フレームワーク

AWS Well-Architected フレームワーク

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Well-Architected フレームワーク: フレームワーク

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

要約と序章 1
序章 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 8
オペレーショナルエクセレンス 8
設計原則 9
定義 10
ベストプラクティス 11
リソース 20
セキュリティ 20
設計原則 20
定義 21
ベストプラクティス 22
リソース 31
信頼性 31
設計原則 32
定義 33
ベストプラクティス 33
リソース 38
パフォーマンス効率 39
設計原則 39
定義 40
ベストプラクティス 40
リソース 46
コスト最適化 46
設計原則 47
定義 48
ベストプラクティス 48
リソース 54
持続可能性 55
設計原則 55
定義 56

ベストプラクティス 57
リソース 63
レビュープロセス 64
結論 66
寄稿者 67
詳細情報 68
ドキュメントの改訂 69
付録: 質問とベストプラクティス 72
オペレーショナルエクセレンス 72
組織 72
準備 128
運用 197
進化 238
セキュリティ 257
セキュリティ基盤 257
ID およびアクセス管理 282
検出 338
インフラストラクチャの保護 352
データ保護 379
インシデントへの対応 411
アプリケーションのセキュリティ 434
信頼性 453
基礎 453
ワークロードアーキテクチャ 492
変更管理 539
障害管理 579
パフォーマンス効率 678
アーキテクチャの選択 678
コンピューティングとハードウェア 693
データ管理 711
ネットワークとコンテンツ配信 736
プロセスと文化 767
コスト最適化 783
クラウド財務管理を実践する 783
経費支出と使用量の認識 807
費用対効果の高いリソース 849

需要を管理しリソースを供給する 891
継続的最適化 903
持続可能性 912
リージョンの選択 912
需要に合わせた調整 914
ソフトウェアとアーキテクチャ 929
[データ] 942
ハードウェアとサービス 962
プロセスと文化 972
注意 980
AWS 用語集 981

AWS Well-Architected フレームワーク

公開日: 2024 年 6 月 27 日 (ドキュメントの改訂)

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。

序章

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、安全で信頼性および有効性が高く、コスト効率に優れた持続可能なワークロードを AWS クラウドで設計し、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。また、このフレームワークは、ベストプラクティスに照らしてアーキテクチャを評価し、改善すべき分野を特定する一貫した方法を提供します。アーキテクチャをレビューするプロセスは、アーキテクチャの決定に関する建設的な話し合いであり、監査メカニズムではありません。当社は、Well-Architected システムによってビジネスの成功の可能性が大いに高まると確信しています。

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する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、最新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修正を提供します。AWS が進化を続けるのにつれて、当社はお客様との共同作業から多くのことを学び、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ます。

このフレームワークは、最高技術責任者 (CTO)、設計者、開発者、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。本ドキュメントは、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」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 ツール) は、AWS Well-Architected フレームワークを使用してアーキテクチャをレビューおよび測定するための一貫したプロセスを提供する、クラウド上のサービスです。AWS WA ツールでは、より信頼性が高く、安全で、効率やコスト効果に優れたワークロード処理を実行するための推奨事項を提供します。

当社は、ベストプラクティスの適用をサポートするために、AWS Well-Architected ラボを作成しました。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。当社は、AWS Well-Architected パートナープログラムのメンバーである、厳選された 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有しており、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、日々、お客様を支援しています。当社は、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。お客様が実際の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パフォーマンスやトレードオフの結果について学んでいます。

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 フレームワークは、オペレーショナルエクセレンス、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、持続可能性という 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
オペレーショナルエクセレンス	開発をサポートし、ワークロードを効率的に実行し、運用に関するインサイトを得て、ビジネス価値をもたらすサポートプロセスと手順を継続的に改善する能力。
セキュリティ	セキュリティの柱では、クラウドテクノロジーを活用して、ユーザーのセキュリティ体制を向

名前	説明
	上させる方法でデータ、システム、アセットを保護する方法について説明します。
信頼性	信頼性の柱には、意図した機能を期待どおりに、正しく、一貫して実行するワークロードの能力が含まれます。これには、ワークロードのライフサイクル全体を通じてワークロードを運用およびテストする能力が含まれます。このホワイトペーパーでは、AWS に信頼性の高いワークロードを実装するための、詳細なベストプラクティスのガイダンスを提供します。
パフォーマンス効率	コンピューティングリソースを効率的に使用してシステム要件を満たし、需要の変化や技術の進歩に合わせてこの効率性を維持する能力。
コスト最適化	最も安価にシステムを実行して、ビジネス価値を実現する能力。
持続可能性	プロビジョニングされたリソースのメリットを最大化し、必要な合計リソースを最小化することにより、エネルギー消費を削減し、ワークロードのすべてのコンポーネントにおいて効率を向上させ、持続可能性への影響を継続的に改善する能力。

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

- コンポーネントとは、要件に合わせて提供されるコード、構成、および AWS リソースです。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。
- ワークロードは、ビジネス価値を実現する一連のコンポーネントを特定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。

- アーキテクチャは、ワークロード内でコンポーネントが連携する方法であると考えられます。多くの場合、コンポーネントが通信や対話をする方法が、アーキテクチャ図の焦点となります。
- 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、ワークロードの複雑性など、追加の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、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は、ビジネスのコンテキストに応じて、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意思決定がエンジニアリングの優先度を左右する可能性があります。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。セキュリティおよび運用上の優秀性は、通常、他の柱に対して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。多くの場合、これらのチームはエンタープライズアーキテクチャ機能の一部として、TOGAF または Zachman Framework を使用します。

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みます。決定権限を分散することは、複数のチームを内部標準に準拠させるという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。まず、各チームがその能力を持てるようにすることに重点を置いたプラクティス (物事のやり方、プロセス、基準、受け入れた規範) を用意

し、チームが満たすべき基準の水準を引き上げているかどうかを検証する専門家を配置します。次に、基準が満たされていることを確認するための、自動チェックを実行するメカニズムを実装します。

> 「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 のリーダーシップ原則によってサポートされており、お客様を起点にして逆算した、すべての役割にわたる文化を確立します。逆算は、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として、取り組みを定義し、推進します。お客様のことを真剣に考えているチームが、お客様のニーズに応じて製品を開発します。

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想定しています。新しいチームがこれらの能力を獲得するために、または既存のチームがそのレベルを上げるために、チームの設計をレビューし、チームが 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有効にします。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。例えば、これを実現する 1 つの方法として、ベストプラクティスを実例に適用することに焦点を当てた、ランチタイムトークが挙げられます。彼らの会話を録音して、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。プリンシパルエンジニアは、新しいベストプラクティスが形成されるにつれて、コミュニティとして協力しながら、チームにそれを徹底させます。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お客様向けに実装したものであり、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を体系化しています。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り、お客様のニーズに基づいて 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 のレビューを実施することで、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、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間で共有することができます。

一般的な設計原則

Well-Architected フレームワークは、クラウドでの適切な設計を可能にするための一般的な設計原則を提供します。

- 容量ニーズの推測が不要: ワークロードのデプロイ時に容量の決定を誤ると、費用のかかるアイドル状態のリソースが発生したり、容量の制約によるパフォーマンスへの影響に対処する必要が生じたりす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。必要な分のみ容量を使用し、自動的にスケールインまたはスケールアウトできます。
- 本稼働スケールでシステムをテストする: クラウドでは、オンデマンドで本稼働規模のテスト環境を作成し、テストが完了したらリソースを削除することができます。テスト環境の支払いは実行時にのみ発生するため、オンプレミスでテストを実行する場合と比べて、わずかなコストで本番環境をシミュレートできます。
- アーキテクチャの実験を念頭に置いた自動化: 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ことが可能になり、手作業による負担を回避できます。自動化に対する変更を追跡し、影響を監査して、必要に応じて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を検討する: 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ジョンがいくつか発生していました。ビジネスとその状況が進化し続けるにしたがって、当初の決定では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ため、設計変更によって生じる影響のリスクを軽減できます。これにより、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、システムを経時的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する: 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行うことができます。クラウドのインフラストラクチャはコードのため、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立ちます。

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

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

オペレーショナルエクセレンス

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については、運用上の優秀性の柱についての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

以下は、クラウド内での運用上の優秀性を実現するための設計原則です。

- ビジネス成果を中心にチームを編成する: チームがビジネス成果を達成する能力は、リーダーシップのビジョン、効果的な運用、ビジネスに沿った運用モデルから得られます。リーダーシップは、チームが最も効率的な方法で業務を行い、ビジネス成果を達成するようチームにインセンティブを与える適切なクラウド運用モデルを用いて、CloudOps の変革に全力で取り組む必要があります。適切な運用モデルでは、人材、プロセス、テクノロジーの能力を活用してスケールと生産性の最適化を実現し、俊敏性、即応性、適応性を通して差別化を図ります。組織の長期的なビジョンは、エンタープライズ全体にわたってステークホルダーおよびクラウドサービスや消費者に伝える目標に変換されます。目標と運用上の KPI はすべてのレベルで一致します。このプラクティスは、以下の設計原則の実装から得られる長期的な価値を維持します。
- オブザーバビリティを実装して実用的なインサイトを得る: ワークロードの動作、パフォーマンス、信頼性、コスト、健全性などを包括的に理解します。主要業績評価指標 (KPI) を設定し、オブザーバビリティのテレメトリを活用して、ビジネス成果の達成が脅かされている場合に情報に基づいた意思決定を行い、迅速に対処します。実用的なオブザーバビリティデータに基づいて、パフォーマンス、信頼性、コストを積極的に改善します。
- 可能な場合は安全に自動化する: 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。ワークロード全体とその運用 (アプリケーション、インフラストラクチャ、設定、手順) をコードとして定義し、更新できます。その後、イベントに応じてワークロードの操作を開始することで、ワークロードの操作を自動化できます。クラウドでは、レート制御、エラーしきい値、承認などのガードレールを設定することで、自動化における安全性を実現できます。効果的な自動化により、イベントへの一貫した対応を実現し、人為的ミスを最小限に抑え、オペレーターの労力を軽減できます。
- 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、スケーラブルで疎結合のワークロードを設計します。デプロイの自動化の手法と併せて、小さく段階的に変更していくことで、障害が発生した場合でも影響範囲を小さく抑え、迅速に復旧することができます。そのため、自信を持ってワークロードに有益な変化を加えられるようになり、一方で品質も維持し、市場の変化にも迅速に適応できます。
- オペレーション手順を頻繁に改善する: ワークロードを進化させるときは、オペレーションを適切に進化させます。運用手順を実施するときに、改善の機会を探します。定期的にレビューを実施し、すべての手順が効果的であり、チームに周知されていることを検証します。ギャップが見つかった場合は、手順を適宜更新してください。手順の更新について、すべてのステークホルダーとチームに伝えます。運用のゲーミフィケーションを行ってベストプラクティスを共有し、チームを教育します。

- 障害を予測する: 障害シナリオを進め、ワークロードのリスクプロファイルとビジネス成果への影響を把握することで、運用の成功を最大化します。こうしてシミュレートした障害に対する手順とチームの対応の有効性をテストします。テストで特定された未解決のリスクを管理するために、情報に基づいた意思決定を行います。
- 運用上のイベントとメトリクスから学ぶ: すべてのオペレーションのイベントや障害から学んだ教訓を通して、改善を推進します。チーム間と組織全体で教訓を共有します。教訓は、運用がビジネス成果にどのように貢献するかについてのデータや工ピソードに焦点を当てたものである必要があります。
- マネージドサービスを使用する: 可能な限り AWS のマネージドサービスを利用して、運用上の負担を軽減します。それらのサービスの操作に関する運用手順を作成します。

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、提供を達成するサービスを実装することで、反復的なプロセスが自動化され、本番環境への有益な変更プロセスの流れが増加します。

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行います。チームはワークロードをサポートできる必要があります。望ましい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の把握、インシデントへの対応ができるようになります。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

> **Note**
>
> 運用上の優秀性の質問にはすべて、柱の省略形であることを示すために OPS のプレフィックスが付いています。

トピック

- 組織
- 準備
- 運用
- 進化

組織

チームは、ビジネスの成功を達成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していることを検証できます。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ていることを検証します。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、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検証します。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 (例えば、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。例えば、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かもしれません。または、リファクタリングせずにシステムの移行を簡素化するため、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る場合もあります。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、リスクが残ることを容認できなくなる場合もあります。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の成功におけるそれぞれの役割、自分たちのチームの成功における他のチームの役割を理解して、目標を共有することが必要です。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検証します。

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。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、および推進者であるべきです。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チームメンバーが措置を講じることができるようにするとともに、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検証します。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やコンプライアンスの要件がある場合は、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、優先順位に与える影響を判別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、実装前のアプローチ、本番稼働前のワークロードの状態、本番稼働時のワークロードの状態を確認するのに役立つ AWS Well-Architected Tool を提供します。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、サステナビリティ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支援を行います。チームを教育するには、AWS Support (AWS ナレッジセンター、AWS ディスカッションフォーラム、AWS Supportセンター) および AWS ドキュメントが提供するリソースを使用します。AWS の質問については、AWS Supportセンターから AWS Supportを参照してください。AWS は、Amazon Builders' Library の AWS の運用を通じて学んだベストプラクティスとパターンも提供しています。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による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。

運用モデルを管理するために役立つ 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することができます。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?

