります。ただし、アプリケーションまたはサービスのフェイルオーバーという運用上の決定を下す場合、イベントが発生する前に、フェイルオーバーのロケーションでのリソースのキャパシティプランニングと、それに関連するクォータについて対処する必要があります。

発生する可能性がある通常のピーク時よりも高いピークを考慮に入れて、各サービスのクォータを確認します。このようなピークは、ネットワークまたはアクセス許可のために到達でき、まだアクティブなリソースに関連している可能性があります。終了していないアクティブなリソースは、サービスクォータ制限に対して引き続きカウントされます。

実装手順

- フェイルオーバーまたはアクセスできなくなった場合に対応するため、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。
- 必要に応じて、静的安定性 (ゾーンおよびリージョン) の計算を含めます。
- 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)。
- 最も重要なワークロードへの静的安定性の影響を考慮します。すべてのリージョンとアベイラビリティーゾーンで静的に安定性のあるシステムに相当するリソースを評価します。
- オンデマンドキャパシティ予約を使用して、フェイルオーバーが発生する前に容量をスケジュールすることを検討します。これは、フェイルオーバー発生時、最も重要なビジネススケジュール中に、適切な量および種類のリソースを取得するうえで予測できるリスクの軽減に役立つ戦略です。

リソース

関連する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
- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL10-BP01 複数の場所にワークロードをデプロイする
- 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
- REL11-BP03 すべてのレイヤーの修復を自動化する
- 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

関連するドキュメント:

- AWS Well Architected フレームワークの信頼性の柱: 可用性
- 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
- クォータの引き上げのリクエスト方法
- サービスエンドポイントとクォータ
- Service Quotas ユーザーガイド
- AWS のクォータモニター
- AWS Fault Isolation Boundaries (AWS の障害分離境界)
- Availability with redundancy (冗長性を備えた可用性)
- データのための AWS
- 継続的インテグレーションとは
- 継続的デリバリーとは
- APN パートナー: 設定管理を支援できるパートナー

- Managing the account lifecycle in account-per-tenant SaaS environments on AWS(AWS のテナント別アカウント SaaS 環境でアカウントのライフサイクルを管理する)
- Managing and monitoring API throttling in your workloads (ワークロードの API スロットリングの管理とモニタリング)
- View AWS Trusted Advisor recommendations at scale with AWS Organizations(AWS Organizations で AWS Trusted Advisor の推奨事項を大規模に表示する)
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower(AWS Control Tower でサービス制限の緩和とエンタープライズサポートを自動化する)
- Actions, Resources, and Condition Keys for Service Quotas Services (AWS サービスのアクション、リソース、および条件キー)

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas (Service Quotas を使用して AWS のサービスのクォータを表示および管理する)
- AWS IAM Quotas Demo (AWS IAM のクォータのデモ)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: ループをクローズして柔軟に対応: サイズを問わず、システムを制御する方法)

関連ツール:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

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

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構築すると、接続の喪失によるダウンタイムを低減し、ワークロードの可用性と SLA を向上できます。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、API ゲートウェイ、負荷分散、またはリバースプロキシを使用します。

期待される成果: パブリックエンドポイントの高可用性ネットワーク接続を計画、構築、運用化することが重要です。接続が失われたためにワークロードにアクセスできなくなった場合、ワークロードが実行中で利用可能であっても、顧客はシステムがダウンしていると見なします。ワークロードのパブリックエンドポイントに対する可用性と回復力に優れたネットワーク接続と、ワークロード自体の回復力のあるアーキテクチャを組み合わせることで、可能な限り最高レベルの可用性とサービスレベルを顧客に提供できます。

AWS Global Accelerator、Amazon CloudFront、Amazon API Gateway、AWS Lambda 関数 URL、AWS AppSync API、Elastic Load Balancing (ELB) はすべて、高可用性のパブリックエンドポイントを提供します。Amazon Route 53 は、ドメイン名解決のための高可用性 DNS サービスを提供し、パブリックエンドポイントアドレスを解決できることを確認できます。

また、ロードバランシングとプロキシ処理のために、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。

一般的なアンチパターン:

- 高可用性に向けて DNS とネットワーク接続を計画せず、高可用性ワークロードを設計する。
- 個別の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。
- サービスの検索に、ドメイン名ではなく、IP アドレスを使用する。
- パブリックエンドポイントへの接続が失われるシナリオをテストしていない。
- ネットワークスループットのニーズと分散パターンを分析していない。
- ワークロードのパブリックエンドポイントへのインターネットにおけるネットワーク接続が中断される可能性を考慮したシナリオをテストおよび計画していない。
- 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、またはメディアファイル) を提供し、コンテンツ配信ネットワークを使用しない。
- 分散サービス拒否 (DDoS) 攻撃に備えた計画をしていない。DDoS 攻撃は、正当なトラフィックを遮断し、ユーザーの可用性を低下させ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: 可用性と回復力に優れたネットワーク接続を設計することで、ユーザーはワークロードにアクセスして利用できます。

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

パブリックエンドポイントへの高可用性ネットワーク接続を構築する際の中核となるのが、トラフィックのルーティングです。トラフィックがエンドポイントに到達できることを確認するには、ドメイン名を DNS が対応する IP アドレスに解決することができる必要があります。ドメインの DNS レコードを管理するには、Amazon Route 53 などの高可用性かつスケーラブルなドメインネームシステム (DNS) を使用します。Amazon Route 53 が提供するヘルスチェックも利用できます。ヘルスチェックは、アプリケーションが到達可能かつ利用可能で、機能していることを確認します。ヘルスチェックは、ウェブページや特定の URL を要求するなどのユーザーの動作を模倣するように設定できます。障害が発生した場合、Amazon Route 53 は DNS 解決リクエストに応答し、トラフィックを正常なエンドポイントのみに転送します。Amazon Route 53 が提供する位置情報 DNS およびレイテンシーベースルーティング機能の使用を検討することもできます。

ワークロード自体の可用性が高いことを確認するには、Elastic Load Balancing (ELB) を使用します。Amazon Route 53 は、トラフィックを ELB にターゲットとするために使用できます。ELB は、このトラフィックを、ターゲットのコンピューティングインスタンスに分散します。サーバーレスソリューションには、Amazon API Gateway と AWS Lambda を組み合わせて使用できます。また、複

数の AWS リージョン でワークロードを実行することもできます。マルチサイトのアクティブ/アクティブパターンを使用すると、ワークロードは複数のリージョンからのトラフィックを処理できます。マルチサイトのアクティブ/パッシブパターンの場合、ワークロードはアクティブリージョンからのトラフィックを処理し、データはセカンダリリージョンにレプリケートされ、プライマリリージョンで障害が発生した場合にアクティブになります。その後、Route 53 のヘルスチェックを使用して、プライマリリージョンの任意のエンドポイントからセカンダリリージョンのエンドポイントへの DNS フェイルオーバーを制御して、ワークロードが到達可能であり、ユーザーが利用できることを確認することができます。

Amazon CloudFront は、世界中のエッジロケーションのネットワークを使用してリクエストを処理することにより、低レイテンシーかつ高データ転送速度でコンテンツを配信する、シンプルな API を提供します。コンテンツ配信ネットワークは、ユーザーの所在地に近いロケーションにあるか、近いロケーションでキャッシュされているコンテンツを提供することで、顧客にサービスを提供します。これにより、コンテンツの負荷がサーバーから CloudFront のエッジロケーションへと移動するため、アプリケーションの可用性も向上します。エッジロケーションとリージョンのエッジキャッシュは、視聴者の近くにコンテンツのキャッシュコピーを保持するため、ワークロードの取得が迅速化し、到達可能性と可用性が向上します。

ユーザーが地理的に分散しているワークロードの場合、AWS Global Accelerator を使用すると、アプリケーションの可用性とパフォーマンスを向上できます。AWS Global Accelerator は、1 つまたは複数の AWS リージョン でホストされている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エニーキャスト静的 IP アドレスを提供します。これにより、トラフィックがユーザーにできるだけ近い AWS グローバルネットワークに入ることができ、ワークロードの到達可能性と可用性が向上します。AWS Global Accelerator を使用すると、TCP、HTTP、および HTTPS ヘルスチェックを使用して、アプリケーションエンドポイントの健全性もモニタリングできます。エンドポイントの正常性または設定に変更が生じると、正常なエンドポイントへのユーザートラフィックのリダイレクトがトリガーされ、最高のパフォーマンスと可用性がユーザーに提供されます。さらに、AWS Global Accelerator は障害を分離するように設計されており、独立したネットワークゾーンによって提供される 2 つの静的 IPv4 アドレスを使用して、アプリケーションの可用性を向上します。

DDoS 攻撃からの保護として、AWS は、AWS Shield Standard を提供しています。Shield Standard は自動的に有効にされており、SYN/UDP フラッド攻撃やリフレクション攻撃などの一般的なインフラストラクチャ (レイヤー 3 および 4) 攻撃から保護し、AWS 上のアプリケーションの高可用性をサポートします。より高度で大規模な攻撃 (UDP フラッド攻撃など)、State-Exhaustion 攻撃 (TCP SYN フラッドなど) に対する追加の保護、および Amazon Elastic Compute Cloud (Amazon EC2)、Elastic Load Balancing (ELB)、Amazon CloudFront、AWS Global Accelerator、Route 53 上で実行されるアプリケーションの保護には、AWS Shield Advanced の使用を検討できます。HTTP POST や GET フラッド攻撃などのアプリケーションレイヤー攻撃からの保護には、AWS WAF を

使用します。AWS WAF を使用すると、IP アドレス、HTTP ヘッダー、HTTP ボディ、URI 文字列、SQL インジェクション、クロスサイトスクリプティング条件を使用して、リクエストをブロックするか許可するかを決定できます。

実装手順

1. 高可用性の DNS の設定: Amazon Route 53 は、高可用性かつスケーラブルなドメインネームシステム (DNS) ウェブサービスです。Route 53 は、AWS またはオンプレミスで実行されるインターネットアプリケーションにユーザーリクエストを接続します。詳細については、「DNS サービスとしての Amazon Route 53 の設定」を参照してください。
2. ヘルスチェックの設定: Route 53 を使用する場合、正常なターゲットのみが解決可能であることを確認します。Route 53 ヘルスチェックの作成と DNS フェイルオーバーの設定から始めます。ヘルスチェックを設定する際には、以下を考慮することが重要です。
   a. Amazon Route 53 でヘルスチェックの正常性を判断する方法
   b. ヘルスチェックの作成、更新、削除
   c. ヘルスチェックステータスのモニタリングと通知の受信
   d. Amazon Route 53 DNS のベストプラクティス
3. DNS サービスをエンドポイントに接続します。
   a. Elastic Load Balancing をトラフィックのターゲットとして使用する場合、Amazon Route 53 を使用してロードバランサーのリージョンエンドポイントを指すエイリアスレコードを作成します。エイリアスレコードの作成中に、[Evaluate Target Health (ターゲットの正常性の評価)] オプションを [Yes (あり)] に設定します。
   b. サーバーレスワークロードまたはプライベート API については、API Gateway を使用する場合、Route 53 を使用してトラフィックを API Gateway にルーティングします。
4. コンテンツ配信ネットワークを決定します。
   a. ユーザーに近いエッジロケーションを使用してコンテンツを配信するには、CloudFront がコンテンツを配信する方法を理解することから始めます。
   b. 簡単な CloudFront ディストリビューションの使用から始めます。これにより、CloudFront は、コンテンツの配信元と、コンテンツ配信の追跡および管理方法に関する詳細を認識できます。CloudFront のディストリビューションを設定する際には、以下の側面を理解して、考慮することが重要です。
      i. CloudFront エッジロケーションでのキャッシュの仕組み
      ii. CloudFront キャッシュから直接提供されるリクエストの比率 (キャッシュヒット率) の向上
      iii. Amazon CloudFront Origin Shield の使用

iv. CloudFront オリジンフェイルオーバーによる高可用性の最適化

5. アプリケーションレイヤー保護の設定: AWS WAF は、可用性低下、セキュリティの侵害、リソースの過剰消費といった、一般的なウェブのエクスプロイトやボットから保護します。詳細を理解するには、「AWS WAF の仕組み」を確認し、アプリケーションレイヤーの HTTP POST および GET フラッド攻撃からの保護を実装する準備が整ったら、「AWS WAF の開始方法」を確認してください。AWS WAF は CloudFront と組み合わせて使用することもできます。ドキュメントについては、「AWS WAF と Amazon CloudFront の機能との連携」を参照してください。
6. 追加の DDoS 保護の設定: デフォルトでは、すべての AWS のお客様が追加料金なしで、ウェブサイトやアプリケーションをターゲットする一般的で最も頻繁に発生するネットワーク層およびトランスポート層の DDoS 攻撃に対する AWS Shield Standard を使用した保護を受けています。Amazon EC2、Elastic Load Balancing、Amazon CloudFront、AWS Global Accelerator、Amazon Route 53 で実行されているインターネット接続アプリケーションの保護を強化するには、AWS Shield Advanced を検討し、「DDoS に対する回復力が高いアーキテクチャの例」を確認してください。ワークロードとパブリックエンドポイントを DDoS 攻撃から保護するには、「AWS Shield Advanced の開始方法」を確認してください。

リソース

関連する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
- 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-BP06 イベントが可用性に影響する場合に通知を送信する

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 Marketplace for Network Infrastructure (ネットワークインフラストラクチャ向け AWS Marketplace)
- What Is AWS Global Accelerator? (AWS Global Accelerator とは)
- Amazon CloudFront とは何ですか?
- Amazon Route 53 とは ?
- Elastic Load Balancing とは?

- Network Connectivity capability - Establishing Your Cloud Foundations (ネットワーク接続機能 - クラウド基盤の確立)
- Amazon API Gateway とは何ですか?
- AWS WAF、AWS Shield、AWS Firewall Manager とは
- What is Amazon Route 53 Application Recovery Controller? (Amazon Route 53 Application Recovery Controller とは)
- Amazon Route 53 ヘルスチェックの作成と DNS フェイルオーバーの設定

関連動画:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator(AWS re:Invent 2022 - AWS Global Accelerator を使用してパフォーマンスと可用性を向上する)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53 (AWS re:Invent 2020: Amazon Route 53 を使用したグローバルトラフィック管理)
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications (AWS re:Invent 2022 - 高可用性のマルチ AZ アプリケーションの運用)
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure (AWS re:Invent 2022 - AWS ネットワークインフラストラクチャの詳細)
- AWS re:Invent 2022 - Building resilient networks (AWS re:Invent 2022 - 回復力のあるネットワークの構築)

関連する例:

- Disaster Recovery with Amazon Route 53 Application Recovery Controller (ARC) (Amazon Route 53 Application Recovery Controller (ARC) を使用したディザスタリカバリ)
- Reliability Workshops (信頼性ワークショップ)
- AWS Global Accelerator ワークショップ

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。

Site-to-Site 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。

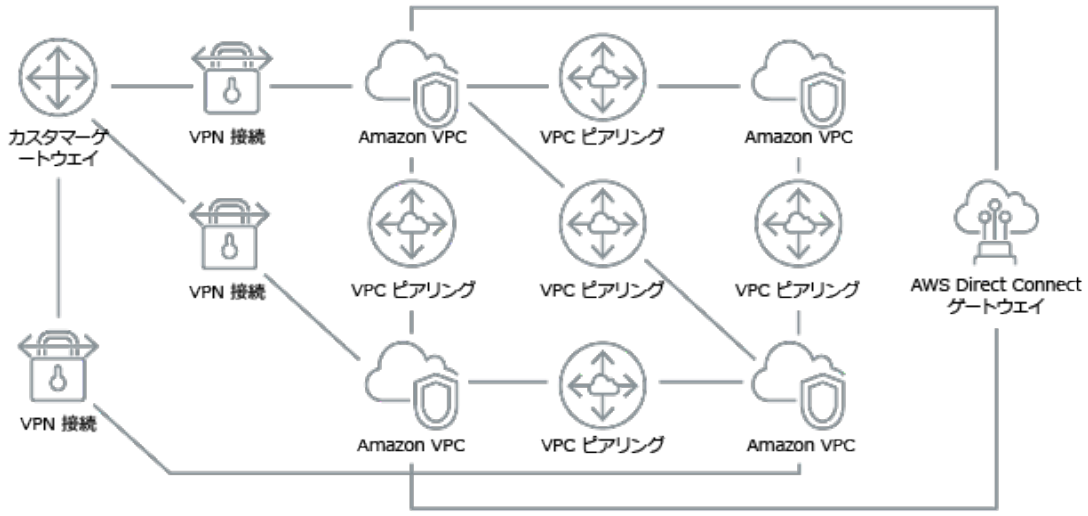


図 1: AWS Transit Gateway なしの場合: VPN 接続を使用して、各 Amazon VPC を相互にピア接続し、オンサイトの各ロケーションをピア接続する必要があります。これは、拡張するにつれ複雑さが増す可能性があります。

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。

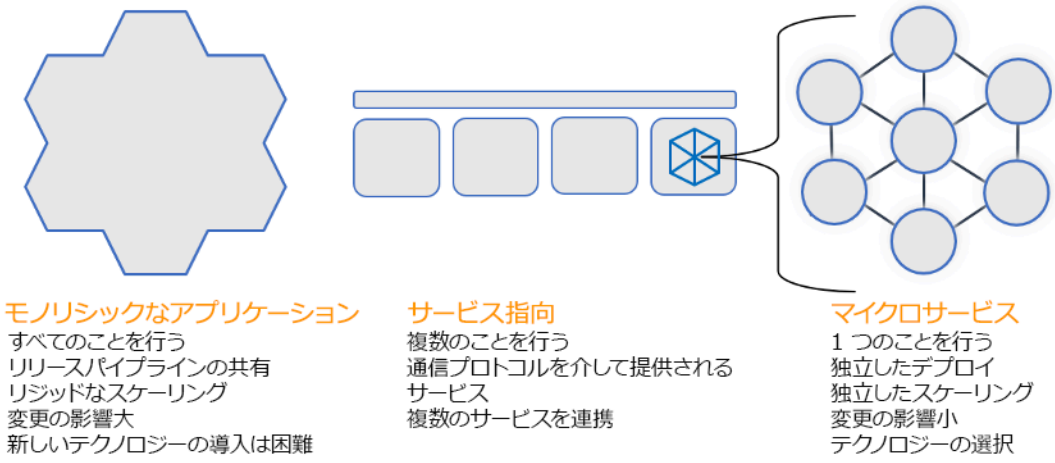
- アプリケーションコンポーネントには異なる可用性要件があり、より特化したセグメント化によってサポートされることがある。
- ワークロードをサポートするチームの責任が明確に定義される。

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定義します。マイクロサービスは、ドメインモデルと制限付きコンテキストを使用して、ビジネスコンテキストの境界に沿ってサービスの境界を描きます。ビジネスドメインと機能に重点を置くことで、チームがサービスの独立した信頼性要件を定義しやすくなります。コンテキストに制限があると、ビジネスロジックが分離されてカプセル化されるため、チームは障害の処理方法についてより的確に判断できるようになります。

期待される成果: エンジニアとビジネス関係者は共同で境界のあるコンテキストを定義し、それを使用して特定のビジネス機能を果たすサービスとしてシステムを設計します。これらのチームは、イベントストーミングなどの確立された手法を使用して要件を定義します。新しいアプリケーションは、境界を明確に定義し、ゆるく結合するサービスとして設計されています。既存のモノリスは境界コンテキストに分解され、システム設計は SOA またはマイクロサービスアーキテクチャに移行します。モノリスをリファクタリングする際には、バブルコンテキストやモノリスの分解パターンなどの確立されたアプローチが適用されます。

ドメイン指向のサービスは、状態を共有しない 1 つ以上のプロセスとして実行されます。需要の変動に独自に対応し、ドメイン固有の要件に照らして障害シナリオを処理します。

一般的なアンチパターン:

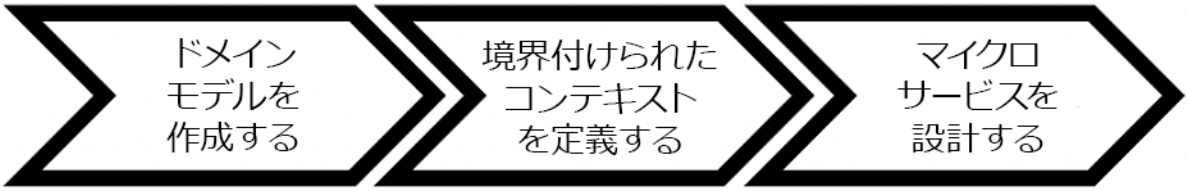
- チームは、特定のビジネスドメインではなく、UI や UX、ミドルウェア、データベースなどの特定の技術ドメインを中心に形成されます。
- アプリケーションはドメインの担当範囲にまたがります。限定されたコンテキストにまたがるサービスは、メンテナンスが難しく、大規模なテスト作業が必要になり、複数のドメインチームがソフトウェアアップデートに参加する必要がある場合があります。
- ドメインエンティティライブラリと同様に、ドメイン依存関係はサービス間で共有されるため、あるサービスドメインを変更すると、他のサービスドメインも変更する必要があります。
- サービス契約とビジネスロジックは、エンティティを共通で一貫性のあるドメイン言語で表現しないため、翻訳層が複雑になり、デバッグ作業が増えます。

このベストプラクティスを活用するメリット: アプリケーションは、ビジネスドメインによって区切られた独立したサービスとして設計され、共通のビジネス言語を使用します。サービスは独立してテストおよびデプロイできます。サービスは、実装されたドメインのドメイン固有の耐障害性要件を満たします。

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

ドメイン主導型意思決定 (DDD) は、ビジネスドメインを中心にソフトウェアを設計および構築するための基本的なアプローチです。ビジネスドメインに焦点を当てたサービスを構築する際には、既存のフレームワークを使用すると便利です。既存のモノリシックアプリケーションを扱う場合は、確立された手法を提供する分解パターンを利用して、アプリケーションをモダナイズしてサービスにすることができます。



ドメイン主導の意思決定

実装手順

- チームは イベントストーミング ワークショップを開催して、軽量の付箋形式でイベント、コマンド、集計、ドメインをすばやく特定できます。
- ドメインのエンティティと機能をドメインコンテキストで形成したら、以下のようにドメインをサービスに分割できます。 境界コンテキスト、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。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。
  - 例えば、Amazon.com ウェブサイトエンティティには、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。
  - パッケージ、配送、スケジュールは出荷コンテキストにグループ化され、料金、割引、通貨は料金設定コンテキストにグループ化されます。
- モノリスのマイクロサービスへの分解 マイクロサービスをリファクタリングするためのパターンを概説します。ビジネス機能、サブドメイン、またはトランザクション別に分解するパターンを使用することは、ドメイン主導のアプローチによく合います。
- バブルコンテキストなどの 戦術的なテクニックを使用すると、 事前に書き直したり、DDD に全面的にコミットしたりすることなく、既存またはレガシーアプリケーションに DDD を導入できます。バブルコンテキストアプローチでは、サービスマッピングと調整、または 破損対策レイヤーを使用して小さな境界コンテキストが確立され、新しく定義されたドメインモデルを外部の影響から保護します。

チームがドメイン分析を行い、エンティティとサービス契約を定義したら、AWS のサービスを活用してドメイン主導型の設計をクラウドベースのサービスとして実装できます。

- ドメインのビジネスルールを実践するテストを定義することから開発を始めましょう。テスト駆動開発 (TDD) と行動主導開発 (BDD) は、チームがサービスをビジネス上の問題の解決に集中させるのに役立ちます。
- ビジネスドメインの要件とマイクロサービスアーキテクチャに最適な AWS のサービスは を 選択する:
  - AWS サーバーレス を使用すると、チームはサーバーやインフラストラクチャを管理するのではなく、特定のドメインロジックに集中できます。
  - AWS でのコンテナ インフラストラクチャの管理が簡素化されるため、ドメイン要件に集中できます。
  - 目的別データベース は、ドメイン要件を最適なデータベースタイプに一致させるのに役立ちます。

- AWS 上でヘキサゴナルアーキテクチャを構築すると、 ビジネスドメインから逆算してサービスにビジネスロジックを構築して機能要件を満たし、統合アダプターを接続するフレームワークの概要が示されます。インターフェイスの詳細と AWS のサービスのビジネスロジックを分離するパターンは、チームがドメインの機能に集中し、ソフトウェアの品質を向上させるのに役立ちます。

リソース

関連する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-BP03 API ごとにサービス契約を提供する

関連するドキュメント:

- AWS マイクロサービス
- AWS でのマイクロサービスの実装
- 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
- “Domain-Driven Design: Tackling Complexity in the Heart of Software”
- AWS 上でヘキサゴナルアーキテクチャを構築すると、
- モノリスのマイクロサービスへの分解
- イベントストーミング
- 制限されたコンテキスト間のメッセージ
- マイクロサービス
- テスト駆動型開発
- 行動主導型開発

関連サンプル:

- エンタープライズクラウドネイティブワークショップ
- AWS でのクラウドネイティブマイクロサービスの設計 (DDD/EventStormingWorkshop より)

関連ツール:

- AWS クラウド データベース
- AWS でのサーバーレス
- AWS でのコンテナ

REL03-BP03 API ごとにサービス契約を提供する

サービス契約とは、機械が読み取れる API 定義で定義された、API プロデューサーとコンシューマー間の文書化された契約です。契約バージョニング戦略により、コンシューマーは既存の 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。プロデューサーのデプロイは、契約がある限りいつでも行うことができます。サービスチームは、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。

期待される成果:

一般的なアンチパターン: サービス指向またはマイクロサービスアーキテクチャで構築されたアプリケーションは、ランタイム依存関係を統合しながら独立して動作できます。API コンシューマーまたはプロデューサーに変更をデプロイしても、双方が共通の API 契約に従っていれば、システム全体の安定性が損なわれることはありません。サービス API を介して通信するコンポーネントは、相互にほとんどまたはまったく影響を与えずに、独立した機能リリース、ランタイム依存関係のアップグレード、またはディザスタリカバリ (DR) サイトへのフェイルオーバーを実行できます。さらに、ディスクリートサービスでは、他のサービスを一斉にスケーリングしなくても、吸収するリソース需要を個別にスケーリングできます。

- 厳密に型指定されたスキーマを使用しないサービス API を作成します。その結果、API バインディングの生成に使用できない API や、プログラムで検証できないペイロードが生成されます。
- API の利用者に更新とリリースを強いるようなバージョニング戦略を採用していない。そうしないと、サービス契約が発展したときに失敗する。
- ドメインコンテキストや言語での統合の失敗を説明するのではなく、基盤となるサービス実装の詳細を漏らすエラーメッセージ。
- サービスコンポーネントを個別にテストできるようにするために、API 契約を使用せずにテストケースを開発したりモック API 実装を使用したりします。

このベストプラクティスを活用するメリット: API サービス契約を介して通信するコンポーネントで構成される分散型システムでは、信頼性を向上させることができます。デベロッパーは、コンパイル中にタイプチェックを行って、リクエストとレスポンスが API 契約に従っていることと、必須フィールドが存在することを確認することで、開発プロセスの早い段階で潜在的な問題を発見できま

す。API 契約は、API 用のわかりやすい自己文書化インターフェイスを提供し、さまざまなシステムやプログラミング言語間の相互運用性を向上させます。

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: 中

実装のガイダンス

ビジネスドメインを特定し、ワークロードのセグメント化を決定したら、サービス API を開発できます。まず、機械が読み取れる API のサービス契約を定義し、次に API バージョニング戦略を実装します。REST、GraphQL、非同期イベントなどの一般的なプロトコルでサービスを統合する準備ができたら、AWSのサービスをアーキテクチャに組み込んで、コンポーネントを厳密に型指定された API 契約と統合できます。

サービス API 契約の AWS のサービス

以下を含む AWS のサービスを Amazon API Gateway、AWS AppSync、 Amazon EventBridge アプリケーションで API サービス契約を使用するようにアーキテクチャに組み込むことができます。Amazon API Gateway は、ネイティブ AWS サービスや他の Web サービスと直接統合するのに役立ちます。API Gateway は、 OpenAPI 仕様 とバージョニングをサポートしています。AWS AppSync は、クエリ、ミューテーション、およびサブスクリプションのサービスインターフェイスを定義する GraphQL スキーマを定義することによって構成するマネージド GraphQL エンドポイントですAmazon EventBridge はイベントスキーマを使用してイベントを定義し、イベントのコードバインディングを生成します。

実装手順

- まず、API の契約を定義します。契約では、API の機能を説明するだけでなく、API の入出力用に厳密に型指定されたデータオブジェクトとフィールドを定義します。
- API Gateway で API を設定すると、エンドポイントの OpenAPI 仕様をインポートおよびエクスポートできます。
  - OpenAPI 定義をインポートする API の作成が簡単になり、次のようなコードツールとしてAWS インフラストラクチャと統合できます AWS Serverless Application Model および AWS Cloud Development Kit (AWS CDK).
  - API 定義をエクスポートすると、 API テストツールとの統合が簡素化され、サービス利用者に統合仕様が提供されます。
- GraphQL API は次の方法で AWS AppSync を使用して定義および管理できます。 GraphQL スキーマファイルを定義して 契約インターフェイスを生成し、複雑な REST モデル、複数のデータベーステーブル、またはレガシーサービスとのやり取りを簡素化します。

- AWS Amplify は、AWS AppSync と統合されたプロジェクトで、アプリケーションで使用するための厳密に型指定された JavaScript クエリファイルと、AWS AppSync GraphQL クライアントライブラリを Amazon DynamoDB テーブル用に生成します。
- Amazon EventBridge からサービスイベントを利用する場合、イベントはスキーマレジストリに既に存在するスキーマや OpenAPI 仕様で定義したスキーマに従います。レジストリでスキーマを定義すると、スキーマ契約からクライアントバインディングを生成して、コードをイベントと統合することもできます。
- API の拡張またはバージョニング。オプションフィールドまたは必須フィールドのデフォルト値で構成できるフィールドを追加する場合、API を拡張する方が簡単なオプションです。
  - REST や GraphQL などのプロトコルの JSON ベースの契約は、契約の拡張に適しています。
  - SOAP のようなプロトコルの XML ベースの契約をサービスコンシューマーとテストして、契約延長の可能性を判断する必要があります。
- API をバージョニングするときは、ロジックを単一のコードベースで管理できるように、ファサードを使用してバージョンをサポートするプロキシバージョニングの実装を検討してください。
  - API Gatewayを使用すると、 リクエストとレスポンスのマッピングを使用して、 新しいフィールドにデフォルト値を提供したり、リクエストまたはレスポンスから削除されたフィールドを削除するファサードを確立したりすることで、契約の変更の吸収を簡素化できます。この方法では、基盤となるサービスは単一のコードベースを維持できます。

リソース

関連する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
- REL04-BP02 疎結合の依存関係を実装する
- REL05-BP03 再試行呼び出しを制御および制限する
- REL05-BP05 クライアントタイムアウトを設定する

関連するドキュメント:

- API (アプリケーションプログラミングインターフェイス) とは。
- 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
- OpenAPI の API Gateway 拡張機能の使用
- OpenAPI 仕様
- GraphQL: スキーマとタイプ
- Amazon EventBridge コードバインディング

関連サンプル:

- Amazon API Gateway: OpenAPI を使用した REST API の設定
- OpenAPI を使用した Amazon API Gateway から Amazon DynamoDB への CRUD アプリケーション
- サーバーレス時代のモダンアプリケーション統合パターン: API Gateway サービス統合
- Amazon CloudFront を使用したヘッダーベースの API Gateway バージョニングの実装
- AWS AppSync: クライアントアプリケーションの構築

関連動画:

- AWS SAM で OpenAPI を使用して API Gateway を管理する

関連ツール:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

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。

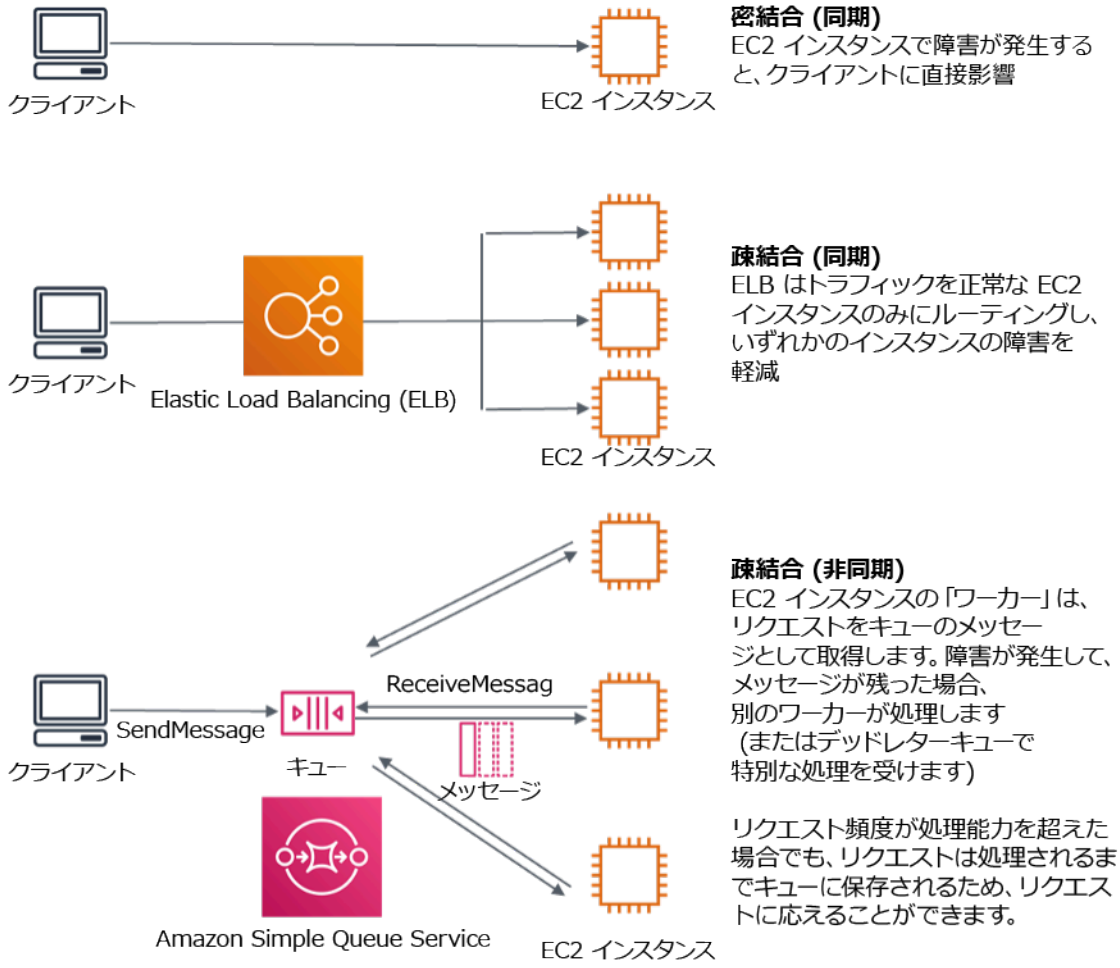


図 4: キューイングシステムやロードバランサーなどの依存関係が疎結合されています。

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

アプリケーションコンポーネントは、依存関係が使用できなくなっても、引き続きコア機能を実行する必要があります。少し古いデータ、代替データ、またはまったくデータを提供していない可能性があります。これにより、局所的な障害によるシステム全体の機能への影響を最小限に抑えながら、中心的なビジネス価値を実現できます。

期待される成果: 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。コンポーネントの故障モードは通常の動作と見なしてください。ワークフローは、このような障害が完全な障害につながらないように、あるいは少なくとも予測可能で回復可能な状態になるように設計する必要があります。

一般的なアンチパターン:

- 必要な中核的なビジネス機能が特定されていない。依存関係に障害が発生してもコンポーネントが機能することをテストしていません。
- エラーに関するデータを提供しない場合や、複数の依存関係のうち 1 つしか使用できず、結果の一部が返される場合もあります。
- トランザクションが部分的に失敗すると、一貫性のない状態になる。
- 中央パラメータストアにアクセスする代替手段がない。
- 更新に失敗した結果、その結果を考慮せずにローカルステートを無効化または空にする。

このベストプラクティスを活用するメリット: グレースフルデグラデーションを行うと、システム全体の可用性が向上し、障害が発生しても最も重要な機能の機能が維持されます。

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: 高

実装のガイダンス

グレースフルデグラデーションを実装することで、依存関係の障害がコンポーネントの機能に与える影響を最小限に抑えることができます。コンポーネントが依存関係の障害を検出し、他のコンポーネントやカスタマーへの影響を最小限に抑える方法で回避するのが理想的です。

グレースフルデグラデーションを考慮した設計とは、依存関係の設計時に潜在的な障害モードを考慮することを意味します。障害モードごとに、コンポーネントのほとんどの機能、または少なくとも最も重要な機能を発信者または顧客に提供する方法を用意してください。これらの考慮事項は、テストや検証が必要な追加要件になる可能性があります。理想的には、1 つまたは複数の依存関係に障害が発生した場合でも、コンポーネントがコア機能を許容範囲内で実行できることが理想的です。

これは技術的な議論であると同時にビジネス上の議論でもあります。すべてのビジネス要件は重要であり、可能であれば満たす必要があります。ただし、すべてが満たされない場合に何が起こるかをたずねることは依然として理にかなっています。システムは可用性と一貫性を保つように設計できますが、1 つの要件を削除しなければならない状況では、どちらの要件がより重要でしょうか。 支払い処理については、一貫性があるかもしれません。リアルタイムアプリケーションの場合、可用性が高

くなる可能性があります。カスタマー向けウェブサイトの場合、答えはカスタマーの期待するものによって異なる場合があります。

これが何を意味するかは、コンポーネントの要件と、そのコア機能と見なすべき内容によって異なります。以下はその例です。

- e コマースウェブサイトでは、パーソナライズされたレコメンデーション、最高ランクの商品、カスタマーの注文状況など、複数の異なるシステムからのデータがランディングページに表示される場合があります。上流システムの 1 つに障害が発生した場合でも、エラーページをカスタマーに表示するのではなく、他のシステムすべてを表示する方が理にかなっています。
- バッチ書き込みを実行するコンポーネントは、個々の操作のいずれかが失敗した場合でも、バッチの処理を続行できます。再試行メカニズムを実装するのは簡単なはずです。これは、どの操作が成功し、どの操作が失敗したか、なぜ失敗したかについての情報を呼び出し元に返すか、失敗したリクエストをデッドレターキューに入れて非同期再試行を実装することで実現できます。失敗した操作に関する情報も記録する必要があります。
- トランザクションを処理するシステムは、個々の更新がすべて実行されたか、まったく実行されないかを確認する必要があります。分散トランザクションでは、同じトランザクションの後の操作が失敗した場合に備えて、Saga パターンを使用して以前の操作をロールバックできます。ここでの中心的な機能は一貫性を維持することです。
- タイムクリティカルなシステムは、タイムリーに応答しない依存関係に対処できるはずです。このような場合は、サーキットブレーカーパターンを使用できます。依存関係からの応答がタイムアウトし始めると、システムは追加の呼び出しが行われないクローズ状態に切り替えることができます。
- アプリケーションはパラメータストアからパラメータを読み取ることができます。デフォルトのパラメータセットを使用してコンテナイメージを作成し、パラメータストアが利用できない場合にこれらを使用すると便利です。

なお、コンポーネントに障害が発生した場合の経路は検査が必要で、主要経路よりも大幅に簡潔でなければなりません。一般的に、フォールバック戦略は避けるべきです。

実装手順

外部依存関係と内部依存関係を特定します。どのような種類の障害が発生する可能性があるかを検討してください。障害発生時に上流と下流のシステムやカスタマーへの悪影響を最小限に抑える方法を考えてください。

依存関係の一覧と、失敗した場合に正常にデグレードする方法は次のとおりです。

1. 依存関係の部分的な障害: コンポーネントは、1 つのシステムへの複数の要求、または複数のシステムへの 1 つの要求のいずれかとして、下流システムに対して複数の要求を行うことができます。ビジネスの状況によっては、これに対するさまざまな処理方法が適切な場合があります (詳細については、実装ガイダンスの前述の例を参照してください)。
2. ダウンストリームシステムは、負荷が高いため、リクエストを処理できません。 ダウンストリームシステムへのリクエストが絶えず失敗している場合、再試行を続行しても意味がありません。これにより、既に過負荷になっているシステムに追加の負荷がかかり、回復が困難になる可能性があります。ここでは、ダウンストリームシステムへのコールの失敗を監視するサーキットブレーカーパターンを利用できます。大量のコールが失敗すると、ダウンストリームシステムへのリクエストの送信が停止され、ダウンストリームシステムが再び使用可能かどうかをテストするコールがたまにしか送信されません。
3. パラメータストアは使用できません: パラメータストアを変換するには、ソフト依存関係キャッシュを使用するか、コンテナイメージやマシンイメージに含まれる適切なデフォルトを使用できます。これらのデフォルトは最新の状態に保ち、テストスイートに含める必要があることに注意してください。
4. モニタリングサービスやその他の機能しない依存サービスは利用できません。 コンポーネントが断続的にログ、メトリクス、またはトレースを中央監視サービスに送信できない場合でも、通常どおりビジネス機能を実行するのが最善策です。メトリクスを長時間ログに記録したりプッシュしたりしないことは、ほとんどの場合受け入れられません。また、ユースケースによっては、コンプライアンス要件を満たすために完全な監査エントリが必要になる場合があります。
5. リレーショナルデータベースのプライマリインスタンスが使用できない可能性があります。Amazon Relational Database Service は、ほとんどすべてのリレーショナルデータベースと同様に、プライマリライターインスタンスは 1 つしか設定できません。これにより、書き込みワークロードの単一障害点が生じ、スケーリングがより困難になります。これは、可用性を高めるためにマルチ AZ 構成を使用するか、スケーリングを向上させるために Amazon Aurora サーバーレス構成を使用することで部分的に軽減できます。可用性要件が非常に高い場合は、プライマリライターにまったく依存しない方が理にかなっています。読み取り専用のクエリには、リードレプリカを使用できます。これにより、冗長性が確保され、スケールアップだけでなくスケールアウトも可能です。書き込みは、例えば、Amazon Simple Queue Service キューにバッファリングできるため、プライマリが一時的に使用できなくなっても、カスタマーからの書き込み要求を引き続き受け付けることができます。

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
- 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

リクエストを制限して、予想外の需要の増加によるリソースの枯渇を緩和します。スロットリングレートを下回るリクエストは処理され、定義された制限を超えるリクエストは拒否され、リクエストがスロットリングされたことを示すメッセージが返されます。

期待される成果: 突然のカスタマートラフィックの増加、フラッディング攻撃、または再試行ストームによる大量のスパイクは、リクエストスロットリングによって軽減され、サポートされているリクエスト量の通常の処理をワークロードが継続できるようになります。

一般的なアンチパターン:

- API エンドポイントのスロットルは実装されていないか、予想される量を考慮せずにデフォルト値のままになっています。
- API エンドポイントは負荷テストされておらず、スロットリング制限もテストされていません。
- リクエストのサイズや複雑さを考慮せずにリクエストレートをスロットリングできます。

- 最大リクエストレートまたは最大リクエストサイズをテストしますが、両方を一緒にテストするわけではありません。
- リソースは、テストで設定したのと同じ制限にプロビジョニングされません。
- アプリケーション (A2A) API コンシューマーへの適用を目的とした使用プランは設定も検討もされていません。
- 水平方向にスケーリングするキューコンシューマーには、最大同時実行設定は設定されていません。
- IP アドレスごとのレート制限は実装されていません。

このベストプラクティスを活用するメリット: スロットル制限を設定したワークロードは、予期しない量のスパイクが発生しても、正常に動作し、受け入れられたリクエストの負荷を正常に処理できます。API やキューへのリクエストの急なスパイクや持続的なスパイクはスロットリングされ、リクエスト処理リソースを使い果たすことはありません。レート制限は、単一の IP アドレスまたは API コンシューマーからの大量のトラフィックがリソースを使い果たして他のコンシューマーに影響を与えないように、個々のリクエスタを制限します。

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: 高

実装のガイダンス

サービスは、既知のキャパシティのリクエストを処理するように設計する必要があります。このキャパシティは、負荷テストによって確立できます。リクエストの到着率が制限を超えると、適切なレスポンスからリクエストがスロットリングされたことが通知されます。これにより、コンシューマーはエラーを処理して後で再試行できます。

サービスにスロットリングの実装が必要な場合は、トークンがリクエストにカウントされるトークンバケットアルゴリズムの実装を検討してください。トークンは 1 秒あたりのスロットルレートで補充され、リクエストごとに 1 つのトークンで非同期に空になります。

トークンバケットアルゴリズム。

Amazon API Gateway は、アカウントとリージョンの制限に従ってトークンバケットアルゴリズムを実装し、使用プランを使用してクライアントごとに構成できます。さらに、Amazon Simple Queue Service (Amazon SQS) および Amazon Kinesis は、リクエストをバッファリングしてリクエストレートをスムーズにし、対応可能なリクエストのスロットリングレートを高くすることができます。最後に、 AWS WAF を使用してレート制限を実装し、異常に高い負荷を発生させる特定の API コンシューマーを制限します。

実装手順

API のスロットリング制限を使用して API Gateway を設定し、制限を超えると `#429 Too Many Requests#(429 ##########)` エラーを返すことができます。AWS AppSync および API Gateway エンドポイントで AWS WAF を使用すると、IP アドレスごとにレート制限を有効にできます。さらに、システムが非同期処理に対応できる場合は、メッセージをキューまたはストリームに入れてサービスクライアントへの応答を高速化できます。これにより、より高いスロットルレートにバーストできます。

非同期処理では、Amazon SQS を AWS Lambda のイベントソースとして設定すると、 最大同時実行数を設定して、 ワークロードやアカウント内の他のサービスに必要な、使用可能なアカウントの同時実行クォータを高いイベント率が消費することを回避できます。

API Gatewayトークンバケットのマネージド実装を提供していますが、API Gateway を使用できない場合は、サービス用のトークンバケットの言語固有のオープンソース実装 (「参考文献」の関連例を参照) を利用できます。

- リージョンごとのアカウントレベル、ステージごとの API、使用プランレベルごとの API キーでの API Gateway スロットリング制限 を理解して設定します。
- 以下の AWS WAF レート制限ルールを API Gateway および AWS AppSync エンドポイントに適用して、フラッドから保護し、悪意のある IP をブロックします。A2A コンシューマー向けの AWS AppSync API キーにレート制限ルールを設定することもできます。
- AWS AppSync API のレート制限よりも高度なスロットリング制御が必要かどうかを検討し、必要な場合は AWS AppSync エンドポイントの前に API Gateway を設定します。
- Amazon SQS キューが Lambda キューコンシューマーに対してアクティブとして設定されている場合は、 最大同時実行数を サービスレベル目標を達成するのに十分な処理を行うが、他の Lambda 関数に影響を与える同時実行数の制限を消費しない値に設定します。Lambda でキューを使用する場合は、同じアカウントおよびリージョン内の他の Lambda 関数に予約された同時実行を設定することを検討してください。
- API Gateway と Amazon SQS または Kinesis へのネイティブサービス統合を使用して、リクエストをバッファリングします。
- API Gateway を使用できない場合は、言語固有のライブラリを調べて、ワークロード用のトークンバケットアルゴリズムを実装してください。サンプルセクションを確認して、適切なライブラリを見つけるために独自の調査を行ってください。
- 設定する予定の、または引き上げを許可する予定の制限をテストし、テストした制限を文書化します。
- テストで設定した上限を超えて制限を増やさないでください。制限を増やす場合は、増やす前に、プロビジョニングされたリソースが既にテストシナリオのものと同等かそれ以上であることを確認してください。

リソース

関連するベストプラクティス:

- REL04-BP03 継続動作を行う
- REL05-BP03 再試行呼び出しを制御および制限する

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 WAF: ルートベースのルールステートメント
- Amazon SQS をイベントソースとして使用する場合の AWS Lambda の最大同時実行数の導入

- AWS Lambda: 最大同時実行数

関連サンプル:

- 最も重要な 3 つの AWS WAF レートベースのルール
- Java Bucket4j
- Python トークンバケット
- ノードトークンバケット
- .NET システムスレッドのレート制限

関連動画:

- AWS AppSync を使用した GraphQL API セキュリティのベストプラクティスの実装

関連ツール:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF

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、各再試行の間隔を徐々に長くしてリクエストを再試行します。再試行間隔をランダム化するために、再試行間にジッターを導入します。最大再試行回数を制限します。

期待される成果: 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。通常の操作では、これらのコンポーネントは一時的なエラーや限定的なエラーを含むリクエストに応答できます。また、再試行してもエラーが続くリクエストにも応答できます。クライアントがサービスにリクエストを行うと、そのリクエストはメモリ、スレッド、接続、ポート、またはその他の限られたリソースを含むリソースを消費します。再試行の制御と制限は、リソースを解放して消費を最小限に抑え、負荷がかかっているシステムコンポーネントに負荷がかからないようにするための戦略です。

クライアントのリクエストがタイムアウトになったり、エラーレスポンスが返されたりした場合は、再試行するかどうかを決定する必要があります。再試行する場合は、ジッターと最大再試行値によるエクスポネンシャルバックオフを行います。その結果、バックエンドのサービスとプロセスの負荷が軽減され、自己修復にかかる時間が短縮されるため、復旧が速くなり、リクエストサービスが正常に処理されます。

一般的なアンチパターン:

- エクスポネンシャルバックオフ、ジッター、最大再試行値を追加せずに再試行を実装します。バックオフとジッターは、同じ間隔で意図せずに調整された再試行による人為的なトラフィックのスパイクを防ぐのに役立ちます。
- その効果をテストしたり、再試行シナリオをテストせずに再試行が既に SDK に組み込まれていることを前提に再試行を実装します。
- 公開されている依存関係のエラーコードを理解できず、許可の欠如、設定エラー、または手動による介入なしでは解決できないと思われるその他の状態を示す明確な原因がある エラーを含め、すべてのエラーを再試行することになります。
- 根本的な問題を明らかにして対処できるように、繰り返し発生するサービス障害の監視や警告など、可観測性のプラクティスには触れていません。
- 組み込みまたはサードパーティの再試行機能で十分な場合は、カスタムの再試行メカニズムを開発します。
- アプリケーションスタックの複数のレイヤーで再試行すると、再試行が複雑になり、再試行の大混乱の中でさらにリソースを消費します。これらのエラーが依存しているアプリケーションの依存関係にどのように影響するかを必ず理解し、再試行は 1 つのレベルでのみ実装してください。
- べき等性を持たないサービスコールを再試行すると、結果が重複するなどの予期しない影響が発生します。

このベストプラクティスを活用するメリット: 再試行は、リクエストが失敗したときにクライアントが希望する結果を得るのに役立ちますが、必要な応答を得るまでにサーバーの時間を多く消費します。障害がまれな場合や一時的な場合は、再試行しても問題ありません。リソースの過負荷が原因で障害が発生した場合、再試行は事態を悪化させる可能性があります。クライアントの再試行にジッターを伴うエクスポネンシャルバックオフを追加することで、リソース過負荷が原因で障害が発生した場合でも、サーバーを回復できます。ジッターを使用すると、リクエストがスパイクに陥るのを防ぎ、バックオフによって通常のリクエスト負荷に再試行を追加することによる負荷のエスカレーションが軽減されます。最後に、メタステーブル障害の原因となるバックログが作成されないように、最大再試行回数または経過時間を設定することが重要です。

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

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。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。

一部の AWS 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。ワークロードに該当する場合は、これらの組み込み AWS 実装を使用してください。べき等性を持たせて再試行することでクライアントの可用性が向上するサービスを呼び出すときも、同様のロジックをワークロードに実装します。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。こうした再試行のユースケースに対応するテストシナリオを構築し、実行してください。

実装手順

- アプリケーションスタック内の最適なレイヤーを決定して、アプリケーションが依存するサービスの再試行を実装してください。
- 選択した言語に対してエクスポネンシャルバックオフとジッターを伴う実証済みの再試行戦略を実装している既存の SDK に注意し、独自の再試行実装を作成するよりもこれらを優先してください。
- 再試行を実装する前に、 サービスがべき等性を持っていることを 確認してください。再試行を実装したら、必ずテストを行い、本番環境で定期的に実行するようにしてください。
- AWS サービス API を呼び出すときは、 AWS SDK と AWS CLI を使用し、再試行設定オプションを理解してください。デフォルトがユースケースに適しているかどうかを判断し、テストし、必要に応じて調整します。

リソース

関連するベストプラクティス:

- ???
- REL05-BP02 リクエストのスロットル
- 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
- 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

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
- 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
- エクスポネンシャルバックオフとジッター
- べき等 API で再試行を安全に行う

関連サンプル:

- Spring Retry
- Resilience4j Retry

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)

関連ツール:

- AWS SDK とツール: 再試行動作
- AWS Command Line Interface: AWS CLI 再試行

REL05-BP04 フェイルファストとキューの制限

サービスがリクエストに正常に応答できない場合は、すぐに失敗します。これにより、リクエストに関連付けられたリソースが解放され、リソースが不足した場合にサービスを復旧できます。フェイルファストは確立されたソフトウェア設計パターンであり、これを活用して信頼性の高いワークロードをクラウド上に構築できます。キューイングは、負荷をスムーズにし、非同期処理が許容できる場合にクライアントがリソースを解放できるようにする、確立されたエンタープライズ統合パターンでもあります。サービスが通常の状態では正常に応答できるが、リクエストのレートが高すぎると失敗する場合は、キューを使用してリクエストをバッファします。ただし、長いキューのバックログの蓄積は許可しないでください。クライアントが既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。

期待される成果: システムにリソースの競合、タイムアウト、例外、またはグレー障害が発生してサービスレベル目標を達成できない場合、フェイルファスト戦略を使用するとシステムをより迅速に

回復できます。トラフィックの急増を吸収する必要があり、非同期処理に対応できるシステムでは、バックエンドサービスへのリクエストをバッファリングするキューを使用して、クライアントがリクエストを迅速にリリースできるようにすることで信頼性を向上できます。リクエストをキューにバッファリングする際には、克服できないバックログを回避するためにキュー管理戦略が実装されます。

一般的なアンチパターン:

- メッセージキューを実装するが、システムに障害が発生したことを検出するデッドレターキュー (DLQ) やアラームを DLQ ボリュームに設定しない。
- キュー内のメッセージの経過時間を測定するのではなく、キューのコンシューマーが遅れたり、エラーが発生して再試行が発生したりするタイミングを把握するためのレイテンシーの測定です。
- 業務上の必要がなくなった場合に、これらのメッセージを処理する価値がない場合に、未処理のメッセージをキューから消去しない。
- 先入れ先出し (FIFO) キューを後入れ先出し (LIFO) キューに設定すると、クライアントのニーズにより適切に対応できます。例えば、厳密な順序付けが不要で、バックログ処理により新規リクエストや時間的制約のあるリクエストがすべて遅延し、その結果、すべてのクライアントでサービスレベル違反が発生するような場合です。
- 仕事の受け入れを管理してリクエストを内部キューに入れる API を公開する代わりに、内部キューをクライアントに公開します。
- 1 つのキューに多数の作業リクエストタイプをまとめると、リソース需要がリクエストタイプ全体に分散され、バックログの状態が悪化する可能性があります。
- 異なるモニタリング、タイムアウト、リソース割り当てが必要な場合でも、複雑なリクエストと単純なリクエストを同じキューで処理します。
- エラーを適切に処理できる上位レベルのコンポーネントに例外をバブリングするフェイルファストメカニズムをソフトウェアで実装するために、入力を検証したり、アサーションを使用しない。
- リクエストルーティングから障害のあるリソースを削除しない。特に、クラッシュや再起動、断続的な依存関係の障害、容量の低下、ネットワークのパケットロスなどにより、障害がグレーで成功と失敗の両方を示している場合。

このベストプラクティスを活用するメリット: すぐに失敗するシステムは、デバッグや修正が容易で、リリースが本稼働環境に公開される前にコーディングや構成の問題が明らかになることがよくあります。効果的なキューイング戦略を組み込んだシステムは、トラフィックの急増や断続的なシステム障害状態に対する回復力と信頼性が向上します。

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: 高

実装のガイダンス

フェイルファスト戦略は、ソフトウェアソリューションにコード化することも、インフラストラクチャに構成することもできます。キューは、高速に障害が発生するだけでなく、システムコンポーネントを切り離してスムーズに負荷をかけるための単純でありながら強力なアーキテクチャ手法です。Amazon CloudWatch には、障害を監視して警告する機能があります。システムに障害が発生していることが判明したら、障害が発生したリソースからフェイルアウェイするなどの緩和策を講じることができます。システムが次の方法でキューを実装する場合 Amazon SQS およびその他のキューテクノロジーでは、負荷をスムーズにするために、キューのバックログやメッセージ消費の失敗を管理する方法を検討する必要があります。

実装手順

- プログラムによるアサーションや特定のメトリクスをソフトウェアに実装し、それらを使用してシステムの問題を明示的に警告します。Amazon CloudWatch は、アプリケーションログパターンと SDK 機器に基づいてメトリクスとアラームを作成するのに役立ちます。
- CloudWatchメトリクスとアラームを使用して、リソースに障害が発生して処理に遅延が発生したり、リクエストの処理が繰り返し失敗しないようにします。
- Amazon SQS を使用してリクエストを受け入れ、リクエストを内部キューに追加し、メッセージ生成クライアントに成功メッセージで応答する API を設計することで非同期処理を使用します。これにより、バックエンドキューのコンシューマーがリクエストを処理している間、クライアントはリソースを解放して他の作業に進むことができます。
- 現在とメッセージのタイムスタンプを比較することで、メッセージをキューから取り出すたびに CloudWatch メトリクスを生成し、キューの処理遅延を測定およびモニタリングします。
- 障害によってメッセージ処理が正常に行われなかった、またはサービスレベル契約の範囲内で処理できない量のトラフィックが急増した場合は、古いトラフィックや過剰なトラフィックをスピルオーバーキューに振り分けます。これにより、キャパシティに空きがあれば、新しい作業や古い作業を優先的に処理できます。この手法は LIFO 処理の近似値であり、すべての新規作業で通常のシステム処理が可能になります。
- 処理できないメッセージをバックログから後で調査して解決できる場所に移動するには、デッドレターキューまたはリドライブキューを使用します。
- 再試行するか、許容範囲内であれば、メッセージのタイムスタンプと現在を比較して、要求元のクライアントに関係のないメッセージは破棄して、古いメッセージを削除してください。

リソース

関連するベストプラクティス:

- REL04-BP02 疎結合の依存関係を実装する
- 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
- REL06-BP02 メトリクスを定義および計算する (集計)
- 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

関連するドキュメント:

- 乗り越えられないキューバックログの回避
- 速やかな失敗
- Amazon SQS キュー内のメッセージのバックログの増加を防ぐにはどうすればよいですか?
- Elastic Load Balancing: ゾーンシフト
- Amazon Route 53 アプリケーションリカバリコントローラ: トラフィックフェイルオーバーのルーティング制御

関連サンプル:

- エンタープライズ統合パターン: デッドレターチャネル

関連動画:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications (AWS re:Invent 2022 - 高可用性のマルチ AZ アプリケーションの運用)

関連ツール:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 クライアントタイムアウトを設定する

接続とリクエストにタイムアウトを適切に設定し、体系的に検証します。また、デフォルト値には依存しないでください。これらはワークロードの詳細を認識していないためです。

期待される成果: クライアントのタイムアウトには、完了までに異常に時間がかかるリクエストを待つことに関連するクライアント、サーバー、およびワークロードにかかるコストを考慮する必要があります。タイムアウトの正確な原因を知ることはできないため、クライアントはサービスの知識を活用して、考えられる原因と適切なタイムアウトを予測する必要があります。

クライアント接続は、設定された値に基づいてタイムアウトします。タイムアウトが発生すると、クライアントはバックオフして再試行するか、 サーキットブレーカーを開くかを決定します。これらのパターンは、根本的なエラー状態を悪化させる可能性のあるリクエストの発行を回避します。

一般的なアンチパターン:

- システムタイムアウトまたはデフォルトタイムアウトを認識していない。
- 通常のリクエスト完了タイミングを認識していない。
- リクエストが完了するまでに異常に時間がかかる原因や、これらの完了を待つことによってクライアント、サービス、またはワークロードのパフォーマンスが低下する原因を認識していない。
- ネットワークに障害が発生して、タイムアウトに達したときだけリクエストが失敗する確率や、より短いタイムアウトを採用しないことでクライアントとワークロードのパフォーマンスにコストがかかることを認識していない。
- 接続とリクエストの両方のタイムアウトシナリオはテストされていません。
- タイムアウトの設定が高すぎると、待機時間が長くなり、リソースの使用率が高くなる可能性があります。
- タイムアウトの設定が低すぎると、人為的な障害が発生します。
- サーキットブレーカーや再試行などのリモート呼び出しのタイムアウトエラーを処理するパターンを見落としています。
- サービス呼び出しエラー率、遅延に関するサービスレベル目標、および遅延異常値のモニタリングは考慮していません。これらのメトリクスから、タイムアウトが積極的または許容範囲が広いかを判断できます。

このベストプラクティスを活用するメリット: リモート呼び出しのタイムアウトは、リモート呼び出しの応答が異常に遅い場合やタイムアウトエラーがサービスクライアントによって適切に処理される場合にリソースを節約できるように、タイムアウトを適切に処理するように設定され、システムがタイムアウトを適切に処理す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: 高

実装のガイダンス

サービス依存関係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、通常プロセス全体のすべての呼び出しにも行います。多くのフレームワークにはタイムアウト機能が組み込まれていますが、デフォルト値が無限であるか、サービス目標の許容範囲を超えているものもあるので注意してください。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。

タイムアウト戦略を決定する際には、次の点を考慮してください。

- リクエストの内容、ターゲットサービスの障害、またはネットワークパーティションの障害により、リクエストの処理に通常よりも時間がかかる場合があります。
- 異常に高価なコンテンツを含むリクエストは、サーバーとクライアントのリソースを不必要に消費する可能性があります。この場合、これらのリクエストをタイムアウトさせて再試行しないことで、リソースを節約できます。また、サービスは、スロットルやサーバー側のタイムアウトにより、異常にコストが大きいコンテンツから身を守る必要があります。
- サービスの障害により異常に時間がかかるリクエストは、タイムアウトして再試行できます。リクエストと再試行のサービスコストを考慮する必要がありますが、原因が局所的な障害である場合は、再試行してもコストはかからず、クライアントリソースの消費量を削減できます。障害の性質によっては、タイムアウトによってサーバーリソースが解放されることもあります。
- リクエストまたはレスポンスがネットワークから配信されなかったために完了までに時間がかかるリクエストは、タイムアウトして再試行できます。リクエストまたはレスポンスが配信されなかったため、タイムアウトの長さに関係なく失敗に終わったことになります。この場合、タイムアウトしてもサーバーリソースは解放されませんが、クライアントリソースが解放され、ワークロードのパフォーマンスが向上します。

再試行やサーキットブレーカーなどの確立された設計パターンを活用して、タイムアウトをスムーズに処理し、フェイルファストアプローチをサポートします。AWS SDK そして AWS CLI 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でき、エクスポネンシャルバックオフとジッターによる再試行も可能です。AWS Lambda 関数はタイムアウトの設定をサポートしており、 AWS Step Functions を使用すると、AWS サービスおよび SDK との事前構築された統合を利用するローコード

サーキットブレーカーを構築できます。AWS App Mesh Envoy はタイムアウトとサーキットブレーカー機能を提供します。

実装手順

- リモートサービス呼び出しのタイムアウトを設定し、組み込みの言語タイムアウト機能またはオープンソースのタイムアウトライブラリを活用してください。
- ワークロードが AWS SDK を使用して呼び出しを行う場合は、ドキュメントで言語固有のタイムアウト設定を確認してください。
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++
- ワークロードで AWS SDK や AWS CLI コマンドを使用する場合、以下の AWS 設定デフォルト を設定し、デフォルトタイムアウト値を設定してください。(`connectTimeoutInMillis` そして `tlsNegotiationTimeoutInMillis`)。
- コマンドラインオプション cli-connect-timeout ### cli-read-timeout ###### AWS サービスへの 1 回限りの AWS CLI コマンドを制御します。
- リモートサービス呼び出しのタイムアウトをモニタリングし、エラーが続く場合はアラームを設定して、エラーシナリオにプロアクティブに対処できるようにします。
- タグ付けを CloudWatch Metrics cli-read-timeout CloudWatch 異常の検出 オンコールエラー率、レイテンシーに関するサービスレベル目標、レイテンシーの異常値から、過度にアグレッシブなタイムアウトや許容範囲のタイムアウトの管理に関するインサイトが得られます。
- タイムアウトを Lambda 関数に設定します。
- API Gateway クライアントは、タイムアウトを処理するときに独自の再試行を実装する必要があります。API Gateway は、 ダウンストリーム統合に対して 50 ミリ秒から 29 秒の統合タイムアウトをサポートし、 統合リクエストのタイムアウト時に再試行しません。
- タイムアウト時のリモート呼び出しを回避するには、 サーキットブレーカーを開くかを決定します パターンを実装します。呼び出しが失敗しないように回線を開き、呼び出しが正常に応答したら回線を閉じます。

- コンテナベースのワークロードの場合は、組み込みのタイムアウトとサーキットブレーカーを活用するための App Mesh Envoy 機能を確認してください。
- AWS Step Functions を使用して、リモートサービス呼び出し用のローコードサーキットブレーカーを構築します。特に、ワークロードを簡素化するために AWS ネイティブ SDK とサポートされている Step Functions 統合を呼び出す場合に使用します。

リソース

関連するベストプラクティス:

- 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
- 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

関連するドキュメント:

- AWS SDK: 再試行とタイムアウト
- 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
- Amazon API Gateway クォータと重要事項
- AWS Command Line Interface: コマンドラインオプション
- AWS SDK for Java 2.x: API タイムアウトの設定
- 設定オブジェクトと設定リファレンスを使用した AWS Botocore
- AWS SDK for .NET: 再試行とタイムアウト
- AWS Lambda: Lambda 関数オプションの設定

関連サンプル:

- AWS Step Functions および Amazon DynamoDB を使用したサーキットブレーカーパターンの使用
- Martin Fowler: サーキットブレーカー

関連ツール:

- AWS SDK

- AWS Lambda
- Amazon SQS
- AWS Step Functions を使用すると
- AWS Command Line Interface

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。

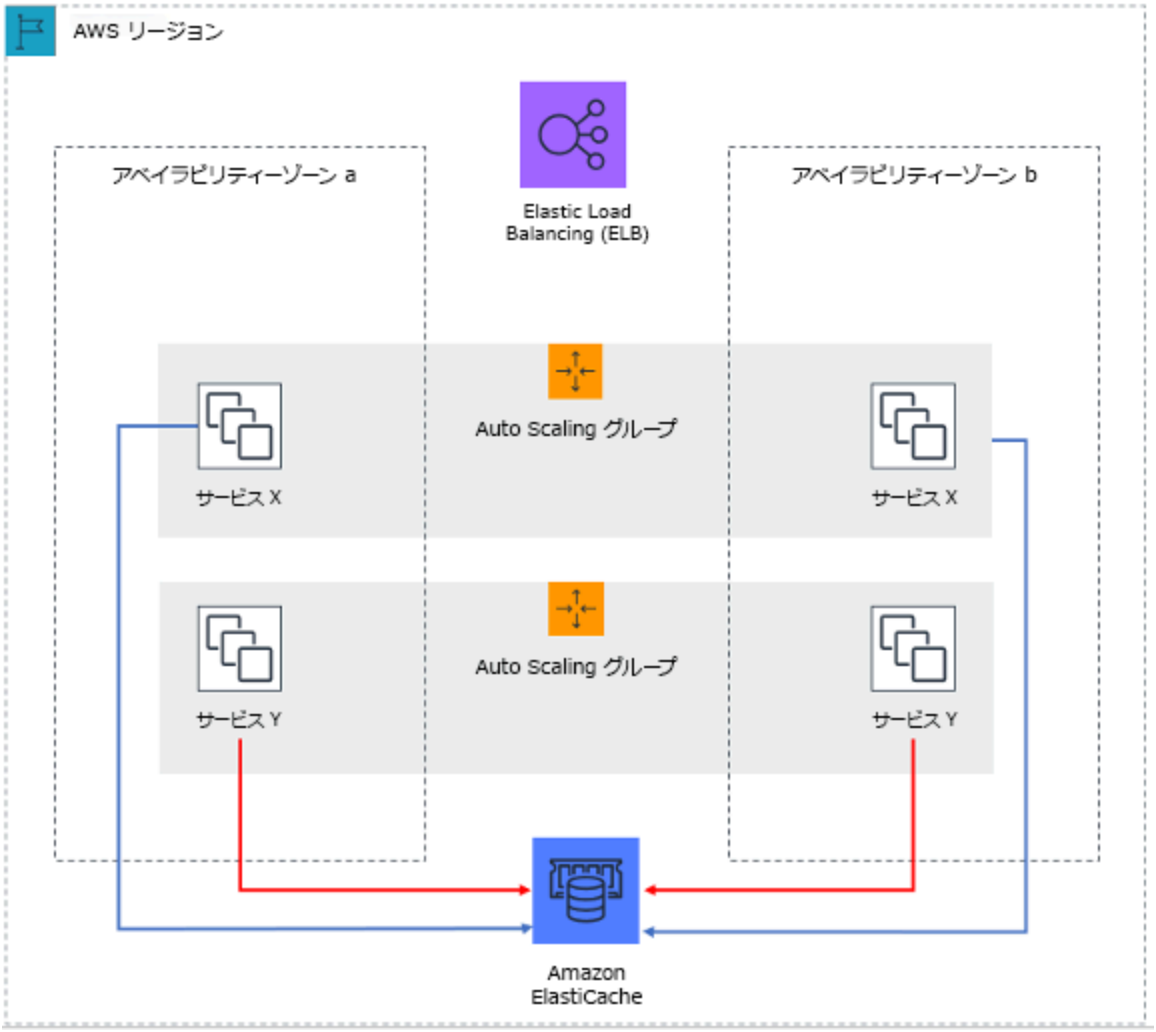


図 7: このステートレスウェブアプリケーションでは、セッション状態は Amazon ElastiCache にオフロードされます。

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 へのログの直接送信)
- 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 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

サービスコンポーネントで処理されるリクエストをトレースすることで、製品チームではより簡単に問題の分析とデバッグを行い、パフォーマンスを向上させることができます。

期待される成果: すべてのコンポーネントを網羅的にトレースできるワークロードは、デバッグが容易で、根本原因の発見を簡略化することで、エラーやレイテンシーの 解決までの平均時間 (MTTR) を改善します。エンドツーエンドのトレースによって影響を受けるコンポーネントを検出し、エラーやレイテンシーの根本原因の詳細調査にかかる時間を短縮できます。

一般的なアンチパターン:

- トレースは一部のコンポーネントに使用されますが、すべてのコンポーネントで使用されるわけではありません。例えば、AWS Lambda のトレースを行わない場合は、ワークロードの急増でのコールドスタートが原因で生じたレイテンシーを明確に把握できない可能性があります。
- Synthetic Canaries やリアルユーザーモニタリング (RUM) には、トレースは設定されていません。Canary や RUM を使用しない場合、クライアントインタラクションのテレメトリがトレース分析から除外され、パフォーマンスプロファイルが不完全な状態になります。
- ハイブリッドワークロードには、クラウドネイティブとサードパーティのトレースツールの両方が含まれていますが、単一のトレースソリューションを選択し、完全に統合する手段は講じられていません。選択したトレースソリューションに基づいて、クラウドネイティブのトレース SDK を使用して、クラウドネイティブではないコンポーネントを測定するか、サードパーティツールを使用してクラウドネイティブのトレーステレメトリを取り込むように設定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: 開発チームでは、問題についてアラートを受けることで、ロギング、パフォーマンス、障害に対するコンポーネントごとの相関関係など、システムコンポーネントの相互作用の全体像を把握できます。トレースによって根本原因を視覚的に把握しやすく

なるため、根本原因の究明に費やす時間を短縮できます。チームではコンポーネントの相互作用を詳細に理解することで、問題を解決する際に、より適切で迅速な意思決定を行うことができます。システムトレースの分析によって、ディザスタリカバリ (DR) フェイルオーバーの開始時期、自己修復戦略を実行する場所などの意思決定を改善することで、最終的にサービスに対する顧客満足度の向上につながります。

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

分散アプリケーションを運用するチームでは、トレースツールを使用して相関識別子を設定し、リクエストのトレースを収集して、接続されたコンポーネントのサービスマップを作成できます。サービスクライアント、ミドルウェアゲートウェイ、イベントバス、コンピューティングコンポーネント、キーバリューストアやデータベースを含むストレージなど、すべてのアプリケーションコンポーネントをリクエストトレースに含める必要があります。エンドツーエンドのトレース設定に Synthetic Canaries とリアルユーザーモニタリングを組み込んで、リモートクライアントとのやり取りやレイテンシーを測定することで、サービスレベル契約や目標に対するシステムのパフォーマンスを正確に評価できます。

そのため、 AWS X-Ray および Amazon CloudWatch アプリケーションモニタリング の測定サービスを使用すると、アプリケーションを通過するリクエストの完全なビューを提供できます。X-Ray は、アプリケーションのテレメトリを収集し、ペイロード、関数、トレース、サービス、API 全般の可視化およびフィルター処理が可能で、ノーコードまたはローコードのシステムコンポーネントに対して有効にできます。CloudWatch アプリケーションのモニタリングには ServiceLens が含まれており、トレースをメトリクス、ログ、アラームと統合します。CloudWatch アプリケーションモニタリングには、エンドポイントと API をモニタリングするための Synthetics や、ウェブアプリケーションクライアントを測定するためのリアルユーザーモニタリングも含まれています。

実装手順

- サポートされているすべてのネイティブサービスで AWS X-Ray (Amazon S3、AWS Lambda、Amazon API Gateway など) を使用します。これらの AWS サービスでは、インフラストラクチャをコードとして、AWS SDK、または AWS マネジメントコンソール を使用して設定を切り替え、X-Ray を有効にできます。
- 測定アプリケーション AWS Distro for Open Telemetry および X-Ray またはサードパーティの収集エージェント。
- プログラミング言語固有の実装については、 AWS X-Ray 開発者ガイド を参照してください。これらのドキュメントでは、HTTP リクエスト、SQL クエリ、アプリケーションのプログラミング言語固有のその他のプロセスを測定する方法について詳しく説明します。

- X-Ray トレース (Amazon CloudWatch Synthetic Canaries および Amazon CloudWatch RUM) を使用して、エンドユーザークライアントからダウンストリームの AWS インフラストラクチャを経由するリクエストパスを分析します。
- リソースの健全性と Canary テレメトリに基づき CloudWatch メトリクスとアラームを設定することで、チームでは迅速に問題についてアラートを発し、ServiceLens でトレースやサービスマップを詳しく調査できます。
- 次のようなサードパーティのトレースツールと X-Ray の統合を有効にします。 Datadog、New Relic、 Dynatrace (主要なトレースソリューションにサードパーティツールを使用している場合)。

リソース

関連する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)
- 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

関連するドキュメント:

- AWS X-Ray とは?
- Amazon CloudWatch: アプリケーションモニタリング
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- 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
- 他の AWS サービスと AWS X-Ray の統合
- AWS Distro for OpenTelemetry および AWS X-Ray
- Amazon CloudWatch: 合成モニタリングを使用
- Amazon CloudWatch: CloudWatch RUM を使用
- Amazon CloudWatch Synthetic Canaries と Amazon CloudWatch アラームの設定
- 可用性を超えて: AWS での分散システムの回復力の理解と向上

関連サンプル:

- 1 つの可観測性ワークショップ

関連動画:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts
- AWS アプリケーションをモニタリングする方法

関連ツール:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

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 マネジメントコンソール 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 = λW

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。

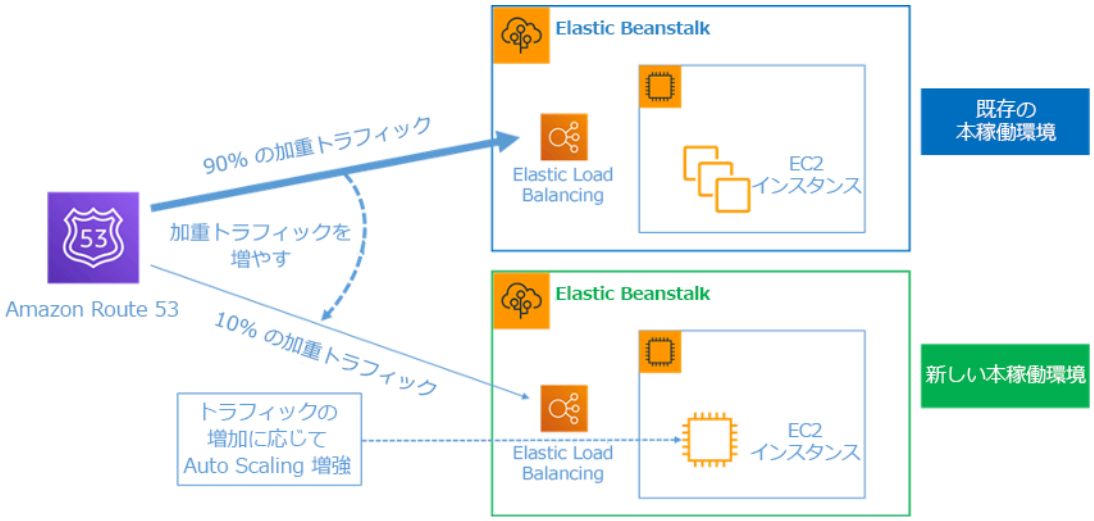


図 8: AWS Elastic Beanstalk と Amazon Route 53 によるブルー/グリーンデプロイ

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

ワークロードが使用するデータ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。

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ます。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活用するメリット: 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

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、AWS サービス全体にわたるデータ保護を一元化して自動化する機能を提供するツールです。AWS Elastic Disaster Recovery を使用すると、サーバーのワークロード全体をコピーして、オンプレミス、クロス AZ、またはクロスリージョンから継続的なデータ保護を維持できます。目標復旧時点 (RPO) は秒単位で測定されます。

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は、AWS Storage Gateway または AWS DataSync を使用して AWS クラウドにバックアップできます。このデータを AWS で保管するには、Amazon S3 バケットを使用できます。Amazon S3 は、Amazon Glacier や Amazon Glacier Deep Archive などの複数のストレージ層を提供し、データストレージコストを低減します。

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 Amazon 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には、バックアップは必要でないことがあります。別の例として、Amazon EMR を使用している場合、データを Amazon S3 から Amazon EMR に再生成できる限り、HDFS データストアをバックアップする必要がないことがあります。

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。

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、ボリューム、ファイルシステム、ログ記録システム、オブジェクトストレージなどのさまざまなリソースに保存できます。「リソース」セクションを参照して、データが保存されているさまざまな AWS サービスに関する関連ドキュメントと、これらのサービスが提供するバックアップ機能を確認してください。
2. 重要度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ー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 で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。AWS Elastic Disaster Recovery は、AWS リージョン への自動サブセカンドデータレプリケーションを処理します。また、AWS 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。さまざまな AWS 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については、以下に記載されているリソース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。

実装計画に必要な工数レベル: 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:

- What Is AWS Backup? (AWS Backup とは)
- What is AWS DataSync? (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: products that can be used for backup (AWS Marketplace: バックアップに使用できる製品)
- Amazon EBS スナップショット
- Backing Up Amazon EFS (Amazon EFS のバックアップ)
- バックアップの使用 - Amazon FSx for Windows File Server
- 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
- Creating an EventBridge Rule That Triggers on a Schedule (スケジュールに従ってトリガーする Amazon EventBridge ルールの作成)
- Cross-Region Replication with Amazon S3 (Amazon S3 を使用したクロスリージョンレプリケーション)
- EFS-to-EFS AWS Backup (AWS の EFS-to-EFS バックアップ)
- 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
- DynamoDB のオンデマンドバックアップと復元の使用
- DynamoDB のポイントインタイムリカバリ
- Amazon OpenSearch Service でのインデックススナップショットの作成
- What is AWS Elastic Disaster Recovery? (AWS Elastic Disaster Recovery とは)

関連動画:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS (AWS re:Invent 2021 - AWS を使用したバックアップ、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 (AWS 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, ft.Rackspace (STG341)

関連する例:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3 (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 Lab - Testing Backup and Restore of Data (Well-Architected ラボ - データのバックアップと復元のテスト)
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload (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 Lab - Disaster Recovery - Backup and Restore (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

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。

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活用するメリット: 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

AWS Identity and Access Management (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。AWS Elastic Disaster Recovery を使用すると、転送中および保管中のすべてのデータが暗号化されます。Elastic Disaster Recovery を使用すると、デフォルトの Amazon EBS 暗号化ボリューム暗号化キーまたはカスタムのカスタマーマネージドキーのいずれかを使用して、保管中のデータを暗号化できます。

実装手順

1. 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。

- Amazon RDS で暗号化を使用します。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 EBS ボリュームで暗号化を使用します。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。
- 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。
- Amazon EFS に保存し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。
- 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。
- デフォルトの暗号化を設定するか、Elastic Disaster Recovery 向けの Amazon EBS 暗号化を使用するかを選択します。このオプションでは、ステージングエリアのサブネットディスクとレプリケートしたディスク上に保存されているレプリケートされたデータを暗号化します。

2. 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。

リソース

関連するドキュメント:

- AWS Marketplace: products that can be used for backup (AWS Marketplace: バックアップに使用できる製品)
- Amazon EBS 暗号化
- Amazon S3: 暗号化を使用したデータの保護
- CRR 追加のレプリケーション設定: AWS KMS に保存された暗号化キーで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
- Encrypting Amazon RDS Resources (Amazon RDS リソースの暗号化)
- Encrypting Data and Metadata in Amazon EFS (EFS でのデータとメタデータの暗号化)
- Encryption for Backups in AWS (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

- AWS Well-Architected Framework - セキュリティの柱
- What is AWS Elastic Disaster Recovery? (AWS Elastic Disaster Recovery とは)

関連する例:

- Well-Architected Lab - Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3 (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。

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活用するメリット: 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を確立しない場合のリスクレベル: 中

実装のガイダンス

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 DynamoDB、Amazon RDS、Amazon Keyspaces (for Apache Cassandra) 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て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 は、Amazon EBS スナップショットの作成、コピー、削除を自動化するために使用できます。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。

AWS Elastic Disaster Recovery は、ソース環境 (オンプレミスまたは AWS) からターゲットの復旧リージョンへの継続的なブロックレベルのレプリケーションを提供します。ポイントインタイム Amazon EBS スナップショットは、このサービスが自動的に作成し、管理します。

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。

実装手順

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。詳細については、以下を参照してください 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.
2. ワークロードの RPO を決定します。詳細については、以下を参照してください 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.
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、AWS サービス、クラウド、オンプレミス全体にわたるデータ保護の一元化と自動化を容易にします。AWS Backup のバックアッププランを使用して、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。AWS Backup を使用した自動バックアップの作成方法に関するハンズオントレーニングについては、「Testing Backup and Restore of Data」(データのバックアップと復元のテスト) を参照してください。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. オンプレミスのデータソースやメッセージキューなどの自動バックアップソリューションやマネージドサービスでサポートされてい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ー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 に基づく頻度で実行することができます。

実装計画に必要な工数レベル: 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: products that can be used for backup (AWS Marketplace: バックアップに使用できる製品)
- Creating an EventBridge Rule That Triggers on a Schedule (スケジュールに従ってトリガーする Amazon EventBridge ルールの作成)
- What Is AWS Backup? (AWS Backup とは)
- What Is AWS Step Functions? (AWS Step Functions とは)
- What is AWS Elastic Disaster Recovery? (AWS Elastic Disaster Recovery とは)

関連動画:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft.Rackspace (STG341)

関連する例:

- Well-Architected Lab - Testing Backup and Restore of Data (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。

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、必要に応じて復元する。

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と、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確立しない場合のリスクレベル: 中

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。実行する必要がある一般的なテストには、すべてのデータが利用可能かどうか、破損していないかどうか、アクセス可能かどうか、データ損失がワークロードの RPO 内に収まるかどうかを確認することなどがあります。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。

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。

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。

AWS Elastic Disaster Recovery は、Amazon EBS ボリュームの継続的なポイントインタイムリカバリのスナップショットを提供します。ソースサーバーがレプリケートされると、設定済みのポリシーに基づいて、ポイントインタイムの状態が経時的に記録されます。Elastic Disaster Recovery を使用すると、トラフィックをリダイレクトせずにテストおよびドリル目的でインスタンスを起動することにより、これらのスナップショットの整合性を検証できます。

実装手順

1. 現在バックアップされているデータソースを特定し、バックアップが保存されている場所を確認します。実装のガイダンスについては、以下を参照してください 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.
2. 各データソースのデータ検証に使用する基準を確立します。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。実装のガイダンスについては、以下を参照してください 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.
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。AWS Resilience Hub を使用して、ワークロードの評価を実行できます。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. 本番環境で現在確立されている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。例えばこれは、AWS Backup などのマネージドサービスを使用している場合は、新しいリソースでバックアップを復元するという簡単な作業となります。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. 前のステップで確立したデータ検証の基準に基づいて、復元したリソースからのデータ回復を検証します。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または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. 復元と復旧に必要とした時間を測定して、決定済みの 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 コールにはタイムスタンプが付けられており、この情報は、AWS CloudTrail で提供されています。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。自動化されたプロセスを使用している場合は、この情報の保存に Amazon DynamoDB などのサービスを使用できます。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 では、バックアップと復元アクションは、ジョブと呼ばれており、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. データの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、ステークホルダーに通知します。このラボのように、このプロセスのオートメーションを実装する場合、Amazon Simple Notification Service (Amazon SNS) などのサービスを使用して、E メールや SMS などのプッシュ通知をステークホルダーに送信できます。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、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ができます。
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。Automate data recovery validation with AWS Backup (AWS Backup を使用して復元データの検証を自動化する) 方法を確認してください。さらに、この Well-Architected ラボでは、いくつかのステップのオートメーション方法の 1 つについてのハンズオンエクスペリエンスを提供しています。

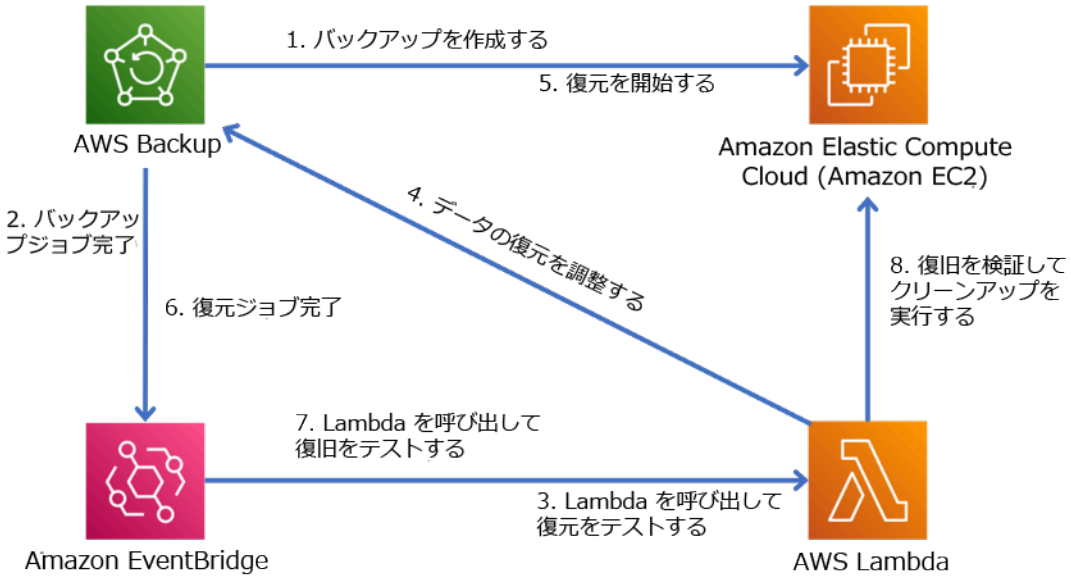


図 9.自動化されたバックアップおよび復元プロセス

実装計画に必要な工数レベル: 中 ~ 高 (検証基準の複雑性に依存)。

リソース

関連するドキュメント:

- Automate data recovery validation with AWS Backup (AWS Backup を使用して復元データの検証を自動化する)
- 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: products that can be used for backup (AWS Marketplace: バックアップに使用できる製品)
- Creating an EventBridge Rule That Triggers on a Schedule (スケジュールに従ってトリガーする Amazon EventBridge ルールの作成)
- DynamoDB のオンデマンドバックアップと復元の使用
- What Is AWS Backup? (AWS Backup とは)
- What Is AWS Step Functions? (AWS Step Functions とは)
- What is AWS Elastic Disaster Recovery (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。

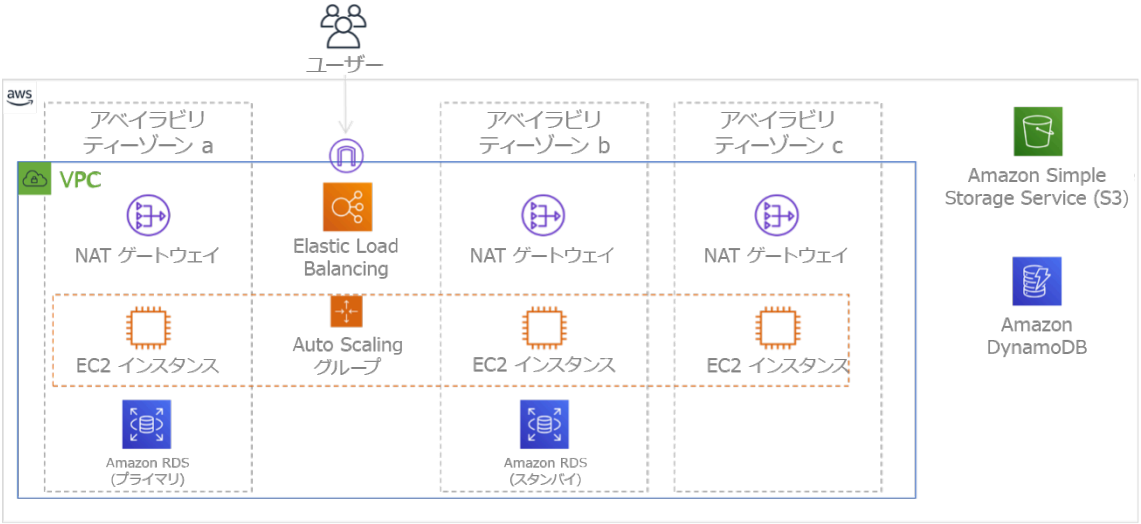


図 9: 3 つのアベイラビリティーゾーンにまたがってデプロイされる多階層アーキテクチャ。Amazon S3 と Amazon DynamoDB は常に自動的にマルチ AZ であることに注意してください。ELB も 3 つのゾーンすべてにデプロイされます。

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。

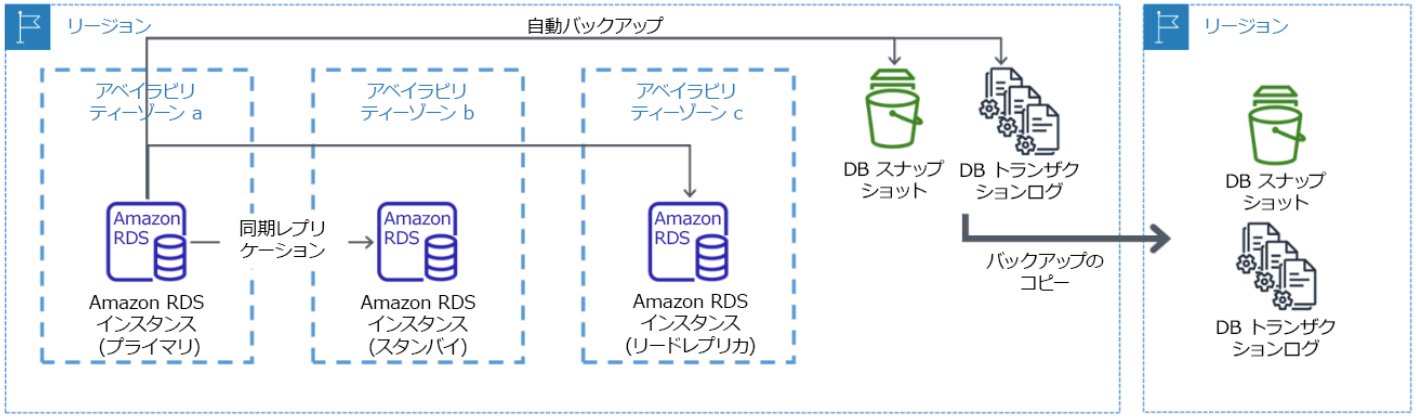


図 10: 別の AWS リージョンへのバックアップを備えた回復力の高いマルチ AZ データベースデプロイ

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を実行できるのが単一のアベイラビリティーゾーンまたはオンプレミスのデータセンターでのみである場合は、定義し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。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

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、回復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。複数のノードをプロビジョニングできます。EMR File System (EMRFS)を使用すると、EMR のデータを Amazon S3 に保存でき、今度はそのデータを複数のアベイラビリティーゾーンまたは複数の AWS リージョン にわたり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。

オンプレミスのデータセンターにデプロイされたサーバベースのステートフルなワークロードの場合、AWS Elastic Disaster Recovery を使用して AWS のワークロードを保護できます既に AWS でホストされてる場合、Elastic Disaster Recovery を使用してワークロードを別のアベイラビリティゾーンまたはリージョンに保護できます。Elastic Disaster Recovery は、軽量のステージングエリアへのブロックレベルの継続的なレプリケーションを行い、オンプレミスおよびクラウドベースのアプリケーションの高速かつ信頼性の高い復旧を提供します。

実装手順

1. 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、Amazon EC2 Auto Scaling グループを使用します。
  - 起動テンプレートの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、Amazon EC2 インスタンスの自己復旧機能を使用します。
  - 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、Amazon EC2 インスタンスのライフサイクルイベントや Amazon ECS イベントを利用して、自己修復を自動化します。
  - 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。
- 単一のロケーションに制限したステートフルワークロードは、AWS Elastic Disaster Recovery を使用して保護します。

リソース

関連するドキュメント:

- Amazon ECS イベント
- Amazon EC2 Auto Scaling の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
- Amazon EC2 Auto Scaling とは
- AWS Elastic Disaster Recovery

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

バルクヘッドアーキテクチャ (セルベースアーキテクチャとも呼ばれる) を実装して、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制限します。

期待される成果: セルベースアーキテクチャでは、複数の分離されたワークロードのインスタンスが使用されます。各インスタンスはセルと呼ばれます。各セルは独立しており、その他のセルとは状態を共有せず、ワークロードのリクエスト全体のサブセットを処理します。これにより、不適切なソフトウェア更新などの障害による個別のセルおよび処理中のリクエストへの予測される影響が軽減されます。例えばワークロードが 10 個のセルを使用して 100 個のリクエストを処理していると、障害が発生した場合、リクエスト全体の 90% は障害の影響を受けません。

一般的なアンチパターン:

- セルを際限なく増殖させる。
- コードの更新やデプロイをすべてのセルに同時に適用する。
- セル間で状態またはコンポーネントを共有する (ルーターレイヤーを除く)。
- 複雑なビジネスロジックやルーティングロジックをルーターレイヤーに追加する。
- セル間のインタラクションを最小に抑えない。

このベストプラクティスを活用するメリット: セルベースアーキテクチャでは、多数の一般的なタイプの障害がセル自体に封じ込めるため、障害分離が向上します。このような障害境界は、コードのデプロイの失敗や、破損したリクエスト、または特定の障害モードをトリガーするリクエスト (ポイズンピルリクエストとも呼ばれる) など、これ以外の方法では封じ込めが困難なタイプの障害への回復力を実現します。

実装のガイダンス

船ではバルクヘッドが使用されており、船体が破損しても、船体の一部のセクションのみに封じ込めることができます。複雑なシステムでは、このパターンを模して障害分離を実現する場合がよくあります。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。AWS では、複数のアベイラビリティーゾーンとリージョンを使用して障害分離を実現できます。この障害分離の概念はワークロードのアーキテクチャにも拡張できます。

ワークロード全体は、パーティションキーによってセルに分割されます。このキーは、サービスの粒度またはサービスのワークロードを最小限のクロスセルインタラクションで分割できる自然な方法に沿って分割を行う必要があります。パーティションキーの例には、カスタマー ID、リソース ID、またはほとんどの API コールで簡単にアクセスできるその他のパラメータなどがあります。セルルーティングレイヤーは、パーティションキーに基づいて個々のセルにリクエストを分散し、クライアントに単一のエンドポイントを提示します。

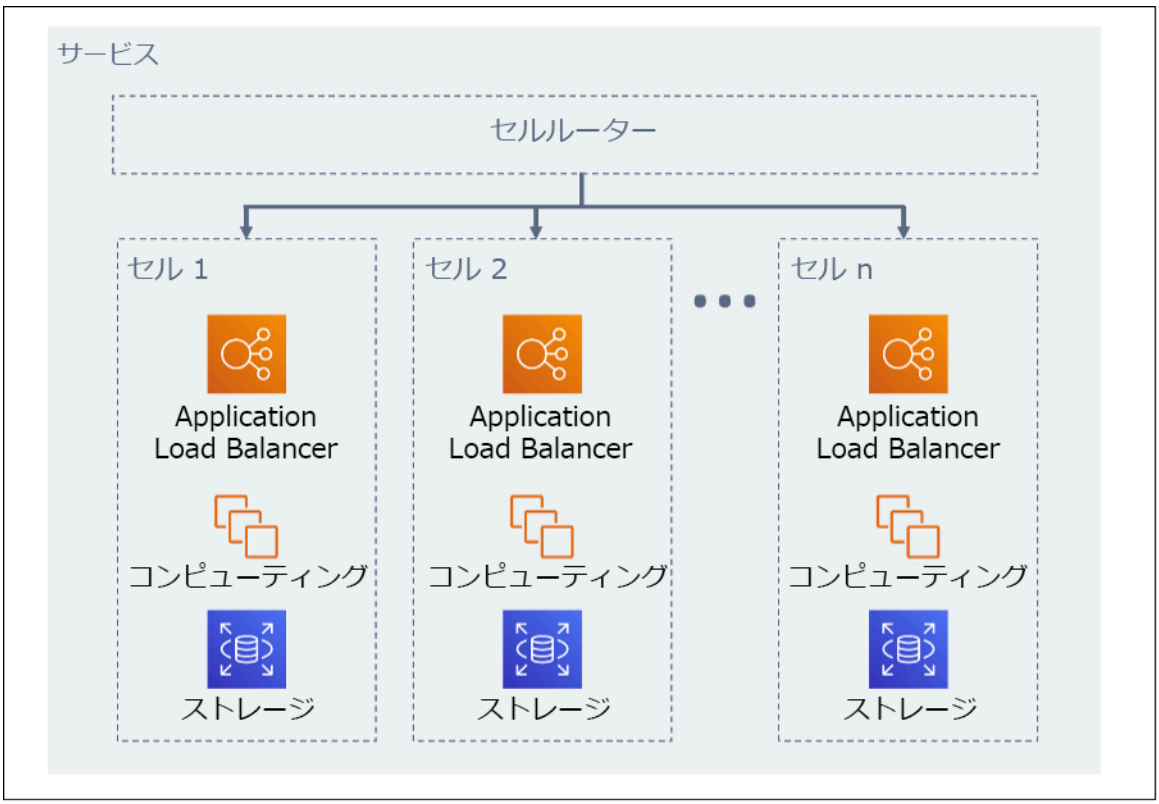


図 11: セルベースアーキテクチャ

実装手順

セルベースアーキテクチャを設計する場合、以下のとおり、考慮すべき設計上の考慮事項がいくつかあります。

1. パーティションキー: パーティションキーを選択する際は、特に配慮が必要です。
   - このキーは、サービスの粒度またはサービスのワークロードを最小限のクロスセルインタラクションで分割できる自然な方法に沿って分割を行う必要があります。例としては、 `#####` ID や `####` ID `#####`.
   - パーティションキーは、あらゆるリクエストにおいて、直接またはその他のパラメータによって決定論的に容易に推測できる方法で使用できる必要があります。
2. 永続的なセルマッピング: アップストリームサービスは、リソースのライフサイクル全体にわたり、単一のセルとのみインタラクションを行う必要があります。
   - ワークロードによっては、あるセルから別のセルにデータを移行するためにセル移行戦略が必要となる場合があります。セルの移行が必要になる可能性のあるシナリオには、ワークロード内の特定のユーザーまたはリソースが過剰に増大して、専用のセルが必要になる、といった場合があります。

- セルは、セル間で状態またはコンポーネントを共有すべきではありません。
- つまり、セル間のインタラクションは回避するか、最小限に抑える必要があります。これは、このようなインタラクションにより、セル間の依存関係が形成され、障害分離による改善が妨げられるためです。

3. ルーターレイヤー: ルーターレイヤーはセル間の共有コンポーネントであるため、セルと同様の区分化戦略に従うことはできません。
   - ルーターレイヤーでは、パーティションマッピングアルゴリズムを計算効率の高い方法で使用して、リクエストを個別のセルに分散することをお勧めします。例えば、暗号化ハッシュ関数と合同算術を組み合わせてパーティションキーをセルにマップするなどの方法があります。
   - マルチセルへの影響を回避するには、ルーティングレイヤーを可能な限りシンプルかつ水平方向にスケーラブルなものとする必要があります。この場合、このレイヤー内での複雑なビジネスロジックは避ける必要があります。この方法には期待される動作をいつでも簡単に把握できるという利点があり、完全なテスト容易性が実現します。Colm MacCárthaigh が『Reliability, constant work, and a good cup of coffee (信頼性、動作の継続、一杯のコーヒー)』で説明しているとおり、シンプルな設計と一定の作業パターンは、信頼性の高いシステムを生み出し、アンチフラジャイルの低減につながります。
4. セルのサイズ: セルにはサイズ制限が必要であり、最大サイズを超えて拡大すべきではありません。
   - 最大サイズを特定するには、限界点に達して安全なオペレーションの限界が明らかになるまで徹底的にテストを実施する必要があります。テストプラクティスの実装方法の詳細については、以下を参照してください。 REL07-BP04 ワークロードの負荷テストを実施する
   - 全体的なワークロードの増大は、セルの追加によるべきです。これにより、需要の拡大に合わせてワークロードをスケーリングできるようになります。
5. マルチ AZ またはマルチリージョン戦略: さまざまな障害ドメインからの保護として、マルチレイヤーの回復力を活用する必要があります。
   - 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。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。詳細については、以下を参照してください 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.