各人が、ビジネスを成功させるうえでの自分の役割を理解しなければなりません。リソースの優先順位を設定するために共通の目標を設定します。そうすることで、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。

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の成功におけるそれぞれの役割、自分たちのチームの成功における他のチームの役割を理解して、目標を共有することが必要です。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すべき場合に遭遇する可能性があります。必要な機能の開発とリスクの管理を検証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すことで、ビジネスの成功を達成します。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設計が可能になります。

ワークロードを設計する際には、オブザーバビリティ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。オブザーバビリティは単なるモニタリングにとどまらず、外部からの情報に基づいてシステム内部の仕組みを包括的に明らかにします。メトリクス、ログ、トレースを柱とするオブザーバビリティは、システムの動作とダイナミクスに関する深いインサイトを提供します。効果的なオブザーバビリティによって、チームはパターン、異常、傾向を見極め、潜在的な問題に積極的に対処し、最適なシステムの状態を維持することができます。主要業績評価指標 (KPI) を特定することは、モニタリングアクティビティと事業目標の連携を確保するうえで非常に重要です。このような連携により、チームは真に重要なメトリクスを使用してデータ主導の意思決定を行い、システムパフォーマンスとビジネス成果の両方を最適化できます。さらに、オブザーバビリティにより、企業は事後的ではなく積極的に対処できるようになります。チームはシステム内の因果関係を理解し、問題に対処するのみでなく、問題を予測して防止することができます。ワークロードが進化するにつれて、オブザーバビリティ戦略を再検討して改善し、戦略の関連性と効果を維持することが重要です。

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、バグ修正を実現するアプローチを採用します。これらにより、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抑制するとともに、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な復旧を達成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を心がけて、変更の範囲を限定します。これにより、迅速なトラブルシューティングと修復が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、より頻繁に有意義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。ワークロードや変更を本番稼働する準備が整うタイミングを明らかにするために、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) を使用しま

す。また、これにより、対処計画を策定するべき領域も明らかにできます。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。変更が本稼働環境に入ることのメリットとリスクを理解し、十分な情報に基づく決定を下します。

AWS では、ワークロード全体 (アプリケーション、インフラストラクチャ、ポリシー、ガバナンス、運用) をコードとして表示できます。つまり、アプリケーションコードに使用しているのと同じエンジニアリング規律をスタックのあらゆる要素に適用し、チームや組織間でこれらを共有することで、開発作業のメリットを拡大できます。クラウド上でコードとしてオペレーションを使用するとともに、安全に実験を行う機能を使用して、ワークロードや運用手順を開発し、障害に備えた練習を実施します。AWS CloudFormation を使用することで、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、本番環境の構築ができます。

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: オブザーバビリティをワークロードに実装するにはどうすればよいですか?

ワークロードにオブザーバビリティを実装することで、ワークロードの状態を把握し、ビジネス要件に基づいてデータ主導の意思決定を行うことができます。

OPS 5: 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、バグ修正を実現し、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導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な復旧を達成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。

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、プロセスと手順、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。

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限に引き上げ、エラーの発生を最小限に抑え、自動応答を実現します。「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 を使用して一貫したタグ付け戦略に従ったメタデータを適用して、リソースの識別を達成します。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

オブザーバビリティにより、意義あるデータに集中して取り組み、ワークロードの相互作用と出力を把握できます。重要なインサイトに重点的に取り組み、不要なデータを排除することで、ワークロードのパフォーマンスを把握するうえで明快なアプローチを維持できます。データの収集のみでなく、データを正しく解釈することも不可欠です。明確なベースラインを定義して、適切なアラートのしきい値を設定し、逸脱がないかを積極的にモニタリングします。主要なメトリクスの変化は、特に他のデータと相関している場合、特定の問題領域を指し示すことができます。オブザーバビリティを使用すると、潜在的な課題の予測や対処がしやすくなり、ワークロードを円滑に動作させ、ビジネスニーズを満たせるようになります。

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した場合に実行する関連プロセスがあり、これに所有者が具体的に指定されていることを検証します。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エスカレーションのプロセスを含めます。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ー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することで、ワークロードの問題を特定して、根本原因の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: 組織でワークロードのオブザーバビリティを活用するにはどうすればよいですか?

オブザーバビリティを活用して、ワークロードの最適な状態を確保します。関連するメトリクス、ログ、トレースを活用して、ワークロードのパフォーマンスを包括的に把握し、問題に効率的に対処します。

OPS 9: オペレーションの正常性を把握するにはどうすればよいですか?

運用メトリクスを定義し、キャプチャして、分析することによって運用イベントへの可視性が得られ、適切な措置を講じることができます。

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手順を準備して検証し、それらによるワークロードの中断を最小限に抑えます。

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

学習、共有、継続的な改善によって、運用上の優秀性を維持します。ほぼ継続した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、実際の運用から教訓を学びます。

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

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

以下の質問は、運用の優秀性に関する考慮事項を焦点としています。

OPS 11: 運用を進化させるにはどうすればよいか?

ほぼ継続的で漸進的な改善に時間とリソースを費やすことで、オペレーションの効果と効率を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

- オペレーショナルエクセレンスの柱

動画

- Amazon での DevOps

セキュリティ

セキュリティの柱では、データ、システム、資産を保護し、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。

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。

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

クラウドにはワークロードのセキュリティ強化に役立つ多くの原則があります。

- 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、お客様の AWS リソースのやり取りごとに適切な承認を得て、職務の分離を強制します。アイデンティティ管理を一元化し、長期的な静的認証情報への依存を排除することを目指します。

- トレーサビリティの維持: 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と統合して、自動的に調査してアクションを実行します。
- すべての層に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、詳細な防御アプローチを適用します。すべての層に適用します (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)。
- セキュリティのベストプラクティスの自動化: 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、迅速かつコスト効果に優れた方法で安全にスケールできます。バージョン管理されたテンプレートのコードとして定義および管理されるコントロールの実装を含む、安全なアーキテクチャを作成します。
- 転送中のデータおよび保管中のデータの保護: データを機密レベルに分類し、必要に応じて暗号化、トークン化、アクセス制御などのメカニズムを使用します。
- 人をデータから遠ざける: メカニズムとツールを使用して、データに直接アクセスしたり、手動でデータを処理したりする必要性を軽減または排除します。これにより、機密データを扱う際の誤処理や変更、人的ミスのリスクが軽減されます。
- セキュリティイベントの準備: 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、ツールと自動化により、検出、調査、復旧のスピードを上げます。

定義

クラウドでのセキュリティには 7 つのベストプラクティス領域があります。

- セキュリティ基盤
- ID およびアクセス管理
- 検出
- インフラストラクチャの保護
- データ保護
- インシデントへの対応
- アプリケーションのセキュリティ

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する必要があります。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、クラウドを導入する組織は、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 は当社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らの目標達成に集中することができます。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 およびアクセス管理
- 検出
- インフラストラクチャの保護
- データ保護
- インシデントへの対応
- アプリケーションのセキュリティ

セキュリティ

次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リストについては、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ためには、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。「運用上の優秀性」で定義された要件とプロセスを組織やワークロードのレベルで作成し、あらゆる領域に適用します。

AWS や業界筋、脅威インテリジェンスからのレコメンデーション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、セキュリティ運用を拡張できます。

AWS における推奨アプローチは、機能、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、アカウントごとにさまざまなワークロードを分離することです。

ID およびアクセス管理

ID 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部分です。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。例えば、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実行するアカウント、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することができます。これらの特権管理要素が、認証と認可の中核を形成します。

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。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の場合、IAM により、ロール、インスタンスプロファイル、ID フェデレーション、一時的認証情報を使用したセキュアなアクセスが可能になります。

次の質問は、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?

安全な AWS ワークロードの運用へのアプローチでは、管理が必要な 2 つのタイプの ID があります。管理し、アクセス権を付与する必要がある ID の種類を理解すると、適切な ID が適切な条件下で適切なリソースにアクセスしていることを検証するのに役立ちます。

人の ID: 管理者、デベロッパー、オペレーター、エンドユーザーには、AWS 環境とアプリケーションにアクセスするための ID が必要です。これらは、組織のメンバーやコラボレーションする外部のユーザーであり、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 リソースとやり取りします。

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。これらの ID には、Amazon EC2 インスタンスや AWS Lambda 関数などの、AWS 環境で実行中のマシンが含まれます。また、アクセスを必要とする外部の関係者のマシン ID を管理することもできます。さらに、AWS 環境へのアクセスを必要とする AWS 外部のマシンが含まれる場合もあります。

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。アクセス許可は、誰が何に、どのような条件でアクセスできるかを制御します。

認証情報は、ユーザーまたはシステム間で共有しないでください。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、最小特権で付与する必要があります。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 Management Console の外部で AWS を操作するには、ユーザーはプログラムによるアクセスが必要です。プログラマチックアクセス権を付与する方法は、AWS にアクセスしているユーザーのタイプによって異なります。

ユーザーにプログラマチックアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラマチックアクセス権を必要とするユーザー	目的	方法
ワークフォースアイデンティティ (IAM アイデンティティセンターで管理されているユーザー)	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラマチックリクエストに署名します。	使用するインターフェイス用の手引きに従ってください。 • AWS CLI については、AWS Command Line Interface ユーザーガイドの「AWS IAM Identity Center を使用するための AWS CLI の設定」を参照してください。 • AWS SDK、ツール、および AWS API については、AWS SDK とツールリファレンスガイドの「IAM Identity Center 認証」を参照してください。

プログラマチックアクセス権を必要とするユーザー	目的	方法
IAM	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	「IAM ユーザーガイド」の「AWS リソースでの一時的な認証情報の使用」の指示に従ってください。
IAM	(非推奨) 長期的な認証情報を使用して、AWS CLI、AWS SDK、AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイス用の手順に従ってください。 • AWS CLI については、AWS Command Line Interface ユーザーガイドの「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDK とツールについては、AWS SDK とツールリファレンスガイドの「長期認証情報を使用して認証する」を参照してください。 • AWS API については、IAM ユーザーガイドの「IAM ユーザーのアクセスキーの管理」を参照してください。

AWS では、ID とアクセスの管理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、人によるアクセスの制御、プログラムによるアクセスの制御に関するハンズオンラボをご覧ください。

検出

検出コントロールにより、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をサポートするために、この機能を使用できます。さまざまな種類の検出コントロールがあります。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、情報システムに関連するコントロールの検査である内部監査を使用して、プラクティスがポリシーと要件を満たしていること、定義された条件に基づいて正しい自動アラート通知が設定されていることを検証することもできます。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、ログとイベントを処理し、監査、自動分析、アラームを可能にするモニタリングを実施することで、検出コントロールを実装できます。CloudTrail ログ、AWS API コール、CloudWatch でアラームによるメトリクスのモニタリングを行い、AWS Config で設定履歴を確認できます。Amazon GuardDuty は、悪意ある動作や不正な挙動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の保護を支援するマネージド脅威検出サービスです。サービスレベルのログも使用できます。例えば、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。

次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?

ログとメトリクスからイベントをキャプチャして分析し、可視性を得ることができます。ワークロードを保護するため、セキュリティイベントと潜在的な脅威に対するアクションを実行します。

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。潜在的なセキュリティインシデントを特定するために、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したりすることで、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の保護

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。これらの手段を使用することは、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し、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。

次の質問は、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は、外部や内部のネットワークベースの脅威から保護するための複数の防御層を必要とします。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の防御層を設置することをお勧めします。インフラストラクチャ保護では、そのコンセプトと方法の多くが、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必須です。

AWS ユーザーは、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あるいは強化し、その設定を維持して、変更不能な Amazon マシンイメージ (AMI) を作成できます。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、自動スケーリングまたは手動で起動して、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に分類し、暗号化は不正なアクセスに対してデータを理解不能にすることでデータの開示を防ぎます。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、次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、お客様は自分のデータを完全に管理することができます。
- AWS により、データの暗号化と定期的なキーのローテーションを含むキーの管理が容易になります。これは、AWS によって簡単に自動化することも、ユーザーが保守することもできます。
- 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が設計されています。例えば、Amazon 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は、間違った上書きや削除によるデータの喪失を防ぎます。
- 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。

次の質問は、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: データは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しますか?

複数のコントロールを実装することで保管中のデータを保護し、不正アクセスや誤操作のリスクを軽減します。

SEC 9: 転送中のデータはどのように保護しますか?

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

AWS には、保管中および転送中のデータを暗号化する複数の手段があります。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。例えば、Amazon S3 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管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 (一般的に SSL termination として知られているプロセス) を調整し、Elastic Load Balancing (ELB) によって処理することもできます。

インシデントへの対応

きわめて優れた予防的な検出コントロール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、インシデントの発生時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 / 封鎖するかどうか、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発生す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、次の取り組みにより、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といった、重要なコンテンツを含む詳細なログを記録できます。
- イベントは自動的に処理することができ、AWS API を使用して対応を自動化するツールを起動できます。
- 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。

次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に対するタイムリーで効果的な調査、対応、復旧を準備することが重要です。

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共に、フォレンジック用のデータと状態のキャプチャを自動化します。

アプリケーションのセキュリティ

アプリケーションセキュリティ (AppSec) は、開発するワークロードのセキュリティ特性の設計、構築、テストの各方法の全体的なプロセスを説明するものです。適切なトレーニングを受けた人材を組織に配置したうえで、ビルドのセキュリティ特性を理解してインフラストラクチャをリリースし、自動化を使用してセキュリティ問題を識別する必要があります。

ソフトウェア開発ライフサイクル (SDLC) とリリース後プロセスの一部としてアプリケーションセキュリティテストを導入することで、本稼働環境でのアプリケーションのセキュリティ問題の識別、修正、防止のための構造的なメカニズムを構築することができます。

ワークロードを設計、構築、デプロイ、運用する際に、アプリケーション開発手法にセキュリティコントロールを含める必要があります。そのうえで、継続的な欠陥削減のプロセスと技術的負債の低減を調整します。例えば、脅威モデリングを設計フェーズで使用すると、設計上の欠陥を早期に発見することができ、後になって問題を軽減するのと比較して、欠陥の修正をより簡単に、より安価に行うことができます。

一般的に、SDLC の早期に欠陥を解決することで、コストと複雑性を抑えられます。最も簡単な問題解決の方法は、そもそも問題を抱えないことです。したがって、最初に脅威モデルを作成することで、設計段階から正しい結果に焦点を合わせることができます。AppSec プログラムが成熟するにつれて、自動化を使用してテストの数を増やしたり、ビルダーへのフィードバックの忠実性を高めたり、セキュリティレビューに必要な時間を短縮したりすることができます。これらすべてのアクションは、構築するソフトウェアの品質を改善し、機能を本稼働環境に実装するまでの時間を短縮します。

これらの実装ガイドラインは、組織と文化、パイプラインのセキュリティ、パイプライン内のセキュリティ、依存関係の管理の 4 つの領域に焦点を当てています。各領域は、実装可能な一連の原則と、ワークロードの設計、開発、構築、デプロイ、運用のエンドツーエンドビューを提供します。

AWS には、アプリケーションのセキュリティプログラムに対処する際に使用できる多くのアプローチがあります。これらのアプローチの一部はテクノロジーに依存し、他のアプローチはアプリケーションのセキュリティプログラムにおける人や組織を重視しています。

以下の質問は、アプリケーションの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 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問題が発生する可能性を軽減するのに役立ちます。

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、次のリソースを参照してください。

ドキュメント

- AWS クラウドのセキュリティ
- AWS コンプライアンス
- AWS セキュリティブログ
- AWS セキュリティ成熟度モデル

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 セキュリティの指針
- 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。本資料では、AWS に信頼性の高いワークロードを実装するための、詳細なベストプラクティスの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。

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には 5 つの設計原則があります。

- 障害から自動的に復旧する: ワークロードの主要業績評価指標 (KPI) をモニタリングすることで、しきい値を超えた場合に自動化を開始できます。この場合の KPI は、サービス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必要があります。これにより、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。より高度な自動化を使用すると、障害が発生する前に予測して、修復できます。
- リカバリ手順をテストする: 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、どのようにワークロードに障害が発生するかをテストし、復旧の手順を検証できます。オートメーションを使用してさまざまな障害をシミュレートすることも、以前に障害が発生したシナリオを再現することもできます。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を行い、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総合的なワークロードの可用性を高める: 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、1 つの障害がワークロード全体に及ぼす影響を軽減します。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。
- 容量を推測しない: 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容量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、需要を満たす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も管理できます (「Service Quotas と制約の管理」を参照してください)。

- 自動化で変更を管理する: インフラストラクチャに対する変更は、自動化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。

- 基礎
- 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 Service Quotas 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。分散システムにおける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

- 基礎
- ワークロードアーキテクチャ
- 変更管理
- 障害管理

基礎

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、このようは基本的な要件のほとんどが既に組み込まれているか、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リストについては、付録を参照してください)。

REL 1: Service Quotas 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、Service Quotas (サービスの制限とも言います) があります。これらのクォータは、サービスを不正使用されないようにするために、必要以上のリソースを誤ってプロビジョニングすることを防ぎ、API オペレーションのリクエストレートを制限するためのものです。また、光ファイバーケーブルにビットをプッシュダウンできる速度や、物理ディスク上のストレージの量などのリソースの制限もあります。

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。その中には、複数のクラウド環境 (パブリックアクセスとプライベートの両方) や、場合によっては既存のデータセンターのインフラストラクチャが含まれます。プランには、システム内およびシステム間の接続性、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、ドメイン名の解決といったネットワークの考慮事項が含まれている必要があります。

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。AWSSDK 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、開発者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。開発者は、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」を読んで学ぶこともできます。

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、スケーラビリティ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ます。

REL 4: 障害を防ぐためには、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に依存しています。これらのネットワークでデータ損失や遅延が発生しても、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及ぼさ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障害を防ぎ、平均故障間隔 (MTBF) を改善します。

REL 5: 障害を緩和するため、または障害に耐えるには、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及ぼさ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、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。変化には、需要の急増といったワークロードに強いられる変化や、機能のデプロイやセキュリティパッチの適用などの内部の変化があります。

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。例えば、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?

ログやメトリクスは、ワークロードの状態に関するインサイトを得るための強力なツールです。ログやメトリクスをモニタリングし、しきい値を超えたときや、重要なイベントが発生したときに通知を送信するようにワークロードを設定することができます。モニタリングにより、ワークロードが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らを認識し、それに応じて自動的に復旧できます。

REL 7: 需要の変化に対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では、リソースを自動的に追加または削除することで、任意の時点での需要により合致できる伸縮性を得られます。

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、ワークロード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検証したりするために必要です。これらの変更がコントロールされていない場合、変更による影響を予測したり、変更によって発生する問題に対応することが困難になります。

需要の変化に応じ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すると、KPI が予測された基準から逸脱した際に、アラートが自動的にチームに送られます。環境の変更が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迅速に特定できます。変更管理を制御することで、必要な信頼性を実現するためのルールを執行することができます。

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必要があります。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、その問題を修復する自動化されたアクションが開始されるように設定できます。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 9: データはどのようにバックアップするのですか?

データ、アプリケーション、設定をバックアップすると、目標復旧時間 (RTO) や目標復旧時点 (RPO) の要件を満たすことができます。

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?

障害を分離した境界によって、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限定できます。境界の外部のコンポーネントは、障害の影響を受けません。障害を分離した複数の境界を使用して、ワークロードへの影響を制限することができます。

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、回復力を考慮して設計する必要があります。

REL 12: 信頼性はどのように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、期待する回復性を実現することを検証する唯一の方法は、テストを行うことです。

REL 13: ディザスタリカバリ (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。これらは、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、データの場所と機能を考慮して、目標を達成するための戦略を実装します。ワークロードのディザスタリカバリを提供することのビジネス価値を伝える際、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 ファイルをテストして、論理的エラーと物理的エラーの両方から回復できることを確認します。障害の管理で重要なのは、頻繁にワークロードに自動化されたテストを実行して障害を発生させ、ワークロードがどのように回復するかを観察することです。これを定期的なスケジュールで実行し、ワークロードの大幅な変更後にもこのテストが開始されることを確認します。KPI、目標復旧時間 (RTO)、目標復旧時点 (RPO) を積極的に追跡し、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、顧客にサービスを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、次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、クラウドリソースを効率的に使用してパフォーマンス要件を満たし、需要の変化や技術の進歩に合わせてこ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。

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設計原則が 5 つあります。

- 高度なテクノロジーを誰でも使えるようにする: 複雑なタスクをクラウドベンダーに委託することによって、チームがより円滑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ト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ます。例えば、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が消費できるサービスとなり、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: 世界各地にある複数の AWS リージョンでのワークロードのデプロイメントは、最小限のコストで、お客様により低いレイテンシーとより良いエクスペリエンスを提供することを可能にします。
- 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。例

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 (ウェブサイト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: 仮想および自動化可能なリソースを使用して、異なるタイプのインスタンス、ストレージ、および設定による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 5 つのベストプラクティス領域があります。

- アーキテクチャの選択
- コンピューティングとハードウェア
- データ管理
- ネットワークとコンテンツ配信
- プロセスと文化

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。高レベルの設計からリソースタイプの選択と設定まで、アーキテクチャのすべての側面におけるデータを収集します。

選択した内容を定期的に見直して、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検証します。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把握していることを検証します。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

- アーキテクチャの選択
- コンピューティングとハードウェア

- データ管理
- ネットワークとコンテンツ配信
- プロセスと文化

アーキテクチャの選択

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し、異なる機能を有効化して、パフォーマンスを向上させます。

AWS のリソースはあらゆるタイプと設定で利用できるため、お客様のニーズを満たす、最も近いアプローチを簡単に見つけることができます。また、オンプレミスのインフラストラクチャでは実現が難しいオプションも見つけることができます。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: ワークロードに適切なクラウドリソースとアーキテクチャを選択するにはどうすればよいでしょうか?

多くの場合、ワークロード全体でのより効果的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。

コンピューティングとハードウェア

特定のワークロードに対する最適なコンピューティングの選択は、アプリケーションの設計、利用パターン、および構成設定に応じて異なります。アーキテクチャでは、各種コンポーネントに異なるコンピューティングを使用し、異なる機能を有効化してパフォーマンスを向上させることができます。アーキテクチャに誤ったコンピューティングを選択することは、パフォーマンス効率の低下につながる可能性があります。

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。
- 関数は、適用するコードに基づいて、実行環境が抽象化されます。例えば、AWS Lambda ではインスタンスを実行せずに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リソースを選択し、ワークロードで使用するにはどうすればよいでしょうか?

ワークロードにとってより効率的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に有効にする機能も異なります。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。

データ管理

特定のシステムに最適なデータ管理ソリューションは、データの種類 (ブロック、ファイル、またはオブジェクト)、アクセスパターン (ランダムまたはシーケンシャル)、必要なスループット、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)、および可用性と耐久性に関する制約に応じて異なります。優れた設計のワークロードは、さまざまな機能によってパフォーマンスを向上させることができる専用のデータストアを使用します。

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、データ可用性、セキュリティ、およびパフォーマンスを提供する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、整合性、低レイテンシーのブロックストレージを提供し、直接アタッチストレージ (DAS) またはストレージエリアネットワーク (SAN) に似ています。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な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のに役立ちます。
- 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 (Amazon 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効率的かつコスト効率よく起動して実行できるため、広く使用されていて、オープンソースであり、商用ライセンスで利用できる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: ワークロード内のデータはどのように保存、管理、アクセスすればよいでしょうか?

システムにとってより効率的なストレージソリューションは、アクセス操作 (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。Well–Architected システムは、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ます。

ネットワークとコンテンツ配信

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。

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。これにより、ネットワークのニーズへの適合が容易になります。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 (拡張ネットワーキング、Amazon EC2 ネットワーク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 (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 4: ワークロード内のネットワークリソースはどのように選択して構成すればよいでしょうか?

この質問には、クラウドで効率的なネットワークおよびコンテンツ配信ソリューションを設計、構成、運用するためのガイダンスとベストプラクティスが含まれています。

プロセスと文化

ワークロードを設計する際には、効率的で高性能なクラウドワークロードをより良く実行するために採用できる原則と慣行があります。クラウドワークロードのパフォーマンス効率を高める文化を採用するには、以下の重要な原則と慣行を検討してください。

この文化を築くために、以下の基本方針を考慮してください。

- コードとしてのインフラストラクチャ: AWS CloudFormation テンプレートなどのアプローチを使用してインフラストラクチャをコードとして定義します。テンプレートを使用すると、アプリケーションコードと設定のほか、インフラストラクチャをソースコントロールに配置できます。これは、ソフトウェアの開発に使用するものと同じ慣行をインフラストラクチャに適用することを可能にし、イテレーションを迅速に行えるようになります。
- デプロイメントパイプライン: 継続的インテグレーション/継続的デプロイメント (CI/CD) パイプライン (ソースコードリポジトリ、ビルドシステム、デプロイメント、およびテストのオートメーションなど) を使用してインフラストラクチャをデプロイします。これにより、イテレーションのたびに、反復可能で、一貫性があり、低コストな方法でデプロイできるようになります。
- 明確に定義されたメトリクス: メトリクスをセットアップおよびモニタリングして主要業績評価指標 (KPI) をキャプチャします。メトリクスについては、技術とビジネス両方のメトリクスの使用をお勧めします。ウェブサイトまたはモバイルアプリの主なメトリクスは、最初の 1 バイトを受信するまでの時間、または最初のレンダリングまでの時間のキャプチャです。一般的に適用されるそ

の他のメトリクスには、スレッド数、ガベージコレクション率、および待機状態などがあります。リクエストあたりの総累積コストなどのビジネスメトリクスでは、コストを削減する方法をアラートできます。メトリクスを解釈する方法を十分に検討してください。例えば、平均値ではなく、最大値または 99 パーセンタイル値を選択できます。