6. コードのデプロイ: コードの変更をすべてのセルに同時にデプロイせずに、タイミングをずらしたコードのデプロイ戦略を優先する必要があります。
   - これにより、不適切なデプロイや人的エラーによる複数のセルでの障害発生可能性を最小限に抑えることができます。詳細については、「安全なハンズオフデプロイメントの自動化」を参照してください。

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: 高

リソース

関連するベストプラクティス:

- REL07-BP04 ワークロードの負荷テストを実施する
- 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

関連するドキュメント:

- Reliability, constant work, and a good cup of coffee (信頼性、動作の継続、一杯のコーヒー)
- AWS and Compartmentalization (AWS と区分化)
- シャッフルシャーディングを使用したワークロードの分離
- 安全なハンズオフデプロイメントの自動化

関連動画:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small (AWS re:Invent 2018: ループをクローズして柔軟に対応: サイズを問わず、システムを制御する方法)
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338) (AWS re:Invent 2018: AWS が障害の影響範囲を最小限に抑える方法 (ARC338))
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328) (AWS re:Invent 2020: Amazon Builders' Library の紹介 (DOP328))
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience (Summit ANZ 2021 - すべてが常に失敗する: 回復力に向けた設計)

関連する例:

- Well-Architected Lab - Fault isolation with shuffle sharding (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-BP07 可用性の目標と稼働時間のサービスレベルアグリーメント (SLA) を満たす製品を設計する

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、複数の口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 つの口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。

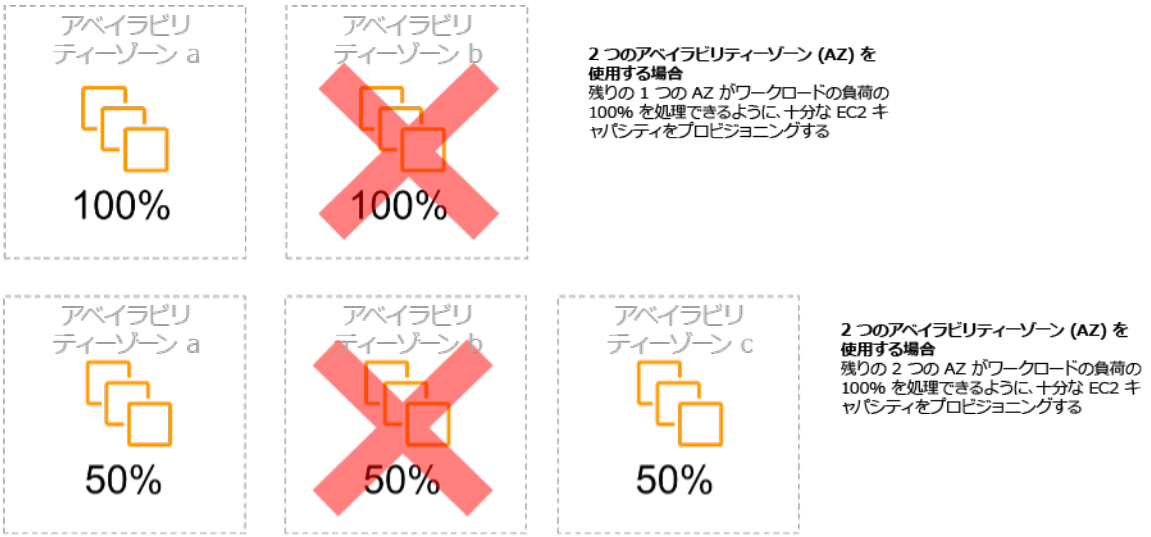


図 14: 複数のアベイラビリティゾーンにわたる EC2 インスタンスの静的安定性

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

REL11-BP07 可用性の目標と稼働時間のサービスレベルアグリーメント (SLA) を満たす製品を設計する

可用性の目標と稼働時間のサービスレベルアグリーメント (SLA) を満たすように製品を設計します。可用性目標またはアップタイム SLA を公開するか、非公開で同意する場合は、アーキテクチャと運用プロセスが SLA をサポートするように設計されていることを確認します。

期待される成果: 各アプリケーションには、可用性の目標とパフォーマンスメトリクスの SLA が定義されています。これらのメトリクスは、ビジネス上の成果を満たすためにモニタリングされ、維持されることがあります。

一般的なアンチパターン:

- SLA を設定せずにワークロードを設計およびデプロイする。
- 合理的な理由やビジネス要件なしに SLA メトリクスが高く設定されている。
- 依存関係とその基盤となる SLA を考慮せずに SLA を設定する。
- 回復力の共有責任モデルを考慮せずにアプリケーションが設計される。

このベストプラクティスを活用するメリット: 主要な復元力の目標に基づいてアプリケーションを設計すると、ビジネス目標と顧客の期待を満たすことができます。このような目標は、さまざまなテクノロジーを評価し、さまざまなトレードオフを考慮に入れたアプリケーション設計プロセスの促進につながります。

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

アプリケーションの設計では、ビジネス、オペレーション、財務上の目的に基づくさまざまな要件を考慮する必要があります。運用上の要件の範囲内で、ワークロードを適切にモニタリングおよびサポートできるように、ワークロードには特定の回復性メトリクス目標が必要です。ワークロードのデプロイ後は、回復性メトリクスの設定や派生の作成を行うべきではありません。これは設計段階で定義し、さまざまな決定とトレードオフのガイドとしての役割を果たす必要があります。

- 各ワークロードには、独自の回復性メトリクスセットが必要です。このようなメトリクスは、その他のビジネスアプリケーションとは異なる場合があります。
- 依存関係を低減すると、可用性にプラスの影響が生まれる可能性があります。各ワークロードは、独自の依存関係と SLA を考慮に入れる必要があります。通常、ワークロードの目標以上の可用性目標を持つ依存関係を選択します。

- 可能であれば、依存関係が損なわれてもワークロードが正常に動作できるように、疎結合の設計を検討します。
- コントロールプレーンの依存関係を減らします。特に、復旧時または機能低下時の依存関係を低減します。ミッションクリティカルなワークロードに対して静的安定性がある設計を評価します。リソースを節約して、ワークロード内のこのような依存関係の可用性を向上します。
- オブザーバビリティと計測は、平均検出時間 (MTTD) と平均修復時間 (MTTR) の短縮と SLA の達成のために重要です。
- 障害の頻度が低い (MTBF が長い)、障害検出時間が短い (MTTD が短い)、修復時間が短い (MTTR が短い) という 3 つの要素が、分散システムの可用性の向上のために使用されます。
- ワークロードの回復性メトリクスを確立し、それを満たすことは、効果的な設計の基本となります。このような設計では、設計の複雑性、サービスの依存関係、パフォーマンス、スケーリング、コストのトレードオフを考慮する必要があります。

実装手順

- 以下の質問を検討し、ワークロードの設計を確認して、文書化します。
  - コントロールプレーンはワークロードのどの個所で使用されますか。
  - ワークロードはどのように耐障害性を実装しますか。
  - どのようなスケーリング、自動スケーリング、冗長性、高可用性コンポーネントの設計パターンがありますか。
  - どのようなデータ整合性と可用性の要件がありますか。
  - リソース節約またはリソースの静的安定性に関する考慮事項はありますか。
  - どのようなサービスの依存関係がありますか。
- ステークホルダーと協力して、ワークロードアーキテクチャに基づいて SLA メトリクスを定義します。ワークロードで使用されるすべての依存関係の SLA を検討します。
- SLA 目標の設定後、SLA を満たすようにアーキテクチャを最適化します。
- SLA を満たす設計が定まったら、運用上の変更、プロセスの自動化、MTTD および MTTR の短縮についても重視するランブックを導入します。
- デプロイ後、SLA についてモニタリングしてレポートを作成します。

リソース

関連するベストプラクティス:

- REL03-BP01 ワークロードをセグメント化する方法を選択する
- REL10-BP01 複数の場所にワークロードをデプロイする
- 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
- REL11-BP03 すべてのレイヤーの修復を自動化する
- 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
- 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
- ワークロードの状態の把握

関連するドキュメント:

- Availability with redundancy (冗長性を備えた可用性)
- 可用性 - 信頼性の柱
- Measuring availability (可用性の測定)
- AWS Fault Isolation Boundaries (AWS の障害分離境界)
- 回復性に関する責任共有モデル
- アベイラビリティーゾーンを使用した静的安定性
- AWSAWS サービスレベルアグリーメント (SLA)
- Guidance for Cell-based Architecture on AWS(AWS のセルベースアーキテクチャについてのガイダンス)
- AWS infrastructure (AWS インフラストラクチャ)
- Advanced Multi-AZ Resiliance Patterns whitepaper (高度なマルチ AZ 回復性パターンについてのホワイトペーパー)

関連サービス:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

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。

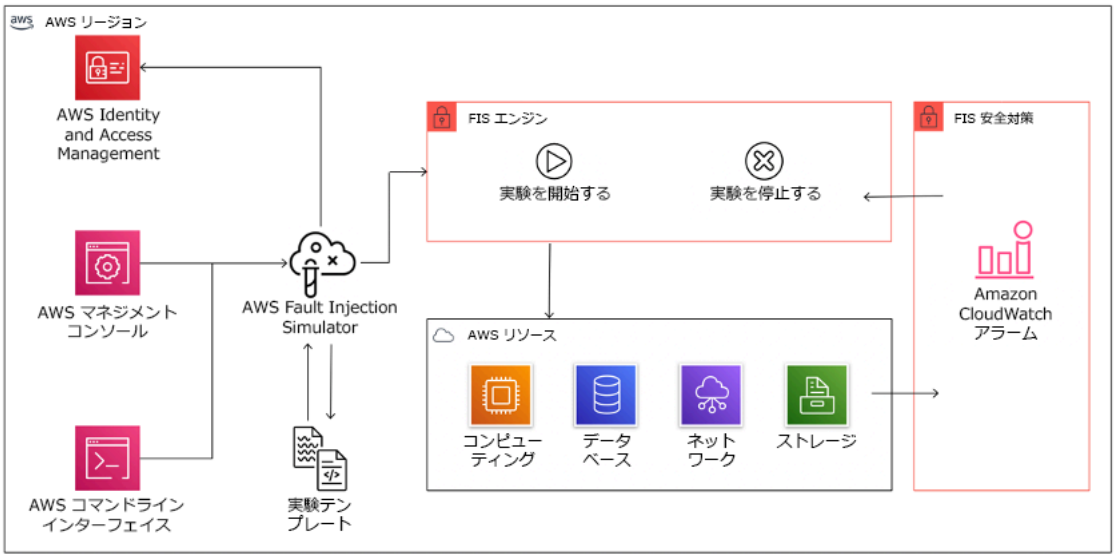
アプリケーションレベルの障害 (クラッシュなど) では、メモリや CPU の枯渇などのストレス要因から始めます。

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。

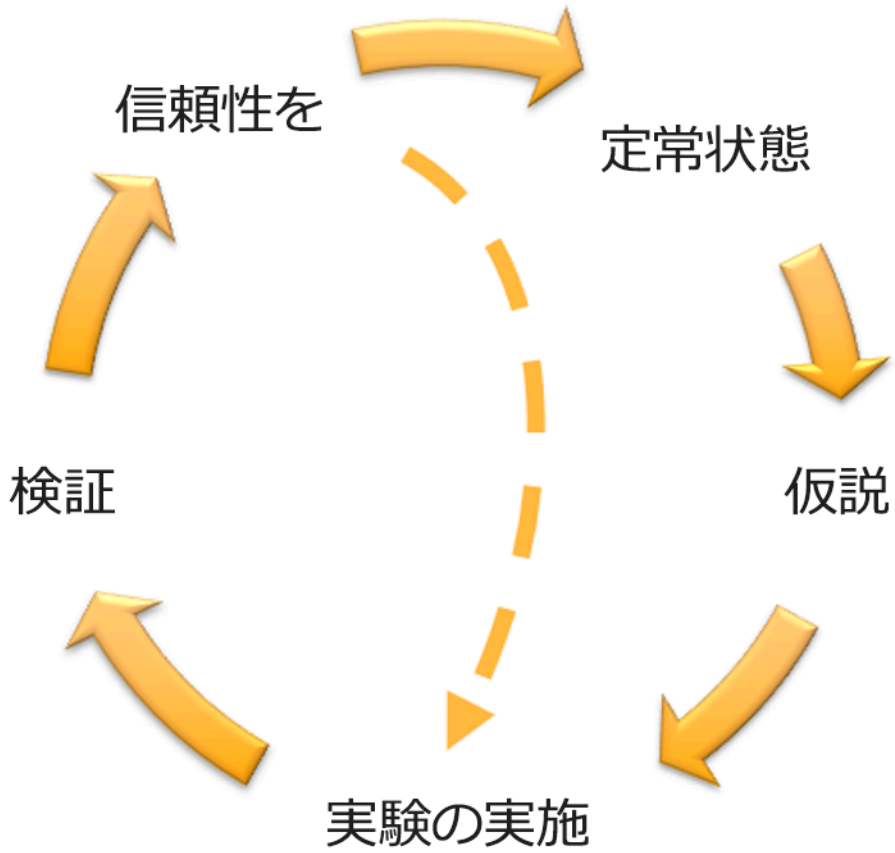
  ある障害の発生頻度を考慮する場合、利用可能であれば、そのワークロードの過去のデータを分析します。利用できない場合は、類似の環境において実行されている他のワークロードのデータを使用します。

  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、ワークロードを回復するための目標です。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)、またはアクティブ/アクティブなどの戦略を選択します。

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。
- ディザスタリカバリが計画されていない。
- 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と、チーム間のナレッジ共有と、チームが所有するワークロードでの DR の実装が改善します。

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

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。

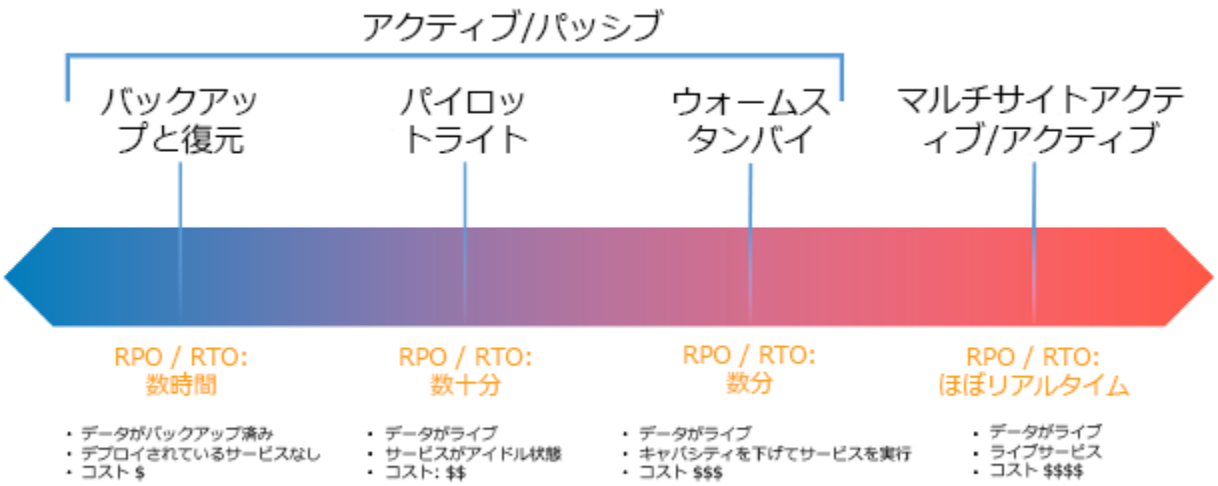


図 17: ディザスタリカバリ (DR) 戦略

- バックアップと復元 (RPO は時間単位、RTO は 24 時間以内): 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。その違いは、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 (すべてが既にデプロイされ、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。
> コストが懸念事項であり、ウォームスタンバイ戦略での定義と同様の RPO および RTO 目標の達成を目指す場合は、パイロットライトアプローチを採用して、改善された RPO および RTO 目標を提供する AWS Elastic Disaster Recovery などのクラウドネイティブソリューションを検討することができます。

実装手順

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) 、戦略を実装するコストと複雑性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。

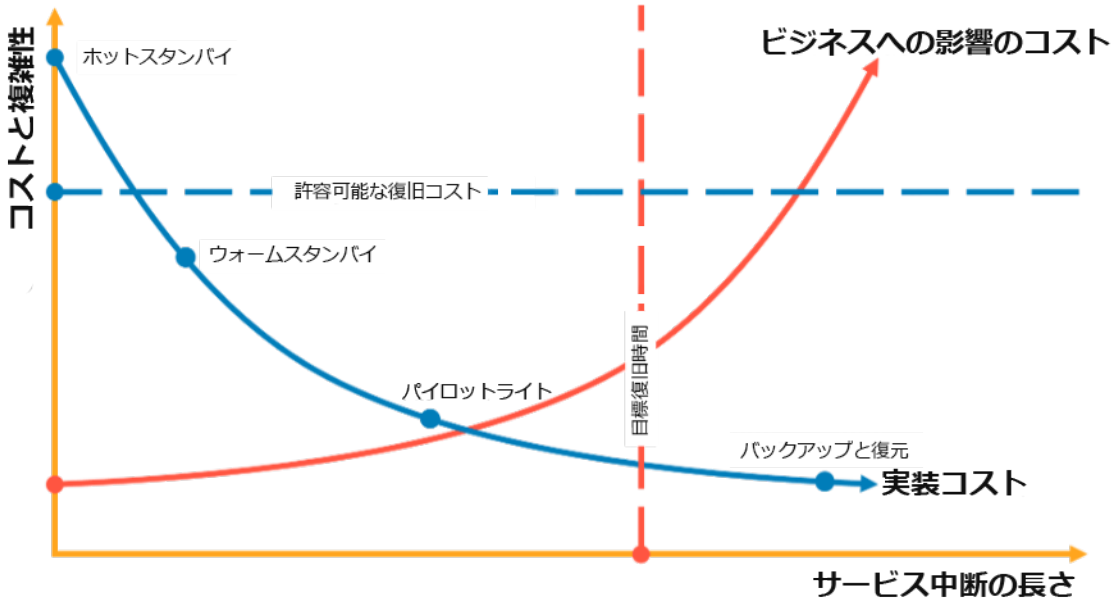


図 18: RTO とコストに基づく DR 戦略の選択を示すグラフ

詳細については、「Business Continuity Plan (BCP)」(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。

以下の手順では、戦略を特定のワークロードに適用できます。

バックアップと復元

バックアップと復元は、最も実装の複雑性が低い戦略であるとはいえ、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。

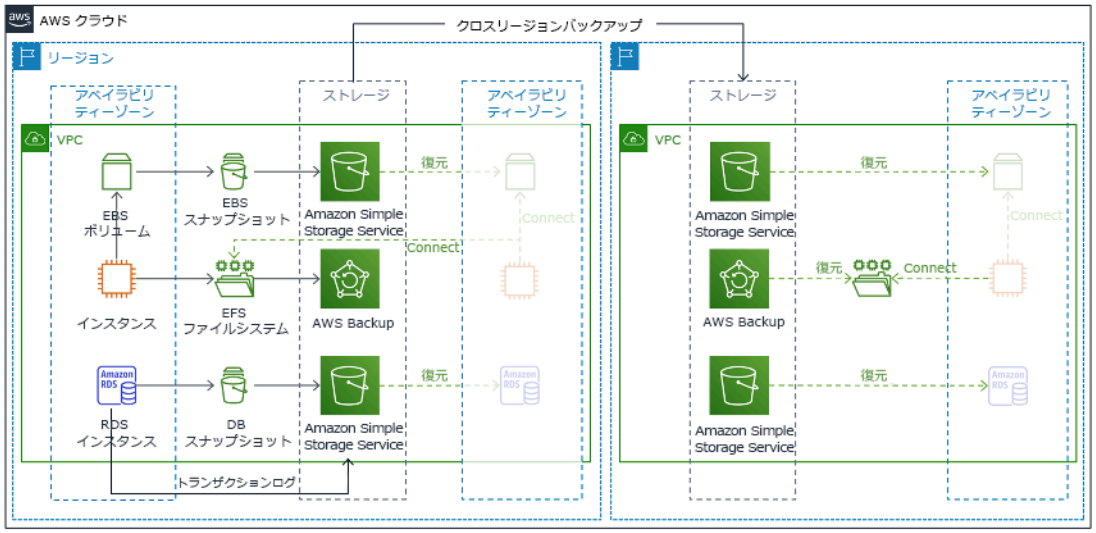


図 19: バックアップと復元アーキテクチャ

この戦略の詳細については、「AWS のディザスタリカバリ (DR) アーキテクチャ、パート II: 迅速なリカバリによるバックアップと復元」を参照してください。

パイロットライト

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。

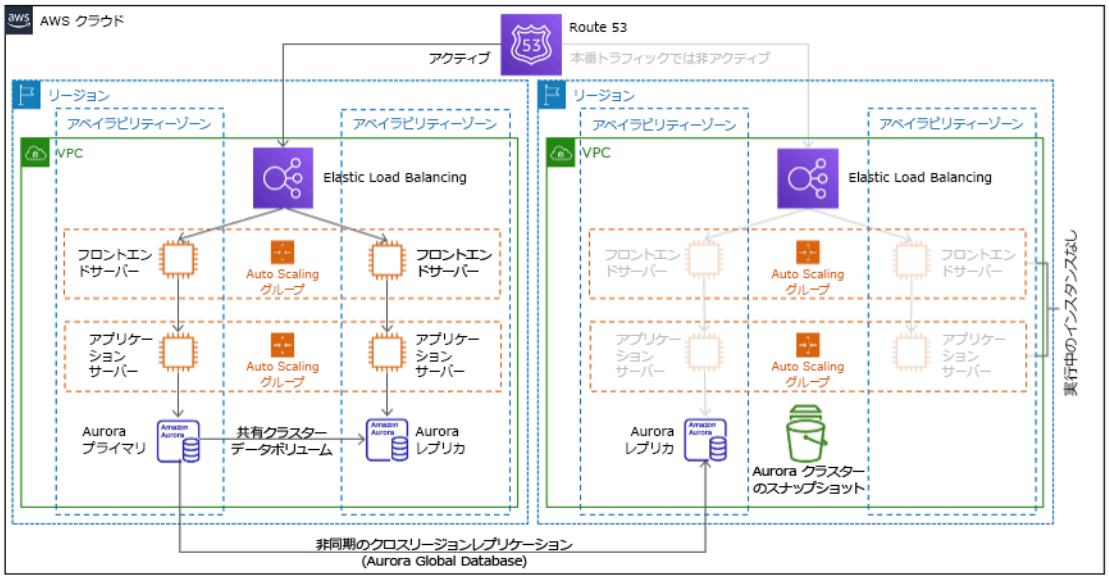


図 20: パイロットライトアーキテクチャ

この戦略の詳細については、「AWS 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」を参照してください。

ウォームスタンバイ

ウォームスタンバイのアプローチでは、本番稼働環境の完全に機能するスケールダウンした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。

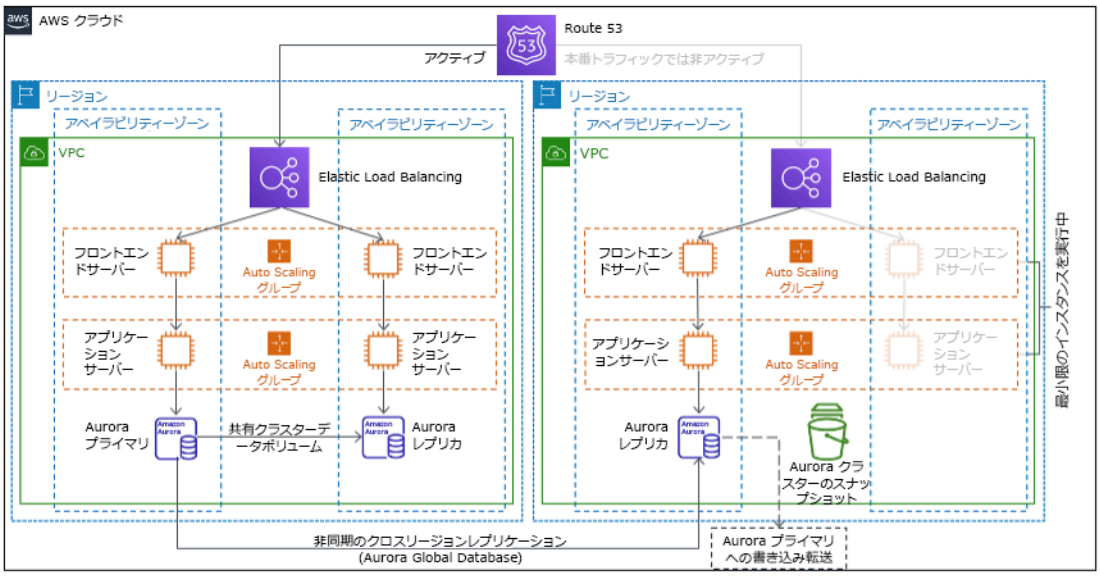


図 21: ウォームスタンバイアーキテクチャ

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利用可能であることを確認するには、EC2 インスタンスのキャパシティ予約の使用を検討します。AWS Lambda を使用する場合は、プロビジョニングした同時実行が、関数の呼び出しにすぐに応答できるように実行環境を提供します。

この戦略の詳細については、「AWS 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」を参照してください。

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環として、ワークロードを複数リージョンで同時に実行することができます。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。

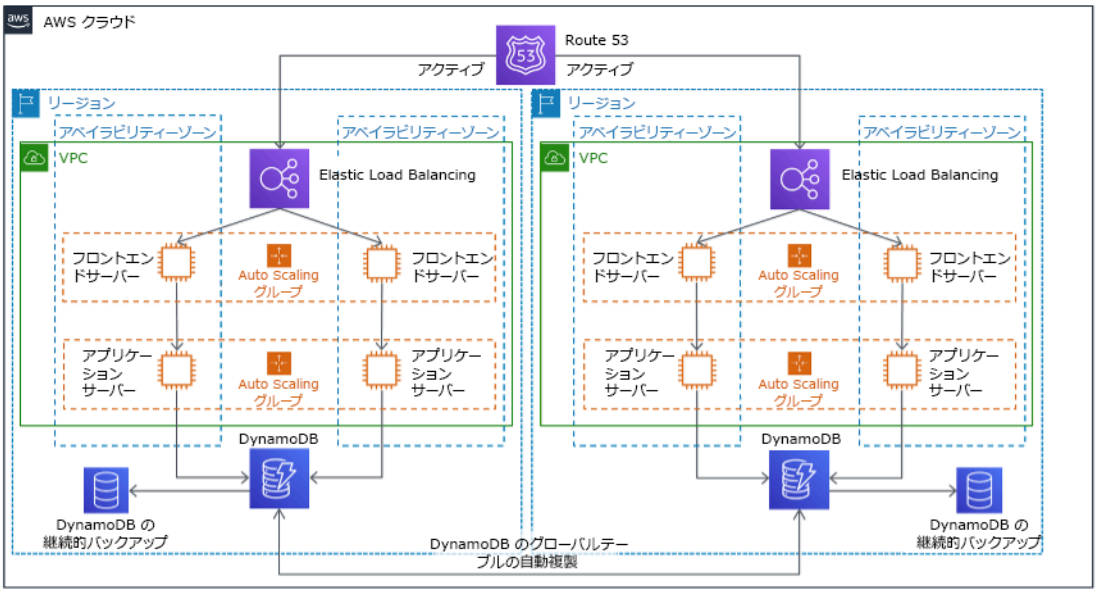


図 22: マルチサイトアクティブ/アクティブアーキテクチャ

この戦略の詳細については、「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」を参照してください。

AWS Elastic Disaster Recovery

ディザスタリカバリのためにパイロットライト戦略またはウォームスタンバイ戦略を検討している場合、AWS Elastic Disaster Recovery を使用すると、さらに優れた利点を提供する代替アプローチが提供される場合があります。Elastic Disaster Recovery は、ウォームスタンバイと同様の RPO および RTO 目標を提供することができ、パイロットライトの低コストアプローチを維持します。Elastic Disaster Recoveryは、継続的なデータ保護を使用して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、秒単位で測定される RPO と分単位で測定できる RTO を達成します。パイロットライト戦略と同様、データのレプリケートに必要となるリソースのみが復旧リージョンにデプロイされるため、コストが抑えられます。Elastic Disaster Recovery を使用する場合、サービスはフェイルオーバーまたはドリルの一環として開始されたコンピューティングリソースの回復を調整します。

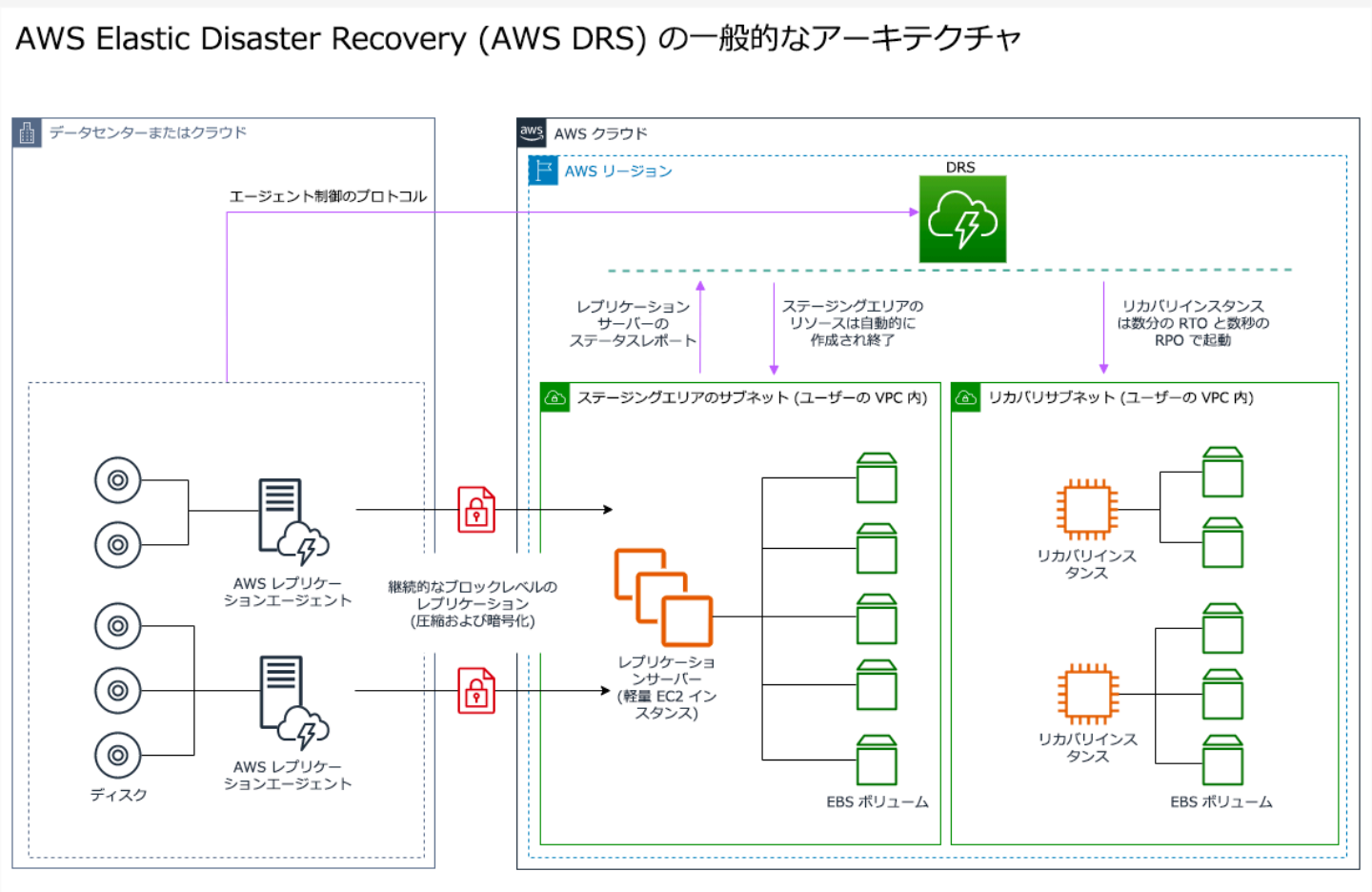


図 23: AWS Elastic Disaster Recoveryアーキテクチャ

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 内での複数のアベイラビリティーゾーン (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 に示されているとおり、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活用し、Amazon EC2 インスタンス と Elastic Load Balancer は複数の AZ にデプロイされたリ

ソースを備え、アクティブにリクエストを処理します。このアーキテクチャは、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 自体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するホットスタンバイについても説明しています。

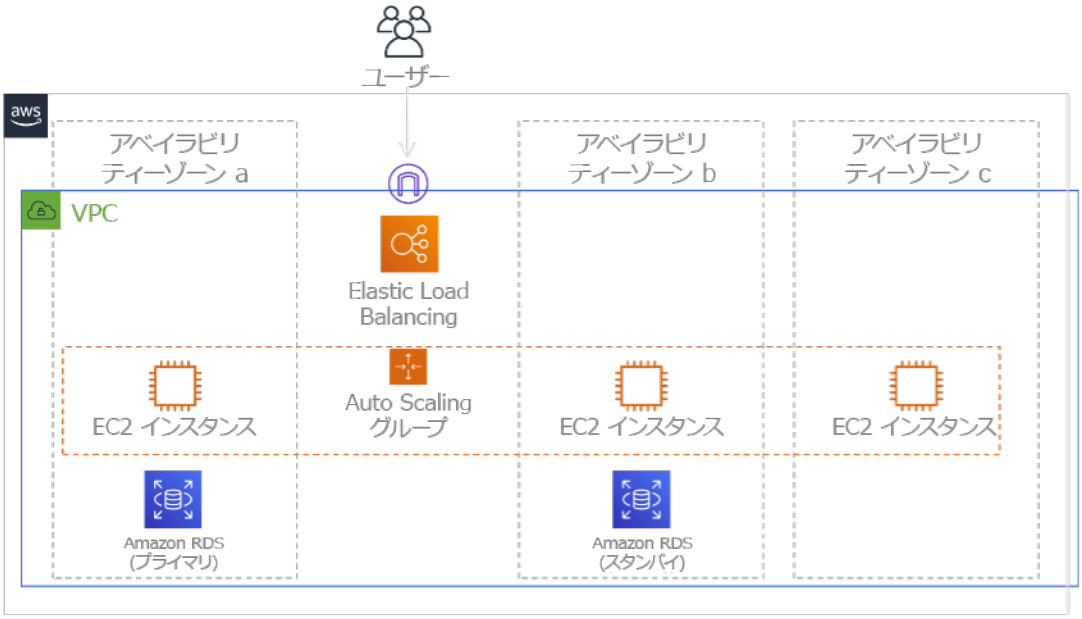


図 24: マルチ AZ アーキテクチャ

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、Amazon EBS ボリュームや Amazon Redshift クラスターなどの単一のゾーンに制約されているデータの場合に特に重要です。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。「Building highly resilient applications using Amazon Route 53 Application Recovery Controller, Part 1: Single-Region stack」(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。

> **Note**
>
> ワークロードによっては、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 や、Hashicorp Terraform のようなサードパーティーツールなどの Infrastructure as Code (IaC) を使用します。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 を使用します。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 のパラメータと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。Elastic Disaster Recovery を使用する場合、サービスがアプリケーション設定とコンピューティングリソースの復元をレプリケートおよび調整します。

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 テーブルなどにもレプリケートする必要があります。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、下記に関するこのブログシリーズを参照してください。Creating a Multi-Region Application with AWS Services (AWS サービスを使用した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)、サブネット、必要となるセキュリティグループに加え、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。方法の詳細については、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。オプションの 1 つに、Amazon Route 53 の使用があります。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。手動で開始する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。Application Recovery Controller 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.

このオプションおよび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に関する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。Amazon DynamoDB の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反映を再開します。Amazon Aurora Global Database を使って、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 の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、計画外のフェイルオーバーを前提として、Amazon Aurora Global Database でこれを行う方法の説明については、以下のラボを参照してください。Fail Back a Global Database (Global Database 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。

Elastic Disaster Recovery を使用する場合、サービスはフェイルバックプロセスの調整と自動化のサポートを提供します。詳細については、「Performing a failback」(フェイルバックの実行) 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:

- 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 Architecture Blog: Disaster Recovery Series (AWS アーキテクチャに関するブログ: 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: クラウド内での復旧 (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
- What Is AWS Backup? (AWS Backup とは)
- What is Route 53 Application Recovery Controller? (Amazon Route 53 Application Recovery Controller とは)
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started - AWS (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: products that can be used for disaster recovery (AWS Marketplace: ディザスタリカバリに活用できる商品)

関連動画:

- Disaster Recovery of Workloads on AWS (AWS 上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) (AWS re:Invent 2018: 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 (ARC209-R2))

- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services (AWS Elastic Disaster Recovery の使用を開始する | Amazon Web Services)

関連する例:

- Well-Architected Lab - Disaster Recovery - Series of workshops illustrating DR strategies (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、適切な動作と、RTO および RPO が満たされることを確認します。

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: ディザスタリカバリ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

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。

実装手順

1. 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します。復旧指向コンピューティングは、回復を強化するシステムの以下の特性を特徴としています。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、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。
2. Amazon EC2 ベースのワークロードの場合、AWS Elastic Disaster Recovery を使用して、DR 戦略のためのドリルインスタンスを導入して起動します。AWS Elastic Disaster Recovery を使用すると、フェイルオーバーイベントに備えて、ドリルを効率的に実行することができます。また、Elastic Disaster Recovery を使用すると、トラフィックをリダイレクトせずに、テストおよびドリル目的でインスタンスを頻繁に起動できます。

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 Architecture Blog: Disaster Recovery Series (AWS アーキテクチャに関するブログ: ディザスタリカバリシリーズ)
- AWS Marketplace: products that can be used for disaster recovery (AWS Marketplace: ディザスタリカバリに活用できる商品)
- AWS Elastic Disaster Recovery
- AWS でのワークロードのディザスタリカバリ: クラウド内での復旧 (AWS ホワイトペーパー)
- AWS Elastic Disaster Recovery Preparing for Failover (フェイルオーバーの準備)
- 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
- What is AWS Fault Injection Simulator? (AWS Fault Injection Simulator とは)

関連動画:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (AWS re:Invent 2018: 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)
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS (AWS re:Invent 2019: AWS を使用したバックアップと復元およびディザスタリカバリソリューション)

関連する例:

- Well-Architected Lab - Testing for Resiliency (Well-Architected ラボ - 回復力テスト)

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

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。実装に関する規範的なガイダンスについては、パフォーマンス効率の柱のホワイトペーパーを参照してください。

ベストプラクティス領域

- 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 サポート 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), Amazon Elastic Kubernetes Service (Amazon EKS)	AWS Lambda
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	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

タイプ	サーバー	コンテナ	関数
	めの専用オプションがある		
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。

        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、それらの特性に対するメモリ、ネットワーク、I/O、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。例えば、データベースなどのメモリ集約型のワークロードの場合、コアあたりのメモリ比率が高いと、多くの利点が期待できます。ただし、コンピューティング集約型のワークロードでは、より多くのコア数とコア周波数が必要になる一方、コアあたりのメモリ量はそれほど必要ない場合があります。

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、利用できるすべてのパフォーマンス特性の最大の値を持つ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。
- 特定のワークロードの実際の要件を検証せずに、標準的な合成ベンチマークに照らして最適化している。
- 新しいサービスを再評価して統合せずに、同じインフラストラクチャを長期間維持している。

このベストプラクティスを活用するメリット: ワークロードの要件を把握していれば、利用できるコンピューティングサービスとこのようなニーズを照らし合わせて、迅速に実験を行い、ワークロードのニーズを最も効率的に満たすソリューションを判断できます。これにより、不要なリソースに過大なコストを割くことなく、最適なパフォーマンスを実現できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

適切なサイジングを行い、ワークロードの設定を変更します。パフォーマンス、全体的な効率性、費用対効果を最適化するには、まずワークロードに必要なリソースを特定します。データベースなどのメモリ集約型のワークロードには、R ファミリーのインスタンスなどのメモリ最適化インスタンスを選択します。より高いコンピューティング能力を必要とするワークロードの場合は、C ファミリーのインスタンスを選択するか、コア数またはコア周波数が高いインスタンスを選択します。標準的な合成ベンチマークとの比較によってではなく、ワークロードのニーズに基づいて I/O パフォーマンスを選択します。より高い I/O パフォーマンスを得るには、I ファミリーのインスタンスを選択するか、I/O が最適化された Amazon EBS ボリュームを選択するか、インスタンスストアを持つインスタンスを選択します。特定のインスタンスタイプの詳細については、「Amazon EC2 インスタンスタイプ」を参照してください。

適切なサイジングを行うことで、不要なワークロードに対して余分な料金を支払うことなく、可能な限り最高レベルのパフォーマンスを達成できます。

実装手順

- ワークロードの特性を把握するか、ワークロードのリソース要件を分析します。
- ワークロードを個別に評価します。AWS クラウド を使用すると、妥協することなく、各ワークロードについて個別に適切なサイジングで調整できる柔軟性と俊敏性が得られます。
- テスト環境を作成して、ワークロードに最適なコンピューティングサービスを探します。
- 継続的に新しいコンピューティングサービスを再評価して、ワークロードのニーズと照らし合わせます。

- 費用対効果の向上のために、新しいサービスを定期的に確認します。
- 定期的に Well-Architected Framework レビューを実施します。

リソース

関連するベストプラクティス:

- 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-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

関連するドキュメント:

- AWS Compute Optimizer
- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- 関数: Lambda 関数の設定

関連動画:

- Amazon EC2 foundations (CMP211-R2) (Amazon EC2 の基礎 (CMP211-R2))
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1) (より良く、より速く、より安価なコンピューティング: Amazon EC2 でのコストの最適化 (CMP202-R1))
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1) (AWS Inferentia を使用して高パフォーマンスの機械学習推論を実現する (CMP324-R1))
- 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)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system (次世代 EC2 の強化: Nitro System の詳細)
- 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) (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)

関連する例:

- Rightsizing with Compute Optimizer and Memory utilization enabled (Compute Optimizer を使用したサイズ適正化とメモリ使用率の有効化)
- AWS Compute Optimizer Demo code (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。この伸縮性を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、ワークロードは変更に自動的に対応して、必要なリソースのみを使用することができます。

一般的なアンチパターン:

- 予想されるスパイクに対応するためにオーバープロビジョニングする。
- 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。
- プロビジョニングにかかる時間を考慮に入れずに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。
- ワークロードの実際の要件を直接反映していないメトリクスをモニタリングする。

このベストプラクティスを活用するメリット: 需要は、一定である場合も、変動する場合も、あるパターンに従う場合も、スパイクが発生しやすい場合もあります。需要と供給をマッチングすることで、ワークロードのコストを最低限に抑えることができます。ワークロードの伸縮性をモニタリング、テスト、設定することで、需要の変化に応じてパフォーマンス最適化、コスト低減、信頼性の向上を実現できます。これに対して手動で対応するアプローチも可能であるとはいえ、大規模な場合は非実用的です。自動化したメトリクスベースのアプローチを採用することで、どの時点でもリソースが確実に需要を満たすようにすることができます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

リソースの供給が、ワークロードが求めるリソースの需要とマッチすることを目標に、伸縮性を活用するために、メトリクスベースのオートメーションを使用する必要があります。例えば、リソースのモニタリングに Amazon CloudWatch メトリクスを使用したり、Auto Scaling グループには Amazon CloudWatch メトリクスを使用したりすることができます。

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。プロビジョニングにかかる時間と予想されるリソース障害を考慮に入れて計画を策定する必要があります。

インスタンス、コンテナ、関数には、伸縮性のためのメカニズムがあり、サービスの一部として、Application Auto Scaling で、または Amazon EC2 Auto Scaling と組み合わせて提供されます。アーキテクチャの伸縮性を活用して、広範囲の規模の使用における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があることを確認します。

デプロイされているワークロードのタイプに対して、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検証されていることを確認します。例えば、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する場合、CPU 使用率は 100% となることが想定されるため、主要な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は、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必要があります。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。

スケーリング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。

実装手順

- 履歴データを活用して、経時的なワークロードのリソース需要を分析します。以下のような具体的な事項を検討します。
  - ワークロードは安定しており、経時的に既知の割合で増加しているか。
  - ワークロードは、季節的な繰り返しのパターンで増減しているか。
  - ワークロードはスパイクが発生しやすいか。 スパイクは予想したり予測したりできるか。
- モニタリングサービスと履歴データを可能な限り活用します。
- リソースにタグ付けすると、モニタリングに役立ちます。タグを使用する場合は、「Tagging Best Practices (タグ付けのベストプラクティス)」を参照してください。また、タグを使用すると、リソースの管理、特定、整理に役立ちます。
- 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ベストプラクティス (COST09-BP01 ~ COST09-03) では、以下のアプローチを採用してコストを低減する方法について説明しています。
  - 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
- 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。
- 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。
- Amazon Elastic Block Store (Amazon EBS) を使用する場合のストレージのニーズについては、ボリュームベースの伸縮性を活用します。
- Amazon Elastic Compute Cloud (Amazon EC2) については、Auto Scaling グループの使用を検討してください。これにより、需要のスパイク発生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して、パフォーマンスとコストを最適化できます。

リソース

関連するベストプラクティス:

- 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-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

関連するドキュメント:

- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- 関数: Lambda 関数の設定

関連動画:

- Amazon EC2 foundations (CMP211-R2) (Amazon EC2 の基礎 (CMP211-R2))
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1) (より良く、より速く、より安価なコンピューティング: Amazon EC2 でのコストの最適化 (CMP202-R1))
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1) (AWS Inferentia を使用して高パフォーマンスの機械学習推論を実現する (CMP324-R1))

- 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)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system (次世代 EC2 の強化: Nitro System の詳細)

関連する例:

- Amazon EC2 Auto Scaling グループの例
- Amazon EFS 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

データ駆動型のアプローチを採用し、ワークロードのコンピューティングリソースを再評価して、最適化します。

期待される成果: システムレベルのメトリクスを使用して、経時的なワークロードの動作と要件を積極的にモニタリングします。収集したデータに基づいて、使用できるリソースと比較してワークロードの需要を評価し、ワークロードのプロファイルに最適なコンピューティング環境となるように変更を加えます。例えば、時間がたつにつれて、当初指定した以上にワークロードがメモリ集約型であることが判明する場合があります。この場合、別のインスタンスファミリーやインスタンスサイズに移行すると、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:

- ワークロードについてのインサイトを得るうえで、コンピューティングのニーズを再評価せずにシステムレベルのメトリクスをモニタリングしている。
- 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している。
- 代替のコンピューティングソリューションに移行する際、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過剰に増やしてワークロードの特性により効率的に適合させている。

このベストプラクティスを活用するメリット: 実際のデータと、コストおよびパフォーマンスの適切なバランスに基づいて最適化されたコンピューティングリソース。

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

データ駆動型アプローチを採用して、観測したワークロードの動作に基づいてコンピューティングリソースを最適化します。パフォーマンスと効率性を最大限に高めるには、時間の経過に伴ってワークロードから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。リソースが過剰にコミットされると、システムのパフォーマンスは低下します。リソースが適切に使用されないと、システムの運用効率は低下し、コストが増大します。

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、運用と利用に関するインサイトを得ることができます。

実装手順

1. コンピューティング関連の経時的メトリクスを収集します。
2. 選択したコンピューティングソリューションで使用できるリソースとワークロードのメトリクスを比較します。
3. 既存のソリューションのサイズを適正化するか、代替のコンピューティングソリューションを評価して、設定で必要となる変更を決定します。

リソース

関連する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

関連するドキュメント:

- AWS を使用したクラウドコンピューティング
- AWS Compute Optimizer
- EC2 インスタンスタイプ

- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- AWS Lambda 関数を使用するためのベストプラクティス

関連動画:

- Amazon EC2 foundations (CMP211-R2) (Amazon EC2 の基礎 (CMP211-R2))
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1) (より良く、より速く、より安価なコンピューティング: Amazon EC2 でのコストの最適化 (CMP202-R1))
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1) (AWS Inferentia を使用して高パフォーマンスの機械学習推論を実現する (CMP324-R1))
- 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)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system (次世代 EC2 の強化: Nitro System の詳細)
- Selecting and optimizing Amazon EC2 instances (Amazon EC2 インスタンスの選択と最適化)

関連する例:

- Rightsizing with Compute Optimizer and Memory utilization enabled (Compute Optimizer を使用したサイズ適正化とメモリ使用率の有効化)
- AWS Compute Optimizer Demo code (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 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	COTS アプリケーション、I/O 集約型アプリケーション、リ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	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

AWS のサービス	主な特徴	一般的なユースケース
	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	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 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 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 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 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 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 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

タイプ	AWS のサービス	主な特徴	一般的なユースケース
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、Amn Aurora	Amazon DynamoD	Amazon DocumentB	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspace	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	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	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	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
	自動スケーリングする	はプロビジョンドキャパシティモードでプロ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					はプロビジョンドキャパシティモードでプロ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	

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
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、Amn 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、An 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とアプリケーションの要件を使用して、使用する最適なデータサービスとテクノロジーを決定します。

期待される成果: 特定および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、必要に

応じた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、データベースエンジンを 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確立するメリット: 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し、パフォーマンス改善の機会が最適化します。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確立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 Services	Amazon RDS	Amazon Aurora	Amazon Dynamol	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
Scaling Storage	Storage can be scaled up manually or	Storage scales automati ally up to maximur	Storage automati ally scales. Tables are	Storage scales automati ally up to maximur	Storage is in-memory , tied to instance	Storage scales automati ally can grow	Organize your time series data to optimize	Scales table storage up and down automati	Storage automatic ally scales. Tables are

AWS Services	Amazon RDS	Amazon Aurora	Amazon Dynamol	Amazon Docume B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
	configure d to scale automati ally to a maximur of 64 TiB based on engine types. Provisior ed storage cannot be decrease .	of 128 TiB and decrease when data is removed Maximur storage size also depends upon specific Aurora MySQL or Aurora Postgres engine versions.	unconstr ined in terms of size.	of 64 TiB. Starting Amazon Docume B 4.0 storage can decrease by compara e amounts for data removal through dropping a collectio n or index. With Amazon Docume B 3.6 allocated space remains same and free space	type or count.	up to 128 TiB (or 64 TiB in few Regions) Upon data removal from, total allocated space remains same and is reused in the future.	query processi g and reduce storage costs. Retentio period can be configure d through in-memory and magnetic tiers.	ally as your applicati on writes, updates, and deletes data.	unconstra ined in terms of size.

AWS Services	Amazon RDS	Amazon Aurora	Amazon DynamoI	Amazon DocumeI B	Amazon ElastiCa he	Amazon Neptune	Amazon Timestre m	Amazon Keyspac	Amazon QLDB
				is reused when data volume increase .					

実装手順:

1. トランザクション、原子性、一貫性、分離、耐久性 (ACID) コンプライアンス、整合性のある読み込みの要件を理解します。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. グローバルに分散されているアプリケーションの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します。
3. クエリパターン、ランダムアクセスパターンおよび 1 回限りのクエリを分析します。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。
4. 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 に移しアーカイブするか、過去のデータを集計し分析するか、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。

5. 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) と一貫性に関するベースライン要件 (ACID および結果整合性)。
6. 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 (グローバルレプリケーション、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など) を文書化します。

実装計画に必要な工数レベル: 低。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必要な工数レベルは低です。

リソース

関連するドキュメント:

- AWS でのクラウドデータベース
- Amazon RDS DB インスタンスのストレージを使用する
- Amazon DocumentDB ストレージ
- AWS Database Caching
- Amazon Timestream ストレージ
- Amazon Keyspaces のストレージ
- Amazon ElastiCache に関するよくある質問
- Amazon Neptune ストレージ、信頼性、および可用性
- Amazon Aurora ベストプラクティス
- Amazon DynamoDB Accelerator
- Amazon DynamoDB ベストプラクティス
- Amazon RDS のストレージタイプ
- Amazon RDS インスタンスクラスのハードウェア仕様
- Aurora ストレージの制限

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)

関連する例:

- 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. ネットワークス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. ユーザーロ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 でオンプレミスのリソー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、接続オプション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キングを必要とするワークロードをデプロイする場合、専用接続や仮想プライベートネットワーク (VPN) などの複数の接続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:

- ワークロードのすべての要件 (帯域幅、レイテンシー、ジッター、暗号化、トラフィックのニーズ) を理解または特定できていない。
- 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。

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

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します。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。専用ネットワーク接続またはインターネットベースの VPN を使用する適切な接続オプションを検討します。

専用接続は、専用回線を介してネットワーク接続を確立し、一貫性あるパフォーマンスを達成しながら、高帯域幅、低レイテンシーが求められる場合に適しています。VPN 接続は、インターネット上で安全な接続を確立し、既存のインターネット接続を使用して暗号化した接続が必要となる場合に適しています。

帯域幅の要件によっては、単一の VPN 接続や専用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の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。

実装手順

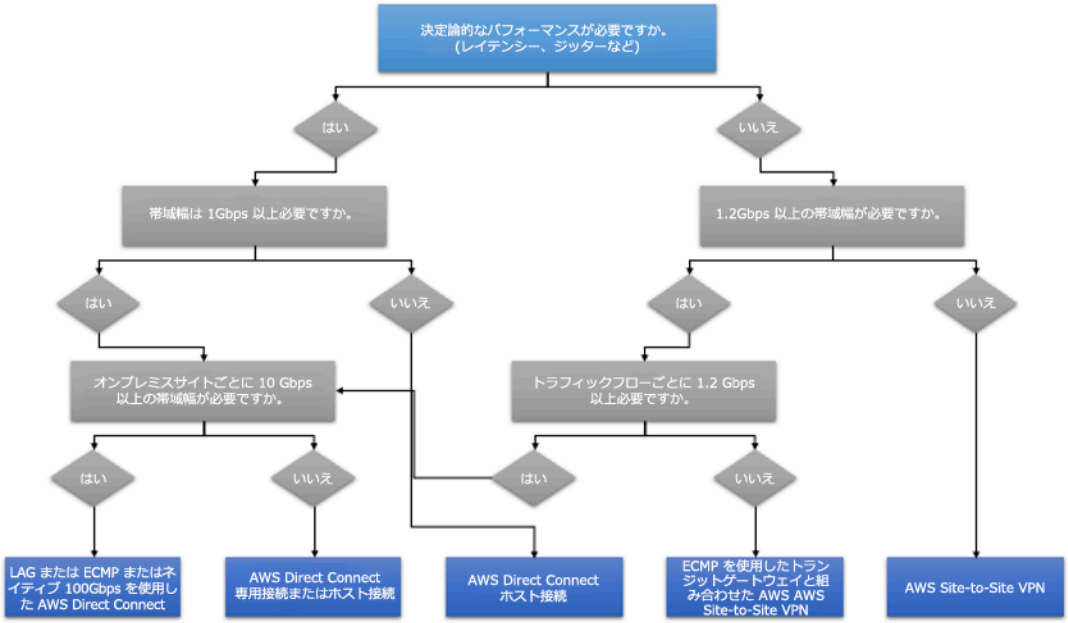
1. ハイブリッドアプリケーションの帯域幅とレイテンシー要件を見積もります。
    a. AWS に移行している既存のアプリケーションについては、内部ネットワークモニタリングシステムからのデータを活用します。

b. モニタリングデータがない新しいアプリケーションまたは既存のアプリケーションについては、プロダクトオーナーと相談のうえ、適切なパフォーマンスメトリクスを導き出し、優れたユーザーエクスペリエンスを提供します。

2. 接続オプションとして専用接続または VPN 接続を選択します。すべてのワークロード要件 (暗号化、帯域幅、トラフィックのニーズ) に基づいて、AWS Direct Connect か AWS Site-to-Site VPN (またはその両方) を選択できます。下の図は、適切な接続タイプの選択に役立ちます。

   a. 専用接続を検討する場合は、AWS Direct Connect が必要になる場合があります。AWS Direct Connect のプライベートネットワーク接続を使用すると、パフォーマンスの予測可能性と一貫性が向上します。AWS Direct Connect は、専用接続またはホスト接続のいずれかを使用して、50 Mbps から最大 100 Gbps までの AWS 環境への専用接続を提供します。これにより、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が提供され、ワークロードが効率の良い方法でその他の環境に接続できます。AWS Direct Connect パートナーを使用すると、複数の環境からエンドツーエンドの接続を確立し、一貫性あるパフォーマンスで拡張ネットワークを実現できます。AWS を使用すると、ネイティブ 100 Gbps、Link Aggregation Group (LAG)、または BGP Equal-cost multipath (ECMP) のいずれかを使用して、直接接続の帯域幅をスケーリングできます。

   b. VPN 接続を検討する場合は、AWS マネージド VPNのオプションをお勧めします。AWS Site-to-Site VPN は、インターネットプロトコルセキュリティ (IPsec) をサポートするマネージド VPN サービスを提供します。VPN 接続が作成されると、高可用性に向けて 各 VPN 接続に 2 つのトンネルが含まれています。AWS Transit Gateway を使用すると、複数の VPC 間の接続を簡素化し、単一の VPN 接続で AWS Transit Gateway にアタッチされた任意の VPC に接続できます。AWS Transit Gateway を使用すると、複数の VPN トンネルでの等コストマルチパス (ECMP) ルーティングサポートを有効化して、1.25Gbps の IPsec VPN スループット制限以上の拡張もできます。



決定論的なパフォーマンスについてのフローチャート

実装計画に必要な工数レベル: 高。ハイブリッドネットワークのワークロードニーズを評価し、ハイブリッドネットワーキングソリューションを実装するには、多大な労力が必要です。

リソース

関連するドキュメント:

- 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
- AWS Site-to-Site VPN
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure (スケーラブルで安全なマルチ VPC AWS ネットワークインフラストラクチャの構築)
- AWS Direct Connect
- Client VPN

関連動画:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1) (AWS への接続性とハイブリッド AWS EC2 ネットワークアーキテクチャ (NET317-R1))
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1) (Amazon EC2 インスタンスのネットワークパフォーマンス最適化 (CMP308-R1))
- AWS Global Accelerator
- AWS Direct Connect
- Transit Gateway Connect
- VPN Solutions
- Security with VPN Solutions

関連する例:

- AWS Transit Gateway and Scalable Security Solutions (AWS Transit Gateway とスケーラブルな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-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

ロードバランサーを使用して、ターゲットリソースの最適なパフォーマンス効率を実現し、システムの応答性を向上します。

期待される成果: トラフィックを処理するコンピューティングリソースの数を低減します。ターゲットにおけるリソース消費の不均衡を回避します。高度なコンピューティング集約型タスクをロードバランサーにオフロードします。クラウドの伸縮性と柔軟性を活用して、パフォーマンスを向上し、アーキテクチャを最適化します。

一般的なアンチパターン:

- ロードバランサーの種類を選択する際にワークロードの要件を考慮していない。
- パフォーマンスの最適化のためにロードバランサー機能を活用していない。
- ロードバランサーを介さずにワークロードをインターネットに直接公開している。

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

ロードバランサーは、ワークロードのエントリポイントとして機能し、そこからコンピューティングインスタンスやコンテナなどのバックエンドターゲットにトラフィックを分散します。適切なロードバランサーの種類を選択することが、アーキテクチャを最適化するうえでの最初のステップとなります。

まず、プロトコル (TCP、HTTP、TLS、または WebSocket など)、ターゲットの種類 (インスタンス、コンテナ、またはサーバーレスなど)、アプリケーション要件 (長時間実行される接続、ユーザー認証、またはスティッキーなど) や配置 (リージョン、ローカルゾーン、アウトポスト、ゾーン分離など) のワークロードの特性をリストアップすることから始めます。

適切なロードバランサーを選択したら、ロードバランサーの機能を活用して、バックエンドが必要とするトラフィック処理のための労力を削減できます。

例えば、Application Load Balancer (ALB) と Network Load Balancer (NLB) の両方を使用して、SSL/TLS 暗号化オフロードを実行できます。これにより、CPU 負荷が高い TLS ハンドシェイクのターゲットによる終了を回避して、証明書管理を改善する機会が得られます。

ロードバランサーで SSL/TLS オフロードを設定すると、暗号化されていないトラフィックをバックエンドに配信すると同時に、クライアントとの間のトラフィックの暗号化の役目を果たすため、バックエンドリソースが解放され、クライアントにとっての応答時間が改善されます。

Application Load Balancer は、ターゲットでのサポートを必要とせずに HTTP2 トラフィックを処理することもできます。このようなシンプルな決断をすることで、HTTP2 は TCP 接続をより効率的に使用するようになり、アプリケーションの応答時間を改善できます。

ロードバランサーを使用すると、コンテナやサーバーレスなどのさまざまな種類のバックエンドにトラフィックを分散させて、アーキテクチャの柔軟性を向上することもできます。例えば、ヘッダー、メソッド、パターンなどのリクエストパラメータに基づいてトラフィックをさまざまなターゲットグループに転送するリスナールールを使用するように Application Load Balancer を設定できます。

アーキテクチャを定義する際は、ワークロードのレイテンシー要件を考慮する必要もあります。例えば、レイテンシーの影響を受けやすいアプリケーションがある場合、非常に低レイテンシーを実現する Network Load Balancer を使用するよう決定することができます。または、AWS Local Zones、あるいは AWS Outposts で Application Load Balancer を活用して、ワークロードをより顧客の近くに配置する方法を採用することもできます。

レイテンシーの影響を受けやすいワークロードに関して考慮すべきもう 1 つの点は、クロスゾーン負荷分散です。クロスゾーン負荷分散を使用すると、各ロードバランサーノードはトラフィックを有効化されたすべてのアベイラビリティーゾーン内の登録済みターゲットにトラフィックを分散しま

す。これにより、ラウンドトリップのレイテンシーが 1 桁ミリ秒単位で増加する場合があるとはいえ、可用性が改善します。

最後に、ALB と NLB の両方により、ログやメトリクスなどのモニタリングリソースが提供されます。モニタリングを適切に設定すると、アプリケーションのパフォーマンスに関するインサイトの収集に役立ちます。例えば、ALB アクセスログを使用して、他のものより応答時間がかかるリクエストや、パフォーマンスの問題を引き起こしているバックエンドターゲットを検出できます。

実装手順

1. ワークロードに適したロードバランサーを選択します。
    a. HTTP/HTTPS のワークロードには、Application Load Balancer を使用します。
    b. TCP または UDP で実行される HTTP 以外のワークロードには、Network Load Balancer を使用します。
    c. 両方の製品の機能を活用したい場合は、両方を組み合わせて (ALB を NLB のターゲットとして) 使用します。例えば、NLB の静的 IP を ALB からの HTTP ヘッダーベースのルーティングと組み合わせて使用する場合や、HTTP のワークロードを AWS PrivateLink に公開する場合に使用します。
    d. すべてのロードバランサー製品の比較については、「ELB の製品の比較」を参照してください。
2. SSL/TLS オフロードを使用します。
    a. AWS Certificate Manager と統合された Application Load Balancer と Network Load Balancer の両方で HTTPS/TLS リスナーを設定します。
    b. ワークロードによっては、コンプライアンス上の理由で、エンドツーエンドの暗号化が必要になる場合があることに注意します。この場合は、ターゲットで暗号化を有効にする必要があります。
    c. セキュリティのベストプラクティスについては、「SEC09-BP02 伝送中に暗号化を適用する」を参照してください。
3. 適切なルーティングアルゴリズムを選択します。
    a. ルーティングアルゴリズムは、バックエンドターゲットの使用状況に変化をもたらし、パフォーマンスへの影響を左右します。例えば、ALB には、以下の2 つのルーティングアルゴリズムのオプションがあります。
    b. 最小未処理リクエスト: アプリケーションのリクエストの複雑性が異なる場合や、ターゲットの処理能力が異なる場合に、バックエンドターゲットへのロードバランシングを改善するために使用します。

c. ラウンドロビン: リクエストとターゲットが同様の場合、またはリクエストをターゲット間で均等に分散する必要がある場合に使用します。

4. クロスゾーンまたはゾーン分離を検討します。

a. レイテンシーの改善とゾーンの障害ドメイン対策として、クロスゾーンをオフ (ゾーン分離) で使用します。NLB ではデフォルトでオフになっていますが、ALB ではターゲットグループごとにオフに設定できます。

b. 可用性と柔軟性の向上のためには、クロスゾーンをオンにします。ALB ではデフォルトでクロスゾーンはオンになっています。NLB ではターゲットグループごとにオフに設定できます。

5. HTTP ワークロードの HTTP キープアライブをオンにします。

a. HTTP ワークロードの場合、バックエンドターゲットのウェブサーバー設定で HTTP キープアライブをオンにします。この機能にを使用すると、ロードバランサーはキープアライブタイムアウトが期限切れになるまでバックエンド接続を再利用できるため、HTTP リクエストと応答時間が改善され、バックエンドターゲットでのリソース使用率も低減します。Apache と Nginx でこれを実行する方法の詳細については、「ELB のバックエンドサーバーとして Apache または NGINX を使用するための最適な設定を教えてください。」を参照してください。

6. コンピューティングリソースのオーケストレーションを改善するには、Elastic Load Balancing 統合を使用します。

a. ロードバランサーと統合された Auto Scaling を使用します。パフォーマンス効率に優れたシステムの重要な側面の 1 つに、バックエンドリソースのサイズの適切な設定があります。これを実現するには、バックエンドターゲットリソースのロードバランサー統合を利用できます。Auto Scaling グループと統合したロードバランサーを使用することで、受信トラフィックに対応して、必要に応じてターゲットがロードバランサーに追加されたり削除されたりします。

b. コンテナ化されたワークロードには、ロードバランサーを Amazon ECS および Amazon EKS と統合できます。

- Auto Scaling グループ内のインスタンス全体にトラフィックを分散するには、Elastic Load Balancing を使用します
- Amazon ECS - サービスの負荷分散
- Amazon EKS でのアプリケーション負荷分散
- Amazon EKS でのネットワーク負荷分散

7. ロードバランサーをモニタリングして、パフォーマンスのボトルネックを検出します。

a. Application Load Balancer と Network Load Balancer のアクセスログを有効にします。

b. ALB の場合に考慮すべき主なフィールドは以下のとおりです。`request_processing_time`、`request_processing_time`、`response_processing_time`.

c. NLB の場合に考慮すべき主なフィールドは以下のとおりです。 `connection_time` および `tls_handshake_time`.

d. 必要な際にログをクエリできるように準備を整えておきます。Amazon Athena を使用すると、ALB ログと NLB ログの両方をクエリできます。

e. ALB の`TargetResponseTime` などのパフォーマンス関連メトリクスのアラームを作成します。

リソース

関連するベストプラクティス:

- SEC09-BP02 伝送中に暗号化を適用する

関連するドキュメント:

- ELB 製品の比較
- AWS グローバルインフラストラクチャ
- Improving Performance and Reducing Cost Using Availability Zone Affinity (アベイラビリティーゾーンのアフィニティを使用したパフォーマンスの向上とコストの削減)
- Step by step for Log Analysis with Amazon Athena (Amazon Athena を使用したログ分析の詳しい手順)
- Application Load Balancer ログのクエリ
- Application Load Balancer を監視する
- Network Load Balancer を監視する

関連動画:

- AWS re:Invent 2018: [REPEAT 1] Elastic Load Balancing: Deep Dive and Best Practices (NET404-R1) (AWS re:Invent 2018: [リピート 1] Elastic Load Balancing: ディープダイブとベストプラクティス (NET404-R1))
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads (AWS re:Invent 2021 - AWS のワークロードに適切なロードバランサーを選択する方法)

- AWS re:Inforce 2022 - How to use Elastic Load Balancing to enhance your security posture at scale (NIS203) (AWS re:Inforce 2022 - セキュリティ体制の大規模な強化のために Elastic Load Balancing を使用する方法 (NIS203))
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads (NET407-R2) (AWS re:Invent 2019: さまざまなワークロードに Elastic Load Balancing を最大限に活用する (NET407-R2))

関連する例:

- CDK and CloudFormation samples for Log Analysis with Amazon Athena (Amazon Athena を使用したログ分析の CDK とCloudFormation の例)

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要件を評価し、ワークロードの全体的なパフォーマンスを最適化するネットワーク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。例えば、ファイル転送で Transmission Control Protocol (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や、User Datagram Protocol (UDP) を使用するものもあります。

Scalable reliable datagram (SRD) プロトコルは、Elastic Fabric Adapter 向けに AWS が構築したネットワークトランスポートプロトコルであり、信頼性に優れたデータグラム配信を実現します。TCP プロトコルとは異なり、SRD はパケットを並べ替えて、順不同で配信できます。SRD によるこの順不同の配信メカニズムは、代替パス経由でパケットを並行送信して、スループットを向上します。

一般的なアンチパターン:

- パフォーマンス要件を問わず、すべてのワークロードに TCP を使用している。

このベストプラクティスを活用するメリット:

- 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。
- ユーザーと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が使用されていることを確認すると、アプリケーションのユーザーエクスペリエンスが全体的に向上します。例えば、TCP と

UDP の両方を組み合わせて使用すると、VDI ワークロードは、重要なデータには TCP の信頼性、リアルタイムデータには UDP の速度を利用できます。

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 (適切でないネットワークプロトコルを使用すると、応答時間の遅延、高レイテンシー、貧弱なスケーラビリティなどのパフォーマンスの低下につながる可能性があります)

実装のガイダンス

レイテンシーとスループットの要件を理解し、パフォーマンスを最適化するネットワークプロトコルを選択することが、ワークロードのパフォーマンスを向上するうえで重要となる考慮事項です。

TCP の使用を検討するべきケース

TCP は信頼性に優れたデータ配信を提供し、データの信頼性と確実な配信が重要となるワークロードのコンポーネント間の通信に利用できます。多くのウェブベースのアプリケーションは、AWS 上のサーバーとの通信で TCP ソケットを開くうえで、HTTP や HTTPS などの TCP ベースのプロトコルを基盤としています。TCP にはデータ交換の速度とネットワークの輻輳を制御する機能があるため、E メールとファイルデータ転送も、TCP を利用する一般的なアプリケーションです。TCP で TLS を使用すると、通信に多少のオーバーヘッドが追加されるため、レイテンシーが増加し、スループットが低下する場合があります。このオーバーヘッドは主にハンドシェークプロセスの追加オーバーヘッドにより発生し、完了するまでに数回のラウンドトリップが必要になる場合があります。ハンドシェイクが完了すると、データの暗号化と復号化のオーバーヘッドは比較的少なくなります。

UDP の使用を検討するべきケース

UDP はコネクションレス型のプロトコルであるため、ログ、モニタリング、VoIP データなどの、高速かつ効率的な転送が必要なアプリケーションに適しています。また、多数のクライアントからの小型のクエリに対応するワークロードのコンポーネントがある場合は、ワークロードの最適なパフォーマンスを確保するために UDP の使用を検討します。UDP では、Datagram Transport Layer Security (DTLS) が TLS に相当します。UDP で DTLS を使用する場合、ハンドシェイクプロセスは簡素化され、オーバーヘッドはデータの暗号化と復号化から発生します。また、DTLS には、セキュリティパラメータを提示し、改ざんを検出するための追加フィールドが含まれているため、UDP パケットに少量のオーバーヘッドが追加されます。

SRD の使用を検討するべきケース

Scalable Reliable Datagram (SRD) は、複数のパス間でトラフィックの負荷分散を行い、パケットドロップやリンク障害から迅速に回復できる機能を備えており、高スループットのワークロード向けに

最適化されたネットワークトランスポートプロトコルです。そのため、SRD は、コンピューティングノード間で高スループットと低レイテンシー通信を必要とするハイパフォーマンスコンピューティング (HPC) ワークロードに最適です。ノード間で大量のデータ転送を伴うシミュレーション、モデリング、データ分析などの並列処理タスクなどで利用される場合があります。

実装手順

1. オンラインファイル転送アプリケーションのスループット向上のために、AWS Global Accelerator と AWS Transfer Family サービスを使用します。AWS Global Accelerator サービスを使用すると、クライアントデバイスと AWS 上のワークロード間のレイテンシーが低減されます。AWS Transfer Family では、Secure Shell File Transfer Protocol (SFTP) と File Transfer Protocol over SSL (FTPS) などの TCP ベースのプロトコルを使用して、AWS ストレージサービスへのファイル転送を安全にスケーリングおよび管理できます。
2. ワークロードコンポーネント間の通信に TCP が適切かどうかを判断するには、ネットワークレイテンシーに注目します。クライアントアプリケーションとサーバー間のネットワークレイテンシーが高い場合、3 方向ハンドシェイクに時間がかかり、アプリケーションの応答性に影響を与える可能性があります。Time to First Byte (TTFB) や Round-Trip Time (RTT) などのメトリクスを使用すると、ネットワークレイテンシーを測定できます。ワークロードが動的コンテンツをユーザーに提供している場合は、Amazon CloudFront の使用を検討します。これにより、動的コンテンツのそれぞれのオリジンへの永続的な接続が確立され、各クライアントリクエストの速度を低下させる接続設定時間が必要なくなります。
3. TCP や UDP で TLS を使用すると、暗号化と復号化の影響により、レイテンシーが増大し、ワークロードのスループットが低下する可能性があります。このようなワークロードの場合、バックエンドインスタンスではなく、ロードバランサーで SSL/TLS 暗号化および復号化プロセスの処理を行ってワークロードのパフォーマンスを向上させるために、Elastic Load Balancing での SSL/TLS オフロードを検討します。これは、バックエンドインスタンスの CPU 使用率の低減、パフォーマンスの向上、キャパシティー増大につながります。
4. 認証と認可、ロギング、DNS、IoT、ストリーミングメディアなどの UDP プロトコルに依存するサービスを展開し、ワークロードのパフォーマンスと信頼性を向上させるには、Network Load Balancer (NLB) を使用します。NLB は着信 UDP トラフィックを複数のターゲットに分散するため、ワークロードの水平方向のスケーリング、キャパシティの増大、単一のターゲットのオーバーヘッドの低減につながります。
5. ハイパフォーマンスコンピューティング (HPC) ワークロードの場合は、SRD プロトコルを使用する Elastic Network Adapter (ENA) Express 機能を利用することを検討します。この機能は、EC2 インスタンス間のネットワークトラフィックでシングルフロー帯域幅を増大し (25Gbps)、テールレイテンシーを短縮して (99.9 パーセンタイル)、ネットワークパフォーマンスを向上させます。

6. ワークロードコンポーネント間または gRPC 対応のクライアントとサービス間で gRPC (リモートプロシージャコール) トラフィックをルーティングして負荷分散するには、Application Load Balancer (ALB) を使用します。gRPC は TCP ベースの HTTP/2 プロトコルをトランスポートに使用しており、ネットワークフットプリントの軽量化、圧縮、効率的なバイナリ形式のシリアル化、多言語サポート、双方向ストリーミングなどのパフォーマンス上の利点が得られます。

リソース

関連するドキュメント:

- 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
- 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 ネットワークアーキテクチャとハイブリッド AWS ネットワークアーキテクチャへの接続性 (NET317-R1))
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1) (Amazon EC2 インスタンスのネットワークパフォーマンス最適化 (CMP308-R1))
- Tuning Your Cloud: Improve Global Network Performance for Application (クラウドのチューニング: アプリケーションのグローバルネットワークパフォーマンスの向上)
- Application Scaling with EFA and SRD (EFA と SRD を使用したアプリケーションのスケーリング)

関連する例:

- AWS Transit Gateway and Scalable Security Solutions (AWS Transit Gateway とスケーラブルな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-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

ネットワークのレイテンシー短縮、スループット向上、ページの読み込み時間とデータ転送時間の短縮による最適なユーザーエクスペリエンス提供に向けて、リソースプレイスメントのオプションを評価します。

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 EC2 インスタンスなどのリソースは、AWS リージョン、AWS Local Zones、AWS Outposts、または AWS Wavelength ゾーンに配置します。このロケーション選択が、特定のユーザーのロケーションからのネットワークレイテンシーとスループットに影響を及ぼします。Amazon CloudFront と AWS Global Accelerator などのエッジサービスは、エッジロケーションにコンテンツをキャッシュするか、AWS グローバルネットワークを介してユーザーにワークロードへの最適なパスを提供することにより、ネットワークパフォーマンス向上のために利用できます。

実装手順

1. 以下の主な要素に基づいて、デプロイに適切な AWS リージョン (単一または複数) を選択します。
   a. ユーザーの所在地: ワークロードのユーザーに近いリージョンを選択して、ユーザーがワークロードを使用する際のレイテンシーを低く抑えます。
   b. 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、レイテンシーがデータ転送の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。
   c. 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 (例えば、データを保存する場所の要件)。
2. 特定のワークロードについて、コンポーネントが低レイテンシーを必要とする相互に依存する Amazon EC2 インスタンスのグループで構成されている場合は、ワークロードの要件を満たすうえでこのようなインスタンスの配置に影響を及ぼすために、クラスタープレイスメントグループの使用を検討します。同じクラスタープレイスメントグループ内のインスタンスでは、TCP/IP トラフィックのフローごとのスループット制限が高くなり、ネットワークで同じ高二分帯域幅セグメントに配置されます。クラスタープレイスメントグループは、ネットワークレイテンシーが

低い場合、ネットワークスループットが高い場合、またはその両方の場合にメリットがあるアプリケーションで使用することをお勧めします。

3. 低レイテンシーやデータ常駐要件など、ロケーションに依存するワークロードの場合は、AWS Local Zones または AWS Outposts を検討します。

    a. AWS Local Zones は、コンピューティング、ストレージ、データベース、およびその他の厳選された AWS サービスを人口密集地および産業の中心地の近くに配置するインフラストラクチャのデプロイタイプの 1 つです。

    b. AWS Outposts は、ほぼすべてのオンプレミスまたはエッジロケーションで真に一貫性あるハイブリッドエクスペリエンスを実現する AWS インフラストラクチャとサービスを提供するフルマネージドソリューションのファミリーです。

4. 高解像度ライブビデオストリーミング、Hi-Fi 音源、拡張現実および仮想現実 (AR/VR) などのアプリケーションでは、5G デバイス向けに超低レイテンシーが必要です。このようなアプリケーションについては、AWS Wavelength を検討します。AWS Wavelength は AWS のコンピューティングおよびストレージサービスを 5G ネットワーク内に組み込み、超低レイテンシーアプリケーションの開発、デプロイ、スケーリングのためのモバイルエッジコンピューティングインフラストラクチャを提供します。

5. ユーザーが地理的に分散している場合、コンテンツ配信ネットワーク (CDN) を使用して、グローバルに分散した Points of Presence (PoP) を介してデータを配信することにより、静的ウェブコンテンツおよび動的ウェブコンテンツの配信を高速化できます。CDN は通常、エッジコンピューティング機能も提供して、HTTP ヘッダーの操作や URL の書き換え、エッジにおける大規模なリダイレクトなど、レイテンシーの影響を受けやすいオペレーションを実行します。Amazon CloudFront は、静的ウェブコンテンツおよび動的ウェブコンテンツの配信を高速化するウェブサービスです。CloudFront のユースケースには、静的ウェブサイトのコンテンツ配信の高速化、ビデオオンデマンドまたはライブストリーミングビデオの提供などがあります。CloudFront を使用すると、視聴者のコンテンツとエクスペリエンスをカスタマイズして、レイテンシーを短縮することもできます。

6. アプリケーションによっては、最初のバイトのレイテンシーとジッターを低減してスループットを向上することによる、固定エントリポイントまたはより高いパフォーマンスが必要となります。このようなアプリケーションは、エニーキャスト IP アドレスとエッジロケーションで TCP ターミネーションを提供するネットワークサービスの利点を活用できます。AWS Global Accelerator を使用すると、アプリケーションのパフォーマンスを最大 60% 向上し、マルチリージョンアーキテクチャの迅速なフェイルオーバーを実現します。AWS Global Accelerator を使用すると、1 つまたは複数の AWS リージョン でホストされるアプリケーションの固定エントリポイントとして機能する静的エニーキャスト IP アドレスが利用できます。このような IP アドレスを使用すると、トラフィックはユーザーのできるだけ近くの AWS グローバルネットワーク

に入ることができます。AWS Global Accelerator は、クライアントとクライアントに最も近い AWS エッジロケーションの間で TCP 接続を確立することにより、初期接続設定時間を短縮します。TCP/UDP ワークロードのパフォーマンス向上、マルチリージョンアーキテクチャの迅速なフェイルオーバーを実現するには、AWS Global Accelerator の使用を検討します。

7. アプリケーションまたはユーザーがオンプレミスにある場合は、ネットワークとクラウド間に専用のネットワーク接続を設けると、利点が得られる場合があります。専用のネットワーク接続を使用すると、輻輳が発生したり、待機時間が予期せず増大したりする可能性を低減できます。AWS Direct Connect を使用すると、ネットワークを直接 AWS に接続してパブリックインターネットをバイパスすることで、アプリケーションのパフォーマンスを向上できます。新しい接続を作成する場合、AWS Direct Connect デリバリーパートナーが提供するホスト接続を選択するか、AWS から専用接続を選択して、世界中の 100 か所を超える AWS Direct Connect ロケーションにデプロイできます。また、AWS の低データ転送速度を使用してネットワークコストを削減したり、オプションでフェイルオーバーのために Site-to-Site VPN を設定したりすることもできます。
8. AWS 内でのリソースへの接続に Site-to-Site VPN を設定する場合は、オプションでアクセラレーションを有効化できます。高速 Site-to-Site VPN 接続は AWS Global Accelerator を使用して、オンプレミスネットワークからカスタマーゲートウェイデバイスに最も近い AWS エッジロケーションにトラフィックをルーティングします。
9. ワークロードのトラフィックとユーザーの所在地を確認して、ワークロードのパフォーマンスを最適化する DNS ルーティングオプションを特定します。Amazon Route 53 を使用すると、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、IP ベースルーティングオプションを利用して、世界中の視聴者向けに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。
    a. Route 53 を利用すると、エンドユーザーのクエリレイテンシーを低く抑えることができます。Route 53 は、世界中の DNS サーバーのグローバルエニーキャストネットワークを使用して、ネットワークの状態に応じて最適なロケーションから自動的にクエリに応答する設計です。

リソース

関連するベストプラクティス:

- COST07-BP02 コストに基づいてリージョンを選択する
- COST08-BP03 データ転送コストを削減するサービスを実装する
- 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

- 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
- SUS02-BP04 ワークロードの地理的な配置を、ユーザーの口ケーションに合わせて最適化する
- SUS04-BP07 ネットワーク間でのデータ移動を最小限に抑える

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 and AWS Outposts, choosing the right technology for your edge workload (AWS Local Zones と AWS Outpost、エッジのワークロードに適したテクノロジーの選択)
- プレイスメントグループ
- AWS Local Zones
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- Site-to-Site VPN
- Amazon Route 53

関連動画:

- AWS Local Zones Explainer Video (AWS Local Zones 説明動画)
- AWS Outposts: Overview and How It Works (AWS Outposts: 概要と仕組み)
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises (AWS re:Invent 2021 - AWS Outposts: AWS エクスペリエンスをオンプレミスで実現)
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge (AWS re:Invent 2020: AWS Wavelength: 5G エッジで超低レイテンシーでアプリケーションを実行する)
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge (AWS re:Invent 2022 - AWS Local Zones: 分散エッジ向けアプリケーションの構築)
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront (AWS re:Invent 2021 - Amazon CloudFront を使用した低レイテンシーのウェブサイトの構築)

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator(AWS re:Invent 2022 - AWS Global Accelerator を使用してパフォーマンスと可用性を向上する)
- AWS re:Invent 2022 - Build your global wide area network using AWS(AWS re:Invent 2022 - AWS を使用してグローバルなワイドエリアネットワークを構築)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53 (AWS re:Invent 2020: Amazon Route 53 を使用したグローバルトラフィック管理)

関連する例:

- AWS Global Accelerator ワークショップ
- Handling Rewrites and Redirects using Edge Functions (Edge 関数を使用した書き換えとリダイレクトの処理)

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

ネットワーク設定が適切でないと、多くの場合、ネットワークのパフォーマンス、効率、およびコストに影響します。一般的なネットワーク環境では、早い段階でデプロイを迅速に完了することが優先され、ネットワークパフォーマンスに関する適切なネットワーク設定が十分に考慮されているとは限りません。ネットワーク設定を最適化するには、まずネットワーク環境に関する可視性とデータが必要です。

ネットワークリソースのパフォーマンスを把握するには、データの収集と分析を行い、ネットワーク設定の最適化に関する十分な情報に基づいた決定を下します。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。

期待される成果: メトリクスとネットワークモニタリングツールを使用して、ワークロードの変化に合わせてネットワーク設定を最適化します。クラウドベースのネットワークは迅速に最適化できるため、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。
- ネットワークサービスの設定はすべてデフォルトにしている。
- 十分なキャパシティを提供するためにネットワークリソースを過剰プロビジョニングしている。

このベストプラクティスを活用するメリット: AWS ネットワークの必要なメトリクスを収集し、ネットワークモニタリングツールを実装すると、ネットワークパフォーマンスを把握し、ネットワーク設定を最適化できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

AWS ネットワークリソースの利用方法とネットワーク設定を最適化する方法を理解するには、VPC、サブネット、またはネットワークインターフェイス間のトラフィックをモニタリングすることが重要です。以下のツールを使用すると、トラフィックの使用状況、ネットワークアクセス状況、ログに関する情報を詳細に調査できます。

実装手順

1. Amazon VPC IP Address Manager を使用します。AWS とオンプレミスのワークロードの IP アドレスのプランニング、追跡、モニタリングには、IPAM を使用できます。これは、IP アドレスの使用と割り当てを最適化するベストプラクティスです。
2. VPC フローログ をオンにします。VPC フローログを使用して、VPC 内のネットワークインターフェイス間で送受信される IP トラフィックに関する詳細をキャプチャします。VPC フローログを使用すると、過度に制限的なセキュリティグループルールや過度に寛容なセキュリティグループルールを診断して、ネットワークインターフェイス間のトラフィックの方向を判断できます。フローログを発行する場合、Vended Logs のデータインジェストとアーカイブ料金が適用されます。
3. DNS クエリのログ記録をオンにします。Route 53 が受信するパブリックまたはプライベート DNS クエリに関する情報を記録するように Amazon Route 53 を設定できます。DNS ログを使用すると、要求されたドメインまたはサブドメイン、または DNS クエリに応答した Route 53 エッジロケーションを把握したうえで DNS の設定を最適化できます。
4. ネットワークの到達可能性を解析およびデバッグするには、Reachability Analyzer を使用します。Reachability Analyzer は、VPC 内の送信元リソースと宛先リソース間の接続テストを実行できる設定分析ツールです。このツールを使用すると、ネットワーク設定が意図した接続になっているかを確認できます。
5. リソースへのネットワークアクセスを把握するには、Network Access Analyzer を使用します。Network Access Analyzer を使用すると、ネットワークのアクセス要件を指定して、指定した要件を満たさない可能性のあるネットワークパスを特定できます。対応するネットワーク設定を最適化すると、ネットワークの状態を理解して検証し、AWS のネットワークがコンプライアンス要件を満たしているかどうかを検証できます。

6. Amazon CloudWatch を使用して、ネットワークオプションの適切なメトリクスを有効にします。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。例えば、VPC Network Address Usage、VPC NAT Gateway、AWS Transit Gateway、VPN tunnel、AWS Network Firewall、Elastic Load Balancing、AWS Direct Connect などのメトリクスを有効にできます。メトリクスの継続的なモニタリングは、ネットワークの状態と使用状況を観測して理解する優れた方法であり、観測に基づいたネットワーク設定の最適化につながります。

実装計画に必要な工数レベル: 中

リソース

関連するドキュメント:

- VPC フローログ
- パブリック DNS クエリのログ記録
- IPAM とは
- What is Reachability Analyzer? (Reachability Analyzer とは)
- What is Network Access Analyzer? (Network Access Analyzer とは)
- VPC の CloudWatch メトリクス
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format (Apache Parquet 形式の VPC フローログを使用して、ネットワーク分析のパフォーマンスを最適化してコストを削減する)
- Monitoring your global and core networks with Amazon Cloudwatch metrics
- Continuously monitor network traffic and resources

関連動画:

- Networking best practices and tips with the Well-Architected Framework
- Monitoring and troubleshooting network traffic

関連する例:

- AWS Networking Workshops (AWS ネットワーキングワークショップ)
- AWS ネットワークモニタリング

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。
10KPI 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
- Quick 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 Quick (Quick 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 Quick
- 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
- Quick 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

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。実装に関する規範的なガイダンスについては、コスト最適化の柱のホワイトペーパーを参照してください。

ベストプラクティス領域

- 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、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。

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-BP08 コスト意識を持つ文化を生み出す
- COST01-BP09 コスト最適化によるビジネス価値を数値化する