- パフォーマンステストのオートメーション: デプロイメントプロセスの一環として、より簡単に実行できるテストが正常に終了した後で、パフォーマンステストを自動的に開始します。自動化により新しい環境が作成され、テストデータなどの初期条件が設定された後に、一連のベンチマークとロードテストが実行されます。これらのテストの結果は、時間の経過に伴うパフォーマンスの変化を追跡できるようにビルドに関連付けてください。テストが長時間実行される場合、パイプラインのこの部分を、他のビルドと同期しないようにできます。または、Amazon EC2 スポットインスタンスを使用して夜間にパフォーマンステストを実行できます。
- ロードの生成: 合成ユーザージャーニーや事前に記録されたユーザージャーニーをレプリケートする一連のテストスクリプトを作成します。これらは結合スクリプトではなく、べき等性スクリプトにしてください。有効な結果を得るために、「プレウォーミング」スクリプトを含める必要がある場合もあります。テストスクリプトは、可能な限り本番での使用状況を再現する必要があります。負荷の生成には、ソフトウェアまたは Software as a Service (SaaS) ソリューションを使用できます。AWS Marketplace ソリューションおよびスポットインスタンスの使用を検討してください。これらを利用することで、コスト効率に優れた方法で負荷を生成できます。
- パフォーマンスの可視性: 主要なメトリクス、特に各ビルドバージョンに対するメトリクスはチームが確認できるようにする必要があります。これにより、時間の経過に伴う大量の肯定的または否定的な傾向の評価を参照できるようになります。また、エラーや例外の数に関するメトリクスも表示して、動作中のシステムをテストしていることを確認するようにしてください。
- 可視化: パフォーマンス問題、ホットスポット、待機状態、または低使用率が発生している箇所を明らかにする可視化手法を使用します。アーキテクチャ図にパフォーマンスメトリクスを重ね合わせます。コールグラフ、またはコードも問題をすばやく特定するために役立ちます。
- 定期的なレビュープロセス: 低パフォーマンスのアーキテクチャは通常、パフォーマンスレビュープロセスが存在しないか、または破綻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、反復的な改善を促すことができます。
- 継続的な最適化: クラウドワークロードのパフォーマンス効率を継続的に最適化する文化を取り入れましょう。

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: ワークロードのパフォーマンス効率を高めるためにどのようなプロセスを使用しますか?

ワークロードを設計する際には、効率的で高性能なクラウドワークロードをより良く実行するために採用できる原則と慣行があります。クラウドワークロードのパフォーマンス効率を高める文化を採用するには、以下の重要な原則と慣行を検討してください。

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)
- AWS re:Invent 2019: Leadership session: Storage state of the union (STG201-L)
- AWS re:Invent 2019: Leadership session: AWS purpose-built databases (DAT209-L)
- AWS re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2)
- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)

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。コスト最適化の柱のホワイトペーパーでは、実装に関する規範的なガイダンスを確認できます。

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。

- クラウド財務管理の実装: クラウドで財務上の成功を達成し、ビジネス価値の実現を加速するには、クラウド財務管理およびコスト最適化に投資します。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します。複雑なコストの予測は必要ありません。例えば、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。この測定値を使って、成果向上とコスト削減から得られる利点を把握します。
- 差別化につながらない手間のかかる作業にコストをかけるのをやめる: サーバーのラッキング、積み上げ、電力供給などのデータセンターの手間のかかる運用作業は 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やビジネスプロジェクトに集中できるようになります。
- コストを分析し帰属関係を明らかにする: クラウドでは、システムの使用状況とコストを正確に確認しやすくなり、IT コストを個々のワークロード所有者に透過的に結び付けることができます。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。市場投入に要する期間の短縮、新機能導入、納期順守といったケースでは、前持ったコスト最適化に投資するよりも、スピードを重視して最適化することが効率的なことがあります。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「リフトアンドシフト」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、およびオプションで Amazon Athena と Amazon QuickSight をコストと使用状況レポート (CUR) とともに使用して、組織全体のコストと使用状況を把握できます。AWSBudgets 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、付録をご覧ください)。

COST 1: クラウド財務管理はどのように実装するのですか?

クラウド財務管理を導入することで、組織はコストと使用量を最適化し、AWS でスケールしながら、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削減することができます。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取り入れることによって、無駄なコストを費やすことなく革新することができます。

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して無駄を減らします。

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

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

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

費用対効果の高いリソース

ワークロードにとって適切なインスタンスとリソースを使用することが、コスト削減の鍵になります。例えば、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よ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。

ニーズにより効果的に適した方法で Amazon EC2 インスタンスやその他のサービスを利用できるように、AWS には柔軟でコスト効率のよいさまざまな料金オプションがあります。オンデマンドインスタンスは、最低限のコミットメントの必要もなく、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 とリザーブドインスタンスは、オンデマンド料金から最大 75% の割引を提供します。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フリート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。例えば、CloudFront を使用するとデータ転送を最小限に抑えられます。コストを削減できる場合もあります。例えば、Amazon 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、基盤となる AWS のサービスです。Amazon RDS や Amazon DynamoDB などのマネージドサービスは、高レベルまたは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、管理または運用のオーバーヘッドを大幅に削減または排除することができ、アプリケーションとビジネス関連の活動に専念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?

費用を最小化するために、リソースロードに最適な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、時間と共に運用コストを大幅に削減できます。

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低減できます。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するか、Amazon SQS を使用してワークロードにキューを実装できます。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、費用をかけたすべてのものが活用されるようにして、使用率が著しく低いインスタンスの発生を回避します。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。

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、コストの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。例えば、Amazon 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および管理する必要がなくなります。

COST 11: 工数のコストを評価する方法

クラウドの運用にかかる労力のコストを評価し、時間のかかるクラウド運用を見直し、それらを自動化して、関連する AWS サービスやサードパーティー製品、カスタムツールを採用することにより人的労力およびコストを削減します。

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

持続可能性

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。

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、ストレージを減らすか、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リングし評価します。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
- 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

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ほぼ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
- 需要に合わせた調整
- ソフトウェアとアーキテクチャ
- データ管理
- 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

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。

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、付録をご覧ください)。

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。

需要に合わせた調整

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。継続的に需要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。

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: 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。継続的に需要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。

ユーザーの負荷に応じてインフラストラクチャをスケールする: 使用率が低い、または使用されていない期間を特定し、リソースをスケールして、余分な容量を削減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性コンテンツ (重複または共通のデータセットと出力が含まれる月次レポートなど) と統合し、出力が重複する際に消費されるリソースを削減します。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

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廃止します。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。顧客がお客様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廃止します。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。顧客がお客様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: お客様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効果的に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。

データ管理

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効果的に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効果的に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。

ハードウェアとサービス

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。

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: 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。

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。

プロセスと文化

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: 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。

持続可能性の改善を迅速に導入できるオペレーションを採用する: 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。低コストのテストオペレーションを開発し、小規模な改善を推進します。

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実現できます。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、スポットインスタンス、伸縮自在なデータベースサービス、コンテナ、その他のテクノロジーを備えたインスタンスタイプ使用して、開発およびテストのキャパシティを使用に合わせて調整します。

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

アーキテクチャのレビューは、徹底的な調査を促進する非難のないアプローチにより、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一連のアクションになります。

「アーキテクチャ」セクションで説明したとおり、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構築するチームメンバーには、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的にレビューすることをお勧めします。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、アーキテクチャのアプローチに影響を与える一連の設計原則と、根本原因分析 (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を前提としています。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、変更が難しい「一方向ドア」を避けるために、設計の早い段階で製品ライフサイクルの主要なマイルストーンで適用し、その後は運用開始日の前に適用する必要があります。(多くの決定は、リバーシブルの双方向ドアです。これらの決定には軽量なプロセスを使用できます。一方向ドアは反転するのが困難または不可能であり、ドアを作成する前にさらに検査する必要があります。) 本番稼働開始後に新しい機能の追加や技術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経時的に変化します。アーキテクチャを進化させ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は、Well-Architected レビューを含めて、一連の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。レビューを実施したことにより、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、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。

会議をすみやかに進行するために、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
- 印刷した図や設計ノート
- 回答するには帯域外の調査が必要な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提示されます。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時間を、ビジネス価値を創出するための時間に充てることができます。課題に対処する際にレビューを更新することで、アーキテクチャの改善具合を確認できます。

レビューの価値は 1 度やってみると明らかになりますが、新しいチームでは最初に抵抗があるかもしれません。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。

- 「忙しすぎる」 (チームが大きな仕事を始める準備の段階でよくある発言)
  - 大きな仕事を始める準備をしているならば、それを円滑に進める必要があります。レビューを実施することで、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。
- 「結果に対して何かをする時間はありません。」 (スーパーボウルなど、動かせないイベントを目標としている場合によくある発言)
  - 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 フレームワークに関する質問をチームに示したら、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、存在するかもしれません。すべてのレビューを総合的に検討し、それらの主題の問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回答を見つける必要があります。

結論

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

このドキュメントには、次の個人および組織が貢献しました。

- Brian Carlson、オペレーションリーダー、優れた設計、Amazon Web Services
- Ben Potter、Well-Architected セキュリティリード、Amazon Web Services
- Seth Eliot、Well-Architected 信頼性リード、Amazon Web Services
- Eric Pullen、シニアソリューションアーキテクト、Amazon Web Services
- Rodney Lester、プリンシパルソリューションズアーキテクト、Amazon Web Services
- Jon Steele、シニアテクニカルアカウントマネージャー、Amazon Web Services
- 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、Amazon Web Services
- Callum Hughes、ソリューションズアーキテクト、Amazon Web Services
- Ben Mergen、シニアコストリードソリューションズアーキテクト、Amazon Web Services
- Chris Kozlowski、Enterprise Support 部門、シニアスペシャリストテクニカルアカウントマネージャー、Amazon Web Services
- Alex Livingstone、Cloud Operations 部門、プリンシパルスペシャリストソリューションズアーキテクト、Amazon Web Services
- Paul Moran、Enterprise Support 部門、プリンシパルテクノロジスト、Amazon Web Services
- Peter Mullen、Professional Services 部門、アドバイザリーコンサルタント、Amazon Web Services
- Chris Pates、Enterprise Support 部門、シニアスペシャリストテクニカルアカウントマネージャー、Amazon Web Services
- Arvind Raghunathan、Enterprise Support 部門、プリンシパルスペシャリストテクニカルアカウントマネージャー、Amazon Web Services
- Sam Mokhtari、シニアエフィシェンシーリードソリューションズアーキテクト、Amazon Web Services

詳細情報

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

セキュリティの柱に関するホワイトペーパー

信頼性の柱に関するホワイトペーパー

パフォーマンス効率の柱のホワイトペーパー

コスト最適化の柱に関するホワイトペーパー

持続可能性の柱に関するホワイトペーパー

Amazon Builders' Library

ドキュメントの改訂

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 フィードにサブスクライブしてください。

変更	説明	日付
ベストプラクティスガイダンスの更新	柱全体で大規模なベストプラクティスの更新が行われました。セキュリティとコストの両方で新しいベストプラクティスが適用されました。	2024 年 6 月 27 日
メジャーな更新	大規模な柱の更新。	2023 年 10 月 3 日
新しいフレームワークの更新	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。セキュリティとコスト最適化の柱に新しい質問が追加されました。	2023 年 4 月 10 日
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。	2022 年 10 月 20 日
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。	2021 年 12 月 2 日
メジャーな更新	持続可能性の柱がフレームワークに追加されました。	2021 年 11 月 20 日
マイナーな更新	非包括的言語を削除しました。	2021 年 4 月 22 日
マイナーな更新	多数のリンクを修正しました。	2021 年 3 月 10 日

マイナーな更新	全体を通した編集のマイナー変更。	2020 年 7 月 15 日
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。	2020 年 7 月 8 日
ホワイトペーパーの更新	AWS Well-Architected Tool、AWS Well-Architected ラボへのリンク、および AWS Well-Architected パートナーの追加と、多言語バージョンのフレームワークを可能にする軽微な修正。	2019 年 7 月 1 日
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。	2018 年 11 月 1 日
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。	2018 年 6 月 1 日
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。	2017 年 11 月 1 日

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。	2016 年 11 月 1 日
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。	2015 年 11 月 1 日
初版発行	AWS Well-Architected フレームワークが公開されました。	2015 年 10 月 1 日

> **Note**
>
> RSS の更新をサブスクライブするには、使用しているブラウザで RSS プラグインを有効にする必要があります。

フレームワークのバージョン:

- 2023-10-03 (現行)
- 2023-04-10
- 2022-03-31

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

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

オペレーショナルエクセレンス

運用上の優秀性の柱には、開発を支援し、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について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

Questions

- 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しますか?
- 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?