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。

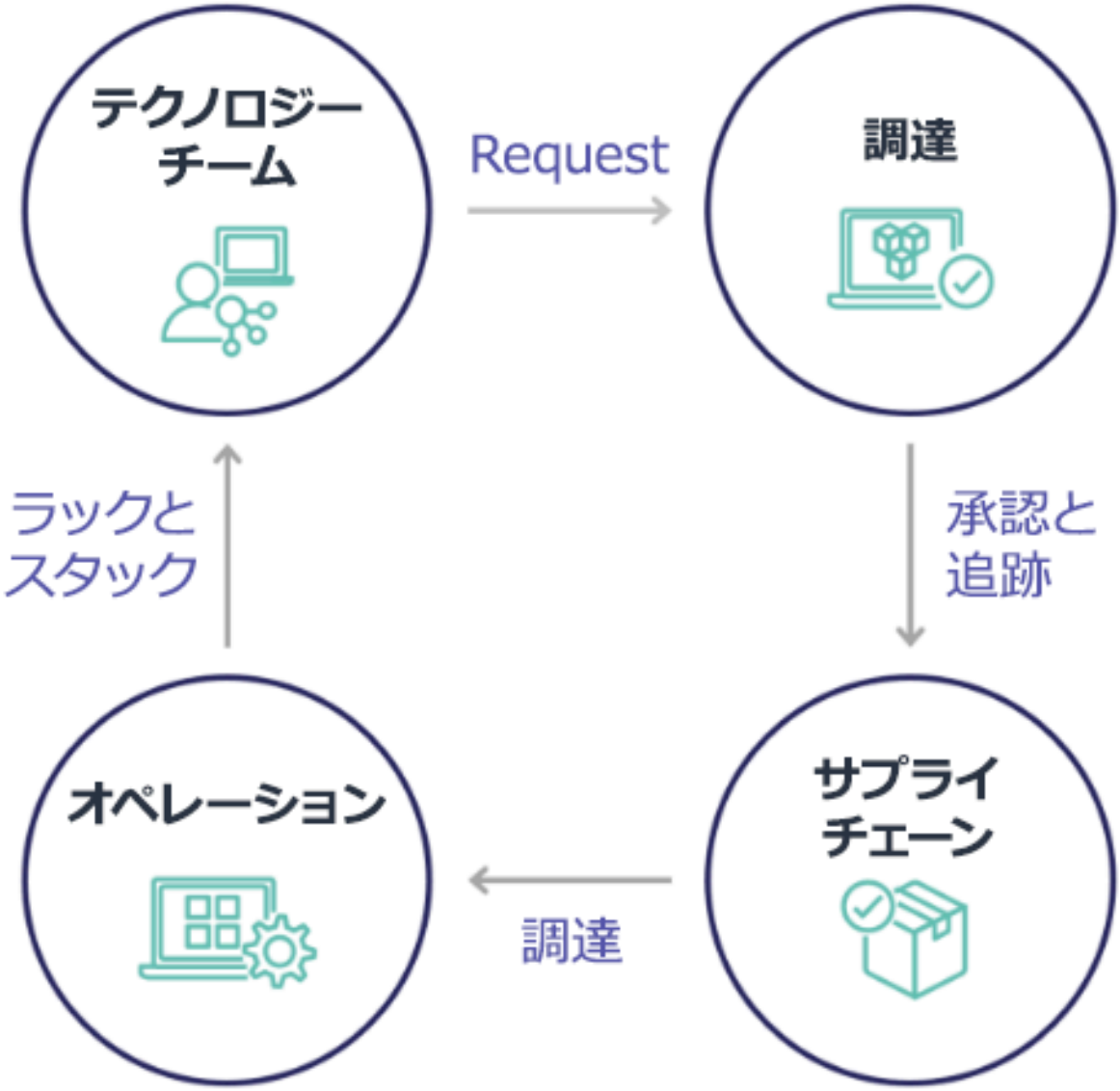
このベストプラクティスを活用しない場合のリスクレベル: 高

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。

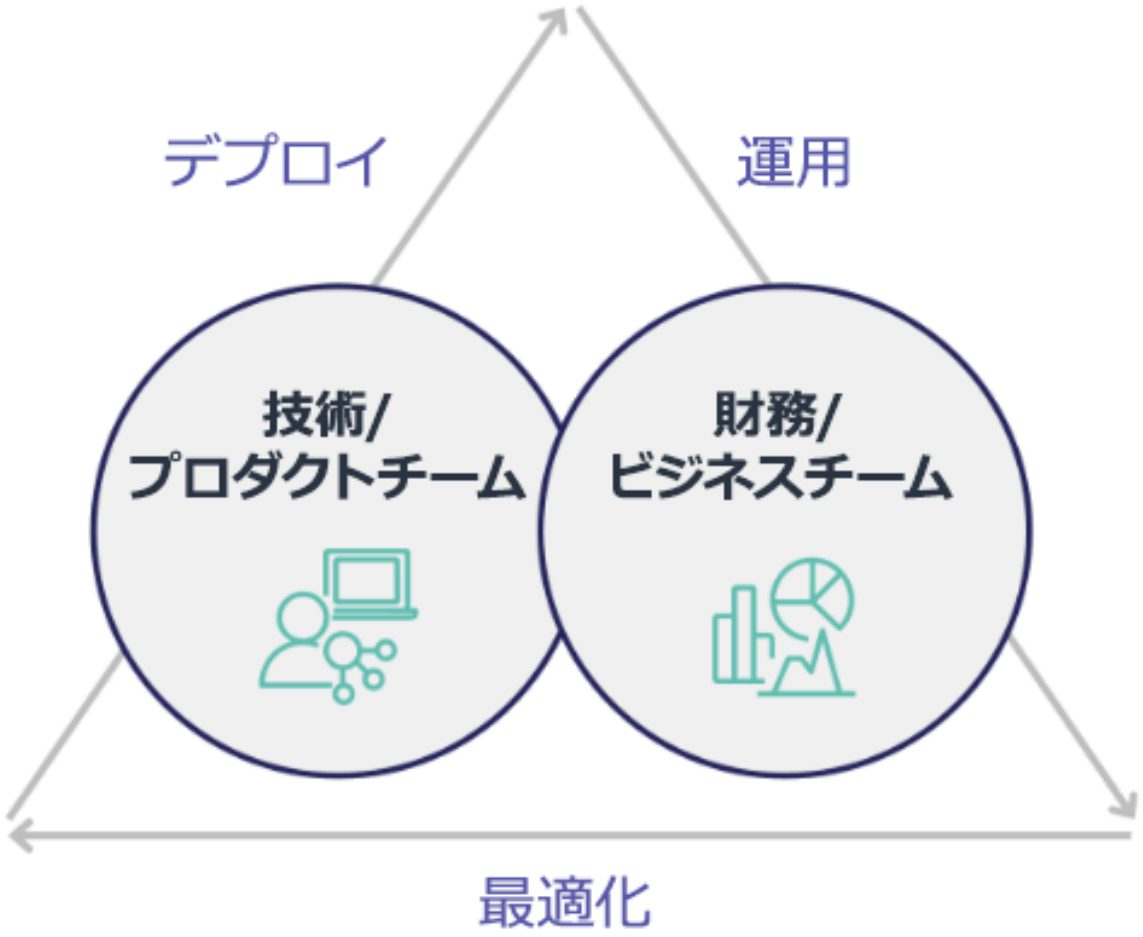
- エンジニアリングチームや IT チームは、通常、依頼主である
- さまざまな財務チームが承認者、調達者として機能する
- 運用チームは、すぐに使えるインフラストラクチャのラック、スタック、および引き渡しを行う



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 料金見積りツール 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 料金見積りツール

- 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 Quick 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 Quick 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: ミディアム

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 Quick 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 Quick 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 Quick 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: ミディアム

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

COST01-BP08 コスト意識を持つ文化を生み出す

コストを意識した企業文化を醸成するために、組織全体で改革やプログラムを実施しましょう。まず小さく始めて、続いて機能や組織でのクラウド利用の増加に合わせて、規模を拡大していき、さまざまなプログラムを運用していくことを推奨します。

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

コスト意識を持つ文化があると、組織全体で有機的かつ分散的に実行されるベストプラクティスを通じて、コストの最適化とクラウド財務管理 (財務運用、Cloud Center of Excellence、クラウド運用

チームなど) の規模を拡大することができます。コスト意識を持つことで、トップダウンで集中的に行う厳格なアプローチと比較して、最小限の労力で組織全体に高いレベルの能力を生み出すことができます。

クラウドコンピューティング、特にクラウドコンピューティングの主なコスト要因についてのコスト意識を持つことで、チームは予想される変更の成果をコスト面で理解できます。クラウド環境にアクセスするチームは、料金モデルと、従来のオンプレミスデータセンターとクラウドコンピューティングの違いを認識する必要があります。

コストを意識する文化の主な利点は、技術チームが必要に応じて消極的なコスト最適化を行うのではなく、積極的かつ継続的にコストを最適化する (たとえば、新しいワークロードを設計する際や、既存のワークロードを変更する際に機能要件でないと見なされる) ことにあります。

この文化のわずかな変化で、現在や将来のワークロードの効率に大きな影響を与える可能性があります。これには、次のような例があります。

- エンジニアリングチームに可視性を与え、意識を高めることで、自分たちが何をしているのか、コスト面でどのような影響があるのかを理解させることができます。
- 組織全体のコストと使用量にゲーム的要素を取り入れる。これは、公開ダッシュボードや、チーム間の標準コストと標準使用量 (例えば、ワークロードあたりのコストとトランザクションあたりのコストなど) を比較するレポートによって実行できます。
- コスト効率を認識する。自発的または独断で行なったコスト最適化の成果を公開または非公開で評価して、間違いから学び、今後繰り返さないようにします。
- あらかじめ設定された予算でワークロードを実行するために、トップダウンの組織的要件を作成します。
- 変更のビジネス要件と、アーキテクチャインフラストラクチャまたはワークロード設定について要求された変更がコストにどのように影響するかを探求して、必要な分だけ支払うようにします。
- 変更計画者は、予想される変更がコストに影響することを意識し、コスト効率的にビジネス成果をもたらすように関係者に確認してもらう必要があります。

実装手順

- クラウドコストをテクノロジーチームに報告する: これによりコスト意識を高め、財務およびビジネス関係者にとって効率的な KPI を確立します。
- 予定されている変更を関係者やチームメンバーに通知する: 予定されている変更とワークロードに対する費用対効果を週次の変更ミーティングで協議するための議題を作成します。

- 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、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、アーキテクト、サポートチームと話します。
- 成功事例を共有する: ワークロード、AWS アカウント、または組織のコスト削減に関する成功事例を共有して、コスト最適化に関する前向きな姿勢と励みにします。
- トレーニング: 技術チームやチームメンバーが AWS クラウド に関するリソースコストについて認識するためのトレーニングを受けるようにします。
- 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す。
- ブログを登録する: 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購読して、AWS が共有する新しいリリース、実装、例、および変更に従います。

リソース

関連ドキュメント:

- AWS ブログ
- AWS コスト管理
- AWS ニュースブログ

関連する例:

- AWS によるクラウド財務管理
- AWS Well-Architected ラボ: クラウド財務管理

COST01-BP09 コスト最適化によるビジネス価値を数値化する

コスト最適化でビジネス価値を数値化することで、組織に対するメリットの全体像を把握できます。コスト最適化は必要な投資であるため、ビジネス価値を数値化することで、各ステークホルダーに投資利益率を説明できます。ビジネス価値の数値化は、将来のコスト最適化投資に関してステークホルダーからより多くの賛同を得ることができます。また、組織のコスト最適化活動の結果を測定するためのフレームワークを提供します。

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

コスト最適化により削減された金額を報告することに加えて、実現された付加価値を数値化することをお勧めします。コスト最適化のメリットは通常、ビジネス成果に対して削減されたコストという観点で数値化されます。例えば、Savings Plans を購入した場合はオンデマンドの Amazon Elastic Compute Cloud (Amazon EC2) のコスト削減を数値化できます。Savings Plans により、コストを削減し、ワークロードの出力レベルを維持できます。アイドル状態の Amazon EC2 インスタンスが終了した場合や、アタッチされていない Amazon Elastic Block Store (Amazon EBS) ボリュームが削除された場合は、AWS の利用料削減を数値化できます。

コスト最適化のメリットは、コスト削減やコスト回避にとどまりません。効率性向上とビジネス価値を測定するために、その他のデータを追加で取得することを検討してください。

実装手順

- コスト最適化のベストプラクティスを実行する: たとえば、リソースのライフサイクル管理によってインフラストラクチャコストと運用コストを削減し、実験のための時間や予定外の予算を作り出すことができます。これにより、組織の俊敏性が向上し、収益創出のための新しい商機の発掘につながります。
- 自動化を実装する: 例えば Auto Scaling では、最小限の労力で伸縮性を確保でき、手作業によるキャパシティプランニング作業を排除することで、スタッフの生産性を向上させることができます。運用の回復力の詳細については、 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。.
- 将来の AWS コストを予測する: 予測により、財務関係者は、組織内外の他の利害関係者と見通しを立て、組織の財務予測の可能性を向上させることができます。AWS Cost Explorer を使用して、コストと使用量を予測できます。

リソース

関連するドキュメント:

- AWS ブログ
- AWS コスト管理
- AWS ニュースブログ
- 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。
- AWS Cost Explorer

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

組織によるリソースの管理方法を定義するポリシーを策定し、定期的に検査します。ポリシーでは、リソースのライフタイム全体にわたる作成、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、ワークロードがどの程度最適化されているか、また組織単

位や製品の収益性がどの程度あるかを理解するのに役立ちます。この知識により、組織内のどこにリソースを割り当てるかについて、より多くの情報に基づいた意思決定が可能になります。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。クラウド環境でのあらゆる変更について、ポリシーと手順の遵守と実装を徹底してください。IT の変更管理会議では、質問を提起して、計画された変更によるコストへの影響 (増加または減少)、ビジネスの正当性、期待される結果を確認します。

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。また、ポリシーは、(使用されるように) 遵守と解釈が容易で、(チーム間で誤解が発生しないように) 具体的である必要があります。さらに、顧客のビジネス状況や優先順位が変わったときにポリシーが古くなるため、(当社のメカニズムと同様に) 定期的に検査し、更新する必要があります。

使用する地理的リージョンや、リソースを稼働する時間帯など、大局的な幅広い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を利用できるか (例えば、テスト環境や開発環境では低パフォーマンスのストレージ)、どのタイプのリソースを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を medium にする)、これらのリソースをどの程度のスパンで使用するか (一時的、短期、特定の期間)、などがあります。

ポリシーの例

次に、コスト最適化に焦点を当てた独自のクラウドガバナンスポリシーを作成するために確認できるポリシーの例を示します。組織の要件と関係者の要求に基づいてポリシーを調整するようにしてください。

- ポリシー名: リソースの最適化やコスト削減ポリシーなど、明確なポリシー名を定義します。
- 目的: このポリシーを使用すべき理由と、期待される結果を説明してください。このポリシーの目的は、ビジネス要件を満たすために必要なワークロードのデプロイと実行に必要な最小限のコストがあると確認することです。
- スコープ: このポリシーを誰が使用すべきか、いつ使用すべきかを明確に定義してください。例えば、DevOps X Team が X 環境 (本番環境または非本番環境) で米国東部のお客様にこのポリシーを使用する、などです。

ポリシーステートメント

1. ワークロードの環境とビジネス要件 (開発、ユーザー受け入れテスト、本番稼働前、または本番稼働) に基づいて us-east-1 または複数の us-east リージョンを選択します。
2. Amazon EC2 と Amazon RDS インスタンスが、午前 6 時から夜 8 時 (東部標準時 (EST)) に実行されるようにスケジュールを立てます。
3. 8 時間後に未使用の Amazon EC2 インスタンスをすべて停止し、24 時間非アクティブ状態の未使用の Amazon RDS インスタンスをすべて停止します。
4. 非本番環境で非アクティブ状態が 24 時間続いたら、未使用の Amazon EC2 インスタンスをすべて終了します。Amazon EC2 インスタンス所有者に (タグに基づいて) 停止した Amazon EC2 インスタンスを本番環境で確認するように促し、使用していない場合は Amazon EC2 インスタンスが 72 時間以内に終了することを通知します。
5. m5.large などの汎用インスタンスファミリーとサイズを使用し、AWS Compute Optimizer を使用して CPU とメモリの使用率に基づいてインスタンスのサイズを変更します。
6. 自動スケーリングの使用を優先して、トラフィックに基づいて実行中のインスタンスの数を動的に調整します。
7. 重要ではないワークロードにはスポットインスタンスを使用します。
8. キャパシティ要件を確認して、予測可能なワークロードに備えて削減プランやリザーブドインスタンスをコミットし、クラウド財務管理チームに通知してください。
9. Amazon S3 ライフサイクルポリシーを使用して、アクセス頻度の低いデータを安価なストレージ階層に移動できます。保存ポリシーが定義されていない場合は、Amazon S3 Intelligent Tiering を使用してオブジェクトをアーカイブ階層に自動的に移動します。
10. Amazon CloudWatch を使用して、リソースの使用状況をモニタリングし、スケーリングイベントをトリガーするアラームを設定します。
11. それぞれの AWS アカウント について、AWS Budgets を使用して、コストセンターとビジネスユニットに基づいてアカウントのコストと使用量の予算を設定します。
12. AWS Budgetsを使用してアカウントのコストと使用量の予算を設定すると、支出を常に把握し、予期しない請求を回避できるため、コストをより適切に制御できます。

プロシージャ: このポリシーを実装するための詳細な手順を提供するか、各ポリシーステートメントの実装方法を説明する他のドキュメントを参照してください。このセクションでは、ポリシー要件を実行するためのステップバイステップの手順を説明する必要があります。

このポリシーを実装するには、さまざまなサードパーティ製ツールや AWS Config ルールを使用してポリシーステートメントの遵守を確認したり、AWS Lambda 関数を使用して自動修復アクションを

トリガーしたりできます。AWS Organizations を使用してポリシーを適用することもできます。さらに、リソースの使用状況を定期的に見直し、必要に応じてポリシーを調整して、ビジネスニーズが引き続き満たされていることを確認する必要があります。

実装手順

- 関係者とのミーティングを設ける: ポリシーを策定するには、組織内の関係者 (クラウドビジネスオフィス、エンジニア、またはポリシー実施部門の意思決定者) に、要件を明記して文書化するよう依頼します。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。
- 確認する: 誰が AWS クラウド にアクセスしてデプロイできるかを指定したポリシーに、チームが同意していることを確認します。チームが組織のポリシーに従っているかどうか、同意したポリシーおよび手順に沿ってチームがリソースを作成しているかどうかを確認してください。
- オンボーディングトレーニングセッションを作成する: 新しい組織メンバーに対し、コスト意識を定着させ、組織の要件を特定するために、オンボーディングトレーニングコースを完了するよう求めます。新しいメンバーは、以前の経験から異なるポリシーを想定している場合や、ポリシーについてまったく考えていない場合があります。
- 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: 組織で、定義されたポリシーが、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:

- Change Management in the Cloud (クラウドにおける変更管理)
- 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
- AWS での管理とガバナンス
- 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

関連動画:

- AWS での大規模な管理とガバナンス

関連サンプル:

- VMware - What Are Cloud Policies? (VMware - クラウドポリシーとは)

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、目標およびターゲットを策定します。目標は、期待される結果に基づく方向性を組織に示し、ターゲットは、ワークロードについて具体的に達成すべき測定可能な成果を表します。

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。AWS で成長を続ける組織にとって、コスト最適化の目標を設定して追跡することは重要です。これらの目標または 主要業績評価指標 (KPI) には、オンデマンドでの支出の割合や、AWS Graviton インスタンスまたは gp3 EBS ボリュームタイプなどの特定の最適化されたサービスの導入などが含まれます。測定可能で達成可能な目標を設定することで、継続的なビジネスオペレーションにとって重要な効率の向上を継続的に測定できるようになります。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。つまり、目標は進みたい方向を示し、ターゲットはその方向へどこまで進み、その目標をいつ達成する必要があるかを表します。このとき、「SMART」つまり具体的 (specific)、測定可能 (measurable)、割り当て可能 (assignable)、現実的 (realistic)、タイムリー (timely) であることをガイダンスとします。「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」などは目標の例です。「プラットフォームの使用量を 20% 増加させ、コスト増は 5% 未満に抑える」などはターゲットの例です。ワークロードを 6 か月ごとに効率化する必要があるというケースも、目標としてはよくあります。付随する目標は、ビジネスあたりのコストメトリクスを 6 か月ごとに 5% 削減する必要があるというものです。

コスト最適化の目標は、ワークロードの効率を高めることです。つまり、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 か月から 1 年ごとに効率を 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、新しいサービスや機能をリリースすることで達成できます。

KPI と関連する節約の機会をほぼリアルタイムで把握し、経時的に進捗状況を追跡することが重要です。KPI 目標の定義と追跡を始めるには、 Cloud Intelligence Dashboards (CID) フレームワークの KPI ダッシュボードをお勧めします。KPI ダッシュボードには、AWS Cost and Usage Report から取得したデータに基づいて一連の推奨コスト最適化 KPI が表示され、カスタム目標の設定や、経時的な進捗状況の追跡ができます。

KPI 目標を設定して追跡できる別のソリューションがある場合は、そのソリューションが組織内のすべてのクラウド財務管理関係者に採用されていることを確認してください。

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。顧客の需要が経時的にどのように変化するか、季節要因による増加やマーケティングキャンペーンによる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うえで、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、使用量、コスト、および両者の関わり方を対象に含めたものである必要があります。目標はシンプルかつ大局的なものにして、そのビジネスで求められる成果を従業員が理解できるような内容にする必要があります (未使用のリソースを特定のコストレベル以下に抑えるなど)。未使用リソースのタイプごとに目標を定義したり、目標とターゲットに対しての損失とするコストを具体的に設定する必要はありません。使用量に変化がない状態でコストの変化が予想される場合は、組織的なプログラム (トレーニングや教育による能力向上など) を用意しておきます。
- 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。ワークロードの効率を高めることが目標である場合、ターゲットでは、改善の量 (通常は 1 ドルあたりのビジネス成果で示す) と、その改善をいつ実現できるかを数値化します。例えば、オーバープロビジョニングによる無駄を最小限に抑えるという目標を設定した場合、ターゲットとしては、「実稼働ワークロードの 1 つ目の階層でコンピューティングのオーバープロビジョニングによる無駄を階層コンピューティングコストの 10% 以下に抑え、実稼働ワークロードの 2 つ目の階層でコンピューティングのオーバープロビジョニングによる無駄を階層コンピューティングコストの 5% 以下に抑える」のような内容が考えられます。

リソース

関連するドキュメント:

- AWS managed policies for job functions
- AWS Control Tower のランディングゾーンに対する AWS マルチアカウント戦略
- Control access to AWS リージョン using IAM policies
- SMART 目標

関連動画:

- Well-Architected Labs: Goals and Targets (Level 100) (Well-Architected ラボ: 目標とターゲット (レベル 100))

関連サンプル:

- Well-Architected Labs: Decommission resources (Goals and Targets) (Well-Architected ラボ: リソースを削除する (目標とターゲット))
- Well-Architected Labs: Resource Type, Size, and Number (Goals and Targets) (Well-Architected ラボ: リソースのタイプ、サイズ、数 (目標とターゲット))

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 では、ワークロードを AWS で拡張する際に環境を一元管理するのに役立つ複数の AWS アカウント を作成できます。組織単位 (OU) 構造で AWS アカウント をグループ化し、各 OU の下に複数の AWS アカウント を作成することで、組織階層をモデル化できます。アカウント構造を作成するには、まず、どの AWS アカウント を管理アカウントにするかを決定する必要があります。その後、管理アカウントのベストプラクティスとメンバーアカウントのベストプラクティスに従って、設計されたアカウント構造に基づいて、新しい AWS アカウント を作成したり、既存のアカウントをメンバーアカウントとして選択したりできます。

組織の規模や使用状況にかかわらず、少なくとも 1 つの管理アカウントとそれに紐づく 1 つのメンバーアカウントを常に持つことをお勧めします。すべてのワークロードリソースはメンバーアカウント内にのみ存在し、管理アカウントにはリソースを作成しないでください。AWS アカウント をいくつ持つべきかということについては、一律に答えることはできません。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。

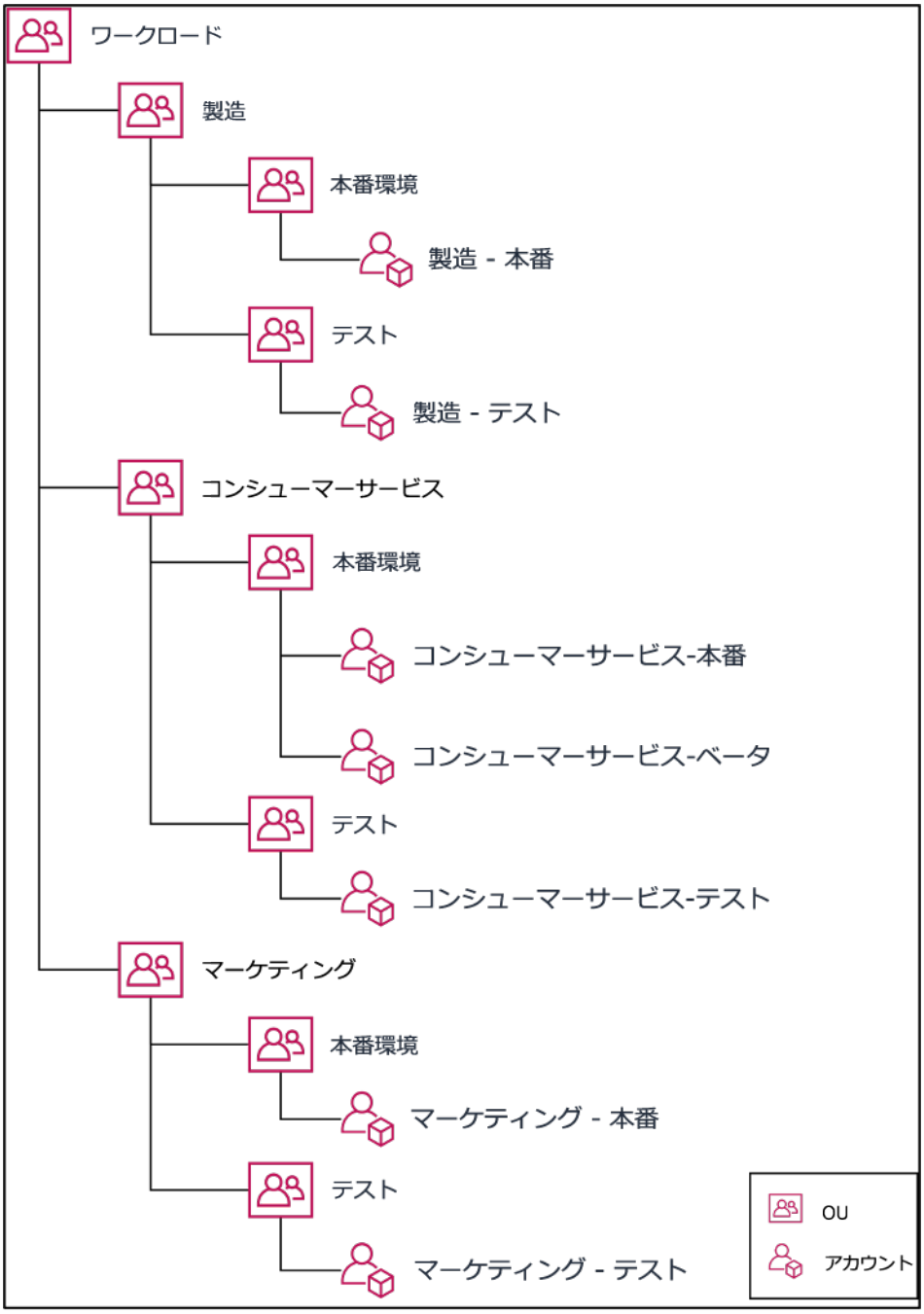
- 組織単位、コストセンター、特定のワークロード間で、管理、会計、請求の職務機能を切り離す必要がある場合。
- AWS のサービスの制限が特定のワークロードのみに設定される場合。
- ワークロードとリソース間の隔離と分離には要件があります。

AWS Organizations 内部では、一括請求により、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。

一括請求機能により、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、リンクされた各アカウントのアクティビティを可視化することができます。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。

次の図は、AWS Organizations を組織単位 (OU) とともに使用して複数のアカウントをグループ化し、各 OU の下に複数の AWS アカウント を配置する方法を示しています。アカウントを整理するためのパターンを提供するために、さまざまなユースケースやワークロードに OU を使用することをお勧めします。



複数の AWS アカウント を組織単位でグループ化する例。

AWS Control Tower は、複数の AWS アカウントを迅速に設定、構成し、組織の要件に沿ったガバナンスを確保することができます。

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、アクセス要件とデータ要件への準拠を促進します。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管理します。Well-Architected フレームワークのセキュリティと信頼性の柱を定期的に見直し、提供されるベストプラクティスに従います。財務構造により、厳格な財務分離 (異なるコストセンター、ワークロードのオーナーシップ、説明責任) が実現します。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、組織内の個々の事業部門や部門、またはアカウントを所有する利害関係者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、分離要件を維持します。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個別に表示させることも可能です。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:

- コスト配分タグの使用
- AWS managed policies for job functions (職務に応じた AWS マネージドポリシー)
- 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 (IAM ポリシーの使用による AWS Regions へのアクセス制御)
- AWS Control Tower
- AWS Organizations

- 管理アカウントとメンバーアカウントに関するベストプラクティス
- Organizing Your AWS Environment Using Multiple Accounts (複数のアカウントを使用した AWS 環境の組織化)
- 共有リザーブドインスタンスと Savings Plans の割引の有効化
- 一括請求
- 一括請求

関連する例:

- CUR の分割とアクセスの共有

関連動画:

- Introducing AWS Organizations(AWS Organizations の紹介)
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations(AWS Organizations のベストプラクティスを使用するマルチアカウント AWS 環境の設定)

関連する例:

- Well-Architected ラボ: AWS 組織の作成 (レベル 100)
- Splitting the AWS Cost and Usage Report and Sharing Access (CUR の分割とアクセスの共有)
- Defining an AWS Multi-Account Strategy for telecommunications companies (通信会社のための AWS マルチアカウント戦略の定義)
- AWS アカウント の最適化のベストプラクティス
- AWS Organizations における組織単位のベストプラクティス

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: ミディアム

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。これらは、リージョンやリソースタイプへのアクセスコントロールなど、組織の要件によって定義されたコストのみが発生することを保証するものです。

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

コスト管理を導入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状況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。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上限を把握したら、ガバナンスを適用できます。AWS Budgets では、通知を設定し、AWS コスト、使用状況、コミットメント割引 (Savings Plans とリザーブドインスタンス) の月間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。

AWS Budgets で予算限度を設定したら、AWS Cost Anomaly Detection を使用して予期せぬコストを削減します。AWS Cost Anomaly Detection は、機械学習を使用してコストと使用状況を継続的に監視し、異常な支出を検出するコスト管理サービスです。これにより、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。まず、AWS Cost Anomaly Detection でコストモニターを作成し、ドルのしきい値 (影響が 1,000 USD を超える異常に対してアラートを出すなど) を設定し、アラートの設定を選択します。アラートを受信したら、異常の背後にある根本原因とコストへの影響を分析できます。また、AWS Cost Explorer では独自の異常解析を監視および実行することもできます。

AWSで AWS Identity and Access Management とAWS Organizations サービスコントロールポリシー (SCP) をとおしてガバナンスポリシーを適用します。IAM により、AWS サービスや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 のリソースを作成または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、定義されたポリシーの範囲を超えてリソースが作成される可能性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、正しい使用を強制します。SCP は、組織内のすべてのアカウントで利用可能な最大権限を一元管理し、アカウントをアクセス制御ガイドラインの範囲内に維持することができます。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 SCP によるメンバーアカウントのアクションの可否を設定できます。アクセス管理の導入の詳細については、Well-Architected 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。

AWSService 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。Service Quotas を最小限のオーバーヘッドで設定し、正確に維持することで、組織の要件以外の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 を使用すると、必要に応じてクォータを引き上げることができます。

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、ポリシーから外れた支出があった場合に通知します。アカウントごとに複数のコスト予算を設定し、アカウント全体の支出を通知します。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。個人の E メールアカウントではなく、E メール配信リストを通知の受信者として設定します。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。特定の IAM または SCP のポリシーを適用したり、ターゲットの Amazon EC2 または Amazon RDS インスタンスを停止したりできるAWS Budget アクションを事前設定することもできます。予算に関するアクションは、自動的に実行するか、ワークフローの承認を得るようにすることができます。
- 異常な支出に関する通知を実装する: AWS Cost Anomaly Detection を使用して、組織内の思いがけないコストを削減し、異常な支出の可能性の根本原因を分析します。指定した粒度で異常な支出を特定するコストモニターを作成し、AWS Cost Anomaly Detection で通知を設定すると、異常な支出が検出された際にアラートが送信されます。これにより、異常の背後にある根本的な原因を分析し、コストへの影響を理解できます。AWS Cost Anomaly Detection を設定する際に AWS Cost Categories を使用して、予想外のコストの根本原因を分析し、必要なアクションをタイムリーに実行できるプロジェクトチームまたはビジネスユニットチームを特定します。
- 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 managed policies for job functions (職務に応じた AWS マネージドポリシー)
- 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 (IAM ポリシーの使用による AWS Regions へのアクセス制御)

- AWS Budgets
- AWS Cost Anomaly Detection
- AWS コストを管理する

関連動画:

- How can I use AWS Budgets to track my spending and usage (AWS Budgets を使用して支出と使用量を追跡するにはどうすればよいですか?)

関連する例:

- Example IAM access management policies (IAM アクセス管理ポリシーの例)
- サービスコントロールポリシーの例
- AWS Budgets アクション
- タグを使って Amazon EC2 リソースへのアクセスを管理する IAM ポリシーを作成する
- IAM アイデンティティのアクセスを特定の Amazon EC2 リソースに制限する
- Create an IAM Policy to restrict Amazon EC2 usage by family (家族ごとに EC2 の利用を制限する IAM ポリシーを作成する)
- Well-Architected ラボ: コストと使用量のガバナンス (レベル 100)
- Well-Architected ラボ: コストと使用量のガバナンス (レベル 200)
- Slack integrations for Cost Anomaly Detection using Amazon Q Developer in chat applications (AWS Chatbot を使用した AWS Cost Anomaly Detection のための Slack の統合)

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 コストと使用状況に組織情報を追加する
- COST03-BP03 コスト属性カテゴリを特定する
- COST03-BP04 組織のメトリクスを確立する
- COST03-BP05 請求およびコスト管理ツールを設定する
- 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 Quick 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 Quick
- 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、および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。これによりコスト管理ツールで、フィルター処理によるリソースの検索や、コストおよび使用状況のモニタリングを行うことができます。目的、チーム、環境、またはビジネスに関連するその他の基準によって、可能な限りすべてのリソースに一貫したタグ付けを実装します。

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

AWS でタグ付けを実装することにより、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。キーは組織全体で一意になるように定義されている必要があります。値はリソースのグループに対して一意になります。キーと値のペアとして、例えばキーを `Environment`、値を `Production` にすることができます。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関するレポート作成および分析を実行することで、社内のコスト管理ポリシーへの準拠を強化し、正確に帰属を特定できます。

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。AWS Organizations で、アカウントの AWS リソースに対してタグを使用する方法のルールを定義するには、AWS Organizations の タグポリシーを使用します。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 タグエディタでは、複数のリソースのタグを追加、削除、管理できます。タグエディタを使用して、タグ付けするリソースを検索し、検索結果でリソースのタグを管理できます。

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: ビジネス全体からすべての関係者を集めて、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、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。効率を向上させるには、CLI、タグエディタ、AWS Systems Manager などのツールを使用します。
- AWS Cost Categories を実装する: タグ付けを実装せずにコストカテゴリを作成できます。Cost Categories では、既存のコストと使用量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、それをコストカテゴリに実装します。
- タグ付けを自動化する: すべてのリソースにわたって大局的なタグ付けを継続するには、タグ付けを自動化して、リソースの作成時にタグ付けが自動的に行われるようにします。リソースの作成時にタグ付けを行うには、AWS CloudFormation などのサービスを使用します。Lambda 関数を使用してタグ付けを自動的に行うカスタムソリューションを作成することも、ワークロードを定期的にスキャンして、タグ付けされていないリソースを削除するマイクロサービスを使用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)。
- タグ付けのモニタリングとレポート作成を行う: 組織全体で大局的なタグ付けを継続するには、ワークロード全体のタグについて、レポート作成およびモニタリングを行います。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確認することも、タグエディタなどのサービスを利用することもできます。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:

- タグ付けに関するベストプラクティス
- AWS CloudFormation のリソースタグ
- AWS Cost Categories
- 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

- AWS コストと使用状況レポートの管理

関連動画:

- How can I tag my AWS resources to divide up my bill by cost center or project (AWS リソースにタグを付けて、コストセンターまたはプロジェクト別に請求書を分割する方法)
- AWS リソースのタグ付け

関連する例:

- Automatically tag new AWS resources based on identity or role (アイデンティティまたはロールに基づいて新しい AWS リソースへのタグ付けを自動的に行う)

COST03-BP03 コスト属性カテゴリを特定する

支出の説明責任を徹底し、消費行動を効果的に推進できるように、組織内のコストを内部消費エンティティに配分するために使用できるビジネスユニット、部門、プロジェクトなどの組織カテゴリを特定してください。

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

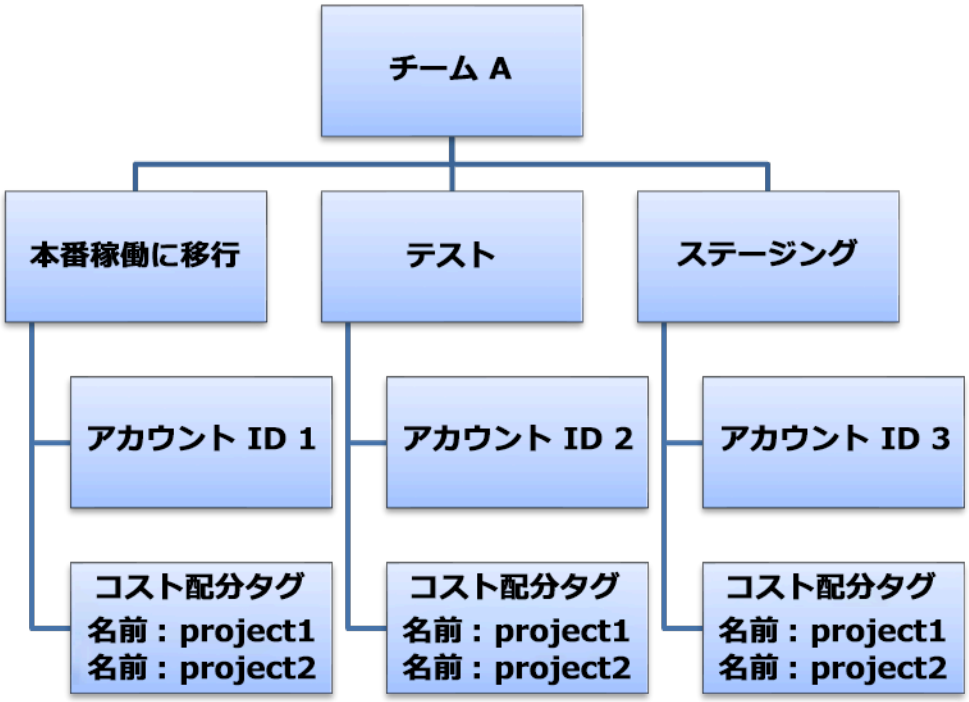
コストを分類するプロセスは、予算編成、会計、財務報告、意思決定、ベンチマーキング、およびプロジェクト管理においてきわめて重要です。費用を分類してカテゴリ化することで、チームはクラウドジャーニーで発生するコストの種類をよりよく理解でき、チームが情報に基づいた意思決定を行い、予算を効果的に管理できるようになります。

クラウド支出の説明責任は、統制の取れた需要とコスト管理に対する強力なインセンティブを確立します。その結果、クラウド支出の大部分を消費するビジネスユニットやチームに割り当てている組織では、クラウドコストを大幅に節約できます。

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。

組織内の関係者と共にコスト属性カテゴリを定義した後、 AWS Cost Categories を使用して、コストと使用状況の情報を AWS クラウド での意味を持つカテゴリ (特定のプロジェクトのコストや部門またはビジネスユニットの AWS アカウント など) にグループ化します。カスタムカテゴリを作成して、アカウント、タグ、サービス、料金タイプ、その他のコストカテゴリなどのさまざまなディメンションを使用して定義したルールに基づき、コストと使用状況の情報をカスタムカテゴリにマッピングすることもできます。コストカテゴリを設定すると、カテゴリごとにコストと使用状況の情報を確認できるようになり、組織の戦略や購入に関する決定をより適切に行うことができます。これらのカテゴリは、AWS Cost Explorer、AWS Budgets、および AWS Cost and Usage Report にも表示されます。

例として、次の図は、組織でコストと使用状況の情報をグループ化する方法を示しています。例えば、複数のチーム (コストカテゴリ) に複数の環境 (ルール) が含まれ、各環境に複数のリソースまたはアセット (ディメンション) が含まれると考えることができます。



コストと使用状況の組織図。

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。同じリソースに複数のカテゴリを割り当てることも、同じリソースを複数の異なるカテゴリに含めることもできるため、必要な数のカテゴリを定義します。これにより、 カテゴリが適用された構造内で AWS Cost Categories を使用して、コストを管理できます。
- AWS Cost Categories を定義する: また、 コストカテゴリを作成すると、 コストと使用状況の情報を整理できます。ここでは AWS Cost Categories を使用して、AWS コストと使用状況を意味のあるカテゴリにマッピングします。コストカテゴリを使用することにより、ルールベースのエンジンを使用してコストを整理できます。ルールを設定することによって、コストがカテゴリに分類されます。ルール内では、特定の AWS アカウント、特定の AWS サービス、特定の料金タイプなどの各カテゴリについて、複数のディメンションを使用してフィルター処理を行うことができます。これらのカテゴリは、 AWS Billing and Cost Management コンソール内で複数の製品にわたって使用できます。これには、AWS Cost Explorer、AWS Budgets、AWS Cost and Usage Report、AWS Cost Anomaly Detection が含まれます。コストカテゴリを使用して、コストのグループを作成することもできます。コストカテゴリの作成後 (使用状況レコードの値が更新されるまでに最長で 24 時間かかります)、作成したコストカテゴリは、次の場所に表示されます AWS Cost Explorer、 AWS Budgets、AWS Cost and Usage Report、 AWS Cost Anomaly Detection。例えば、ビジネスユニット (DevOps チーム) のコストカテゴリを作成し、各カテゴリの下に、複数のルール (各サブカテゴリのルール) を作成します。各ルールでは、定義したグループに基づいて、複数のディメンション (AWS アカウント、コスト配分タグ、サービス、料金タイプ) を使用します。AWS Cost Explorer および AWS Budgets では、コストカテゴリが追加の請求ディメンションとして表示されます。これを使用すると、特定のコストカテゴリ値をフィルター処理することも、コストカテゴリ別にグループ化することもできます。

リソース

関連するドキュメント:

- AWS リソースのタグ付け
- コスト配分タグの使用
- 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)

- AWS Cost Categories
- AWS Cost Categories を用いてコストを管理する
- コストカテゴリを作成する
- コストカテゴリのタグ付け
- コストカテゴリ内で料金を分割する
- AWS Cost Categories Features (AWS Cost Categories の機能)

関連サンプル:

- Organize your cost and usage data with AWS Cost Categories (AWS Cost Categories でコストと使用状況のデータを整理する)
- AWS Cost Categories を用いてコストを管理する

COST03-BP04 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: 社内の関係者との会議により、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: 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、必要に応じて、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
- 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
- AWS コストと使用状況レポートの管理

COST03-BP05 請求およびコスト管理ツールを設定する

クラウド支出を管理および最適化するには、組織のポリシーに沿ってコスト管理ツールを設定します。これに含まれるサービス、ツール、リソースを使用すると、コストと使用状況のデータを整理して追跡し、統合された請求とアクセス許可で制御を強化できます。また、予算編成と予測を通じた計画の改善、通知またはアラートの受信、リソースと価格の最適化によるさらなるコスト削減を行うこともできます。

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: 高

実装のガイダンス

確固たる説明責任を確立するには、まずアカウント戦略をコスト配分戦略の一部として検討する必要があります。これを正しく行えば、それ以上先に進む必要はないかもしれません。正しく行えない場合、認識の欠如が発生し、さらに問題点が増えます。

クラウド支出の説明責任を高めるには、ユーザーがコストと使用状況を可視化するツールにアクセスできるようにする必要があります。すべてのワークロードとチームに、以下の詳細と目的に合わせてツールを設定することをお勧めします。

- 整理: 独自のタグ付け戦略と分類を使用して、コスト配分とガバナンスのベースラインを確立します。サポートされている AWS リソースにタグを付け、組織構造 (ビジネスユニット、部門、プロジェクト) に基づいてわかりやすく分類します。特定のコストセンターのアカウント名にタグ付けし、それを AWS Cost Categories とマッピングして、特定のビジネスユニットのアカウントをコ

ストセンターのグループにまとめることで、ビジネスユニットの所有者が複数のアカウントの消費を 1 か所で確認できるようにします。

- アクセス: 組織全体の請求情報を 一括請求 (コンソリデーティッドビリング) で追跡し、適切な利害関係者とビジネスオーナーがアクセスできることを確認します。
- 制御: 適切なガードレールを使用して、効果的なガバナンスメカニズムを構築し、SCP、タグポリシー、予算アラートを使用する際の予期しないシナリオを回避します。例えば、効果的な制御メカニズムがあれば、サポートされていないリージョンでチームがリソースを作成するのを防ぐことができます。
- 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。ここでは Cloud Intelligence Dashboard (CID) またはその他のサポート対象製品を使用して、このような可視性を実現します。
- 通知: コストまたは使用量が定義された制限を超えた場合、および AWS Budgets または AWS Cost Anomaly Detection で異常が発生した場合に通知します。
- レポート: すべてのコストと使用状況の情報を要約し、詳細で割り当て可能なコストデータを使用して、クラウド支出の認識と説明責任の意識を高めます。レポートは、それを利用するチームと関連性があり、推奨事項を含めるのが理想的です。
- 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、すべての可能な側面を使用して、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できるようにします。
- 調査: リソースのデプロイとコストの最適化を行う機会について最新情報を入手します。組織レベルでのリソースデプロイに関する通知 (Amazon CloudWatch、Amazon SNS、または Amazon SES を使用) を受け取り、コスト最適化の推奨事項 (AWS Compute Optimizer、AWS Trusted Advisor、など) を確認します。
- 傾向: 指定した期間のコストと使用量の変動を、指定の詳細度で示します。
- 予測: 作成した予測ダッシュボードで、将来の推定コストを示し、リソースの使用量と支出を見積もります。

次のような AWS ツール、例えば AWS Cost Explorer、AWS Billing、 AWS Budgets を必須として使用することができ、または CUR データを Amazon Athena および Quick と統合して、より詳細なビューにこの機能を提供することもできます。組織に必須のスキルや処理能力がない場合、 AWS ProServ、AWS Managed Services (AMS)、 AWS Partner と連携して、ツールを使用することができます。サードパーティ製ツールを使用することもできますが、組織にとって、そのツールの価値がコストに見合っているかどうかを確認する必要があります。

実装手順

- ツールに対するチームベースのアクセスを許可する: アカウントを設定してグループを作成し、必要なコストと使用状況レポート (グループの使用状況に関するもの) へのアクセスを許可します。また、 AWS Identity and Access Management を 使用して AWS Cost Explorer などのツールへのアクセスを制御します。これらのグループには、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状況の情報にアクセスして、使用を追跡できます。
- AWS Budgets を設定する: 設定 AWS Budgets をワークロードのすべてのアカウントで設定します。タグを使用して、アカウント全体の支出に対する予算とワークロードに対する予算を設定します。AWS Budgets で通知を設定すると、予算額を超えたときや、推定コストが予算を超えているときにアラートを受信できます。
- AWS Cost Explorer を設定する: 設定 AWS Cost Explorer をワークロードとアカウントについて設定し、さらに分析を行うためにコストデータを視覚化します。ワークロードのダッシュボードを作成することにより、全体的な支出、ワークロードの主要な使用状況メトリクス、過去のコストデータに基づく将来のコストの予測を追跡できます。
- AWS Cost Anomaly Detection を設定する: ここでは AWS Cost Anomaly Detection をアカウント、コアサービス、または作成した Cost Categories について使用することにより、コストと使用状況をモニタリングし、通常と異なる支出を検出できます。集計レポートでアラートを個別に受信したり、E メールまたは Amazon Simple Notification Service トピックでアラートを受信したりすることで、異常の根本原因を分析および特定し、コストの増加を引き起こしている要因を特定できます。
- 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 Quick。共有データへのデータアクセスの管理を簡素化するには、 Cloud Intelligence Dashboards (CID) を事前設定された高度なダッシュボードに使用することを検討してください。クラウド支出の監視と最適化を 1 か所で簡単に実現するには、 AWS パートナー との連携により、クラウド管理ソリューションを導入することもできます。

リソース

関連するドキュメント:

- AWS コスト管理
- 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)
- AWS Cost Categories
- AWS によるクラウド財務管理
- AWS APN パートナー - コスト管理

関連動画:

- Deploying Cloud Intelligence Dashboards (Cloud Intelligence Dashboards のデプロイ)
- Get Alerts on any FinOps or Cost Optimization Metric or KPI (FinOps またはコスト最適化のメトリクスまたは KPI に関するアラートを受け取る)

関連サンプル:

- 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
- Well-Architected ラボ: Cloud Intelligence Dashboards

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

使用量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。インサイトとチャージバック機能が利用できる分析サービスにより、コストと使用状況データを分析する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: 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 (恒久的に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 (例えば、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。

共有リソースのワークロードメトリクスの作成は、Amazon Elastic Container Service (Amazon ECS) および Amazon API Gateway のコンテナ化されたアプリケーションなどのリソースに比べて難しい場合があります。ただし、使用量を分類してコストを追跡する方法はあります。Amazon ECS および AWS Batch の共有リソースを追跡する必要がある場合は、AWS Cost Explorer で分割コスト配分データを有効にできます。分割コスト配分データを使用すると、コンテナ化されたアプリケーションのコストと使用状況を把握して最適化し、共有コンピューティングリソースとメモリリソースの消費状況に基づいてアプリケーションコストを個々のエンティティに配分できます。共有 API Gateway および AWS Lambda 機能がある場合は、 AWS Application Cost Profiler を使用し、以下に基づいて消費量を分類できます。 `#### ID` や `##### ID`。

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を使用して、ワークロードの出力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 Quick 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)

関連サンプル:

- AWS 分割コスト配分データにより Amazon ECS および AWS Batch のコストの可視性を向上する

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-BP05 データ保持ポリシーを適用する

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。タグを使用してリソースを追跡する (およびそれらのタグに関するレポートを実行する) ことで、使用されなくなったり、ライセンスの有効期限が切れたりした場合に、廃止する資産を特定するのに役立ちます。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。別の例としては、削除するタグのキー名や値などのリソースに `LifeSpan` または TTL を使用して、廃止する期間や特定の時間を定義する方法があります。

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。タグ付けにより、目的、チーム、環境など、ビジネスに関連した基準でリソースを分類することができます。タグ付けのユースケース、戦略、テクニックの詳細については、AWS タグ付けに関するベストプラクティスを参照してください。

- 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: 入力リクエストまたは出力完了のいずれかをトリガーとして、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ます。ワークロードのリクエストまたは出力がゼロになったときに、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。未使用または使用率の低いリソースの詳細については、AWS Trusted Advisor コストの最適化チェックを参照してください。
- AWS リソースをグループ化する:AWS リソースのグループを作成します。AWS Resource Groups を使用して、同じ AWS リージョン にある AWS リソースを整理および管理できます。ほとんどのリソースにタグを追加して、組織内でのリソースの識別と分類に役立てることができます。Tag Editorを使用して、サポートされているリソースに一括でタグを追加できます。AWS Service Catalog を使用して、承認済みの製品のポートフォリオを作成、管理、およびエンドユーザに配布し、製品ライフサイクルを管理することを検討してください。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor コストの最適化チェック
- AWS リソースのタグ付け
- カスタムメトリクスの発行

関連動画:

- AWS Trusted Advisor を使用してコストを最適化する方法

関連する例:

- AWS リソースを整理する
- AWS Trusted Advisor を使用してコストを最適化する

COST04-BP02 削除プロセスを実装する

未使用の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

組織全体で標準化されたプロセスを導入し、未使用のリソースを特定し、排除します。このプロセスでは、組織のすべての要件が満たされていることを検証するために、検索を実行する頻度と、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、およびワークロードの各リソースが使用中であるかどうかを検証する方法を網羅する必要があります。リソースを削除するために必要なステップを詳述し、サービスから削除すると同時に、規制要件の遵守を確保します。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含める必要があります。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します。

  プロセスの一部として何を確認する必要があるかについては、以下の削除手順を使用してください。

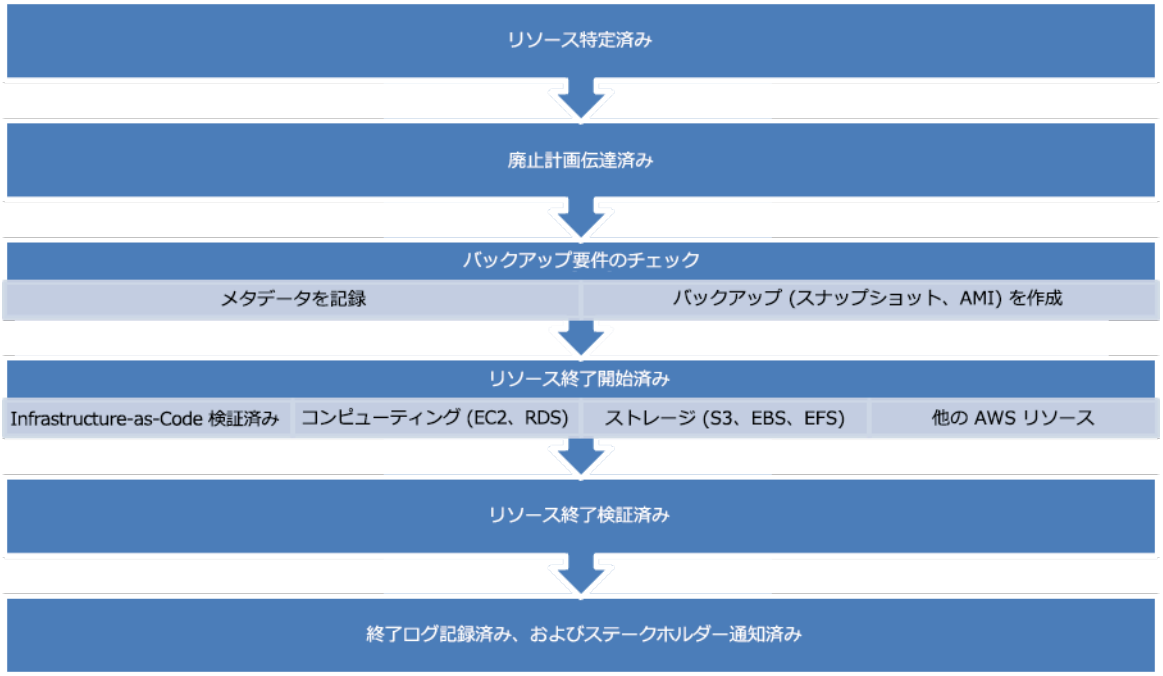
  - 削除されるリソースを特定する: AWS クラウド で削除対象となるリソースを特定します。必要な情報をすべて記録し、削除スケジュールを設定します。タイムラインでは、プロセス中に予期せぬ問題が発生した場合 (およびそのタイミング) を考慮してください。
  - 調整し、コミュニケーションを取る: ワークロード所有者と協力し、削除されるリソースを確認します。
  - メタデータを記録し、バックアップを作成する: 本番環境のリソースに必要な場合、または重要なリソースである場合、メタデータ (パブリック IP、リージョン、AZ、VPC、サブネット、セキュリティグループなど) の記録とバックアップ (Amazon Elastic Block Store スナップショットや AMI の取得、キーのエクスポート、証明書のエクスポートなど) を作成します。
  - Infrastructure-as-Code を検証する: 必要に応じて再デプロイできるように、リソースが、CloudFormation、Terraform、AWS Cloud Development Kit (AWS CDK)、またはその他の Infrastructure-as-Code デプロイツールでデプロイされたかどうかを判断します。
  - アクセスを禁止する: リソースが必要かどうかを判断する間、一定期間、制限的な制御を適用し、リソースの使用を禁止します。必要に応じて、リソース環境を元の状態に戻せることを確認します。
  - 組織内の削除プロセスに従う: 組織のドメインからのリソースの削除、DNS レコードの削除、構成管理ツール、監視ツール、自動化ツール、セキュリティツールからのリソースの削除など、組織の管理タスクと削除プロセスに従います。

リソースが Amazon EC2 インスタンスの場合、次のリストを参照してください。詳細については、「Amazon EC2 リソースを削除または終了するには」をご参照ください。

- Amazon EC2 インスタンスとロードバランサーをすべて停止または終了します。終了後、Amazon EC2 インスタンスは、終了後、少しの間だけコンソールに表示されます。実行状態にないインスタンスには課金されません。
- Auto Scaling インフラストラクチャを削除します。
- すべての専有ホストを解放します。
- すべての Amazon EBS ボリュームおよび Amazon EBS スナップショットを削除します。
- すべての Elastic IP アドレスを解放します。
- すべての Amazon Machine Images (AMI) の登録を解除します。
- すべての AWS Elastic Beanstalk 環境を終了します。

リソースが Amazon Glacier ストレージ内のオブジェクトであり、最小保存期間を満たす前にアーカイブを削除した場合、日割り計算による早期削除料が課金されます。Amazon Glacier の最小保存期間は、使用するストレージクラスによって異なります。各ストレージクラスの最小保存期間の概要については、「Amazon S3 ストレージクラス全体のパフォーマンス」をご参照ください。早期削除料金の計算方法の詳細については、Amazon S3 の料金を参照してください。

次の簡単な削除プロセスのフローチャートは、削除手順を概説しています。リソースを削除する前に、削除対象として特定したリソースが組織で使用されていないことを確認します。



リソースの削除フロー。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

関連動画:

- CloudFormation スタックは削除するが、一部のリソースを保持する
- どのユーザーが Amazon EC2 インスタンスを起動したかを確認する

関連する例:

- Amazon EC2 リソースの削除および終了
- どのユーザーが Amazon EC2 インスタンスを起動したかを確認する

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、手動または自動で実行できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

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

- リソースの削除: これは、不要になった AWS リソースやライセンス契約の終了の減価償却段階です。スナップショットやバックアップの取得などの不要な中断を防ぐために、削除段階に移行して

リソースを削除する前に完了したすべての最終チェックを完了します。削除プロセスを使用して、未使用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

関連する例:

- Well-Architected ラボ: リソースを削除する (レベル 100)

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 Auto Scaling を使用して、削除プロセスを実行できます。また、API または SDK を使用してカスタムコードを運用して、ワークロードリソースの自動削除もできます。

モダンアプリケーションは、サーバーレスサービスの採用を優先する戦略であるサーバーレスファーストで構築されています。AWS は、スタックの 3 つのレイヤー (コンピューティング、インテグレーション、データストア) すべてに対応するサーバーレスサービスを開発しました。サーバーレスアーキテクチャを使用すると、トラフィックの少ない間、自動的にスケールアップおよびスケールダウンしてコストを節約できます。

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 を使用して設定します。AWS サービスを利用する際に、AWS Auto Scaling を使用することで利用率とコスト効率を最適化することができます。AWS Auto Scaling は需要が低下すると、余分なリソースを自動的に削除するため、過剰な支出を避けることができます。

- 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 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。
- サーバーレスサービスを使用する: AWS で、サーバーレスアーキテクチャやイベント駆動型アーキテクチャを優先的に構築し、アプリケーションの構築と実行を行います。AWS では、本来、自動的に最適化されたリソース使用率と自動削除 (スケールインとスケールアウト) を提供する複数のサーバーレステクノロジーサービスが用意されています。サーバーレスアプリケーションでは、リソース使用率が自動的に最適化され、過剰プロビジョニングの費用が発生しません。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS でのサーバーレス
- 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
- Amazon EC2 Auto Scaling の開始方法
- Amazon CloudWatch アラームへの終了アクションの追加

関連する例:

- Scheduling automatic deletion of AWS CloudFormation stacks (AWS CloudFormation スタックの自動削除のスケジューリング)
- Well-Architected ラボ – 自動的にリソースを削除する (レベル 100)
- Servian AWS Auto Cleanup

COST04-BP05 データ保持ポリシーを適用する

サポートされるリソースでデータ保持ポリシーを定義し、組織の要件に従ってオブジェクト削除を処理します。不要または孤立したリソースや不要になったオブジェクトを、特定して削除します。

このベストプラクティスを確立しない場合のリスクレベル: 中

データ保持ポリシーとライフサイクルポリシーを使用して、特定されたリソースの廃止プロセスに関連するコストとストレージコストを削減します。データ保持ポリシーとライフサイクルポリシーを定義してストレージクラスの移行や削除を自動で実行すると、ライフタイム期間の全体的なストレージコストが削減されます。Amazon Data Lifecycle Manager を使用して、Amazon Elastic Block Store スナップショットおよび Amazon EBS 搭載 Amazon マシンイメージ (AMI) の作成と削除を自動化できます。また、Amazon S3 Intelligent-Tiering または Amazon S3 ライフサイクル設定を使用して、Amazon S3 オブジェクトのライフサイクルを管理できます。ほかにも、API または SDK を使用してカスタムコードを実装し、ライフサイクルポリシーとポリシールールを作成して、オブジェクトを自動で削除することもできます。

実装手順

- Amazon Data Lifecycle Manager の使用: Amazon Data Lifecycle Manager でライフサイクルポリシーを使用して、Amazon EBS スナップショットと Amazon EBS 搭載 AMI の削除を自動化します。
- バケットにライフサイクル設定をセットアップする: バケットで Amazon S3 ライフサイクル設定を使用して、ビジネス要件に基づき、オブジェクトのライフサイクル中に Amazon S3 が実行するアクション、およびオブジェクトのライフサイクル終了時の削除を定義します。

リソース

関連するドキュメント:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- Amazon S3 バケットにライフサイクル設定を設定する方法

関連動画:

- Automate Amazon EBS Snapshots with Amazon Data Lifecycle Manager (DLM を使用して EBS スナップショットを自動化する)
- Empty an Amazon S3 bucket using a lifecycle configuration rule (ライフサイクル設定ルールを使用して Amazon S3 バケットを空にする)

関連する例:

- Empty an Amazon S3 bucket using a lifecycle configuration rule (ライフサイクル設定ルールを使用して Amazon S3 バケットを空にする)
- Well-Architected Lab: Decommission resources automatically (Level 100) (Well-Architected ラボ: 自動的にリソースを削除する (レベル 100))

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、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。

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: 高

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 料金見積りツール 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 料金見積りツール

- 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、組織にかかる全体的なコストを調べます。運用および管理のコスト、特にクラウドプロバイダーが提供するマネージドサービスを使用するコストを考慮して、総保有コストを計算します。レビューを行う際には、潜在的利益 (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) を織り込む必要があります。

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: 高

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、ビジネスの差別化要素の構築に集中できるようにします。例えば、データベースをできるだけ迅速にオンプレミス環境からクラウドにリフトアンドシフト (リホストともいいます) して、後で最適化する必要がある場合です。AWS でマネージドサービスを利用して、ライセンスコストの排除または削減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。AWS のマネージドサービスによって、OS のパッチ適用やアップグレードなど、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、イノベーションとビジネスに集中できます。

マネージドサービスはクラウド規模で運用されるため、トランザクションまたはサービス単位でコストを削減できます。アプリケーションのコアアーキテクチャを変更せずに、具体的なメリットを生み出すための潜在的最適化作業を行うことができます。例えば、Amazon Relational Database Service (Amazon RDS) などの database-as-a-service プラットフォームに移行するか、AWS Elastic Beanstalk などのフルマネージド型のプラットフォームにアプリケーションを移行することで、データベースインスタンスの管理に費やす時間を削減できます。

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 マネジメントコンソール 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 RDS、Amazon Redshift、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 Athena、Amazon EMR、Amazon OpenSearch Service はマネージド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。
- 管理されているリソースと管理されていないリソースを比較する: 自社で管理するリソースの運用コストを考え、それを AWS が管理するリソースと比較します。例えば、Amazon EC2 インスタンスで実行しているデータベースをレビューし、Amazon RDS (AWS マネージドサービス) というオプションと比較します。または、Amazon EMR と、Amazon EC2 で Apache Spark を実行する場合を比較します。セルフマネージドワークロードから AWS のフルマネージドワークロードに移行する際は、オプションを慎重に研究してください。最も重要な 3 つの考慮事項は、使用するマネージドサービスの種類、データ移行に使用するプロセス、AWS 責任共有モデルの理解です。

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
- AWS クラウド 製品
- AWS 責任共有モデル

関連動画:

- Why move to a managed database? (マネージドデータベースに移行する理由)
- What is Amazon EMR and how can I use it for processing data? (EMR の概要とこれを使用したデータの処理方法)

関連する例:

- Why move to a managed database? (マネージドデータベースに移行する理由)
- Consolidate data from identical SQL Server databases into a single Amazon RDS for SQL Server database using AWS DMS (AWS を使用してさまざまな SQL サーバーデータベースから単一の Amazon RDS for SQL Server データベースにデータを統一する)
- Deliver data at scale to Amazon Managed Streaming for Apache Kafka (Amazon MSK) (Amazon Managed Streaming for Apache Kafka (Amazon MSK) に大規模にデータを配信する)
- Migrate an ASP.NET web application to AWS Elastic Beanstalk (ASP.NET ウェブアプリケーションを AWS Elastic Beanstalk に移行する)

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

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

ワークロードの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、またはサーバーレス、コンテナ、イベント駆動型アーキテクチャを使用して、全体のコストを削減することが含まれます。オープンソースソフトウェアやライセンス料金がかからないソフトウェア、または代替品を使用して、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

すべてのコンポーネントを選択する際は、サービスのコストとオプションを考慮します。これには、Amazon Relational Database Service (Amazon RDS)、Amazon DynamoDB、Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Email Service (Amazon SES) などの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。コンピューティングには AWS Lambda などのサーバーレスやコンテナを使用し、静的ウェブサイトには Amazon Simple Storage Service (Amazon S3) を使用します。可能であればアプリケーションをコンテナ化し、Amazon Elastic Container Service (Amazon ECS) や Amazon Elastic Kubernetes Service (Amazon EKS) などの AWS マネージドコンテナサービスを使用します。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。

AWS Lambda、Amazon Simple Queue Service (Amazon SQS)、Amazon SNS、Amazon SES など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。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インするため、コスト配分とコストの帰属が効率的になることです。

サーバーレスサービスでは、イベント駆動型アーキテクチャ (EDA) を使用することもできます。イベント駆動型アーキテクチャはプッシュベースであるため、イベントはルーターで発生してもオンデマンドで取得されます。この方法では、イベントをチェックするために定期的にポーリングする費用が発生しません。つまり、ネットワーク帯域幅の消費を抑え、CPU 使用率は低く、アイドルなフリートキャパシティは少なくなり、SSL/TLS ハンドシェイクも減ります。

サーバーレスの詳細については、Well-Architected Serverless Application Lens (Well-Architected サーバーレスアプリケーションレンズ) 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。需要に合わせてキャパシティーを増やすのではなく、より低いコストでより優れたパフォーマンスを得られる可能性がある他のオプションを検討します。例えば、AWS 上のデータベースに対する予想されるトラフィックをレビューし、インスタンスサイズを増やすか、Amazon ElastiCache サービス (Redis または Memcached) を使用してデータベースにキャッシュメカニズムを提供するかを検討します。
- イベント駆動型アーキテクチャを評価する: サーバーレスアーキテクチャを使用すると、分散マイクロサービスベースのアプリケーション向けにイベント駆動型アーキテクチャを構築することも

できます。これを利用すると、スケーラブルで回復性が高く、迅速かつコスト効果の高いソリューションを構築できます。

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
- AWS サーバーレス
- EDA とは
- Amazon S3 ストレージクラス
- クラウド製品
- Amazon ElastiCache (Redis OSS)

関連する例:

- Getting started with event-driven architecture (イベント駆動型アーキテクチャで開始する)
- イベント駆動型アーキテクチャとは
- How Statsig runs 100x more cost-effectively using Amazon ElastiCache (Redis OSS) (Amazon ElastiCache for Redis を使用して Statsig をコスト効果が 100 倍高い方法で実行するには)
- AWS Lambda 関数を使用するためのベストプラクティス

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 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。

データを AWS クラウドに移動する必要がある場合、AWS が提供するバリエーション豊かな製品やパートナーツールを選択して、データセットを移行できます。データセットは、ファイル、データベース、マシンイメージ、ブロックボリューム、あるいはテープバックアップであっても構いません。例えば、大量のデータを AWS に対して入出力する場合や、エッジでデータを処理する場合、AWS の目的別デバイスのいずれかを使用して、コスト効果が高い方法でペタバイト規模のデータをオフラインで移動できます。別の例としては、より速いデータ転送速度が必要な場合、VPN よりも、ビジネスに必要な安定した接続性能を提供する直接接続サービスの方が安価な場合があります。

さまざまな使用状況において繰り返したコスト分析を基にして、スケーリングアクティビティをレビューします。結果を分析して、複数のインスタンスタイプと購入オプションを使用したインスタンスの追加に合わせてスケーリングポリシーを調整できるかを確認します。設定をレビューして、最小限を削減してもユーザーリクエストを処理できる (ただしより小さなフリートサイズで) かを確認し、予想される高需要を満たすためにリソースを追加します。

組織の関係者と話し合って、さまざまな使用状況について繰り返しコスト分析を実行し、AWS Cost Explorer の予測機能を使用して、サービス変更によって発生する可能性のある影響を予測します。AWS Budgets、CloudWatch 請求アラーム、AWS Cost Anomaly Detection を使用して使用状況レベルのトリガーをモニタリングし、最もコスト効果が高いサービスをなるべく迅速に特定して実装します。

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。これまでと今後両方のコストと使用量の増加についてビジネス上の関係者と話し合い、増加がビジネス要件に沿ったものであることを確認します。自社の AWS リソースを使用するユーザーが増える日、週、月を特定します。そのタイミングで既存のリソースのキャパシティを増やすか追加サービスを導入して、コストを削減しパフォーマンスを向上させる必要があります。
- 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、コストと変化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。つまり、コストが増えていれば、ビジネスにおける使用量も同様に増えているはずです。

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
- Amazon EC2 Auto Scaling
- クラウドデータの移行
- AWS Snow Family

関連動画:

- AWS OpsHub for Snow Family

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 (ビジネスニーズや既存のコミットメントなど) を特定して、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 (全体コスト) 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。コストの考慮事項を理解すると、計画されたワークロードのデプロイにおける価値実現の成果評価時に、組織のビジネスケースや意思決定プロセスがわかります。

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。このプロセスのベストプラクティスについては、AWS Well-Architected フレームワークのパフォーマンス効率の柱に関するレビューセクションを参照してください。

例えば、コンピューティングリソースからなるワークロードのコストモデリングを作成する場合、AWS Compute Optimizer がワークロード実行におけるコストの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。CloudWatch エージェントが Amazon EC2 インスタンスにデプロイされていることを確認し、AWS Compute Optimizer 内でより正確な推奨事項を得ることができるメモリメトリクスを収集します。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であるため、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す。

他のサービスやワークロードコンポーネントのサイズを適正化するために、カスタムログをデータソースとして使用できるサービスが複数あります。例えば、AWS Trusted Advisor、Amazon CloudWatch、Amazon CloudWatch Logs などです。AWS Trusted Advisor はリソースをチェックして使用率が低いリソースにフラグを立てるため、リソースのサイズを適正化しコストモデリングを作成するのに役立ちます。

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。

- モニタリングはユーザー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。
- 既存のコミットメント、他のワークロード用に選択された料金モデル、イノベーションを迅速化しコアビジネスバリューに集中する能力を考慮し、計画されたワークロードに対して適切な AWS サービスを選択します。

実装手順

- リソースの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。その後、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、テストをもう一度実行します。コストモデリングの際には、これらのリソースで使用する可能性のある製品のライセンス料と、これらのリソースをデプロイおよび管理する推定運用 (作業またはエンジニア) コストを含めます。一定期間 (時間単位、日次、月次、年次、三年次) のコストモデリングを考慮します。

リソース

関連するドキュメント:

- AWS Auto Scaling
- Identifying Opportunities to Right Size (サイズ適正化の機会を特定する)
- Amazon CloudWatch の機能
- コスト最適化: Amazon EC2 サイズの適正化
- AWS Compute Optimizer
- AWS 料金見積りツール

関連する例:

- Perform a Data-Driven Cost Modelling (データ駆動型のコストモデリングを実行する)
- 計画した AWS リソース設定のコストを見積もる
- 適切な AWS ツールの選択

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。コンピューティング、ストレージ、データ、ネットワーキングなどのサービスに対して、適切なスループット、サイジング、ストレージのプロビジョニングを行います。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。AWS Auto Scaling などのマネージドサービスを使用して、適切なサイジング操作を実行できます。AWS では、API や SDK のほか、最小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 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、

適切なサイジング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。

AWS サービスの中には、Amazon Simple Storage Service Intelligent-Tiering のように、タイプやサイズを自動的に選択する機能が組み込まれているものがあります。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

- ワークロードのメトリクスを設定して可観測性を高める: ワークロードの主要なメトリクスを取得します。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。コンピューティングリソースの場合、パフォーマンスデータを分析して Amazon EC2 インスタンスのサイズを適切に設定します。アイドル状態のインスタンスと利用率の低いインスタンスを特定します。注目すべき重要なメトリクスは、CPU 使用率とメモリ使用率です (たとえば、AWS Compute Optimizer およびメモリ使用率の有効化による適切なサイジングで説明したように、90% の時間で 40% の CPU 使用率)。4 週間の最大 CPU 使用率およびメモリ使用率が 40% 未満のインスタンスを特定します。これらのインスタンスは、コスト削減のために適切なサイズを設定する必要があります。Amazon S3 などのストレージリソースについては、Amazon S3 ストレージレンズを使用することで、バケットレベルでさまざまなカテゴリの 28 のメトリクスを確認でき、デフォルトで 14 日間の履歴データをダッシュボードで確認することが可能です。Amazon S3 ストレージレンズのダッシュボードは、要約、コスト最適化、またはイベントごとにフィルタリングして特定のメトリクスを分析できます。
- 適切なサイジングのレコメンデーションを表示する:AWS Compute Optimizer の適切なサイジングのレコメンデーションとコスト管理コンソールの Amazon EC2 の適切なサイジングツールを使用するか、リソースの適切なサイジングを行う AWS Trusted Advisor を確認してワークロードを調整します。さまざまなリソースの適切なサイジングを行う場合、適切なツールを使用し、Amazon EC2 インスタンス、AWS ストレージクラス、または Amazon RDS インスタンスタイプのいずれであっても、適切なサイジングのガイドラインに従うことが重要です。ストレージリソースには、Amazon S3 ストレージレンズを使用できます。これにより、オブジェクトストレージの使用状況、アクティビティの傾向を可視化し、コストを最適化してデータ保護のベストプラクティスを適用するための実用的な推奨事項を作成できます。Amazon S3 ストレージレンズが組織全体のメトリクスの分析から得た状況に応じた推奨事項を使用して、ストレージを最適化するための手順をすぐに実行できます。
- メトリクスに基づいて自動的にリソースタイプとサイズを選択する: 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で選択します。コンピューティングリソースの

場合、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。1 つの Auto Scaling グループ内でオンデマンドインスタンスとスポットインスタンスのフリートを起動し、自動的にスケールできます。スポットインスタンスの利用による割引に加え、リザーブドインスタンスや Savings Plans を利用して、通常のオンデマンドインスタンスの価格から割引を受けることができます。これらの要素をすべて組み合わせることで、Amazon EC2 インスタンスのコスト削減を最適化し、アプリケーションに必要なスケールとパフォーマンスを決定できます。また、Auto Scaling グループ (ASG) で属性ベースのインスタンスタイプ選択 (ABS) 戦略を使用すると、vCPU、メモリ、ストレージなどの属性セットとしてインスタンスの要件を表現できます。新しい世代のインスタンスタイプがリリースされると自動的に使用し、さらに Amazon EC2 スポットインスタンスでより広い範囲のキャパシティにアクセスできます。Amazon EC2 フリートと Amazon EC2 Auto Scaling が指定した属性に適合するインスタンスを選択して起動するため、手動でインスタンスタイプを選択する必要がなくなります。ストレージリソースには、Amazon S3 Intelligent Tiering と Amazon EFS 低頻度アクセス機能を利用できます。これらの機能のおかげで、データアクセスのパターンが変化しても、パフォーマンスへの影響や運用上のオーバーヘッドなしに、ストレージコストを自動的に削減するストレージクラスを自動選択できるようになります。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS の適切なサイジング
- AWS Compute Optimizer
- Amazon CloudWatch の機能
- CloudWatch の準備
- CloudWatch カスタムメトリクスの発行
- Amazon EC2 Auto Scaling の開始方法
- Amazon S3 ストレージレンズ
- Amazon S3 Intelligent-Tiering
- Amazon EFS 低頻度アクセス
- SDK を使用して Amazon EC2 インスタンスを起動する

関連動画:

- サービスの適切なサイズ設定

関連する例:

- Amazon EC2 フリート向け Auto Scaling 用の属性ベースのインスタンスタイプの選択
- スケジュールされたスケーリングを使用したコストに対する Amazon Elastic Container Service の最適化
- Amazon EC2 Auto Scaling で予測スケーリング
- Optimize Costs and Gain Visibility into Usage with Amazon S3 Storage Lens (Amazon S3 ストレージレンズを使用してコストを最適化し、使用状況を可視化する)
- Well-Architected ラボ: 適切なサイジングのレコメンデーション (レベル 100)
- Well-Architected ラボ: AWS Compute Optimizer および　メモリ使用率の有効化によるサイズ適正化 (レベル 200)

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 (スポットまたはオンデマンド) とするかを決定します。コスト管理ツールのレコメンデーションを使用して、ワークロードに対して分析を行います。これらのレコメンデーションにはビジネスルールを適用して、高いリターンを実現します。

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、組織のニーズと製品に合った最も費用対効果の高い方法でリソース料金を支払うことができます。チームと協力して最適な料金モデルを決定します。可用性にもとづいて決定すると、複数のオプションを組み合わせた料金モデルになることもよくあります

オンデマンドインスタンスを利用すると、コンピューティング容量またはデータベース容量の料金を、長期コミットメントや前払い金なしで、実行するインスタンスによって 1 時間単位または秒単位 (最小 60 秒) で支払うことができます。

Savings Plans は、1 年または 3 年の期間で一定の使用量 (1 時間あたりのドルで計測) をコミットする代わりに、Amazon EC2、Lambda、AWS Fargate を低価格で使用できる柔軟な料金モデルです。

スポットインスタンスは Amazon EC2 の料金の仕組みであり、予備のコンピューティング容量を割引価格の時間単価 (オンデマンド価格の最大 90% オフ) でリクエストでき、前払い金は不要です。

リザーブドインスタンスを利用すると、容量に対して前払いすることで、最大 75% の割引を受けることができます。詳細については、予約を利用したコストの最適化を参照してください。

本稼働、品質、開発の各環境に関連するリソースに、Savings Plans を含めることもできます。あるいは、サンドボックスリソースは必要なときにのみ稼働するため、そういう環境のリソースにオンデマンドモデルを選択することもできます。Amazon スポットインスタンスを使用して Amazon EC2 のコストを削減したり、コンピューティング Savings Plans を使用して Amazon EC2、Fargate、Lambda のコストを削減したりします。AWS Cost Explorer レコメンデーションツールは、Savings Plans を利用したコミットメント割引の機会を提供します。

過去に Amazon EC2 のリザーブドインスタンスを購入したり、組織内で既にコスト配分を実践したりしている場合は、当面は Amazon EC2 リザーブドインスタンスを引き続き使用できます。ただし、より柔軟にコストを節約できる仕組みとして、いずれは Savings Plans を使用する戦略に則ることをお勧めします。AWS Cost Management の Savings Plans (SP) レコメンデーションを更新して、いつでも新しい Savings Plans レコメンデーションを作成できます。リザーブドインスタンス (RI) を使用して、Amazon RDS、Amazon Redshift、Amazon ElastiCache、Amazon OpenSearch Service のコストを削減します。Savings Plans とリザーブドインスタンスには、全額前払い、一部前払い、前払いなしの 3 つのオプションがあります。AWS Cost Explorer RI および SP 購入レコメンデーションで提供されたレコメンデーションを使用します。

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、定期に生じる使用量や伸縮性の変化を探します。スポットインスタンスは、耐障害性と柔軟性が高いさまざまなアプリケーションに使用できます。これには、ステートレスウェブサーバー、API

エンドポイント、ビッグデータアプリケーションや分析アプリケーション、コンテナ化されたワークロード、CI/CD、その他柔軟性の高いワークロードなどがあります。

Amazon EC2 および Amazon RDS インスタンスを、使用していないとき (就業後や週末) にオフにできるか、分析します。このアプローチによって、24 時間 365 日使用する場合と比較して 70% 以上のコスト削減になります。特定の時間にのみ使用できるようにする必要がある Amazon Redshift クラスターがある場合は、そのクラスターを一時停止して、後で再開できます。Amazon Redshift クラスターや Amazon EC2 および Amazon RDS インスタンスが停止すると、コンピューティングに対する請求が停止され、ストレージ料金のみが適用されます。

オンデマンドキャパシティ予約 (ODCR) は料金割引ではないことに注意してください。キャパシティ予約は、リザーブドキャパシティでインスタンスを実行しているかどうかにかかわらず、同等のオンデマンド料金で課金されます。キャパシティ予約は、実行する予定のリソースに対して十分な容量を提供する必要がある場合に、検討します。ODCR は、必要がなくなればキャンセルできるため、長期コミットメントと結びつける必要はありませんが、Savings Plans またはリザーブドインスタンスが提供する割引のメリットを受けることもできます。

実装手順

- 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て、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 Lab: Cost Explorer (Well-Architected ラボ: コストの可視化)
  - Well-Architected Lab: Cost Visualization (Well-Architected ラボ: コストの可視化)
- 既存の料金契約を見直す: 長期にわたる必要性について、現在の契約やコミットメントを見直します。現在締結しているものと、それらのコミットメントをどの程度使用しているかを分析します。既存の契約による割引やエンタープライズ契約を活用します。エンタープライズ契約では、お客様のニーズに最適になるように契約を調整するオプションがあります。長期コミットメントの場合、リザーブド料金割引、リザーブドインスタンス、特定のインスタンスタイプ、インスタンスファミリー、AWS リージョン、アベイラビリティーゾーンにおける Savings Plans を検討します。
- 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
- AWS エンタープライズ

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 Lab: Cost Explorer (Well-Architected ラボ: コストの可視化)
- Well-Architected Lab: Cost Visualization (Well-Architected ラボ: コストの可視化)
- Well-Architected Lab: Pricing Models (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。リージョンによるコストの差異を特定し、レイテンシー、データレジデンシーおよびデータ主権に関する要件を満たす場合にのみ、よりコストの高いリージョンにデプロイします。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: 中

実装のガイダンス

それらの AWS クラウド インフラストラクチャ はグローバルであり、 世界中の複数の場所でホストされており、AWS リージョン、アベイラビリティーゾーン、Local Zones、AWS Outposts、Wavelength Zones を中心に構築されています。リージョンとは世界中の物理的な場所であり、各リージョンは、AWS が複数のアベイラビリティーゾーンを設置している地理的に離れた地域です。各リージョン内の複数の独立した場所である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、接続を備えています。

各 AWS リージョン は現地マーケットの条件内で運用されており、土地代、回線代、電気代、税金などのコストが異なるため、リソース料金は各リージョンで異なります。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。ここでは AWS 計算ツール を使用して、ロケーションタイプ (リージョン、Wavelength Zone、ローカルゾーン) とリージョンごとにサービスを検索し、さまざまなリージョンでワークロードのコストを見積もります。

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。ビジネス、データプライバシー、パフォーマンス、セキュリティの要件に基づいて、地理的場所を選択します。エンドユーザーが世界中にいるアプリケーションの場合は、複数の場所を使用します。

データプライバシー、セキュリティ、ビジネス要件に義務がない場合は、AWS のサービスの料金がより低いリージョンを使用してワークロードをデプロイします。例えば、デフォルトのリージョンが ap-southeasth-2 (シドニー) であり、他のリージョンを使用するにあたっての制約 (データプライバシー、セキュリティなど) がない場合、重要ではない (開発とテスト) Amazon EC2 インスタンスを north-east-1 (バージニア北部) リージョンにデプロイすると、コストを抑えることができます。

	コンプライアンス	レイテンシー	コスト	サービス/機能
リージョン 1	✓	15 MS	$$	✓
リージョン 2	✓	20 MS	$$$	X
リージョン 3	✓	80 MS	$	✓
リージョン 4	✓	15 MS	$$	✓
リージョン 5	✓	20 MS	$$$	X
リージョン 6	✓	15 MS	$	✓
リージョン 7	✓	80 MS	$	✓
リージョン 8	✓	15 MS	$	X

リージョン機能マトリックステーブル

前述のマトリックステーブルから、他のリージョンに比べてレイテンシーが低く、サービスが利用可能で、コストが最も低いリージョンであるため、このシナリオではリージョン 4 が最適なオプションであることがわかります。

実装手順

- AWS 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。
- 複数のリージョンにデプロイする場合の要件を確認する: ビジネス要件と義務 (データプライバシー、セキュリティ、パフォーマンス) を分析して、複数リージョンを使用すべきでない制約があるかどうかを確認します。単一リージョンを使用するよう制限する義務がない場合は、複数のリージョンを使用します。
- 必要なデータ転送を分析する: リージョンを選択する際は、データ転送コストを考慮します。データは顧客に近く、またリソースに近いところに置いてください。データ転送が最小限でデータの流れがいい、よりコストの低い AWS リージョンを選択します。データ転送のビジネス要件によって、 Amazon CloudFront、AWS PrivateLink、 AWS Direct Connect、 AWS Virtual Private Network を使用して、ネットワークコストの削減、パフォーマンスの向上、セキュリティの強化を実現できます。

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

関連サンプル:

- Overview of Data Transfer Costs for Common Architectures (一般的なアーキテクチャでのデータ転送コストの概要)
- Cost Considerations for Global Deployments (グローバルデプロイにおけるコストの考慮事項)

- 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)
- Well-Architected ラボ: Restrict service usage by Region (Level 200) (リージョンごとにサービスの仕様を制限する (レベル 200))

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

請求やコスト管理ツールをチェックして、コミットメントや予約を利用した推奨される割引を確認し、管理アカウントレベルで定期的に分析を実行します。

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 のレコメンデーションツールを使用して、管理アカウントでコミットメント割引を適用する機会を見つけます。管理アカウントレベルでのレコメンデーションは、リザーブドインスタンスまたは Savings Plans (SP) 割引共有が有効になっている AWS 組織内のすべてのアカウントにまたがる使用量を考慮して計算され、アカウント全体で節約を最大化できるコミットメントを推奨します。

管理アカウントレベルでの購入は、多くの場合、最大限の節約を目指して最適化されますが、特定の連結アカウントでの利用に最初に割引を適用したい場合など、連結アカウントレベルで SP を購入することを検討する状況もあります。メンバーアカウントのレコメンデーションは個別のアカウントレベルで計算され、それぞれのアカウントごとに節約を最大化します。アカウントが RI と SP の両方のコミットメントを所有している場合、それらは次の順序で適用されます。

ゾーン RI > 標準 RI > コンバーチブル RI > Instance Savings Plan > Compute Savings Plan

管理アカウントレベルで SP を購入した場合、割引率の高いものから低いものに基づいて節約が適用されます。管理アカウントレベルの SP は、すべての連結アカウントを調べて、割引が最も高いところで節約が適用されます。節約が適用される場所を制限したい場合は、連結アカウント単位で Savings Plan を購入すると、そのアカウントが対象となるコンピューティングサービスを実行しているときはいつでも、割引が最初に適用されます。アカウントが対象となるコンピューティングサービスを実行していない場合、割引は同じ管理アカウントにある他の連結アカウント間で分配されます。割引共有はデフォルトでオンになっていますが、必要に応じてオフにできます。

一括請求ファミリーでは、Savings Plans は最初に所有者アカウントの使用に適用され、次に他のアカウントの使用に適用されます。これは、共有を有効にしている場合にのみ発生します。Savings Plans は、最も高い節約率に最初に適用されます。節約率が等しい使用量が複数ある場合は、Savings Plans のレートが最も低い最初の使用に Savings Plans が適用されます。Savings Plans は、残りの使用量がなくなるか、コミットメントが使い果たされるまで引き続き適用されます。残りの使用量はオンデマンド料金で請求されます。AWS Cost Management の Savings Plans レコメンデーションを更新して、いつでも新しい Savings Plans レコメンデーションを作成できます。

インスタンスの柔軟性を分析すると、レコメンデーションに沿ってコミットできます。コストモデリングを作成するにあたって、さまざまなリソースオプションの可能性を含めたワークロードの短期コストを分析し、AWS 料金モデルの分析を行い、それらをビジネス要件に合わせて、総保有コストとコスト最適化 の機会を見出します。

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。Savings Plan のレコメンデーションを理解し、月次費用の概算、月次節約額の概算を行っていることを確認します。リザーブドインスタンスまたは Savings Plans 割引共有が有効になっている AWS 組織内のすべてのアカウントにまたがる使用量を考慮して計算された管理アカウントレベルでのレコメンデーションをレビューし、アカウント全体で節約を最大化します。Well-Architected ラボに従って、必要な割引を適用し、リスクを認識したうえで、正しいレコメンデーションを実装していることを確認します。

リソース

関連するドキュメント:

- AWS の料金の仕組みはどのようになっていますか?
- インスタンス購入オプション

- Saving Plan Overview (Savings Plans の概要)
- Saving Plan recommendations (Savings Plans のレコメンデーション)
- 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
- How Savings Plans apply to your AWS usage (AWS の使用状況に Savings Plans が適用される仕組み)
- Savings Plans with Consolidated Billing (一括請求を利用した Savings Plans)
- 共有リザーブドインスタンスと Savings Plans の割引の有効化

関連動画:

- Save up to 90% and run production workloads on Spot

関連サンプル:

- Well-Architected Lab: Pricing Models (Level 200) (Well-Architected ラボ: 料金モデル (レベル 200))
- Well-Architected Labs: Pricing Model Analysis (Level 200) (Well-Architected ラボ: 料金モデル分析 (レベル 200))
- Savings Plans を購入する前にどのようなことを考慮すべきですか?
- How can I use rolling Savings Plans to reduce commitment risk> (ローリング Savings Plans を使用してコミットメントのリスクを軽減する方法)
- When to Use Spot Instances (スポットインスタンスの使用が適している場合)

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: ミディアム

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: ミディアム

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。

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。

Site-to-Site 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 Quick 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 Quick

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: ミディアム

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

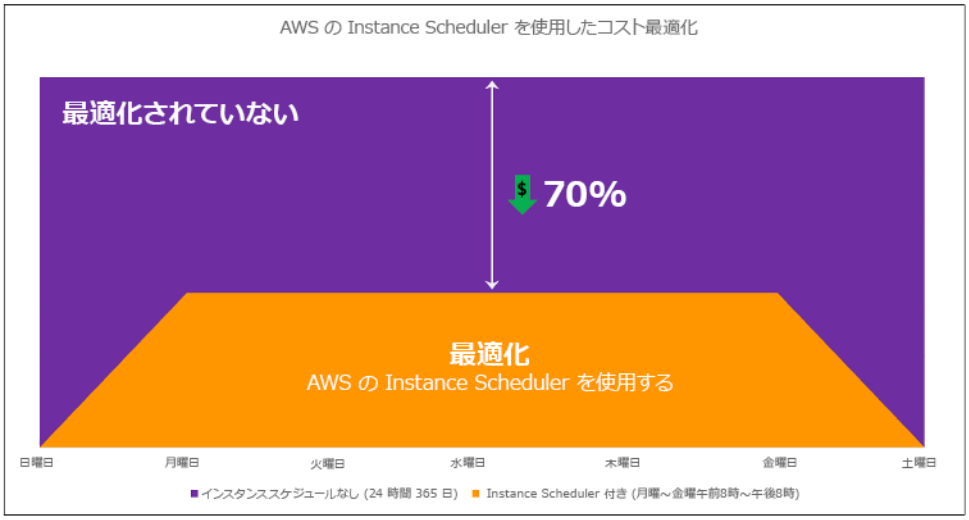
リソースを計画的なやり方でプロビジョニングします。これは、自動スケーリングなどの需要ベース、または需要が予測可能でリソースが時間に基づいて提供される時間ベースで行います。これらの手法を使用すると、過剰プロビジョニングやプロビジョニング不足を最小限に抑えることができます。

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

AWS のお客様がアプリケーションに利用できるリソースを増やし、需要に合わせてリソースを供給する方法はいくつかあります。これらのオプションの 1 つは、Amazon Elastic Compute Cloud (Amazon EC2) および Amazon Relational Database Service (Amazon RDS) インスタンスの起動と停止を自動化する AWS インスタンススケジューラを使用することです。もう 1 つのオプションは AWS Auto Scaling を使用することです。この方法では、アプリケーションやサービスの需要に基づいてコンピューティングリソースを自動的にスケーリングできます。需要に応じてリソースを供給することで、使用したリソースに対してのみ支払いを行い、必要なときにリソースを起動してコストを削減し、必要でないときにリソースを終了することができます。

AWS での Instance Scheduler を使用すると、Amazon EC2 インスタンスや Amazon RDS インスタンスを決まった時間に停止および開始するように設定できます。これにより、例えばユーザーが毎朝 8 時に Amazon EC2 インスタンスにアクセスし、夜 6 時以降は必要としないなど、一貫した時間パターンがある同一リソースの需要に応えることができます。この解決方法では、リソースを使用しないときは停止し、必要なときに開始することで、運用コストを削減できます。



AWS Instance Scheduler によるコストの最適化。

また、AWS Systems Manager クイックセットアップを使用してシンプルなユーザーインターフェイス (UI) を使用して、アカウントやリージョン全体で Amazon EC2 インスタンスのスケジュールを簡単に設定できます。AWS Instance Scheduler を使用して Amazon EC2 または Amazon RDS インスタンスをスケジュールでき、既存のインスタンスを停止および起動できます。ただし、Auto Scaling グループ (ASG) の一部であるインスタンスや、Amazon Redshift や Amazon OpenSearch Service などのサービスを管理しているインスタンスは、停止および開始できません。Auto Scaling グループにはグループ内のインスタンス対して独自のスケジュールがあり、これらのインスタンスが作成されます。

AWS Auto Scaling により、変化する需要に対応するためにキャパシティを調整して、最低限のコストで安定かつ予測可能なパフォーマンスを維持できます。これは、Amazon EC2 インスタンス、スポットフリート、Amazon ECS、Amazon DynamoDB、Amazon Aurora と統合されたアプリケーションのキャパシティを拡張するための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 グループをスケーリングするには、複数のスケーリングオプションを使用できます。