誰もが、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に共通の目標を設定します。そうすることで、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。

ベストプラクティス

- OPS01-BP01 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 メリットとリスクを管理しながらトレードオフを評価する

OPS01-BP01 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ステークホルダーと協力して、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、目的のビジネス成果達成に必要なオペレーションサポートについて十分に理解していることを確かめることができます。

期待される成果:

- 顧客の成果を起点に考える。
- 運用体制がビジネス成果と目標をどのようにサポートしているかを理解する。
- すべての関係者を関与させる。
- 顧客のニーズを捉えるメカニズムがある。

一般的なアンチパターン:

- 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。
- 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利点: 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、運用の各チームを含むステークホルダー全体で目標を共有し、理解を深め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

関連するベストプラクティス:

- OPS11-BP03 フィードバックループを実装する

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。

期待される成果:

- 確立された優先順位に基づいて、改善の努力を最も影響があるところに集中させる (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)。
- ニーズの変化に応じて優先順位を更新する。

一般的なアンチパターン:

- ネットワーク管理を容易にするため、製品チームに相談せず、IP アドレスの割り当てを変更することになった。製品チームに与える影響は未知数です。
- 新しい開発ツールを実装しようとしているが、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ない。
- 新しいモニタリングシステムを実装しようとしているが、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ない。

このベストプラクティスを活用するメリット: 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、開発、運用の各チームを含むステークホルダー全体で目標を共有し、理解を深めることでビジネスは成功します。
- 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ステークホルダーと連携し、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

関連するベストプラクティス:

- OPS11-BP03 フィードバックループを実装する

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が、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 のようなサービスを使用して、governance-as-code を作成し、ガバナンス要件が順守され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、組織外から取得されます。

関連ドキュメント:

- AWS 管理とガバナンスのクラウド環境ガイド
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- Governance in the AWS クラウド: The Right Balance Between Agility and Safety
- GRC (ガバナンス、リスク、コンプライアンス) とは何ですか?

関連動画:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1)
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)

関連する例:

- AWS Config 適合パックのサンプル

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

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 などの サービスとの AWS 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 などのサービスを使用して、コンプライアンスの検証と監査レポートを生成します。

b. AWS セキュリティおよびコンプライアンスドキュメントは、AWS Artifact で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 - セキュリティ統制目標は、全体的なコンプライアンスの重要な部分です。
- SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する - パイプラインの一部として、セキュリティコントロールを検証します。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 データの機密性に基づいてデータ保護統制を適用する - 多くのコンプライアンスフレームワークには、データ処理とストレージポリシーがベースになっています。
- 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、監査コンプライアンスに使用できる場合があります。

関連ドキュメント:

- AWS コンプライアンスセンター
- AWS コンプライアンスのリソース
- AWS リスクとコンプライアンスのホワイトペーパー
- 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 2021 - Cloud compliance, assurance, and auditing
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (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

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。

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設計された運用レビューを受けることができます。

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。

期待される成果:

- Well-Architected と Trusted Advisor 出力を定期的に確認し、これに基づいて対応する
- サービスの最新のパッチステータスを把握する
- 既知の脅威のリスクと影響を理解し、適宜対応する
- 必要に応じて緩和策を実施する
- アクションと背景情報を伝える

一般的なアンチパターン:

- 自社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。未対応のコンプライアンス要件への対応に優先順位が付けてられていません。

このベストプラクティスを活用するメリット: 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、対処すべき脅威、その優先度、対処に必要なリソースを判断しやすくなります。

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

- 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

関連ドキュメント:

- 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

関連動画:

- AWS re:Inforce 2023 - A tool to help improve your threat modeling

OPS01-BP06 メリットとリスクを管理しながらトレードオフを評価する

複数の関係者の利害が対立している場合、労力の優先順位付け、機能の構築、ビジネス戦略に沿った結果の実現が難しくなることがあります。例えば、IT インフラストラクチャコストの最適化よりも、新機能の市場投入までの時間短縮を優先させるよう求められる場合があります。これにより、2つの利害関係者の間で対立が発生します。このような場合、対立を解消するには、より上位の権限者に決断を委ねる必要があります。意思決定プロセスから感情的な固執を取り除くには、データが必要です。

戦術レベルでも同様の問題が発生する可能性があります。例えば、リレーショナルデータベースまたは非リレーショナルデータベースのどちらを使用するかという選択が、アプリケーションの運用に大きな影響を及ぼす場合があります。さまざまな選択肢で予想される結果を理解することが重要です。

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支援を行います。チームを教育するには、AWS Support (AWS ナレッジセンター、AWS ディスカッションフォーラム、AWS Supportセンター) および AWS ドキュメントが提供するリソースを使用します。さらに質問がある場合は、AWS Supportまでお問い合わせください。

また、AWS は Amazon Builders' Library の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。

期待される成果: クラウドデリバリー組織内のあらゆるレベルでの重要な意思決定を促進する、意思決定ガバナンスフレームワークが明確に定義されています。このフレームワークには、リスク登録簿、意思決定の権限を持つ定義済みの役割、考えられる意思決定の各レベルに対する定義済みモデルなどの機能が含まれています。このフレームワークでは、対立の解決方法、提示すべきデータ、オプ

ションの優先順位付けの方法が事前に定義されているため、決定が下されたらすぐに決定にコミットできます。意思決定のフレームワークには、すべての意思決定のメリットとリスクを確認して比較検討し、トレードオフを理解するための標準的アプローチが含まれています。これには、規制コンプライアンス要件の順守などの外部要因が含まれる場合があります。

一般的なアンチパターン:

- 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。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。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、会社に対する支援を停止します。
- 開発者の 1 人がインターネットで見つけたライブラリを含めることにしました。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。
- 当初ビジネスが移行を正当化した理由は、アプリケーションワークロードの 60% のモダナイゼーションに基づくものでした。しかし、技術的な問題により、モダナイゼーションは 20% に留めるという決断が下されました。これにより、計画していた長期的メリットは減少し、インフラストラクチャチームがレガシーシステムを手動でサポートするためにオペレーターの労力が増え、この変更を予定していなかったインフラストラクチャチームでの新しいスキルセットの構築に大きく依存することになりました。

このベストプラクティスを活用するメリット: 取締役会レベルでのビジネスの優先順位を十分に調整し、これをサポートできます。成功の達成に伴うリスクを理解し、十分な情報に基づいた意思決定を行うと共に、リスクが成功のチャンスを妨げる場合に適切な措置を取ることができます。意思決定がもたらす影響と結果を理解することで、選択肢に優先順位を付けやすくなり、リーダーはより迅速に合意に達することができるため、ビジネス成果の向上につながります。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、事例に頼った意思決定ではなく、データドリブンな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: 中

実装のガイダンス

メリットとリスクの管理は、主要な意思決定の要件を決定する運営組織が定義すべきです。関連するリスクを理解したうえで、決定が組織にもたらすメリットに基づいて意思決定を行い、優先順位を付けます。組織の意思決定には正確な情報が不可欠です。この情報は、信頼性の高い測定に基づき、費用対効果分析という一般的な業界慣行によって定義されたものである必要があります。このような決

定を下すには、中央集権型と権限分散型のバランスを取ります。必ずトレードオフはあるため、それぞれの選択肢が定義された戦略と期待されるビジネス成果にもたらす影響を理解することが重要です。

実装手順

1. 包括的なクラウドガバナンスフレームワーク内でメリットの測定方法を定式化します。
   a. 中央集権型の意思決定と、権限分散型の意思決定のバランスを取ります。
   b. あらゆる意思決定で負担の大きい意思決定プロセスを実施することが遅延につながることを理解します。
   c. 意思決定プロセスに外部要因 (コンプライアンス要件など) を組み込みます。
2. さまざまなレベルの意思決定について、合意に基づいた意思決定フレームワークを確立します。このフレームワークには、利害の対立がかかわる意思決定を解決すべき人物が含まれます。
   a. 取り消しが効かない可能性のある「ワンウェイドア」(一方通行の扉) の意思決定を一元化します。
   b. 下位レベルの組織リーダーが「ツーウェイドア」(双方向に行き来できる扉) の意思決定を行えるようにします。
3. メリットとリスクを理解し、管理します。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。
   a. 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、ビジネスケースへの影響、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。
   b. 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。
   c. 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。例えば、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。
4. コンプライアンス要件の順守を自動化する主な意思決定をプログラム的に実施します。
5. バリューストリーム分析や LEAN など既知の業界フレームワークと機能を活用して、現状のパフォーマンスやビジネスメトリクスのベースラインを定め、これらのメトリクスの改善に向けた進捗の反復を定義します。

実装計画に必要な工数レベル: 中 ~ 高

リソース

関連するベストプラクティス:

- OPS01-BP05 脅威の状況を評価する

関連ドキュメント:

- Amazon 1 日目の文化の要素 | 高品質で高速な決定を下す
- クラウドガバナンス
- 管理とガバナンスのクラウド環境ガイド
- クラウドの、そしてデジタル時代のガバナンス: パート 1 および 2

関連動画:

- Podcast | Jeff Bezos | On how to make decisions

関連する例:

- データを使用して情報に基づいた意思決定を下す (DevOps サーガ)
- 開発価値ストリームマッピングを使用して DevOps 成果の制約を特定する

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しますか?

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の成功におけるそれぞれの役割、自分たちのチームの成功における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 責任と所有権を管理するためのメカニズムが存在する
- OPS02-BP05 追加、変更、除外をリクエストするメカニズムが存在する
- OPS02-BP06 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を活用し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。同社は 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離れた場合でも、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、代替のアカウント連絡先があれば、必要に応じて AWS が適切な担当者と連絡を取ることができます。アカウントの代替連絡先は、個人ではなくグループを指定して設定してください。
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 のタグ付けの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. 生成 AI を活用する会話型アシスタント、Amazon Q Business を使用して、従業員の生産性を高め、質問に回答し、エンタープライズシステムの情報に基づいてタスクを完了します。
    a. Amazon Q Business を会社のデータソースに接続します。Amazon Q Business には、Amazon Simple Storage Service (Amazon S3)、Microsoft SharePoint、Salesforce、Atlassian Confluence など、40 を超えるサポート対象のデータソースへの事前構築済みのコネクタが用意されています。詳細については、「Amazon Q Business のコネクタ」を参照してください。

5. 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
- OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する

関連ドキュメント:

- AWS アカウント管理 - 連絡先情報の更新
- AWS Organizations - 組織の代替連絡先の更新
- AWS のタグ付けのベストプラクティス (ホワイトペーパー)
- Amazon Q Business と AWS IAM Identity Center を利用して、プライベートでセキュアなエンタープライズ生成 AI アプリケーションを開発する
- 生成 AI を使用して従業員の生産性向上を支援する Amazon Q Business の一般提供開始
- AWS クラウド運用および移行ブログ - AWS Config と AWS Organizations で自動かつ一元的なタグ付けコントロールを実装する
- AWS セキュリティブログ - AWS CloudFormation Guard で pre-commit フックを拡張する
- AWS DevOps ブログ - AWS CloudFormation Guard を CI/CD パイプラインに統合する

関連するワークショップ:

- AWS Workshop - タグ付け

関連する例:

- AWS Config ルール - Amazon EC2 with required tags and valid values

関連サービス:

- AWS Config ルール - required-tags
- 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、改善の機会を見極めることができます。

期待される成果: 組織において、運用タスクのための一連のプロセスと手順が明確に定義され、維持されています。プロセスと手順は一元的に保管され、チームメンバーが利用できます。プロセスと手順は、所有権が明確に割り当てられ、頻繁に更新されます。可能な場合は、スクリプト、テンプレート、オートメーションドキュメントがコードとして実装されます。

一般的なアンチパターン:

- プロセスが文書化されていない。断片化されたスクリプトが、隔離されたオペレータワークステーションに分散する場合がある。
- スクリプトの使用方法に関する知識が一部の個人によって保持されているか、チームの知識として非公式に共有されている。
- レガシープロセスの更新が必要であるのに、更新の所有権が不明であり、当初の作成者が既に組織を離れている。
- プロセスとスクリプトが検出可能になっていないため、必要なとき (インシデントへの対応時など) にすぐに利用できない。

このベストプラクティスを活用するメリット:

- プロセスと手順が整備されていると、ワークロードの運用努力の効果が上がります。
- 新しいチームメンバーがより早く成果を出せるようになります。
- インシデントを軽減するための時間が短縮されます。
- さまざまなチームメンバー (やチーム) が同じプロセスと手順を一貫した方法で使用できます。
- 繰り返し使用可能なプロセスを持つことで、チームがプロセスをスケールすることができます。
- プロセスと手順が標準化されているため、チーム間でワークロードの責任を移転することによる影響を軽減できます。

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

- プロセスと手順に対し、その定義に責任を持つ所有者が指定されています。
  - 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。
  - 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のある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。
  - AWS Systems Manager などのサービス、ドキュメント、AWS Lambda を通じて、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をキャプチャします。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をキャプチャします。
- 時間が経つにつれて、これらの手順はコードとして実行できるように進化し、人的介入の必要が減るはずです。
  - 例えば、AWS Lambda 関数、CloudFormation テンプレート、AWS Systems Manager オートメーションのドキュメントを検討します。
  - 適切なリポジトリでバージョン管理を行います。
  - 所有者と文書を簡単に識別できるように、適切なリソースタグを付けてください。

お客様事例

AnyCompany Retail では、所有権を 1 つまたは (共通のアーキテクチャプラクティスとテクノロジーを共有する) 複数のアプリケーションのプロセスを所有するチームまたは個人と定義しています。最初にプロセスと手順をステップバイステップガイドとしてドキュメント管理システムに文書化し、アプリケーションをホストする AWS アカウントと、アカウント内の特定のリソースグループのタグを使用して手順を検出可能にしています。同社は AWS Organizations を活用して AWS アカウントを管理しています。時間の経過に伴い、これらのプロセスはコードに変換され、リソースは Infrastructure as Code (CloudFormation または AWS Cloud Development Kit (AWS CDK) テンプレートなど) を使用して定義されます。運用プロセスは AWS Systems Manager または AWS Lambda 関数でオートメーションドキュメントとなります。これらの関数は、スケジュールされたタスクとして、AWSCloudWatch アラームや AWS EventBridge イベントなどのイベントに応じて開始する場合

や、IT サービス管理 (ITSM) プラットフォーム内の要求に応じて開始する場合があります。すべてのプロセスには、所有者を識別するタグが付いています。オートメーションとプロセスのドキュメントは、プロセスのコードリポジトリによって生成された Wiki ページ内で管理されます。

実装手順

1. 既存のプロセスと手順を文書化します。
    a. レビューを行い、最新の状態に保ちます。
    b. 各プロセスまたは手順の所有者を特定します。
    c. それらをバージョン管理下に置きます。
    d. 可能な場合は、アーキテクチャ設計を共有するワークロードや環境全体でプロセスと手順を共有します。
2. フィードバックと改善のためのメカニズムを確立します。
    a. プロセスをレビューする頻度に関するポリシーを定義します。
    b. レビュー担当者と承認者用のプロセスを定義します。
    c. フィードバックを提供し、追跡するための問題やチケットキューを実装します。
    d. プロセスと手順は、可能な限り、変更承認委員会 (CAB) による事前承認とリスク分類を受ける必要があります。
3. プロセスと手順は、それらを実行するユーザーがアクセスおよび検出できることを確認します。
    a. タグを使用して、ワークロードのプロセスと手順にアクセスできる場所を示します。
    b. 有意義なエラーやイベントのメッセージを活用して、問題に対処するための適切なプロセスや手順を示します。
    c. Wiki とドキュメント管理を使用して、プロセスと手順を組織全体で一貫して検索できるようにします。
4. 必要に応じて自動化します。
    a. サービスやテクノロジーが API を提供している場合は、オートメーションを開発する必要があります。
    b. プロセスに関する指導を十分に行います。これらのプロセスを自動化するためのユーザーストーリーと要件を作成します。
    c. プロセスと手順の使用状況を適切に評価し、問題があれば反復的な改善に役立てます。

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:

- OPS02-BP01 リソースには特定の所有者が存在する
- OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する
- OPS11-BP04 ナレッジ管理を実施する

関連ドキュメント:

- AWS ホワイトペーパー - AWS での DevOps の概要
- AWS ホワイトペーパー - AWS リソースのタグ付けのベストプラクティス
- AWS ホワイトペーパー - 複数のアカウントで AWS 環境を構成する
- AWS クラウド運用および移行ブログ - クラウドオートメーションプラクティスを構築して運用上の優秀性を実現する: AWS Managed Services 提供のベストプラクティス
- AWS クラウド運用および移行ブログ - AWS Config と AWS Organizations で自動かつ一元的なタグ付けコントロールを実装する
- AWS セキュリティブログ - AWS CloudFormation Guard で pre-commit フックを拡張する
- AWS DevOps ブログ - AWS CloudFormation Guard を CI/CD パイプラインに統合する

関連するワークショップ:

- AWS Well-Architected 運用上の優秀性ワークショップ
- AWS Workshop - タグ付け

関連動画:

- How to automate IT Operations on AWS
- AWS re:Invent 2020 - Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 - Automating patch management and compliance using AWS (NIS306)
- AWS Supports You - Diving Deep into AWS Systems Manager

関連サービス:

- AWS Systems Manager - Automation

- AWS Service Management Connector

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のが誰かを理解すると、アクティビティを実行する人物、結果を検証する人物、アクティビティの所有者にフィードバックを提供する人物を把握できます。

期待される成果:

組織において、定義されたワークロードで特定のアクティビティを実行し、そのワークロードによって生成されたイベントに対応する責任が明確に定義されています。組織は、プロセスの所有権と履行を文書化し、この情報を検出可能にしています。組織で変更があった場合は責任を見直して更新し、チームは欠点や非効率を特定するアクティビティのパフォーマンスを追跡して測定します。フィードバックメカニズムを実装して、欠点や改善を追跡し、反復的な改善をサポートします。

一般的なアンチパターン:

- 責任を文書化しない。
- 断片化されたスクリプトが、隔離されたオペレータワークステーションに分散している。一部の個人だけがスクリプトの使用方法を知っている、またはチーム内の知識として非公式に参照している。
- レガシープロセスの更新が必要であるのに、プロセスの所有者が誰かを誰も把握しておらず、当初の作成者が既に組織を離れている。
- プロセスとスクリプトが検出可能になっていないため、必要なとき (インシデントへの対応時など) にすぐに利用できない。

このベストプラクティスを活用するメリット:

- アクティビティを実行する責任を持つのは誰か、アクションが必要なときに誰に通知すべきか、アクションを実行し、結果を検証してアクティビティの所有者にフィードバックを提供するのは誰かを理解できます。
- プロセスと手順が整備されていると、ワークロードの運用努力の効果が上がります。
- 新しいチームメンバーがより早く成果を出せるようになります。
- インシデントを軽減するための時間を短縮できます。

- さまざまなチームが同じプロセスと手順を使用して、一貫した方法でタスクを実行できます。
- 繰り返し使用可能なプロセスを持つことで、チームがプロセスをスケールすることができます。
- プロセスと手順が標準化されているため、チーム間でワークロードの責任を移転することによる影響を軽減できます。

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

責任の定義を始めるには、まず責任マトリックス、プロセスと手順、ロールと責任、ツールとオートメーションなどの既存のドキュメントから始めます。文書化されたプロセスについて責任をレビューし、ディスカッションを設けます。複数のチームとレビューして、文書化されている責任とプロセスの間の不一致を特定します。提供されるサービスについてそのチームの内部顧客と話し合い、チーム間での期待事項のギャップを特定します。

相違点を分析して対処します。改善の機会を特定し、頻繁にリクエストされ、リソースを大量に消費するアクティビティを探します。こうしたアクティビティは、一般的に有力な改善候補と考えられます。改善を簡素化し標準化するためのベストプラクティス、パターン、規範ガイダンスを調べます。改善の機会を記録し、改善が完了するまで追跡します。

経時的に、これらの手順をコードとして実行できるよう変換し、人的介入の必要性を減らします。例えば、AWS Lambda 関数、AWS CloudFormation テンプレート、または AWS Systems Manager Automation ドキュメントとして手順を開始できます。これらの手順が適切なリポジトリでバージョン管理されていることを確認し、チームが所有者とドキュメントを簡単に特定できるように適切なリソースタグを付けます。アクティビティの実行責任を文書化し、オートメーションの正常な起動と稼働、期待される成果のパフォーマンスをモニタリングします。

お客様事例

AnyCompany Retail では、所有権を 1 つまたは (共通のアーキテクチャプラクティスとテクノロジーを共有する) 複数のアプリケーションのプロセスを所有するチームまたは個人と定義しています。同社は、最初にプロセスと手順をステップバイステップガイドとしてドキュメント管理システムに文書化しています。アプリケーションをホストする AWS アカウントと、アカウント内の特定のリソースグループのタグを使用して手順を検出可能にし、AWS Organizations を使用して AWS アカウントを管理しています。時間の経過に伴って、AnyCompany Retail はこれらのプロセスをコードに変換し、Infrastructure as Code を使用して (CloudFormation または AWS Cloud Development Kit (AWS CDK) テンプレートなどのサービスを通じて) リソースを定義します。運用プロセスは AWS Systems Manager または AWS Lambda 関数でオートメーションドキュメントとなります。これらの関数は、

スケジュールされたタスクとして、Amazon CloudWatch アラームや Amazon EventBridge イベントなどのイベントへの応答として、または IT サービス管理 (ITSM) プラットフォーム内のリクエストによって起動できます。すべてのプロセスには、所有者を識別するタグが付いています。チームは、プロセスのコードリポジトリによって生成された Wiki ページ内で、オートメーションとプロセスのドキュメントを管理しています。

実装手順

1. 既存のプロセスと手順を文書化します。
    a. レビューを行い、最新の状態であることを確認します。
    b. 各プロセスまたは手順に所有者がいることを確認します。
    c. 手順をバージョン管理下に置きます。
    d. 可能な場合は、アーキテクチャ設計を共有するワークロードや環境全体でプロセスと手順を共有します。
2. フィードバックと改善のためのメカニズムを確立します。
    a. プロセスをレビューする頻度に関するポリシーを定義します。
    b. レビュー担当者と承認者用のプロセスを定義します。
    c. フィードバックを提供して追跡するための問題やチケットキューを実装します。
    d. プロセスと手順は、可能な限り、変更承認委員会 (CAB) による事前承認とリスク分類を受けます。
3. プロセスと手順を実行する必要がある人が、これにアクセスでき、検出できるようにします。
    a. タグを使用して、ワークロードのプロセスと手順にアクセスできる場所を示します。
    b. 有意義なエラーやイベントのメッセージを活用して、問題に対処するための適切なプロセスや手順を示します。
    c. Wiki またはドキュメント管理を使用して、プロセスと手順を組織全体で一貫して検索できるようにします。
4. 適切である場合は自動化します。
    a. サービスやテクノロジーが API を提供している場合は、オートメーションを開発します。
    b. プロセスが十分に理解されていることを確認し、これらのプロセスを自動化するためのユーザーストーリーと要件を作成します。
    c. プロセスと手順の適切な使用を評価し、問題を追跡して反復的な改善に役立てます。

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:

- 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-BP04 責任と所有権を管理するためのメカニズムが存在する
- 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
- OPS11-BP04 ナレッジ管理を実施する

関連ドキュメント:

- AWS ホワイトペーパー | AWS での DevOps の概要
- AWS ホワイトペーパー | AWS リソースのタグ付けのベストプラクティス
- AWS ホワイトペーパー | 複数のアカウントで AWS 環境を構成する
- AWS クラウド運用および移行ブログ | クラウドオートメーションプラクティスを構築して運用上の優秀性を実現する: AWS Managed Services 提供のベストプラクティス
- AWS Workshop - タグ付け
- AWS Service Management Connector

関連動画:

- AWS Knowledge Center Live | Tagging AWS Resources
- AWS re:Invent 2020 | Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 | Automating patch management and compliance using AWS (NIS306)
- AWS Supports You | Diving Deep into AWS Systems Manager

関連する例:

- AWS Well-Architected 運用上の優秀性ワークショップ

OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する

自分の役割の責任と、ビジネスの成果に自分が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。チームメンバーが各自の役割を把握していると、所有権を確立し、改善の機会を特定できるとともに、影響を与える方法や適切な変更を行う方法を理解できます。

責任の所有者が明確に定められていない場合もあります。このような場合は、このギャップを解消するメカニズムを構築します。所有権の割り当て権限を持つ個人への明確なエスカレーションパスを設定するか、ニーズに対処するための計画を立てます。

望ましい結果: 組織内のチームには、リソース、実行するアクション、プロセス、手順との関係性を含む、明確に定義された責任があります。これらの責任は、チームの責任と目標、および他のチームの責任と一致しています。エスカレーションパスを一貫性のある検出可能な方法で文書化し、決定内容を責任マトリクス、チーム定義、Wiki ページなどのドキュメントアーティファクトに反映させます。

一般的なアンチパターン:

- チームの責任が不明瞭、または明確に定義されていない。
- チームが役割と責任を一致させていない。
- チームが目標や目的と責任を一致させていないため、成功の測定が難しい。
- チームメンバーの責任が、チームや広範の組織と一致しない。
- チームが責任を最新の状態に保っていないため、チームが実行するタスクとの整合性が取れていない。
- 責任決定のためのエスカレーションパスが定義されていない、または不明瞭である。
- エスカレーションパスに、適時の対応を保証するシングルスレッド (専任) の所有者がいない。
- 役割、責任、エスカレーションパスが検出可能でないため、必要なとき (インシデントへの対応時など) にすぐには利用できない。

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が誰かを理解することで、適切なチームまたはチームメンバーに連絡して、リクエストをしたり、タスクを移行したりすることができる。
- 不作為やニーズへの未対応というリスクを低減するために、責任や所有権の割り当て権限を持つ個人を特定できる。
- 責任範囲が明確に定義されている場合、チームメンバーの自主性と所有権が高まる。
- 責任を理解することで、決定すべき事項、実行すべきアクション、および適切な所有者に引き渡すべきアクティビティが明確になる。

- 何がチームの責任範囲外かをしっかりと理解しているため、放棄された責任を特定しやすくなり、明確化のためにエスカレーションできるようになる。
- チームの混乱や緊張を防ぎ、チームがワークロードとリソースをより適切に管理できるようになる。

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

チームメンバーの役割と責任を特定し、チームメンバーが自らの役割に対する期待事項を理解していることを確認します。この情報を検出可能にして、組織のメンバーが、特定のニーズについ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。組織が AWS での移行とモダナイズの機会活用を目指す中で、役割と責任も変わる可能性があります。チームとそのメンバーにそれぞれの責任を認識させ、こうした変化の中で各自のタスクを遂行できるように適切にトレーニングを行います。

責任と所有権を特定するためのエスカレーションを受ける役割またはチームを決定します。このチームは、決定を下すためにさまざまなステークホルダーと連携できます。ただし、意思決定プロセスの管理責任はこのチームが負います。