- 常に現在のインスタンスレベルを維持
- 手動スケーリング
- スケジュールに基づくスケーリング
- 需要に応じたスケーリング

- 予測スケーリングを使用する

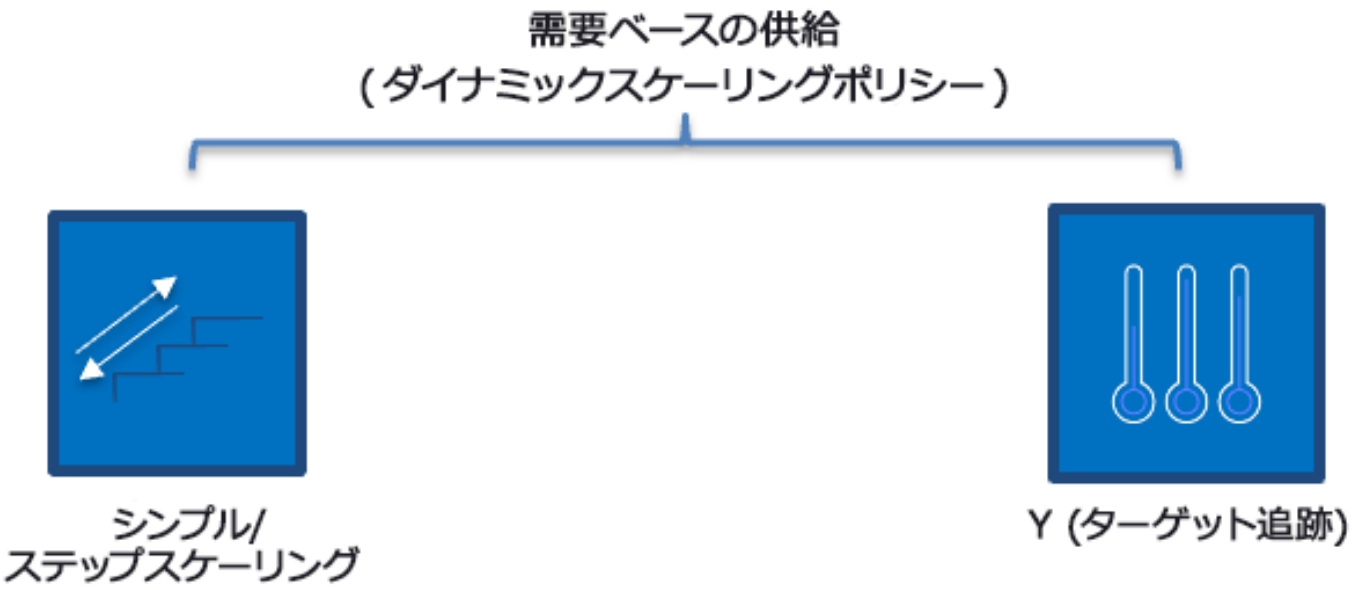
Auto Scaling ポリシーは異なり、動的スケーリングポリシーとスケジュールスケーリングポリシーに分類できます。動的ポリシーには、手動または動的スケーリング、スケジュールスケーリングまたは予測スケーリングがあります。スケーリングポリシーは、動的スケーリング、スケジュールスケーリング、予測スケーリングに使用できます。また、 Amazon CloudWatch のメトリクスとアラームを使用して、ワークロードのスケーリングイベントをトリガーできます。最新の機能や改善点にアクセスできる 起動テンプレートを使用することをお勧めします。起動設定を使用する場合、すべての Auto Scaling 機能が利用できるわけではありません。例えば、スポットインスタンスとオンデマンドインスタンスの両方を起動する、または複数のインスタンスタイプを指定する Auto Scaling グループを作成することはできません。これらの機能を設定するには、起動テンプレートを使用する必要があります。起動テンプレートを使用するときは、それぞれバージョンを作成することをお勧めします。起動テンプレートのバージョニングにより、すべてのパラメータのサブセットを作成できます。その後、それを再利用して同じ起動テンプレートの他のバージョンを作成できます。

AWS Auto Scaling を使用するか、 AWS API または SDK を使用してコードにスケーリングを組み込むことができます。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、変更をより迅速に実行できるようになります。またこれにより、いつでもワークロードのリソースを需要に合わせて調達できます。ベストプラクティスに従って組織に動的にリソースを供給するには、AWS クラウド の水平スケーリングおよび垂直スケーリングと、Amazon EC2 インスタンスで実行されるアプリケーションの特性を理解する必要があります。このベストプラクティスに従うには、クラウド財務管理チームとテクニカルチームが協働することをお勧めします。

Elastic Load Balancing (Elastic Load Balancing) 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。ASG と Elastic Load Balancing を使用する、トラフィックを最適にルーティングして受信リクエストを管理し、Auto Scaling グループ内の 1 つのインスタンスに負荷がかかりすぎないようにすることができます。リクエストは、キャパシティや使用率を考慮せずに、ラウンドロビン方式でターゲットグループのすべてのターゲットに分散されます。

一般的な Amazon EC2 メトリクスは、CPU 使用率、ネットワークスループット、Elastic Load Balancing 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の標準メトリクスです。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。このドキュメントでは、需要を動的に満たす方法を詳しく説明するために、Auto Scaling を需要ベースの供給モデルと時間ベースの供給モデルの 2 つのカテゴリに分類し、それぞれについて詳しく説明します。

需要ベースの供給: クラウドの伸縮性を活用して、ほぼリアルタイムの需要状況に応じて、変化する需要に対応するリソースを供給できます。需要ベースの供給の場合、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を減少させてコストを節減させたりできます。



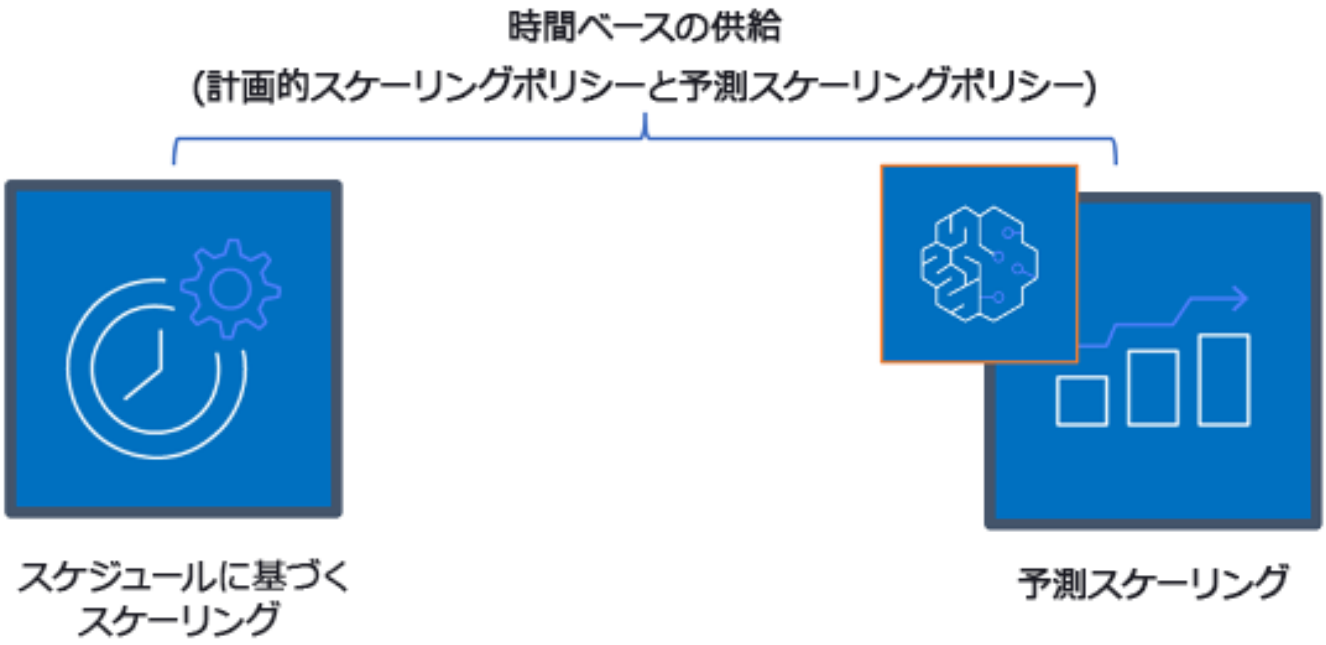
需要ベースの動的スケーリングポリシー

- シンプル/ステップスケーリング: メトリクスをモニタリングし、カスタマーが手動で定義したステップに従ってインスタンスを追加/削除します。
- ターゲット追跡: サーモスタットのような制御メカニズムで、インスタンスを自動的に追加または削除して、メトリクスをカスタマー定義の目標に維持します。

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。

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、キャパシティを拡大したりできます。



時間ベースのスケーリングポリシー

スケジュールされたまたは予測される 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、特定の時間にワークロードを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増加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。予測スケーリングでは、パターンを使用してスケールアウトします。一方スケジュールに基づくスケーリングでは、事前に定義された時間を使用してスケールアウトします。属性ベースの インスタンスタイプの選択 (ABS) 戦略を Auto Scaling グループで使用することもできます。これにより、インスタンスの要件を vCPU、メモリ、ストレージなどの属性のセットとして表現できます。これにより、新しい世代のインスタンスタイプがリリースされると自動的に使用し、さらに Amazon EC2 スポットインスタンスでより広い範囲のキャパシティにアクセスできます。Amazon EC2 フリートと Amazon EC2 Auto Scaling が指定した属性に適合するインスタンスを選択して起動するため、手動でインスタンスタイプを選択する必要がなくなります。

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。例えば、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 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

- スケジュールされたスケーリングを設定: 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。時間ベースのスケジューリングを設定するには、予測スケーリングまたはスケジュールに基づくスケーリングを使用し、予想される、または予測可能な負荷の変化に合わせて、事前に Auto Scaling グループの Amazon EC2 インスタンス数を増やすことができます。
- 予測スケーリングの設定: 予測スケーリングを使用すると、トラフィックフローの日次および週次パターンに先立って、Auto Scaling グループ内の Amazon EC2 インスタンス数を増やすことができます。定期的にトラフィックのスパイクがあり、アプリケーションの起動に時間がかかる場合は、予測スケーリングの使用を考慮すべきです。予測スケーリングを使用すると、見積もられた負荷の前にキャパシティを初期化できるため、性質上後手に回る動的スケーリング単体と比較して、より迅速にスケールできます。例えば、ユーザーが始業時間とともにワークロードの仕様を開始し、終業時間後は使用しない場合、予測スケーリングを使用すれば、始業時間前にキャパシティを追加できるため、トラフィックの変化に反応する動的スケーリングで生じる遅延を排除できます。
- 動的自動スケーリングの設定: 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、Auto Scaling を使用してください。分析を使用して、正しいリソースレベルで起動するように 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検証します。1 つの Auto Scaling グループ内でオンデマンドインスタンスとスポットインスタンスのフリートを起動し、自動的にスケールできます。スポットインスタンスの利用による割引に加え、リザーブドインスタンスや Savings Plans を利用して、通常のオンデマンドインスタンスの価格から割引を受けることができます。これらの要素をすべて組み合わせることで、Amazon EC2 インスタンスのコスト削減を最適化し、アプリケーションに必要なスケールとパフォーマンスを実現できます。

リソース

関連するドキュメント:

- AWS Auto Scaling
- AWS での Instance Scheduler
- Auto Scaling グループのサイズをスケールする
- 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
- Amazon EC2 Auto Scaling の予測スケーリング

関連動画:

- Auto Scaling のターゲット追跡スケーリングポリシー
- AWS での Instance Scheduler

関連サンプル:

- Amazon EC2 フリート向け Auto Scaling 用の属性ベースのインスタンスタイプの選択
- スケジュールされたスケーリングを使用したコストに対する Amazon Elastic Container Service の最適化
- Amazon EC2 Auto Scaling で予測スケーリング
- CloudFormation で Instance Scheduler を使用して Amazon EC2 インスタンスをスケジュールするにはどうすればよいですか?

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 11　工数のコストを評価する方法

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、コアワークロードや、請求の 10% 超に値するワークロードは四半期または 6 か月ごとにレビューし、10% 以下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

ワークロードの費用対効果を最大にするには、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。例えば、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。

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定義し、ワークロードコストを最適化します。パフォーマンスの柱のレビューや信頼性の柱のレビューのプロセスと同様、最適化と改善作業を特定、検証、優先順位付けし、修正を公開してバックログに組み込みます。

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。継続的な改善とレビューの周期のための時間とリソースを割り当て、ワークロードの効率性と最適化を向上させます。これは要因の組み合わせであり、組織内のワークロード、またワークロード内のコンポーネントによって、異なる場合があります。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、それぞれ確認することができます。
- 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。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。同時に、それらの活動に必要な作業量を見積もります。予想される結果が目標に達しておらず、作業コストがさらにかかる場合は、代わりの一連のアクションを使用して作業を繰り返します。レビュープロセスには、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。例えば、データベースコンポーネントの分析に 1 週間、コンピューティングリソースの分析に 1 週間、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決めます。

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

関連する例:

- AWS サポートからのプロアクティブなサービス
- SAP ワークロードの定期的なワークロードレビュー

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、それぞれ定義されたプロセスに基づいて定期的に見直され、新しいサービスを導入できるか、既存のサービスを置き換えることができるか、またはワークロードをリアーキテクトできるかを確認します。

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

AWS は定期的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、より迅速に実験やイノベーションできます。AWS の最新情報には、AWS が行っている追加の詳細や、AWS のサービス、機能、リージョン拡大のお知らせの概要が、随時提供されています。発表されたリリースの詳細を確認して、既存のワークロードの見直しや分析にそれらを使用できます。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を行い、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。つまり、場合によっては、ワークロードに使用している既存のサービスを置き換えたり、ワークロードをモダナイズして新しい AWS のサービスを導入したりする必要があるということです。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 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。

ワークロードを分析して潜在的な機会を見出すには、新しいサービスだけではなく、ソリューション構築における新しい方法も考慮する必要があります。AWS の This is My Architecture の動画を確認して、他のお客様のアーキテクチャの設計、課題、解決策について学びます。All-In series を確認して、AWS のサービスの実際のアプリケーションやお客様事例を参照します。また、基本的なクラウドアーキテクチャパターンのベストプラクティスを説明、検証、詳説している Back to Basics 動画シリーズも視聴できます。他のソースとして How to Build This 動画もあります。これは、AWS のサービスを使用して、実用最小限の製品 (MVP) を実現するための素晴らしいアイデアを持つ人々を

支援するように設計されています。確固たるアイデアを持った世界中の構築者が、経験豊富な AWS のソリューションアーキテクトからのアーキテクチャに関するガイダンスを得ることができます。最後に、使用開始のリソース資料を確認します。ステップバイステップのチュートリアルがあります。

レビュープロセスを実行する前に、合意されたレビュープロセスに従いながら、ワークロードにおけるビジネスの要件、特定のサービスまたはリージョンを使用するためのセキュリティおよびデータのプライバシー要件、パフォーマンス要件に従います。

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。
- 新しいサービスの実装: 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
- AWS の最新情報
- AWS ドキュメント
- AWS の使用開始
- AWS の一般的なリソース

関連動画:

- AWS - This is My Architecture
- AWS - Back to Basics
- AWS - All-In series
- How to Build This

COST 11 工数のコストを評価する方法

ベストプラクティス

- COST11-BP01 オペレーションのオートメーションを実行する

COST11-BP01 オペレーションのオートメーションを実行する

クラウドでのオペレーションの労力コストを評価します。管理タスク、デプロイ、その他オペレーションにおけるオートメーションを使用した時間と労力の削減を定量化します。オペレーションの労力に必要な時間とコストを評価し、可能な部分は管理タスクを自動化して、人間による手作業を削減します。

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

オペレーションを自動化すると、安定性とスケーラビリティが向上し、可視性、信頼性、柔軟性が上がります。また、人的リソースを解放しメトリクスを改善することで、コストを削減し、イノベーションを加速できます。ワークロードをデプロイ、管理、運用する際に安定した信頼性の高いエクスペリエンスを提供することで、手動タスクの頻度を減らし、効率を高め、企業にメリットをもたらします。インフラストラクチャのリソースを手動の運用タスクから解放し、それらをより価値の高いタスクやイノベーションに使用できるため、ビジネス成果が向上します。企業は、クラウドでワークロードを管理するための、実績がありテスト済みの方法を求めています。そのようなソリューションは安全、高速でありコスト効果が高く、最小限のリスクで最大限の信頼性を得られるものでなくてはなりません。

クラウドにおけるオペレーションコスト全体を確認して、必要な労力に基づきオペレーションに優先順位をつけることから始めます。例えば、クラウドに新しいリソースをデプロイするのにかかる時間、既存のものを最適化する変更にかかる時間、必要な構成を設定するのにかかる時間はどのくらいでしょうか。 オペレーションと管理のコストを考慮に入れて、人間による操作の合計コストを確認します。管理タスクのオートメーションに優先順位を付けて、人間の手作業を減らします。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、タスクを手動で実行する場合にかかる時間を自動で実行する場合と比較します。繰り返し実行する価値の高いアクティビティを優先します。通常、人的エラーを起こすリスクが高いアクティビティから自動化するのが良い方法です。このようなリスクは望ましくない追加の運用コスト (オペレーションチームの追加作業時間など) を発生させることが多いためです。

AWS のサービス、ツール、サードパーティ製製品を使用して、特定の要件に対してどの AWS オートメーションを実装しカスタマイズするかを選択します。次の表は、AWS のサービスを使用して管

理とオペレーションを自動化することで実現できる主なオペレーション機能や性能の一部を示したものです。

- AWS Audit Manager: AWS の使用状況を継続的に監査し、リスクとコンプライアンスの評価を簡易化します
- AWS Backup: データ保護を一元的に管理および自動化します。
- AWS Config: コンピューティングリソースを設定し、設定とリソースインベントリを査定、監査、評価します。
- AWS CloudFormation: 高可用性リソースを Infrastructure as Code とともに起動します。
- AWS CloudTrail: IT の変更管理、コンプライアンス、コントロール。
- Amazon EventBridge: イベントをスケジュールし、AWS Lambda をトリガーしてアクションを実行します。
- AWS Lambda: 繰り返し実行するプロセスを、イベントでトリガーしたり、Amazon EventBridge を使用して固定スケジュールで実行したりすることで、プロセスを自動化します。
- AWS Systems Manager: ワークロードの開始と停止、オペレーションシステムへのパッチ適用、設定の自動化、管理の続行。
- AWS Step Functions: ジョブをスケジュールし、ワークフローを自動化します。
- AWS Service Catalog: コンプライアンスとコントロールを備えたテンプレート消費および Infrastructure as Code。

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。Amazon Relational Database Service、Amazon EMR、Amazon WorkSpaces、Amazon SageMaker AI など、ライセンスコストを排除または削減する AWS の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。

AWS の製品やサービスの使用と同時にオートメーションを導入したいが、組織にそのスキルがない場合、AWS Managed Services (AMS)、AWS プロフェッショナルサービス、または AWS パートナー に連絡して、オートメーションの導入を増やし、クラウドにおけるオペレーショナルエクセレンスを向上させます。

AWS Managed Services (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。

AWS プロフェッショナルサービス を利用すると、AWS を使用して目的のビジネス成果を達成し、オペレーションを自動化することもできます。AWS プロフェッショナルサービスは、エンタープライズクラウドコンピューティングの該当分野におけるお客様の業務をサポートする、グローバルで専門性の高いプラクティスを提供します。専門性の高いプラクティスは、ソリューション、テクノロジー、業界の対象分野にわたるベストプラクティス、フレームワーク、ツール、サービスを通じて、対象を絞ったガイダンスを提供します。このガイダンスを使用して、自動化された堅牢で俊敏な IT オペレーションや、クラウドセンターに最適化されたガバナンス機能をデプロイできます。

実装手順

- 構築は 1 回、デプロイは多数: AWS CloudFormation、AWS SDK、AWS Command Line Interface (AWS CLI) などの infrastructure-as-Code を使用して 1 回デプロイし、同じ環境やディザスタリカバリのシナリオで何度も使用します。デプロイ中にタグを付け、他のベストプラクティスで定義されている消費を追跡します。AWS Launch Wizard を使用して、多数の一般的なエンタープライズワークロードをデプロイする回数を削減します。AWS Launch Wizard は、AWS のベストプラクティスに従って、エンタープライズワークロードのサイズ決定、設定、デプロイにわたってガイドを提供します。また、AWS Service Catalog を使用することもできます。これを利用すると、Infrastructure-as-Code の承認を受けたテンプレートを作成および管理して AWS で使用できるため、承認を受けたセルフサービス型クラウドリソースを誰でも利用できます。
- オペレーションのオートメーション: ルーチンオペレーションを、人が介入せずに自動的に実行します。AWS のサービスやツールを使用すると、実装する AWS のオートメーションを選択し、固有の要件に合わせてカスタマイズできます。例えば、EC2 Image Builder を使用して仮想マシンやコンテナイメージを構築、テスト、デプロイし、AWS またはオンプレミスで使用できるようにします。目的のアクションが AWS のサービスで実現できない場合や、リソースのフィルタリングを伴うより複雑なアクションを必要とする場合は、AWS CLI または AWS SDK ツールを使用して、オペレーションを自動化します。AWS CLI は、AWS コンソールを使用せず、スクリプト経由で、AWS のサービスのコントロールおよび管理プロセス全体を自動化する機能を提供します。AWS のサービスとやり取りする希望する AWS SDK を選択します。その他のコード例については、AWS SDK Code examples repository を参照してください。

リソース

関連するドキュメント:

- Modernizing operations in the AWS クラウド(AWS クラウドでのオペレーションのモダナイゼーション)
- AWS Services for Automation (オートメーションのための AWS のサービス)
- AWS Systems Manager Automation
- AWS automations for SAP administration and operations (SAP の管理とオペレーションのための AWS のオートメーション)
- AWS Managed Services
- AWS プロフェッショナルサービス
- Infrastructure and automation (インフラストラクチャとオートメーション)

関連する例:

- Reinventing automated operations (Part I) (オペレーション自動化の再発明 (パート I))
- Reinventing automated operations (Part I) (オペレーション自動化の再発明 (パート II))
- AWS automations for SAP administration and operations (SAP の管理とオペレーションのための AWS のオートメーション)
- AWS Lambda による IT オートメーション
- AWS Code Examples Repository
- AWS Samples (AWS のサンプル)

サステナビリティ

クラウドワークロードを構築するときの持続可能性の柱には、使用しているサービスの影響の理解、ワークロードのライフサイクル全体における影響の数値化、および設計原則とベストプラクティスの適用によるそれらの影響の軽減が含まれます。実装に関する規範的なガイダンスについては、持続可能性の柱のホワイトペーパーを参照してください。

ベストプラクティス領域

- リージョンの選択
- 需要に合わせた調整
- ソフトウェアとアーキテクチャ
- データ
- ハードウェアとサービス

- プロセスと文化

リージョンの選択

質問

- SUS 1 ワークロードにリージョンを選択する方法

SUS 1 ワークロードにリージョンを選択する方法

ワークロードのためのリージョンの選択は、パフォーマンス、コスト、カーボンフットプリントなどの KPI に大きく影響します。これらの KPI を効果的に改善するには、ビジネス要件と持続可能性の目標の両方に基づいて、ワークロードのリージョンを選択する必要があります。

ベストプラクティス

- SUS01-BP01 ビジネス要件と持続可能性の目標の両方に基づいてリージョンを選択する

SUS01-BP01 ビジネス要件と持続可能性の目標の両方に基づいてリージョンを選択する

パフォーマンス、コスト、カーボンフットプリントなどの KPI を最適化するために、ビジネス要件と持続可能性の目標の両方に基づいて、ワークロードのリージョンを選択します。

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。
- 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。

このベストプラクティスを確立するメリット: Amazon の再生可能エネルギープロジェクトや公開されている炭素強度の低いリージョンの近くにワークロードを配置することで、クラウドワークロードのカーボンフットプリントを削減できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

AWS クラウド は、リージョンと Point of Presence (PoP) のネットワークを常に拡大し、それらをグローバルなネットワークインフラストラクチャでつないでいます。ワークロードのためのリージョンの選択は、パフォーマンス、コスト、カーボンフットプリントなどの KPI に大きく影響します。

これらの KPI を効果的に改善するには、ビジネス要件と持続可能性の目標の両方に基づいて、ワークロードのリージョンを選択する必要があります。

実装手順

- 以下の手順に従って、コンプライアンス、利用可能な機能、コスト、レイテンシーなどのビジネス要件に基づき、ワークロードに適したリージョンを評価し、候補をリストアップします。
  - 必要なリージョンの規制に基づいて、これらのリージョンがコンプライアンスに準拠していることを確認します。
  - AWS リージョンサービスリストを使用して、ワークロードの実行に必要なサービスと機能をリージョンが備えているかどうかを確認します。
  - AWS 料金見積りツール を使用して、各リージョンのワークロードのコストを計算します。
  - エンドユーザーの拠点と各 AWS リージョン 間のネットワークレイテンシーをテストします。
- 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。
  - 関連する持続可能性ガイドラインを特定し、温室効果ガスプロトコル (市場ベースとロケーションベースの方法) に基づいて年間の炭素排出量を追跡および比較します。
  - 炭素排出量の追跡に使用する方法に基づいてリージョンを選択します。持続可能性のガイドラインに基づいてリージョンを選択する方法の詳細については、持続可能性の目標に基づいてワークロードのリージョンを選択する方法を参照してください。

リソース

関連するドキュメント:

- 炭素排出量の推定の理解
- Amazon Around the Globe (世界中の Amazon)
- Renewable Energy Methodology (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)

関連動画:

- Architecting sustainably and reducing your AWS carbon footprint (持続可能なアーキテクチャ設計と AWS カーボンフットプリントの削減)

需要に合わせた調整

質問

- SUS 2 クラウドリソースを需要に合わせる方法

SUS 2 クラウドリソースを需要に合わせる方法

ユーザーとアプリケーション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。需要に合うように継続的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を使用していることを検証します。サービスレベルをお客様のニーズと整合させます。ユーザーとアプリケーション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。未使用のアセットを削除します。チームメンバーには、ニーズ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 ワークロードインフラストラクチャを動的にスケールする
- SUS02-BP02 SLA を持続可能性の目標に合わせる
- SUS02-BP03 未使用アセットの創出と維持の停止
- SUS02-BP04 ネットワーク要件に基づいてワークロードの地理的配置を最適化する
- 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
- SUS02-BP06 需要曲線を平坦化するためにバッファリングまたはスロットリングを実装する

SUS02-BP01 ワークロードインフラストラクチャを動的にスケールする

クラウドの伸縮性を利用してインフラストラクチャを動的にスケールすることにより、需要に合わせてクラウドリソースを供給し、ワークロード容量の過剰なプロビジョニングを回避します。

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確立するメリット: ワークロードの伸縮性を設定およびテストすることで、クラウドリソースの供給を需要に効率的に一致させ、容量の過剰なプロビジョニングを回避できます。クラウドの伸縮性を利用して、需要の急増時や急増後に容量を自動的にスケールすることにより、ビジネス要件を満たすために必要となる適切な数のリソースのみを運用できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

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。最適な形で需要と供給を一致させることで、ワークロードに対する環境の影響を最小限に抑えることができます。

需要が一定の場合も変動する場合もあり、管理面の負担を回避するには、メトリクスと自動化が必要になります。アプリケーションのスケールは、インスタンスのサイズを変更して垂直方向 (スケールアップ/スケールダウン)、インスタンス数を変更して水平方向 (スケールイン/スケールアウト)、あるいはこれらの組み合わせで調整できます。

リソースの需要と供給は、さまざまなアプローチで一致させることができます。

- ターゲット追跡アプローチ: スケーリングメトリクスを監視し、必要に応じて容量を自動的に増減します。
- 予測スケーリング: 日単位および週単位の傾向を見越してスケールします。
- スケジュールベースのアプローチ: 予測できる負荷の変化に従って、独自のスケーリングスケジュールを設定します。
- サービススケーリング: 設計によってネイティブにスケールされるか、機能として自動スケーリングを提供するサービス (サーバーレスなど) を選択します。

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して余分な容量を排除し効率性を改善します。

実装手順

- 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。AWS では、ユーザー負荷が低い期間には迅速かつ簡単にワークロードをスケールダウンできるように、幅広い自動スケーリングメカニズムを提供しています。自動スケーリングメカニズムの例を次に示します。

Auto scaling mechanism	Where to use
Amazon EC2 Auto Scaling	アプリケーションのユーザー負荷を処理するために、Amazon EC2 インスタンスを適切な数に調整します。
Application Auto Scaling	Amazon EC2 だけでなく、個々の AWS サービス (Lambda 関数や Amazon Elastic Container Service (Amazon ECS) サービスなど) のリソースを自動的にスケールします。
Kubernetes Cluster Autoscaler	AWS の Kubernetes クラスターを自動的にスケールします。

- スケーリングに関する議論は、Amazon EC2 インスタンスや AWS Lambda 関数などのコンピューティングサービスに関連していることが一般的です。需要に合わせて、Amazon DynamoDB の読み取り/書き込みキャパシティユニットや Amazon Kinesis Data Streams シャードなどの非コンピューティングサービスに関する設定も検討してみてください。
- 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。必要に応じて、カスタマイズされたメトリクス (メモリ使用率など) をスケーリングポリシーに使用することもできます。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。
- Auto Scaling グループの手動スケーリングではなく、動的スケーリングを使用します。また、動的スケーリングで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。
- スケールアウトとスケールインの両方のイベントに対処できるようにワークロードをデプロイします。スケールイ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し、ユーザーエクスペリエンスに影響しない (スティッキーセッションが失われない) ことを確認します。アクティビティ履歴を使用すると、Auto Scaling グループのスケーリングアクティビティを検証できます。

- 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。予測スケーリングを使用すると、容量を過剰にプロビジョニングする必要がなくなります。詳細については、Amazon EC2 Auto Scaling を使用した予測スケーリングに関するブログを参照してください。

リソース

関連するドキュメント:

- Amazon EC2 Auto Scaling の使用を開始する
- 機械学習を利用した EC2 の予測スケーリング
- Amazon OpenSearch Service、Amazon Data Firehose、および Kibana を使用してユーザーの行動を分析する
- Amazon CloudWatch とは
- 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
- Karpenter の概要 - オープンソースの高性能 Kubermetes Cluster Autoscaler
- Amazon ECS クラスター Auto Scaling の詳細

関連動画:

- Build a cost-, energy-, and resource-efficient compute environment (コスト面、エネルギー面、リソース面で効率的なコンピューティング環境の構築)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1) (より良く、より速く、より安価なコンピューティング: Amazon EC2 でのコストの最適化 (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

持続可能性の目標に基づいてワークロードのサービスレベルアグリーメント (SLA) をレビューし最適化して、ビジネスニーズ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を最小化します。

一般的なアンチパターン:

- ワークロード SLA がわからない、またはあいまいである。
- SLA を可用性とパフォーマンスのためにのみ定義している。
- すべてのワークロードに同じ設計パターン (マルチ AZ アーキテクチャなど) を使用している。

このベストプラクティスを活用するメリット: SLA を持続可能性の目標に合わせることで、ビジネスニーズを満たしながら最適なリソース使用量を実現できます。

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: 低

実装のガイダンス

SLA は、クラウドワークロードで期待できるサービスのレベルを定義します。応答時間、可用性、データ保持などです。アーキテクチャ、リソース使用量、クラウドワークロードの環境への影響に関わります。定期的に SLA をレビューして、リソースの使用量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。

実装手順

- ビジネス要件を超えるのではなく満たしながら、持続可能性の目標をサポートする SLA を定義または再設計します。
- 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。
  - 持続可能性と信頼性: 可用性が高いワークロードはより多くのリソースを消費する傾向があります。
  - 持続可能性とパフォーマンス: より多くのリソースを使用してパフォーマンスを強化すると、環境への影響が大きくなる場合があります。
  - 持続可能性とセキュリティ: 過剰に保護されたワークロードは環境への影響が大きくなる場合があります。
- 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 (AWS のマイクロサービスなど) 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 (SaaS プロバイダーにとってのサービスレベルアグリーメントの重要性)

関連動画:

- Delivering sustainable, high-performing architectures (持続可能でパフォーマンスが高いアーキテクチャを実現する)
- Build a cost-, energy-, and resource-efficient compute environment (コスト面、エネルギー面、リソース面で効率的なコンピューティング環境の構築)

SUS02-BP03 未使用アセットの創出と維持の停止

ワークロードの未使用アセットを廃止して、需要をサポートするために必要なクラウドリソース数を削減し、無駄を最小限に抑えます。

一般的なアンチパターン:

- アプリケーションを分析して冗長または不要になったアセットを見つけていない。
- 冗長または不要になったアセットを削除していない。

このベストプラクティスを活用するメリット: 不要なアセットを削除すると、リソースが解放され、ワークロード全体の効率が向上します。

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: 低

実装のガイダンス

未使用のアセットは、ストレージ容量やコンピューティングパワーなどのクラウドリソースを消費します。このようなアセットを特定して排除することで、これらのリソースを解放できるため、クラウドアーキテクチャの効率性が向上します。事前コンパイル済みのレポート、データセット、静的イメージなどのアプリケーションアセットと、アセットのアクセスパターンを定期的に分析し、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。このような冗長アセットを削除して、ワークロード内のリソースの無駄を削減します。

実装手順

- モニタリングツールを使用して、不要になった静的アセットを特定します。
- アセットを削除する前に、削除によるアーキテクチャに対する影響を評価します。
- 計画を立て、不要になったアセットは削除します。
- 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。
- 不要なアセットをこれ以上生成および保存しないようにアプリケーションを更新します。
- 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。
- ワークロードを定期的に見直して、未使用のアセットを特定し削除します。

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)
- AWS アカウントで不要になったアクティブなリソースを終了するにはどうすればよいですか。

関連動画:

- How do I check for and then remove active resources that I no longer need on my AWS アカウント? (AWS アカウントで不要になったアクティブなリソースを確認し削除する方法を教えてください)

SUS02-BP04 ネットワーク要件に基づいてワークロードの地理的配置を最適化する

ワークロード向けに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クラウドのロケーシ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。
- 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。

- 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。

このベストプラクティスを活用するメリット: ワークロードをユーザー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: 中

実装のガイダンス

AWS クラウドインフラストラクチャは、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 および AWS ローカルゾーンといったエッジロケーションなどのロケーションオプションを中心に構築されています。これらのロケーションオプションは、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、オンプレミスのデータセンター間の接続を維持する役割を担っています。

ワークロードのネットワークアクセスパターンを分析して、このようなクラウドロケーションオプションの使用方法や、ネットワークトラフィックが経由する距離を減らす方法を特定します。

実装手順

- ワークロードのネットワークアクセスパターンを分析して、ユーザーがアプリケーションをどのように使用しているかを特定します。
  - ネットワーク活動に関するデータを収集するため、 Amazon CloudWatch および AWS CloudTrailのようなツールを使用します。
  - データを分析して、ネットワークアクセスパターンを特定します。
- 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に近いリージョン (または複数のリージョン) を選択します。
  - その他の制約: 以下で説明されているように、コスト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)。
- 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、頻繁に使用するアセットに使用すると、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。

サービス	使用する状況
Amazon CloudFront	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のキャッシュに使用します。
Amazon ElastiCache	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。
DynamoDB Accelerator	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。

サービス	使用する状況
Lambda@Edge	オブジェクトがキャッシュにないときに開始される、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。
Amazon CloudFront 関数	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。
AWS IoT Greengrass	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。

- 接続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

関連動画:

- Demystifying data transfer on AWS
- 次世代 Amazon EC2 インスタンスでのネットワークパフォーマンスのスケーリング

関連サンプル:

- AWS Networking Workshops
- 持続可能性を考慮したアーキテクチャ - ネットワーク間のデータ移動を最小限に抑える

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環境の持続可能性への影響を最小限に抑えます。

一般的なアンチパターン:

- クラウドアプリケーションの全体的な効率性に関して、チームメンバーが使用するデバイスの影響を無視する。
- チームメンバーが使用するリソースを手動で管理および更新している。

このベストプラクティスを活用するメリット: チームメンバーのリソースを最適化すると、クラウド対応アプリケーションの全体的な効率性が向上します。

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: 低

実装のガイダンス

サービスを利用するためにチームメンバーが使用しているリソース、その予想ライフサイクル、および金融および持続可能性に対する影響を理解します。これらのリソースを最適化する戦略を策定します。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、使用率の高いスケーラブルなインフラストラクチャで行います。

実装手順

- 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て、その伸縮性を活用します。
- 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。
  - AWS Systems Manager Fleet Manager は、AWS やオンプレミスで実行されているノードをリモートで管理できる、統合ユーザーインターフェイス (UI) エクスペリエンスです。

リソース

関連するドキュメント:

- Amazon WorkSpaces とは
- Cost Optimizer for Amazon WorkSpaces (Amazon WorkSpaces の Cost Optimizer)
- Amazon AppStream 2.0 のドキュメント
- NICE DCV

関連動画:

- Managing cost for Amazon WorkSpaces on AWS (AWS での Amazon WorkSpaces のコストを管理する)

SUS02-BP06 需要曲線を平坦化するためにバッファリングまたはスロットリングを実装する

バッファリングやスロットリングは、需要曲線を平坦化し、ワークロードに必要なプロビジョンドキャパシティを削減します。

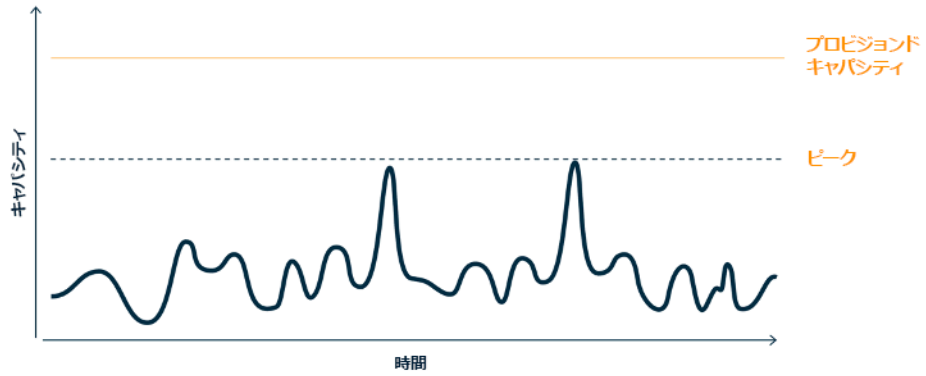
一般的なアンチパターン:

- 即時対応が不要なクライアントのリクエストを即時処理している。
- クライアントのリクエストの要件を分析していない。

このベストプラクティスを活用するメリット: 需要曲線を平坦化することで、ワークロードに必要なプロビジョンドキャパシティを削減できます。プロビジョンドキャパシティが削減されると、エネルギーの消費量が少なくなり、環境への影響が小さくなります。

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: 低

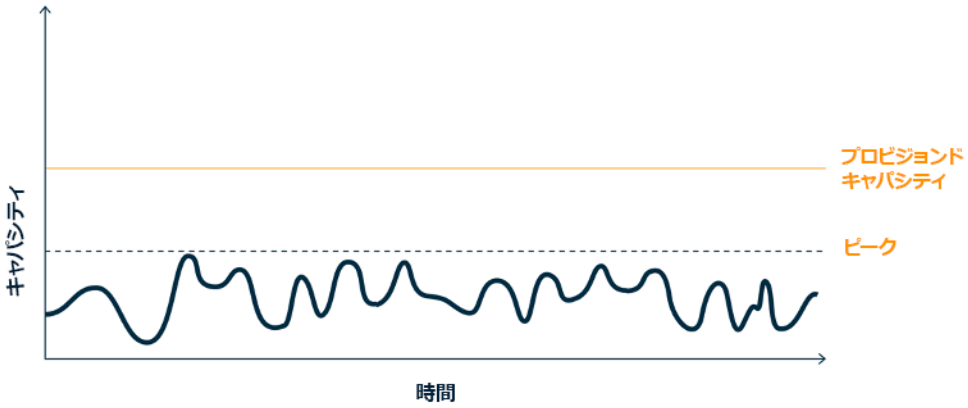
ワークロードの需要曲線を平坦化することで、ワークロードに必要なプロビジョンドキャパシティを削減し、環境への影響を減らすことができます。以下の図に示す需要曲線を持つワークロードがあるとします。このワークロードには 2 つのピークがあり、これらのピークを処理するために、オレンジの線で示されるリソース容量がプロビジョニングされます。このワークロードで使用されるリソースとエネルギーは需要曲線の下の領域ではなく、プロビジョンドキャパシティのラインの下の領域で示されます。これら 2 つのピークを処理するには、プロビジョンドキャパシティが必要であるためです。



大きな容量をプロビジョニングする必要がある 2 つの大きなピークがある需要曲線

バッファリングやスロットリングを使用して需要曲線を変化させ、ピークをならすことができます。つまり、プロビジョンドキャパシティや消費されるエネルギーを減らすことができます。クライアン

トが再試行を実行できるときはスロットリングを実装します。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。



需要曲線とプロビジョンドキャパシティに対するスロットリングの効果

実装手順

- クライアントのリクエストを分析して、それらに応答する方法を決定します。考慮すべき問題は以下のとおりです。
  - このリクエストは非同期で処理できるか?
  - クライアントは再試行できるか?
- クライアントが再試行できる場合、スロットリングを実装できます。これにより、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送信元に通知されます。
  - Amazon API Gateway を使用してスロットリングを実装できます。
- 再試行できないクライアントの場合は、バッファを実装して需要曲線を平坦化する必要があります。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、キューまたはストリーミングを使用して、プロデューサーからメッセージを受信します。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。
  - 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。
- 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適正化します。

リソース

関連するドキュメント:

- Getting started with Amazon SQS (Amazon SQS の開始方法)
- Application integration Using Queues and Messages (キューとメッセージを使用したアプリケーション統合)

関連動画:

- Choosing the Right Messaging Service for Your Distributed App (分散アプリケーションに適したメッセージングサービスを選択する)

ソフトウェアとアーキテクチャ

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
- 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
- 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

- SUS03-BP04 デバイスや機器への影響を最適化する
- 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

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

キュー駆動型などの効率的なソフトウェアおよびアーキテクチャパターンを使用して、デプロイされたリソースの使用率を一貫して高く維持します。

一般的なアンチパターン:

- 予期せぬ需要の急増に対応するために、クラウドワークロードのリソースを過剰にプロビジョニングしています。
- お使いのアーキテクチャでは、メッセージングコンポーネントによって非同期メッセージの送信者と受信者が切り離されていません。

このベストプラクティスを活用するメリット:

- 効率的なソフトウェアとアーキテクチャのパターンは、ワークロード内の未使用リソースを最小限に抑え、全体的な効率を向上させます。
- 非同期メッセージの受信とは無関係に処理を拡張できます。
- メッセージングコンポーネントを使用することで、可用性要件が緩和され、より少ないリソースで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

イベント駆動型アーキテクチャなどの効率的なアーキテクチャパターンを使用することで、コンポーネントを均等に使用し、ワークロードの過剰プロビジョニングを最小限に抑えることができます。効率的なアーキテクチャパターンを使用することで、時間の経過に伴う需要の変化により、使用されずにアイドル状態になるリソースを最小限に抑えることができます。

ワークロードコンポーネントの要件を理解し、リソース全体の利用率を高めるアーキテクチャパターンを採用します。不要になったコンポーネントは廃止します。

実装手順

- ワークロードの需要を分析し、それらに対応する方法を決定します。
- 同期応答を必要としないリクエストやジョブには、キュー駆動型アーキテクチャとオートスケーリングワーカーを使用して使用率を最大化します。キュー駆動型アーキテクチャを検討する場合の例を次に示します。

Queuing mechanism	Description
AWS Batch ジョブキュー	AWS Batch ジョブはジョブキューに送信され、コンピューティング環境で実行されるようにスケジューリングされるまで、そこに留まります。
Amazon Simple Queue Service および Amazon EC2 スポットインスタンス	Amazon SQS とスポットインスタンスのペアリングにより、耐障害性が高く効率的なアーキテクチャを構築します。

- いつでも処理できるリクエストやジョブについては、スケジューリングメカニズムを利用してジョブをバッチ処理することで効率化を図ります。AWS でのスケジューリングメカニズムの例を次に示します。

Scheduling mechanism	Description
Amazon EventBridge Scheduler	スケジュールされたタスクを大規模に作成、実行、管理できるようにするための Amazon EventBridge の機能です。
AWS Glue の時間ベースのスケジュール	AWS Glue で、クローラーやジョブに対して時間ベースのスケジュールを定義します。
Amazon Elastic Container Service (Amazon ECS) のスケジュールされたタスク	Amazon ECS では、スケジュールされたタスクの作成をサポートします。スケジュールされたタスクは、Amazon EventBridge ルールを使用して、スケジュールに従って、または EventBridge イベントへの応答としてタスクを実行します。

Scheduling mechanism	Description
Instance Scheduler	Amazon EC2 および Amazon Relational Database Service インスタンスの開始および停止スケジュールを設定します。

- ご使用のアーキテクチャでポーリングやウェブフックのメカニズムを使用している場合、それらをイベントに置き換えます。イベント駆動型アーキテクチャを使用して、効率性の高いワークロードを構築します。
- AWS のサーバーレスを活用し、過剰にプロビジョニングされたインフラストラクチャを排除します。
- アーキテクチャの個別のコンポーネントの適切なサイズを設定し、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。

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:

- Moving to event-driven architectures (イベント駆動型アーキテクチャへの移行)

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

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、ワークロードの無駄を最小化します。

一般的なアンチパターン:

- ワークロードの個別のコンポーネントの使用率レベルを定期的に確認していない。
- AWS Compute Optimizer 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分析しない。

このベストプラクティスを活用するメリット: 未使用のコンポーネントを削除すると、無駄が最小化され、クラウドワークロード全体の効率が向上します。

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: ミディアム

実装のガイダンス

ワークロードを見直して、アイドルや未使用のコンポーネントを特定します。これは、需要の変化や新しいクラウドサービスのリリースに伴う、反復的な改善プロセスです。例えば、AWS Lambda 関数の実行時間が大幅に低下した場合は、メモリーサイズを小さくする指標になり得ます。また、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やアーキテクチャが変化する可能性があります。

ワークロードのアクティビティを継続的にモニターして、個別のコンポーネントの使用率レベルを改善する機会を見逃さないようにします。アイドルのコンポーネントを削除しアクティビティのサイズ最適化を行って、最小限のクラウドリソースでビジネス要件を満たすようにします。

実装手順

- ワークロードの重要なコンポーネントの使用率メトリクス (Amazon CloudWatch メトリクスの CPU 使用率、メモリ使用率、ネットワークスループットなど) をモニターし、キャプチャします。
- 安定したワークロードの場合は、AWS Compute Optimizer などの AWS サイズ最適化ツールを定期的に確認して、アイドル、未使用、または使用率の低いコンポーネントを特定します。
- 一次的なワークロードについては、使用率メトリクスを評価して、アイドル、未使用、または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や関連アセット (Amazon ECR イメージなど) を廃止します。
- 使用率の低いコンポーネントをリファクタリングまたは他のリソースと統合して、使用効率を改善します。例えば、使用率の低い個別のインスタンスでデータベースを実行するのではなく、単体の Amazon RDS データベースインスタンスに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。
- 作業の単位を完了するためにワークロードによってプロビジョニングされたリソースを理解します。

リソース

関連するドキュメント:

- AWS Trusted Advisor
- Amazon CloudWatch とは
- 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)

関連する例:

- Well-Architected Lab - Rightsizing with AWS Compute Optimizer(Well-Architected ラボ - Compute Optimizer を使用したサイズ最適化)
- Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs (Well-Architected 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)

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

アーキテクチャの異なる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。

一般的なアンチパターン:

- リソースの使用量のためにコードを最適化しない。
- 通常、パフォーマンスの問題にはリソースを増やすことで対処している。
- コードの見直しおよび開発プロセスで、パフォーマンスの変化を追跡していない。

このベストプラクティスを活用するメリット: 効率的なコードは、リソースの使用量を最小化し、パフォーマンスを向上させます。

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: 中

実装のガイダンス

クラウドに構築されたアプリケーションのコードを含むあらゆる機能領域を精査して、そのリソース使用量とパフォーマンスを最適化することが重要です。ビルド環境および本稼働環境でワークロードのパフォーマンスを継続的にモニターし、リソースの使用量が特に高いコードスニペットを改善する機会を特定します。定期的な見直しプロセスを導入して、コードの中でリソースを効率的に使用して

いないバグまたはアンチパターンを特定します。自分のユースケースに合わせて、同じ結果になるシンプルで効率的なアルゴリズムを活用します。

実装手順

- ワークロードを開発する際に、自動化されたコードレビュープロセスを導入して、品質を向上させ、バグやアンチパターンを特定します。
  - Automate code reviews with Amazon CodeGuru Reviewer (Amazon CodeGuru Reviewer を使用したコードレビューの自動化)
  - Detecting concurrency bugs with Amazon CodeGuru (Amazon CodeGuru を使用した並列バグの検出)
  - Raising code quality for Python applications using Amazon CodeGuru (Amazon CodeGuru を使用して Python アプリケーションのコード品質を向上させる)
- ワークロードを実行しながら、リソースをモニターして、作業単位でリソースを多く必要とするコンポーネントを特定し、コードレビューの対象とします。
- コードレビューでは、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。
  - Reducing your organization's carbon footprint with Amazon CodeGuru Profiler (Amazon CodeGuru Profiler を使用して組織のカーボンフットプリントを削減する)
  - Understanding memory usage in your Java application with Amazon CodeGuru Profiler (Amazon CodeGuru Profiler を使用して Java アプリケーションのメモリ使用量を理解する)
  - Improving customer experience and reducing cost with Amazon CodeGuru Profiler (Amazon CodeGuru Profiler を使用してカスタマーエクスペリエンスを改善しコストを削減する)
- 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。エネルギー効率に優れたプログラム言語 (Rust など) の詳細については、 Sustainability with Rust を参照してください。
- コンピューティング負荷が高いアルゴリズムを、結果が同じであり、よりシンプルでより効率的なバージョンに置き換えます。
- ソートや書式設定などの不要なコードを削除します。

リソース

関連するドキュメント:

- 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:

- Improve Code Efficiency Using Amazon CodeGuru Profiler
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru

SUS03-BP04 デバイスや機器への影響を最適化する

アーキテクチャで使用されているデバイスや機器を理解し、それらの使用量を削減する戦略を使用します。これにより、環境に対するクラウドワークロードの全体的な影響を最小化できます。

一般的なアンチパターン:

- 顧客が使用するデバイスの環境に対する影響を無視する。
- 顧客が使用するリソースを手動で管理および更新している。

このベストプラクティスを活用するメリット: 顧客のデバイスに合わせて最適化されたソフトウェアパターンや機能を実装することで、クラウドワークロード全体の環境に対する影響を削減できます。

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: ミディアム

実装のガイダンス

顧客のデバイスに合わせて最適化されたソフトウェアパターンや機能を実装することで、複数の方法で環境に対する影響を削減できます。

- 後方互換性がある新機能を実装することで、ハードウェアの置換数を削減できます。
- アプリケーションを最適化してデバイスで効率的に実行できるようにすることで、エネルギー消費を削減し、バッテリー寿命を延ばすことができます (バッテリー駆動の場合)。
- また、アプリケーションをデバイスに合わせて最適化すると、ネットワーク経由のデータ転送も削減できます。

アーキテクチャで使用されているデバイスや機器、それらの予想ライフサイクル、およびそれらコンポーネントを置換した場合の影響を理解します。デバイスのエネルギー消費、顧客がデバイスを置換

する必要性、およびデバイスを手動でアップグレードする必要性を最小限にできるソフトウェアパターンや機能を実装します。

実装手順

- アーキテクチャで使用されているデバイスをリストアップします。デバイスには、モバイル、タブレット、IoT デバイス、スマートライト、さらに工場のスマートデバイスも含まれます。
- デバイスで実行されているアプリケーションを最適化します。
  - バックグラウンドでのタスク実行などの戦略を使用して、エネルギーの消費量を削減します。
  - 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。
  - ペイロードやファイルを、デバイスが必要とする最適な形式に変換します。例えば、Amazon Elastic Transcoder や AWS Elemental MediaConvert を使用して、サイズが大きい高品質デジタルメディアファイルを、ユーザーがモバイルデバイス、タブレット、ウェブブラウザ、コネクテッドテレビで再生できる形式に変換します。
  - 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。
- 自動化された無線通信 (OTA) の仕組みを使用して、1 つ以上のデバイスに更新をデプロイします。
  - モバイルアプリケーションの更新には、CI/CD パイプラインを使用できます。
  - AWS IoT Device Management を使用して、コネクテッドデバイスを大規模にリモートで管理できます。
- 新機能や更新をテストするには、ハードウェアの代表的なセットを備えたマネージド型 Device Farm を使用し、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。詳細については、SUS06-BP04 マネージド型 Device Farm を使用してテストする を参照してください。

リソース

関連するドキュメント:

- AWS Device Farm とは何ですか?
- Amazon AppStream 2.0 のドキュメント

- NICE DCV
- FreeRTOS を実行するデバイスでファームウェアを更新する OTA チュートリアル

関連動画:

- Introduction to AWS Device Farm(AWS Device Farm の紹介)

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。データへのアクセスと保存を最適にサポートするソフトウェアパターンとアーキテクチャを使用して、ワークロードのサポートに必要なコンピューティング、ネットワーク、ストレージのリソースを最小化します。

一般的なアンチパターン:

- すべてのワークロードのデータの保存とアクセスのパターンが類似していると考えている。
- 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。
- 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。
- アーキテクチャはデータアクセスの高バーストの可能性をサポートしているが、その結果リソースがほとんどの時間でアイドルのままになる。

このベストプラクティスを活用するメリット: データのアクセスパターンおよびストレージパターンにもとづいてアーキテクチャを選択し最適化すると、開発の複雑性が下がり、全体の使用率が増加します。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

データの特性やアクセスパターンに最も合うソフトウェアやアーキテクチャのパターンを使用します。例えば、お客様独自の分析ユースケースに合わせて最適化された目的別サービスを使用できる、AWS のモダンデータアーキテクチャを使用します。このようなアーキテクチャパターンを使用すると、データ処理が効率的になり、リソースの使用量を削減できます。

実装手順

- データの特性やアクセスパターンを分析して、クラウドリソースに最適な構成を特定します。考慮する主な特徴には次のものがあります。
  - データタイプ: 構造、半構造、非構造
  - データ成長: 制限あり、制限なし
  - データ保存期間: 永続、一時的、一過性
  - アクセスパターン: 読み取りまたは書き取り、更新頻度、急増、または安定
- 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アーキテクチャパターンを使用します。
  - Let's Architect! Modern data architectures (構築してみよう! モダンデータアーキテクチャ)
  - Databases on AWS: The Right Tool for the Right Job (AWS のデータベース: 適材適所で使い分けるツール)
- 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。
- アーキテクチャ内のデータ処理に、目的別の分析サービスを使用します。
- 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、効率的なクエリ実行を確実にします。詳細については、AWS のデータベースを参照してください。
- アーキテクチャで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
- Converting Your Input Record Format in Firehose (Kinesis Data Firehose の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

- Amazon S3 Intelligent-Tiering storage class (Amazon S3 Intelligent-Tiering ストレージクラス)

関連動画:

- Building modern data architectures on AWS(AWS にモダンデータアーキテクチャを構築する)

データ

質問

- SUS 4 データ管理のポリシーとパターンをどのように利用して、持続可能性目標を達成しますか?

SUS 4 データ管理のポリシーとパターンをどのように利用して、持続可能性目標を達成しますか?

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
- 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
- SUS04-BP03 ポリシーを使用してデータセットのライフサイクルを管理する
- SUS04-BP04 伸縮性とオートメーションを使用してブロックストレージまたはファイルシステムを拡張する
- SUS04-BP04 不要なデータや重複するデータを削除する
- SUS04-BP06 共有ファイルシステムまたはストレージを使用して共通データにアクセスする
- SUS04-BP07 ネットワーク間でのデータ移動を最小限に抑える
- 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

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対する重要度を理解し、データの保存にエネルギー効率の高い適切なストレージ層を選択します。

一般的なアンチパターン:

- 処理または保存されているデータアセットの中で、類似の特徴 (機密度、ビジネス上の重要度、規制要件など) を持つものを特定していない。
- データアセットのインベントリにデータカタログを実装していない。

このベストプラクティスを活用するメリット: データ分類ポリシーを実装すると、データに対して最もエネルギー効率の高いストレージ層を検討できます。

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

データ分類には、組織が所有または運用する情報システムで処理中または保存中のデータのタイプの特定を含めます。また、データの重要度と、データの侵害、損失、誤使用によって考えられる影響についても検討します。

データ分類ポリシーは、データを使用する流れから逆算して実装し、あるデータセットの組織の運営における重要度のレベルを考慮に入れて、カテゴリ分けのスキームを作成します。

実装手順

- ワークロードに存在するさまざまなデータタイプのインベントリを実施します。
  - データ分類カテゴリの詳細については、Data Classification whitepaper (データ分類ホワイトペーパー) をご覧ください。
- 組織に対するリスクにもとづいて、データの重要度、機密度、整合性、可用性を判断します。このような要件を使用して、導入するデータ分類層のいずれかにデータをグループ分けします。
  - 例として、Four simple steps to classify your data and secure your startup (データを分類しスタートアップを保護する 4 つのシンプルなステップ) を参照してください。
- 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。
  - 例として、AWS Glue のデータカタログとクローラーを参照してください。
- 監査およびガバナンス機能があるデータカタログを作成します。

- データクラスごとに処理手順を決定して文書化します。
- 自動化を使用し、環境を継続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:

- Leveraging AWS クラウド to Support Data Classification (AWS クラウドを活用したデータ分類のサポート)
- AWS Organizations のタグポリシー

関連動画:

- Enabling agility with data governance on AWS(AWS 上でのデータガバナンスで俊敏性を実現する)

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最もよくサポートするストレージ技術を使用し、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。

一般的なアンチパターン:

- すべてのワークロードのデータの保存とアクセスのパターンが類似していると考えている。
- 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。
- 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。

このベストプラクティスを活用するメリット: データのアクセスとストレージのパターンに基づいてストレージ技術を選択し最適化すると、ビジネスニーズを満たすために必要なクラウドリソースが削減し、クラウドワークロードの全体的な効率が向上します。

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: 低

実装のガイダンス

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、パフォーマンス効率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。

- データの特徴とアクセスパターンを評価し、ストレージのニーズにおける主な特徴を収集します。考慮する主な特徴には次のものがあります。
  - データタイプ: 構造、半構造、非構造
  - データの増加: 制限あり、制限なし
  - データの耐久性: 永続、一過性、一時的
  - アクセスパターン: 読み取りまたは書き込み、頻度、急増、または安定
- データの特徴とアクセスパターンをサポートする適切なストレージ技術にデータを移行します。AWS ストレージ技術とその主な特徴を例としていくつか挙げます。

タイプ	テクノロジー	主な特徴
オブジェクトストレージ	Amazon S3	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るオブジェクトストレージサービスです。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。
アーカイブストレージ	Amazon Glacier	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。
共有ファイルシステム	Amazon Elastic File System (Amazon EFS)	さまざまなタイプのコンピューティングソリューショ

タイプ	テクノロジー	主な特徴
		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。
共有ファイルシステム	Amazon FSx	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。

タイプ	テクノロジー	主な特徴
ブロックストレージ	Amazon Elastic Block Store (Amazon EBS)	Amazon Elastic Compute Cloud (Amazon EC2) 向けに設計された、スケーラブルで高パフォーマンスのブロックストレージサービスです。Amazon EBS には、処理が多く IOPS が高いワークロード向けの SSD 搭載ストレージと、スループットが高いワークロード向けの HDD 搭載ストレージが含まれます。
リレーショナルデータベース	Amazon Aurora、Amazon RDS、Amazon Redshift	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 コマースシステムの多くは、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。

タイプ	テクノロジー	主な特徴
key-value データベース	Amazon DynamoDB	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。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。

- Amazon EBS や Amazon FSx など固定サイズのストレージシステムの場合、利用可能なストレージ容量をモニタリングして、しきい値に達した場合のストレージ割り当てを自動化します。Amazon CloudWatch を活用して、 Amazon EBS および Amazon FSx のさまざまなメトリクスを収集および分析できます。
- Amazon S3 ストレージクラスは、オブジェクトレベルで設定でき、単一のバケットのすべてのストレージクラスのオブジェクトを含めることができます。
- また、Amazon S3 ライフサイクルポリシーを使用して、アプリケーションを変更せずにストレージクラス間でオブジェクトを自動的に移動したり、データを削除したりすることができます。一般的に、このようなストレージメカニズムを考える場合、リソース効率、アクセスのレイテンシー、信頼性の間でトレードオフを行う必要があります。

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 EBS I/O の特性
- Amazon S3 のストレージクラスを使用する
- Amazon Glacier とは?

関連動画:

- Architectural Patterns for Data Lakes on AWS
- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)
- Building modern data architectures on AWS

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

SUS04-BP03 ポリシーを使用してデータセットのライフサイクルを管理する

すべてのデータのライフサイクルを管理し、自動的に削除を実行することで、ワークロードに必要なストレージの総量を最小限に抑えます。

一般的なアンチパターン:

- データを手動で削除する。
- ワークロードデータは削除しない。
- データ保持やアクセス要件に基づいて、よりエネルギー効率の高いストレージ階層にデータを移動することがない。

このベストプラクティスを活用するメリット: データライフサイクルポリシーを使用することで、ワークロードのデータアクセスと保持を効率的に行うことができます。

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

データセットには通常、そのライフサイクルにおいて異なる保持要件とアクセス要件があります。例えば、限られた期間のみ頻繁にデータセットにアクセスする必要があるアプリケーションもあります。その後、それらのデータセットにアクセスすることはほとんどありません。

データセットをライフサイクル全体で効率的に管理するには、データセットの処理方法を定義するルールであるライフサイクルポリシーを設定します。

ライフサイクル設定ルールを使用すると、特定のストレージサービスに対して、データセットをよりエネルギー効率の高いストレージ層に移行する、アーカイブする、または削除するように指示できます。

実装手順

- ワークロード内のデータセットを分類します。
- データクラスごとに処理手順を定義します。
- 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。さまざまな AWS ストレージサービスの自動ライフサイクルポリシーを設定する方法の例を次に示します。

Storage service	How to set automated lifecycle policies
Amazon S3	Amazon S3 ライフサイクルを使用して、オブジェクトのライフサイクル全体を管理できます。アクセスパターンが不明、変化している、または予測できない場合、アクセスパターンを監視し、アクセスされていないオブジェクトをより低コストのアクセス層に自動的に移動する Amazon S3 Intelligent-Tiering を使用できます。Amazon S3 ストレージレンズのメトリクスを活用し、ライフサイクル管理における最適化の機会やギャップを特定できます。
Amazon Elastic Block Store	Amazon Data Lifecycle Manager を使用すると、Amazon EBS スナップショットや Amazon EBS でサポートされた AMI の作成、保持、削除を自動化できます。
Amazon Elastic File System	Amazon EFS ライフサイクル管理は、ファイルシステムのファイルストレージを自動的に管理します。

Storage service	How to set automated lifecycle policies
Amazon Elastic Container Registry	Amazon ECR ライフサイクルポリシーは、経過時間やカウントに基づいてイメージを期限切れにし、コンテナイメージのクリーンアップを自動化するものです。
AWS Elemental MediaStore	オブジェクトライフサイクルポリシーを使用して、MediaStore コンテナ内にオブジェクトを保存する期間を管理できます。

- 未使用のボリューム、スナップショット、保存期間を過ぎたデータを削除します。削除には、Amazon DynamoDB の有効期限や Amazon CloudWatch ログ保持などのネイティブサービス機能を活用します。
- 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および圧縮します。

リソース

関連するドキュメント:

- Optimize your Amazon S3 Lifecycle rules with Amazon S3 Storage Class Analysis (Amazon S3 ストレージクラス分析によって Amazon S3 ライフサイクルルールを最適化する)
- AWS Config ルール を使用してリソースを評価する

関連動画:

- Simplify Your Data Lifecycle and Optimize Storage Costs With Amazon S3 Lifecycle (Amazon S3 ライフサイクルによってデータライフサイクルを簡素化し、ストレージコストを最適化する)
- Reduce Your Storage Costs Using Amazon S3 Storage Lens (Amazon S3 ストレージレンズを使用してストレージコストを削減する)

SUS04-BP04 伸縮性とオートメーションを使用してブロックストレージまたはファイルシステムを拡張する

伸縮性とオートメーションを使用して、データの増加につれてブロックストレージまたはファイルシステムを拡張し、プロビジョニングされるストレージの合計を最小化します。

一般的なアンチパターン:

- 将来必要になるかもしれない大きなブロックストレージやファイルシステムを調達している。
- ファイルシステムの IOPS (input and output operations per second、入出力操作毎秒) を過剰プロビジョニングしている。
- データボリュームの使用率をモニターしていない。

このベストプラクティスを活用するメリット: ストレージシステムの過剰プロビジョニングを最小化すると、アイドル状態のリソースが削減され、ワークロード全体の効率が向上します。

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: ミディアム

実装のガイダンス

ワークロードに適したサイズ割り当て、スループット、レイテンシーで、ブロックストレージやファイルシステムを作成します。伸縮性とオートメーションを使用して、データの増加につれてブロックストレージまたはファイルシステムを拡張し、これらのストレージサービスを過剰プロビジョニングしないようにします。

実装手順

- Amazon EBS 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ていることを検証します。
- 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。例えば、Amazon EBS Elastic Volumes を使用して、Amazon EBS ボリュームのボリュームサイズやボリュームタイプを変更したり、パフォーマンスを調整したりできます。
- ファイルシステムに適したストレージクラス、パフォーマンスモード、スループットモードを選択して、ビジネスニーズを超えることなく対処できるようにします。
  - Amazon EFS パフォーマンス
  - Linux インスタンスでの Amazon EBS ボリュームのパフォーマンス
- 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最適化します。
- 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。

- 伸縮自在なボリュームやファイルシステムを定期的に見直して、アイドルなボリュームを停止し、現在のデータサイズに合わせて過剰プロビジョンされたリソースを縮小します。

リソース

関連するドキュメント:

- Amazon FSx ドキュメント
- What is Amazon Elastic File System? (Amazon Elastic File System とは)

関連動画:

- Deep Dive on Amazon EBS Elastic Volumes (Amazon EBS Elastic Volumes の詳細)
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings (パフォーマンスとコスト節約の向上を目指した Amazon EBS とスナップショットの最適化戦略)
- Optimizing Amazon EFS for cost and performance, using best practices (ベストプラクティスを使用した Amazon EFS のコストとパフォーマンスの最適化)

SUS04-BP04 不要なデータや重複するデータを削除する

不要なデータや重複するデータを削除し、データセットの保存に必要なストレージリソースを最小限に抑えます。

一般的なアンチパターン:

- 簡単に取得または再作成できるデータを複製している。
- データの重要性を考慮せず、すべてのデータをバックアップしている。
- データの削除は、不定期、運用イベント時のみ、または全く行わない。
- ストレージサービスの耐久性に関係なく、データを冗長に保存している。
- ビジネス上の正当な理由なく Amazon S3 バージョニングを実行している。

このベストプラクティスを確立するメリット: 不要なデータを削除することで、ワークロードに必要なストレージサイズを縮小し、ワークロードの環境に対する影響も軽減します。

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: ミディアム

実装のガイダンス

不要なデータを保存しない。不要なデータの削除を自動化する。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テクノロジーを使用する。サービスのネイティブデータレプリケーションと冗長性機能を活用する。

実装手順

- AWS Data Exchange およびOpen Data on AWSで公開されている既存のデータセットを利用することで、データの保存を回避できないかを評価します。
- 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。AWS でデータの重複をなくす方法の例を次に示します。

Storage service	Deduplication mechanism
Amazon S3	新しい FindMatches ML Transform を使用して、データセット全体 (識別子のないレコードを含む) で一致するレコードを検索するには、AWS Lake Formation FindMatches を使用します。
Amazon FSx	Windows 向けの Amazon FSx でデータ重複排除を有効にします。
Amazon Elastic Block Store スナップショット	スナップショットは増分バックアップです。つまり、直近のスナップショットの後に変更されたデバイス上のブロックのみが保存されます。

- データアクセスを分析し、不要なデータを特定します。ライフサイクルポリシーを自動化します。削除のための Amazon DynamoDB 有効期限、Amazon S3 ライフサイクル、Amazon CloudWatch ログ保持などのネイティブサービス機能を活用します。
- AWS のデータ仮想化機能を使用してデータをソースに保持し、データの重複を回避します。
  - AWS でのクラウドネイティブデータ仮想化
  - 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
- 増分バックアップが可能なバックアップテクノロジーを使用します。

- セルフマネージドテクノロジー (RAID (Redundant Array of Independent Disks) など) の代わりに、Amazon S3 の耐久性と Amazon EBS のレプリケーションを活用して、耐久性の目標を達成します。
- 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。
- キャッシュの事前入力は、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:

- 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

関連動画:

- Fuzzy Matching and Deduplicating Data with ML Transforms for AWS Lake Formation (AWS Lake Formation の機械学習トランスフォームによるファジーマッチングとデータの重複排除)

関連する例:

- Amazon 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?

SUS04-BP06 共有ファイルシステムまたはストレージを使用して共通データにアクセスする

共有ファイルシステムまたはストレージを導入して、データの重複を避け、ワークロードのインフラストラクチャの効率を向上させます。

一般的なアンチパターン:

- クライアントそれぞれにストレージをプロビジョンしている。
- 非アクティブなクライアントからデータボリュームをデタッチしていない。
- プラットフォームやシステムを横断してストレージに対するアクセスを提供していない。

このベストプラクティスを活用するメリット: 共有のファイルシステムやストレージを使用すると、データをコピーすることなく、1 人以上のコンシューマーがデータを共有できます。これにより、ワークロードに必要なストレージリソースを削減できます。

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: ミディアム

実装のガイダンス

同じデータセットにアクセスするユーザーやアプリケーションが複数の場合、共有ストレージ技術を使用することが、ワークロードの効率的なインフラストラクチャを実現するために重要です。共有ストレージ技術を利用すると、データセットを 1 か所で保存および管理し、データの重複を避けることができます。また、異なるシステム間でデータの一貫性を維持できます。さらに、共有ストレージ技術を利用すると、複数のコンピューティングリソースが並列して同時にデータにアクセスして処理できるため、コンピューティング性能をより効率的に使用できます。

必要なときにのみ、このような共有ストレージサービスからデータを取得し、未使用のボリュームはデタッチしてリソースを解放します。

実装手順

- 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。AWS の共有ストレージ技術の例をいくつか示します。

Storage option	When to use
Amazon EBS Multi-Attach	Amazon EBS Multi-Attach を利用すると、単一のプロビジョンド IOPS SSD (io1 または io2) ボリュームを、同一アベイラビリティー

Storage option	When to use
	ゾーン内の複数のインスタンスに添付できます。
Amazon EFS	When to Choose Amazon EFS (Amazon EFS を選択するケース) を参照してください。
Amazon FSx	Amazon FSx ファイルシステムの選択を参照してください。
Amazon S3	ファイルシステム構造を必要とせず、オブジェクトストレージを使用して動作するように設計されたアプリケーションは、非常にスケーラブルで耐久性が高い低コストのオブジェクトストレージソリューションである Amazon S3 を使用できます。

- 必要なときにのみ、共有ファイルシステムにデータをコピーしたり、共有ファイルシステムからデータを取得したりします。例えば、Amazon S3 を搭載した Amazon FSx for Lustre ファイルシステムを作成して、ジョブの処理に必要なデータのサブセットのみを Amazon FSx にロードできます。
- SUS04-BP03 ポリシーを使用してデータセットのライフサイクルを管理するに概説されているように、使用パターンに応じてデータを削除します。
- 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 S3 バケットにファイルシステムをリンクする
- Using Amazon EFS for AWS Lambda in your serverless applications (サーバーレスアプリケーションの AWS Lambda に Amazon EFS を使用する)
- Amazon EFS Intelligent-Tiering Optimizes Costs for Workloads with Changing Access Patterns (Amazon EFS Intelligent-Tiering はアクセスパターンを変更しワークロードのコストを最適化する)
- オンプレミスデータリポジトリで Amazon FSx を使用する

関連動画:

- Storage cost optimization with Amazon EFS (Amazon EFS を使用したストレージコストの最適化)

SUS04-BP07 ネットワーク間でのデータ移動を最小限に抑える

共通データへのアクセスに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、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。

一般的なアンチパターン:

- データユーザーの所在地とは別の、同じ AWS リージョンにすべてのデータを保存している。
- データをネットワーク経由で移動する前に、データサイズや形式を最適化していない。

このベストプラクティスを活用するメリット: ネットワーク経由のデータの移動を最適化すると、ワークロードに必要なネットワークリソースの総量を削減でき、環境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: 中

実装のガイダンス

組織のあちこちにデータを移動するには、コンピューティング、ネットワーキング、ストレージのリソースが必要です。データ移動を最小限にするテクニックを使用して、ワークロード全体の効率を向上させます。

実装手順

- ワークロードのリージョンを選択する際は、データまたはユーザーの近接性を 意思決定の要素として考慮します。
- リージョン固有のデータが消費されるリージョン内に保存されるよう、リージョン内で消費されるサービスをパーティションします。
- 効率的なファイル形式 (Parquet や ORC など) を使用してデータを圧縮してから、ネットワーク経由で移動します。
- 未使用のデータは移動しないようにします。未使用のデータ移動を防止するために参考となる事例をいくつかご紹介します。
  - API リソースを関連データのみに削減します。

  - データは詳細 (レコードレベルの情報が不要) を集約します。
  - 詳細は、 Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing を参照してください。
  - 検討 AWS Lake Formation のクロスアカウントデータ共有を考慮します。
- 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。

サービス	使用する状況
Lambda@Edge	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。
CloudFront 関数	HTTP(s) 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。
AWS IoT Greengrass	コネクテッド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。

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

関連動画:

- Demystifying data transfer on AWS

関連サンプル:

- 持続可能性を考慮したアーキテクチャ - ネットワーク間のデータ移動を最小限に抑える

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

ビジネス価値のないデータのバックアップを避け、ワークロードに必要なストレージリソースを最小化します。

一般的なアンチパターン:

- データのバックアップ戦略がない。
- 簡単に再作成できるデータをバックアップしている。

このベストプラクティスを活用するメリット: 重要ではないデータのバックアップを避けると、ワークロードに必要なストレージリソースを削減し、環境への影響を減らすことができます。

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: ミディアム

実装のガイダンス

必要ではないデータのバックアップを避けると、コストを下げ、ワークロードが使用するストレージリソースを削減できます。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工フェメラルストレージを除外します。

実装手順

- SUS04-BP01 データ分類ポリシーを実装するで概説されているように、データ分類ポリシーを実装します。
- データ分類の重要度を使用し、目標復旧時間 (RTO) および目標復旧時点 (RPO) に基づいてバックアップ戦略を策定します。重要ではないデータのバックアップを避けます。
  - 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。
- 自動化されたソリューションまたはマネージドサービスを使用してビジネスクリティカルなデータをバックアップします。

- AWS Backup は、フルマネージドサービスで、AWS サービス、クラウド、オンプレミス全体にわたるデータ保護の一元化と自動化を容易にします。AWS Backup を使用した自動バックアップの作成方法に関するハンズオントレーニングについては、Well-Architected Labs - Testing Backup and Restore of Data (Well-Architected ラボ - データのバックアップと復元のテスト) を参照してください。
- Automate backups and optimize backup costs for Amazon EFS using AWS Backup (AWS Backup を使用して Amazon EFS のバックアップを自動化しコストを最適化する)。

リソース

関連する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-BP03 データバックアップを自動的に実行する
- 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 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
- 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
- Backing Up Amazon EFS (Amazon EFS ファイルシステムのバックアップ)
- Amazon FSx for Windows File Server のバックアップ
- Amazon ElastiCache (Redis OSS) のバックアップと復元

関連動画:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS(AWS re:Invent 2021 - 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 (AWS 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, ft.Rackspace (STG341) (AWS re:Invent 2019: AWS Backup の詳細、Rackspace 特集 (STG341))

関連する例:

- Well-Architected Lab - Testing Backup and Restore of Data (Well-Architected ラボ - データのバックアップと復元のテスト)
- Well-Architected Lab - Backup and Restore with Failback for Analytics Workload (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 Lab - Disaster Recovery - Backup and Restore (Well-Architected ラボ - ディザスタリカバリ - バックアップと復元)

ハードウェアとサービス

質問

- SUS 5 アーキテクチャでクラウドのハードウェアとサービスをどのように選択して、持続可能性目標を達成しますか?

SUS 5 アーキテクチャでクラウドのハードウェアとサービスをどのように選択して、持続可能性目標を達成しますか?

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とサービスを選択します。

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
- SUS05-BP02 影響が最も少ないインスタンスタイプを使用する
- SUS05-BP03 マネージドサービスを使用する
- SUS05-BP04 ハードウェアベースのコンピューティングアクセラレーターの使用を最適化する

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

ワークロードには最小限のハードウェアを使用し、ビジネスニーズを効率的に満たします。

一般的なアンチパターン:

- リソースの使用率をモニターしていない。
- アーキテクチャに使用率が低いリソースがある。
- 静的ハードウェアの使用率を見直してサイズを変更するかどうかを判断していない。
- ビジネス KPI に基づいたコンピューティングインフラストラクチャのハードウェア使用率目標を設定していない。

このベストプラクティスを活用するメリット: クラウドリソースのサイズを最適化すると、環境に対するワークロードの影響を削減し、費用を節約して、パフォーマンスベンチマークを維持できます。

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

ワークロードに必要なハードウェアの総数を適切に選択して、全体の効率を改善します。AWS クラウドは、AWS Auto Scaling などのさまざまなメカニズムを通じて、リソース数を動的に拡張または縮小する柔軟性を提供し、需要の変化に対応します。AWS には、最低限の労力でリソースを変更できる API や SDK も用意されています。これらの機能を使用して、ワークロードの実装を頻繁に変更できます。さらに AWS ツールが提供するサイズ最適化のガイドラインを使用して、クラウドリソースを効率的に運用し、ビジネスニーズを満たすことができます。

実装手順

- ニーズに最適なインスタンスタイプを選択します。
  - ワークロードに適切な Amazon EC2 インスタンスタイプを選択する方法を教えてください。
  - Amazon EC2 フリートの属性ベースのインスタンスタイプの選択
  - 属性ベースのインスタンスタイプの選択を使用して Auto Scaling グループを作成します。
- 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。
- 複数のコンピューティング購入オプションを使用して、インスタンスの柔軟性、スケーラビリティ、コスト節約のバランスを取ります。
  - オンデマンドインスタンスは、インスタンスタイプ、ロケーション、処理時間に柔軟性がないワークロードで、新規かつステートフルな、スパイクが発生するワークロードに最適です。
  - スポットインスタンスは、耐障害性が高く、柔軟性があるアプリケーションにおいて、他のオプションを補完する優れた方法です。

    - 自分のニーズ (AZ、リージョン、インスタンスファミリー、インスタンスタイプ) が変化した場合に柔軟に対応できる安定した状態のワークロードには、Compute Savings Plans を活用します。
- さまざまなインスタンスやアベイラビリティーゾーンを使用して、アプリケーションの可用性を最大化し、可能なときは余剰キャパシティを活用します。
- AWS ツール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。
    - AWS Compute Optimizer
    - AWS Trusted Advisor
- 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:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute (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 I : コンピューティング)
- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet (EC2 フリート向け Auto Scaling 用の属性ベースのインスタンスタイプの選択)
- AWS Compute Optimizer ドキュメント
- Operating Lambda: Performance optimization (Lambda の操作: パフォーマンス最適化 – パート 2)
- Auto Scaling のドキュメント

関連動画:

- Build a cost-, energy-, and resource-efficient compute environment (コスト面、エネルギー面、リソース面で効率的なコンピューティング環境の構築)

関連する例:

- Well-Architected Lab - Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled (Level 200) (Well-Architected ラボ - AWS Compute Optimizer およびメモリ使用率の有効化によるサイズ最適化 (レベル 200))

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を継続的にモニターして使用し、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 (例えば、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: 中

実装のガイダンス

クラウドワークロードに効率的なインスタンスを使用することは、リソースの使用量を下げ、コスト効率を高めるために重要です。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ーし、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。

実装手順

- 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
    - Considerations when transitioning workloads to AWS Graviton-based Amazon Elastic Compute Cloud instances
    - AWS Graviton2 for ISVs を参照
  - 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。
  - 機械学習のワークロードには、 AWS Trainium、 AWS Inferentia、および Amazon EC2 DL1 など、ワークロードに特化した専用ハードウェアを利用します。 Inf2 インスタンスなどの AWS Inferentia インスタンスは、同等の Amazon EC2 インスタンスと比較してワットあたりのパフォーマンスが最大 50% 向上します。
  - Amazon SageMaker AI 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。
  - ワークロードの急増 (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。
  - Well-Architected Lab - Rightsizing Recommendations (Well-Architected ラボ - サイズ最適化のレコメンデーション)
  - Well-Architected Lab - Rightsizing with Compute Optimizer (Well-Architected ラボ - Compute Optimizer を使用したサイズ最適化)
  - Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs (Well-Architected 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)

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
- Amazon EC2 DL1 など、ワークロードに特化した専用ハードウェアを利用します
- Amazon EC2 キャパシティ予約フリート
- Amazon EC2 スポットフリート
- 関数: Lambda 関数の設定
- Amazon EC2 フリートの属性ベースのインスタンスタイプの選択
- AWS での持続可能で効率的、コストが最適化されたアプリケーションの構築
- Contino サステナビリティダッシュボードがお客様のカーボンフットプリントの最適化に役立つ仕組み

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
- Build a cost-, energy-, and resource-efficient compute environment (コスト面、エネルギー面、リソース面で効率的なコンピューティング環境の構築)

関連サンプル:

- Solution: Guidance for Optimizing Deep Learning Workloads for Sustainability on AWS (ソリューション: AWS における持続可能性に向けた深層学習ワークロードの最適化ガイダンス)
- Well-Architected Lab - Rightsizing Recommendations (Well-Architected ラボ - サイズ最適化のレコメンデーション)
- Well-Architected Lab - Rightsizing with Compute Optimizer (Well-Architected ラボ - Compute Optimizer を使用したサイズ最適化)
- Well-Architected Lab - Optimize Hardware Patterns and Observe Sustainability KPIs (Well-Architected 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)
- Well-Architected Lab - Migrating Services to Graviton (Well-Architected ラボ - サービスの Graviton への移行)

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を使用して、クラウドでより効率的に運用します。

一般的なアンチパターン:

- アプリケーションの実行に使用率が低い Amazon EC2 インスタンスを使用している。
- 社内チームはワークロードの管理のみを行っており、イノベーションや簡易化に焦点を当てる時間がない。
- マネージドサービスではより効率的に実行できるタスク向けの技術をデプロイして維持している。

このベストプラクティスを活用するメリット:

- マネージドサービスを使用すると、AWS に責任を移行できます。AWS は、数百万のお客様から得られたインサイトで、新規イノベーションと効率性を促進しています。
- マネージドサービスは、マルチテナントコントロールプレーンのおかげで、サービスの環境に対する影響を、多くのお客様に分散します。

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

マネージドサービスは、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。また、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、チームに時間の余裕ができイノベーションに集中できます。

ワークロードを見直して、AWS マネージドサービスに置き換えることができるコンポーネントを特定します。例えば、Amazon RDS、Amazon Redshift、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 Athena、Amazon EMR、Amazon OpenSearch Service はマネージド分析サービスを提供します。

実装手順

1. サービスとコンポーネントのワークロードをリストアップします。
2. コンポーネントを評価して、マネージドサービスに置き換えることができるものを特定します。マネージドサービスの使用を検討する場合の例を次に示します。

Task	What to use on AWS
データベースのホスティング	Amazon Elastic Compute Cloud (Amazon EC2) で独自の Amazon RDS インスタンスを維持するのではなく、マネージド Amazon Relational Database Service (Amazon RDS) インスタンスを使用します。
コンテナワークロードのホスティング	独自のコンテナインフラストラクチャを実装するのではなく、AWS Fargate を使用します。
ウェブアプリケーションのホスティング	フルマネージド CI/CD、および静的ウェブサイトとサーバー側のレンダリング済みウェブアプリケーションのホスティングサービスとして、AWS Amplify ホスティングを使用します。

3. 依存関係を特定して移行計画を作成します。同様にランブックやプレイブックも更新します。
    - AWS Application Discovery Service は、アプリケーションの依存関係と使用状況に関する詳細な情報を自動的に収集して提示するサービスです。これにより、充分な情報に基づいて移行計画の意思決定を行うことができます。
4. サービスをテストしてから、マネージドサービスに移行します。
5. 移行計画を使用して、自己ホスト型サービスをマネージドサービスに置き換えます。
6. 移行完了後は、サービスを継続的にモニターして、必要に応じて調整しサービスを最適化します。

リソース

関連するドキュメント:

- AWS クラウド 製品
- AWS 総保有コスト (TCO) 計算ツール
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

関連動画:

- Cloud operations at scale with AWS Managed Services(AWS マネージドサービスを使用した大規模なクラウド運用)

SUS05-BP04 ハードウェアベースのコンピューティングアクセラレーターの使用を最適化する

高速コンピューティングインスタンスの使用を最適化することで、ワークロードの物理インフラストラクチャの需要を低減します。

一般的なアンチパターン:

- GPU の使用状況を監視していない。
- 専用インスタンスがより高い性能、低コスト、ワットあたりの性能を実現できるのに対し、ワークロードに汎用インスタンスを使用している。
- CPU ベースのコンピューティングアクセラレーターを使用した方が効率的なタスクに、ハードウェアベースのコンピューティングアクセラレーターを使用している。

このベストプラクティスを活用するメリット: ハードウェアベースのアクセラレーターの使用を最適化することで、ワークロードの物理インフラストラクチャの需要を低減できます。

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: 中

実装のガイダンス

高い処理能力が必要な場合、高速コンピューティングインスタンスを使用すると、グラフィック処理ユニット (GPU) やフィールドプログラマブルゲートアレイ (FPGA) などのハードウェアベースのコ

ンピューティングアクセラレーターを利用できるというメリットが得られます。これらのハードウェアアクセラレーターは、グラフィック処理やデータパターンマッチングなどの特定の機能を、CPU ベースの代替手段よりも効率的に実行します。レンダリング、トランスコーディング、機械学習など、多くの高速ワークロードは、リソースの使用量に大きなばらつきがあります。このハードウェアは必要な時間だけ実行し、不要になったら自動で廃止することで、消費されるリソースを最小化します。

実装手順

- どの 高速コンピューティングインスタンスが お客様の要件に対応できるかを特定します。
- 機械学習のワークロードには、 AWS Trainium、 AWS Inferentia、 Amazon EC2 DL1 など、ワークロードに特化した専用ハードウェアを利用します。Inf2 インスタンスなどの AWS Inferentia インスタンスは、 Amazon EC2 インスタンスと比較して、ワットあたりのパフォーマンスが最大 50% 向上します。
- 高速コンピューティングインスタンスの使用状況メトリクスを収集します。例えば、CloudWatch エージェントを使用して、GPU の `utilization_gpu` および `utilization_memory` などのメトリクスを収集できます (「Amazon CloudWatch で NVIDIA GPU メトリクスを収集する」を参照)。
- ハードウェアアクセラレーターのコード、ネットワーク操作、設定を最適化し、基盤となるハードウェアが十分に活用されるようにします。
  - GPU 設定を最適化する
  - Deep Learning AMI での GPU のモニタリングと最適化
  - Amazon SageMaker AI における深層学習トレーニングの GPU パフォーマンスチューニングのための I/O の最適化
- 最新の高性能ライブラリと GPU ドライバーを使用します。
- 使用しないときは、自動化を使用して GPU インスタンスを解放します。

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
- Let's Architect! Architecting with custom chips and accelerators
- ワークロードに適切な Amazon EC2 インスタンスタイプを選択する方法を教えてください。
- Amazon EC2 VT1 インスタンス

- Amazon Elastic Graphics
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

関連動画:

- 深層学習用の Amazon EC2 GPU インスタンスの選択方法
- Amazon EC2 Elastic GPU の詳細
- Deploying Cost-Effective Deep Learning Inference (費用対効果の高い深層学習推論の導入)

プロセスと文化

質問

- SUS 6 組織のプロセスは、持続可能性目標の達成にどのように役立ちますか?

SUS 6 組織のプロセスは、持続可能性目標の達成に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
- SUS06-BP02 ワークロードを最新に保つ
- SUS06-BP03 ビルド環境の利用率を高める
- SUS06-BP04 マネージド型 Device Farm を使用してテストする

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

改善の可能性の検証、テストコストの最小化、小規模な改善の提供を行う手段やプロセスを導入します。

一般的なアンチパターン:

- 持続可能性についてアプリケーションをレビューするのは、プロジェクトの開始時に 1 回だけである。

- リリースプロセスが複雑すぎてリソース効率化のための小規模な変更を導入しづらいため、ワークロードが古くなった。
- 持続可能性のためにワークロードを改善する仕組みがない。

このベストプラクティスを活用するメリット: 持続可能性に関する改善を導入および追跡するプロセスを確立することで、継続的に新しい機能や能力を導入し、問題を排除して、ワークロードの効率を向上させることができます。

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

本稼働環境にデプロイする前に、持続可能性を改善できるか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。

実装手順

- 持続可能性の改善に関する要件を開発バックログに追加します。
- 反復的な改善プロセスを使用して、これらの改善を特定、評価、優先順位付け、テスト、デプロイします。
- 開発プロセスを継続的に改善および合理化します。例えば、継続的な統合および配信 (CI/CD) パイプラインを使用してソフトウェア配信プロセスを自動化して、工数レベルを削減し手動プロセスで発生するエラーを減らす可能性のある改善をテストしデプロイします。
- 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、潜在的な改善を開発およびテストし、テストのコストを削減します。
- 改善の影響を継続的に評価し、必要に応じて調整します。

リソース

関連するドキュメント:

- AWS がサステナビリティソリューションを実現
- Scalable agile development practices based on AWS CodeCommit(AWS CodeCommit に基づいたスケーラブルなアジャイル開発の実践)

関連動画:

- Delivering sustainable, high-performing architectures (持続可能でパフォーマンスが高いアーキテクチャを実現する)

関連する例:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports (Well-Architected 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

ワークロードを最新の状態に保ち、効率的な機能を導入し、問題を排除し、ワークロード全体の効率性を向上させます。

一般的なアンチパターン:

- 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予定がない。

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

最新のオペレーティングシステム、ランタイム、ミドルウェア、ライブラリ、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。定期的に最新の機能やリリースを導入し、ワークロードを最新に保ちます。

実装手順

- 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。
    - 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更新する必要があるかを迅速に把握できます。
- 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。

Workload component	How to update
マシンイメージ	EC2 Image Builder を使用して、Linux または Windows サーバーイメージの Amazon マシンイメージ (AMI) の更新を管理します。
コンテナイメージ	既存のパイプラインに Amazon Elastic Container Registry (Amazon ECR) を使用して、Amazon Elastic Container Service (Amazon ECS) イメージを管理します。
AWS Lambda	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。
    - CI/CD を使用して、AMI、コンテナイメージ、その他クラウドアプリケーションに関連するアーティファクトを自動的に更新できます。
    - AWS Systems Manager Patch Manager 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 Maintenance Windows を使用してアクティビティをスケジュールできます。

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 のデベロッパー用ツール

関連する例:

- Well-Architected Labs: Inventory and Patch Management (Well-Architected ラボ: インベントリおよびパッチ管理)
- Lab: AWS Systems Manager (ラボ: AWS Systems Manager)

SUS06-BP03 ビルド環境の利用率を高める

リソースの使用率を上げて、ワークロードを開発、テスト、構築します。

一般的なアンチパターン:

- ビルド環境を手動でプロビジョニングおよび停止している。
- テスト、ビルド、リリースアクティビティとは無関係にビルド環境を実行し続けている (例えば、開発チームメンバーの就業時間外に環境を実行している)。
- ビルド環境にリソースを過剰プロビジョニングしている。

このベストプラクティスを活用するメリット: ビルド環境の使用率を上げることで、構築者が効率的に開発、テスト、構築できるようにリソースを配分しながら、クラウドワークロード全体の効率を上げることができます。

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

オートメーションと Infrastructure as Code を使用して、必要に応じてビルド環境を起動し、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。テスト環境は、本稼働構成とよく似たものにする必要があります。ただし、バーストキャパシティ、Amazon EC2 スポットインスタンス、自動スケールするデータベースサービス、コンテナ、サーバーレステクノロジーを備えたインスタンスタイプを使用して、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できる機会を探ります。データ量を、テスト要件を満たす量だけに制限します。本稼働データをテストに使用する場合は、データを移動するのではなく、本稼働環境とデータを共有できる可能性を探ります。

実装手順

- Infrastructure-as-Code を使用してビルド環境をプロビジョニングします。
- 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、ビルド用リソースの効率を最大化します。
- 開発環境とテスト環境を最大限に活用する戦略を使用します。
  - 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。
  - 可能な限り、サーバーレス技術を利用します。
  - オンデマンドインスタンスを使用して開発者のデバイスを補完します。
  - 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。
  - 踏み台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。
  - ビルドジョブに合わせてビルドリソースを自動的にスケールします。

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 の概要
- AWS CodeBuild とは
- AWS での Instance Scheduler

関連動画:

- Continuous Integration Best Practices (継続的インテグレーションのベストプラクティス)

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 Device Farm を使用して、ハードウェアの代表的なセットで新機能を効率的にテストします。

一般的なアンチパターン:

- 個別の物理デバイス上で、アプリケーションを手動でテストおよびデプロイしている。
- アプリケーションテストサービスを使用せずに、実際の物理デバイス上でアプリケーションをテストおよび操作している (Android、iOS、ウェブアプリケーションなど)。

このベストプラクティスを活用するメリット: マネージド型 Device Farm を使用してクラウド対応アプリケーションをテストすると、多くのメリットがあります。

- 幅広い種類のデバイスでアプリケーションをテストする、より効率的な機能などです。
- これにより、テスト用の社内インフラストラクチャが必要なくなります。
- あまり使われない古いハードウェアを含む、さまざまデバイスタイプが提供されているため、不要なデバイスをアップグレードする必要がなくなります。

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: 低

実装のガイダンス

マネージド型 Device Farm を使用すると、代表的な一連のハードウェアで新機能をテストするプロセスを合理化できます。マネージド型 Device Farm は、あまり使われない古いハードウェアを含むさまざまなデバイスタイプを提供するため、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。

実装手順

- テストの要件と計画 (テストの種類、オペレーティングシステム、テストのスケジュールなど) を定義します。
  - Amazon CloudWatch RUM を使用して、クライアント側のデータを収集および分析し、テスト計画を策定できます。
- テスト要件をサポートできるマネージド型 Device Farm を選択します。例えば、AWS Device Farm を使用して、代表的な一連のハードウェア上での変更の影響をテストし把握できます。
- 継続的統合/継続的デプロイ (CI/CD) を使用して、テストをスケジュールし実行します。
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests (AWS Device Farm を CI/CD パイプラインに統合してブラウザ間で Selenium のテストを実行する)
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services (AWS DevOps およびモバイルサービスを使用して iOS および iPadOS アプリケーションを構築およびテストする)

- テスト結果を継続的に見直し、必要な改善を行います。

リソース

関連するドキュメント:

- AWS Device Farm デバイスリスト
- CloudWatch RUM ダッシュボードの表示

関連する例:

- AWS Device Farm Sample App for Android
- AWS Device Farm Sample App for iOS
- Appium Web tests for AWS Device Farm

関連動画:

- Optimize applications through end user insights with Amazon CloudWatch RUM (Amazon CloudWatch RUM を使用したエンドユーザーインサイトを通してアプリケーションを最適化する)

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。

Copyright © 2021 Amazon Web Services, Inc. or its affiliates.