組織のメンバーが所有権と責任を知り、特定するために、メンバーがアクセス可能なメカニズムを提供します。こうしたメカニズムによって、特定のニーズについて誰に連絡すべきかがわかります。

お客様事例

AnyCompany Retail は最近、リフトアンドシフトアプローチによって、オンプレミス環境から AWS のランディングゾーンへのワークロードの移行を完了しました。同社は、運用レビューを実施して、一般的な運用タスクをどのように達成するかを検討し、既存の責任マトリックスに新しい環境での運用が反映されていることを確認しました。オンプレミスから AWS に移行したことで、ハードウェアと物理インフラストラクチャに関連するインフラストラクチャチームの責任が軽減されました。この変化により、ワークロードの運用モデルを発展させる新たな機会があることも明らかになりました。

責任の大半の特定、対処、文書化を行うと同時に、見落とされた責任や、運用体制の発展に伴って変更が必要となる可能性のある責任についてのエスカレーションパスも定義しました。ワークロード全体にわたる標準化と効率向上のための新たな機会を探るには、AWS Systems Manager などの運用ツールや、AWS Security Hub および Amazon GuardDuty などのセキュリティツールへのアクセスを提供します。AnyCompany Retail では、最初に取り組むべき改善点に基づいて、責任と戦略の見直しをまとめました。同社は、新しい働き方や技術パターンの導入に合わせて、責任マトリックスを更新しています。

実装手順

1. 既存のドキュメントから始めます。一般的なソースドキュメントには以下が含まれます。
   a. 責任または実行責任者 (responsible)、説明責任者 (accountable)、相談先 (consulted)、報告先 (informed) (RACI) のマトリクス
   b. チーム定義または Wiki ページ
   c. サービスの定義とサービス内容
   d. 役割または職務内容
2. 文書化された責任をレビューし、ディスカッションを設けます。
   a. チームとレビューを行って、文書化された責任と、チームが通常遂行している責任との間の不一致を特定します。
   b. 内部顧客が提供している可能性のあるサービスについて話し合い、チーム間での期待事項のギャップを特定します。
3. 相違点を分析して対処します。
4. 改善の機会を特定します。
   a. 頻繁にリクエストされ、リソースを大量に消費するリクエストを特定します。こうしたリクエストは一般的に有力な改善候補と考えられます。
   b. ベストプラクティス、パターン、規範ガイダンスを探し、このガイダンスを基に改善を簡素化し標準化します。
   c. 改善の機会を記録し、完了まで追跡します。
5. チームが責任の割り当てを管理および追跡する責任を負っていない場合、その責任を担うチームメンバーを指定します。
6. チームが責任の明確化を求めるためのプロセスを定義します。
   a. プロセスを見直し、明確で使いやすいことを確認します。
   b. 必ず誰かがエスカレーションに責任を持ち、完了まで追跡するようにします。
   c. 効果を測定するための運用メトリクスを設定します。
   d. フィードバックメカニズムを作成して、チームが改善の機会を強調できるようにします。
   e. 定期的なレビューのメカニズムを導入します。
7. 検出とアクセスが可能な場所にドキュメントを保管します。
   a. Wiki またはドキュメントポータルが一般的です。

リソース

関連するベストプラクティス:

- OPS01-BP06 トレードオフを評価する
- 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-BP07 チームに適正なリソースを提供する
- OPS09-BP01 メトリクスを使用して業務目標と KPI を測定する
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け
- OPS11-BP01 継続的改善のプロセスを用意する

関連ドキュメント:

- AWS ホワイトペーパー - AWS での DevOps の概要
- AWS Whitepaper - AWS クラウド Adoption Framework: Operations Perspective
- AWS Well-Architected フレームワーク運用上の優秀性 - ワークロードレベルの運用モデルトポロジ
- AWS 規範的ガイダンス - Building your Cloud Operating Model
- AWS 規範的ガイダンス - Create a RACI or RASCI matrix for a cloud operating model
- AWS クラウド運用および移行ブログ - クラウドプラットフォームチームを編成してビジネス価値を提供する
- AWS クラウド運用および移行ブログ - クラウド運用モデルを導入すべき理由
- AWS DevOps ブログ - 組織のクラウドオペレーションをいかにモダナイズするか

関連動画:

- AWS Summit Online - Cloud Operating Models for Accelerated Transformation
- AWS re:Invent 2023 - Future-proofing cloud security: A new operating model

OPS02-BP05 追加、変更、除外をリクエストするメカニズムが存在する

プロセス、手順、およびリソースの所有者にリクエストを送信できます。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。メリットとリスク

を評価した後、実行可能かつ適切であると判断したリクエストを、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、熟考のうえで行われます。

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。
- 変更は熟考のうえで行われます。

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: 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構いません。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) マトリクスを使用して、プロセス、手順、リソース変更の所有者を特定しています。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、まずそれらから始めます。
2. 組織の関係者と協力して、変更管理プロセスを策定します。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 は、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策定では、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、リソースに特定された所有者が必要です。
- 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、プロセスに特定された所有者が必要です。
- 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された所有者が必要です。

関連ドキュメント:

- AWS 規範的ガイダンス - Foundation playbook for AWS large migrations: Creating RACI matrices
- Change Management in the Cloud Whitepaper

関連サービス:

- AWS Systems Manager Change Manager

OPS02-BP06 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を活用し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 - チーム間で契約を設定する前に、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 - チーム間で契約を設定する前に、運用アクティビティの所有権を特定する必要があります。

関連ドキュメント:

- AWS Executive Insights - ピザ 2 枚チームでイノベーションを促進する
- AWS での DevOps の概要 - 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を提供する

- 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-BP01 エグゼクティブスポンサーシップを提供する

トップレベルでは、シニアリーダーシップがエグゼクティブスポンサーの役割を果たし、成功の評価を含む、組織の成果に対する期待と方向性を明確に設定しています。スポンサーは、ベストプラクティスの採用と組織の進化を提唱し、推進しています。

望ましい成果: クラウド運用の導入、変革、最適化に取り組む組織は、望ましい成果に対して明確なリーダーシップと説明責任を確立します。このような組織は、新しい成果を達成するために組織が必要とする各能力を把握し、開発のために機能チームに所有権を割り当てます。リーダーシップは積極的にこの方向性を定め、所有権を割り当て、説明責任を担い、業務を定義します。その結果、組織全体にわたって個人が準備を整え、インスピレーションを受けて、期待される目標に向かって積極的に取り組むことができます。

一般的なアンチパターン:

- クラウド運用のスポンサーや計画が明確にされないまま、ワークロードの所有者にワークロードを AWS に移行することが義務付けられている。これにより、チームは運用能力の改善や成熟に向けて意識的に協力することがなくなっている。運用上のベストプラクティス基準が欠如しているため、チームに負担がかかり (オペレーターの労力、緊急対応、技術的負債など)、イノベーションの制約となっている。
- リーダーシップのスポンサーや戦略を提供せずに、新しいテクノロジーの導入という、組織全体にわたる新しい目標が設定されている。チームによって目標の解釈が異なるため、注力すべき点、それが重要である理由、影響の測定方法について混乱が生じている。結果として、組織はテクノロジーの導入に関する推進力を失っている。

このベストプラクティスを活用するメリット: エグゼクティブスポンサーシップが明確にコミュニケーションをとり、ビジョン、方向性、目標を共有することで、チームメンバーは自分に何が期待さ

れているかを理解します。リーダーが積極的に関与すると、個人とチームは、定義された目標を達成するために同じ方向へ集中的に注力し始めます。この結果、組織は成功に向けて能力を最大限に発揮できます。成功を評価すると、成功への障壁をより適切に特定して、エグゼクティブスポンサーの介入によって対処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

- クラウドジャーニーのあらゆる段階 (移行、導入、または最適化) で成功を得るには、指名されたエグゼクティブスポンサーによるトップレベルのリーダーシップの積極的な関与が必要です。エグゼクティブスポンサーは、定義された戦略に沿ってチームの考え方、スキルセット、作業方法を調整します。
  - 理由を説明する; 明確さを保ち、ビジョンと戦略の背後にある論理について説明します。
  - 期待値を設定する: 進捗状況や成功の測定方法など、組織の目標を定義して公開します。
  - 目標の達成度を追跡する: (タスクの完了だけでなく) 目標の段階的な達成度を定期的に測定します。成果が危ぶまれる場合に適切な措置を講じることができるように、結果を共有します。
  - 目標を達成するために必要なリソースを提供する: 人とチームを集めてコラボレーションを行い、定義された成果をもたらす適切なソリューションを構築します。これにより、組織の摩擦が軽減または排除されます。
  - チームを支援する: チームと常にかかわり、チームのパフォーマンスと、チームに影響を与える外的要因があるかどうかを理解します。チームの進捗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。チームが外的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。
  - ベストプラクティスの導入を促進する: 定量的なメリット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。
  - チームの進化を促す: 継続的な改善の文化を創造し、達成した進歩と失敗から積極的に学びます。個人と組織の両方の成長と発展を奨励します。データやエピソードを利用して、ビジョンと戦略を進化させます。

お客様事例

AnyCompany Retail は、生成 AI による顧客体験の迅速な改革、生産性の向上、成長の加速を通じたビジネス変革の途上にあります。

実装手順

1. シングルスレッドリーダーシップを確立して、変革を主導し、推進する主要エグゼクティブスポンサーを割り当てます。
2. 変革のビジネス成果を明確に定義して、所有権と説明責任を割り当てます。主要エグゼクティブに、重要な決定を主導して下す権限を付与します。
3. 変革戦略が極めて明確であり、エグゼクティブスポンサーから組織のあらゆるレベルに広く伝えられていることを確認します。
   a. IT とクラウドイニシアチブのビジネス目標を明確に定義します。
   b. IT とクラウドトランスフォーメーションを推進するための主要なビジネスメトリクスを文書化します。
   c. 戦略の一端を担うすべてのチームと個人に、継続的にビジョンを伝えます。
4. 特定のリーダー、マネージャー、個人の貢献者にどのようなメッセージを伝える必要があるかを明記した、コミュニケーション計画メトリクスを策定します。このようなメッセージを発信する個人またはチームを指定します。
   a. コミュニケーション計画は、一貫性をもって確実に遂行します。
   b. 定期的に対面イベントを開催し、期待される内容を明確化して管理します。
   c. コミュニケーションの有効性に関するフィードバックを受け入れ、これに応じてコミュニケーションを調整し、計画を策定します。
   d. コミュニケーションイベントをスケジュールして、チームが抱える課題を積極的に把握し、必要に応じて方針を修正できるような一貫性のあるフィードバックループを確立します。
5. リーダーシップの視点から各イニシアチブに積極的に関与して、影響を受けるすべてのチームが達成すべき成果を理解していることを確認します。
6. 各ステータスミーティングでは、エグゼクティブスポンサーは障害となる要因を探し、設定されたメトリクス、エピソード、またはチームからのフィードバックを調べ、目標に向けた進捗状況を測定する必要があります。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- 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
- OPS11-BP01 継続的改善のプロセスを用意する

- OPS11-BP07 オペレーションメトリクスのレビューを実行する

関連ドキュメント:

- 組織の毛玉をほどく: 統制をとる
- 生きている変革: 実用本位で変更にアプローチする
- 未来に対応できる企業になる
- CCOE を構築するときに避けるべき 7 つの落とし穴
- クラウドへの道しるべ: 成功のための重要業績評価指標 (KPI) とは

関連動画:

- AWS re:Invent 2023: A leader's guide to generative AI: Using history to shape the future (SEG204)

関連する例:

- Prosci: Primary Sponsor's Role & Importance

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

リーダーシップが植え付けた所有権文化の行動により、すべての従業員が、定義された役割と説明責任の範囲を超えて、会社全体のために行動する権限を与えられていると感じるようになります。従業員は、リスクが発生するに従ってプロアクティブにリスクを特定し、適切なアクションを取るように行動できます。このような文化により、従業員は状況を認識したうえで価値の高い意思決定を行うことができます。

例えば Amazon では、従業員が状況に従って前進し、問題を解決し、対立に対処し、アクションを起こすために必要な行動を推進するためのガイドラインとして、リーダーシップ原則を採用しています。

期待される成果: リーダーシップは、監査可能な許可と安全メカニズムで決定が定義される限り、個人やチームが組織の下位レベルでも重要な意思決定を行えるようにする新しい文化に影響を与えます。失敗を恐れないことが奨励され、チームは将来的に同様の状況に対処できるように、意思決定と対応を改善する方法を繰り返し学びます。その他のチームに利益をもたらすような改善につながったアクションがあれば、このようなアクションから学んだ知識を積極的に共有します。リーダーシップ

は、オペレーションの改善を測定し、個人や組織が同様のパターンを採用するようにインセンティブを提供します。

一般的なアンチパターン:

- リスクが特定された際に対処すべき内容についての明確なガイダンスやメカニズムが組織に存在しません。例えば、従業員がフィッシング攻撃を発見したときにセキュリティチームへの報告を怠った場合、組織の大部分が攻撃を受けてしまいます。これはデータ侵害の原因となります。
- サービスが利用できないことについて、顧客が不満を訴えています。サービスが利用できない主な原因は、デプロイの失敗です。デプロイツールは SRE チームが担当しており、デプロイの自動ロールバックは長期的なロードマップの対象となっています。最近のアプリケーションロールアウトで、エンジニアの 1 人がアプリケーションを以前のバージョンに自動的にロールバックするソリューションを考案しました。このソリューションは、SRE チームが採用するパターンとなる可能性があります。ただし、このような改善を追跡するプロセスがないため、その他のチームはこの方法を採用していません。組織は、顧客に影響を及ぼしてさらにマイナスのセンチメントを引き起こすデプロイの失敗に引き続き悩まされることになります。
- コンプライアンス維持のため、社内の情報セキュリティチームは、Amazon EC2 Linux インスタンスに接続するオペレーターに代わって、共有 SSH キーを定期的にローテーションするプロセスを長年管理しています。情報セキュリティチームがキーローテーションを完了するまでに数日かかり、その間の対象インスタンスへの接続はブロックされます。情報セキュリティにもその他のチームにも、同様の結果を得るために AWS のその他のオプションを利用することを提案する者はいません。

このベストプラクティスを活用するメリット: 意思決定を行う権限を分散し、チームが主要な意思決定を行えるようにすることで、成功率を上げ、より迅速に問題に対処できます。さらに、チームは当事者意識を持ち始め、失敗を受け入れられるようになります。実験が文化の中軸となります。マネージャーやディレクターは、業務のあらゆる面で細かく管理されているようには感じません。

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: 中

実装のガイダンス

1. 失敗が起こり得ることが予想される文化を育みます。
2. 組織内のさまざまな業務領域について、明確な所有権と説明責任を定義します。
3. 所有権と説明責任を全員に伝え、分散型の意思決定を円滑に進めるうえで支援を提供する人物が誰であるかを各自が把握できるようにします。

4. 単一の方向と双方向の意思決定を定義し、より高いレベルのリーダーシップにエスカレーションする必要があるケースを各自が把握できるようにします。
5. 成果がリスクに直面した場合、すべての従業員がさまざまなレベルで対処する権限を付与されているという意識を組織全体で高めます。ガバナンス、アクセス許可レベル、ツール、機会に関するドキュメントをチームメンバーに提供して、効果的に対応するために必要なスキルを練習します。
6. さまざまな意思決定に対応するために必要なスキルを練習する機会をチームメンバーに提供します。意思決定レベルを定義したら、ゲームデーを開催して、各貢献者がプロセスを理解し、実際に実行できることを確認します。
   a. プロセスと手順のテストとトレーニングを実行できる安全な代替環境を用意します。
   b. 成果に既に定義されているレベルのリスクがある場合、チームメンバーにはアクションを起こす権限があるという意識を受け入れ、育みます。
   c. 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定義します。
7. チームが学んだこと (運用上の成功と失敗) を共有できるようにします。
8. チームが現状に問題意識を持てるようにして、改善点と組織に及ぼす影響を追跡して測定するメカニズムを提供します。

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:

- OPS01-BP06 メリットとリスクを管理しながらトレードオフを評価する
- 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

関連ドキュメント:

- AWS ブログ記事 | 俊敏なエンタープライズ
- AWS ブログ記事 | 成功を測定する: パラドックスと計画
- AWS ブログ記事 | 吹っ切る: チームの自律性を育む
- 集中化か分散化か?

関連動画:

- re:Invent 2023 | How to not sabotage your transformation (SEG201)
- re:Invent 2021 | Amazon Builders' Library: Operational Excellence at Amazon
- Centralization vs. Decentralization

関連する例:

- Using architectural decision records to streamline technical decision-making for a software development project

OPS03-BP03 エスカレーションが推奨されている

リーダーシップは、期待される成果がリスクにさらされ、期待される基準が満たされないと判断された場合にチームメンバーが問題や懸念事項を上位レベルの意思決定者やステークホルダーにエスカレーションするよう奨励します。これは組織内文化の特徴となり、あらゆるレベルで推進されます。リスクを特定し、インシデントの発生を防ぐため、エスカレーションは、早期かつ頻繁に実行する必要があります。リーダーシップは、問題をエスカレーションした個人を叱責することはありません。

期待される成果: 組織全体の個人は、問題を直上のリーダーシップにエスカレーションすることに慣れています。チームがいかなる問題であっても安心して エスカレーションできるはずだという期待を、リーダーシップは意図的かつ意識的に確立しています。組織内の各レベルで問題をエスカレーションするメカニズムが施行されています。従業員がマネージャーにエスカレーションする場合、影響レベルと問題をエスカレーションすべきかどうかを連携して決定します。従業員がエスカレーションを開始するには、問題に対処するための推奨される作業計画を含める必要があります。直属のリーダーシップがタイムリーにアクションを起こさない場合、組織へのリスクがエスカレーションに値すると確信する従業員は、トップレベルのリーダーシップに問題を提起するよう奨励されます。

一般的なアンチパターン:

- エグゼクティブリーダーシップは、クラウドトランスフォーメーションプログラムのステータスミーティング中に、十分な質問をしておらず、問題や障害が発生している個所を発見することができません。ステータスとして良好な報告のみが提示されます。いかなる課題が提起されても、CEO はプログラムが失敗していると判断するため、良好な報告のみを発表したいと CIO が明言したためです。
- クラウド運用エンジニアが、新しいナレッジ管理システムがアプリケーションチームによって広く採用されていないことに気づきました。この企業では、この新しいナレッジ管理システムに数百万 USD を投資し、1 年かけて実装しました。しかし、チームは依然としてランブックをローカルで作成し、組織のクラウド共有で共有しているため、サポートされているワークロードに関連す

るナレッジを検索するのが困難となっています。このシステムを継続的に使用することで業務効率を向上できるため、クラウド運用エンジニアは、この件についてリーダーシップに報告しようとします。ナレッジ管理システムの実装を主導するディレクターにこの件について伝えたところ、この報告により投資が問題視されるという理由で、ディレクターはクラウド運用エンジニアを叱責します。

- コンピューティングリソースの強化を担当する情報セキュリティチームは、リソースを本番環境でリリースする前に、EC2 インスタンスが完全に保護されていることを確認するために必要となるスキャンをコンピューティングチームが実行する必要があるプロセスを導入することを決定しました。これにより、リソースのデプロイがこれまでより 1 週間遅延し、SLA に違反することになります。コンピューティングチームは、情報セキュリティ担当 VP の評判が悪くなることを懸念して、この問題についてクラウド担当の VP にエスカレーションすることを躊躇しています。

このベストプラクティスを活用するメリット:

複雑な問題や重大な問題が、ビジネスに影響を及ぼす前に対処されます。無駄な時間が低減します。リスクが最小限に抑えられます。問題を解決する際、チームがより積極的になり、結果を重視するようになります。

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

組織のあらゆるレベルで自由にエスカレーションする意欲と能力は、組織と文化の基盤であり、重点的なトレーニング、リーダーシップのコミュニケーション、期待値設定、組織全体のあらゆるレベルでのメカニズムのデプロイを通じて、意識的に発展させる必要があります。

実装手順

1. 組織のポリシー、基準、期待される内容を定義します。
   a. ポリシー、期待される内容、基準を幅広く採用し、理解されていることを確認します。
2. 基準が満たされない場合、早期かつ頻繁にエスカレーションを行うよう従業員を奨励し、トレーニングを行い、権限を付与します。
3. 早期かつ頻繁にエスカレーションすることがベストプラクティスであるという組織的な認識を確立します。エスカレーションした内容が事実ではないと判明する場合があること、および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好ましいということを受け入れます。
   a. エスカレーションのメカニズムを構築します (Andon コードシステムなど)。
   b. 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。

   c. アクションを実行または承認する人々を権限順に定義します。また、各ステークホルダーの連絡先の情報も追加します。
4. エスカレーションが行われた場合、リーダーシップが促進する一連のアクションによりリスクが軽減されたとチームメンバーが認めるまで、エスカレーションを続行する必要があります。
   a. エスカレーションには以下を含める必要があります。
      i. 状況およびリスクの性質の説明
      ii. 状況の重要度
      iii. 影響が及ぶ人々や事項
      iv. 影響の規模
      v. 影響が発生した場合の緊急性
      vi. 推奨される救済策と緩和計画
   b. エスカレーションする従業員を保護します。対処を行わない意思決定者やステークホルダーを避けてエスカレーションしたチームメンバーを報復行為から保護するポリシーを施行します。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。
5. 組織が生み出すすべてのものに、継続的な改善のフィードバックループを取り入れる文化を奨励します。フィードバックループは責任者への軽微なエスカレーションとして機能し、エスカレーションが必要ない場合でも改善の機会を特定できます。継続的な改善の文化は、誰もがより積極的に行動することを押し進めます。
6. リーダーシップは、ポリシー、基準、メカニズム、報復されることのないオープンなエスカレーションと継続的なフィードバックループを奨励することを定期的に繰り返し強調すべきです。

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:

- OPS02-BP05 追加、変更、除外をリクエストするメカニズムが存在する

関連ドキュメント:

- How do you foster a culture of continuous improvement and learning from Andon and escalation systems?
- The Andon Cord (IT Revolution)
- AWS DevOps ガイダンス | 明確なエスカレーションパスを確立し建設的な反対意見を奨励する

関連動画:

- Jeff Bezos on how to make decisions (& increase velocity)
- 【トヨタ生産方式】自働化: 異常が発生したらまず止める「呼び出しボタンとアンドン」
- Andon Cord in LEAN Manufacturing

関連する例:

- Working with escalation plans in Incident Manager

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

リーダーシップには、特に組織が新しい戦略、テクノロジー、または働き方を採用する場合、強力かつ効果的なコミュニケーションを創出する責任があります。リーダーシップは、スタッフ全員が企業の目標を目指して業務を行えるように、期待するものを明らかにする必要があります。リーダーシップが資金を提供し、スポンサーとなっている計画の実施を担当するチームにおける意識を向上し、維持するためのコミュニケーションメカニズムを考案します。組織間の多様性を活用して、複数の独自の視点での意見に注意深く耳を傾けます。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。有益な視点が得られるように、チーム内でのインクルージョン、多様性、アクセシビリティを向上します。

期待される成果: 組織は、組織への変化の影響に対処するためのコミュニケーション戦略を設計します。チームには常に情報が提供され、反目し合うのではなく、相互に協力し合う意欲があります。個人は、明文化された目標を達成するうえで、自身の役割がいかに重要であるかを理解しています。E メールは受動的な通信手段に過ぎません。この点を踏まえて使用します。経営陣は、個別の貢献者と話す時間を取り、各自の責任や完了すべきタスク、担当業務がミッション全体にどのように貢献するのかを理解してもらいます。リーダーシップは必要に応じて、小規模な場所で直接従業員とのエンゲージメントの機会を持ち、メッセージを伝え、メッセージが効果的に伝わっていることを確認します。優れたコミュニケーション戦略があれば、組織はリーダーシップの期待と同等かそれ以上の成果を上げることができます。リーダーシップは、チーム内およびチーム間で多様な意見を出すことを奨励し、多様な意見を求めます。

一般的なアンチパターン:

- 組織には、すべてのワークロードを AWS に移行する 5 か年計画があります。クラウドのビジネスケースには、すべてのワークロードの 25% をモダナイズしてサーバーレステクノロジーを活用することが含まれています。CIO は、この戦略を直属部下に伝え、各リーダーが対面でのコミュニケーションなしにマネージャー、ディレクター、個別の貢献者にこのプレゼンテーションを伝達す

ることを期待しています。CIO は現場に関与せず、この組織で新しい戦略が実行されることを期待しています。

- リーダーシップはフィードバックの仕組みを提供したり、利用したりすることはなく、期待のギャップが広がり、プロジェクトが行き詰まってしまいます。
- セキュリティグループに変更を加えるよう求められますが、どのような変更が必要か、変更がすべてのワークロードにどのような影響を及ぼす可能性があるか、いつ実行すべきかについての詳細は提供されていません。マネージャーは、情報セキュリティの VP からの E メールに、「これを実行すること」と付け加えて転送しています。
- 移行戦略に変更が加えられ、計画されているモダナイゼーションの件数が 25% から 10% に低減しました。これはオペレーション組織の下流に影響を及ぼします。この戦略的変更についての知らせはなかったため、AWS にリフトアンドシフトするワークロード数の増加分をサポートするのに十分なスキルを備えたリソースの準備が整っていません。

このベストプラクティスを活用するメリット:

- 組織は、新しい戦略や変更された戦略について十分な情報を得ており、リーダーシップが設定した全体的な目標とメトリクスを相互に支援して達成するうえで、高いやる気を持って行動します。
- 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。
- 必要なスキルとともに、新しい働き方 (人、組織、プロセス、またはテクノロジーの変更を含む) を採用することで、より効果的に組織に導入でき、組織はビジネス上の利点をより迅速に実現できるようになります。
- チームメンバーは、受けとったコミュニケーションについて必要なコンテキストを把握できるため、より効果的に業務を進めることができます。

このベストプラクティスを活用しない場合のリスクレベル: 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。大規模な IT 移行の場合、このようなベストプラクティスを考慮に入れない組織よりも、確立されたプランニングチームの方が、変更による従業員への影響をうまく管理できます。大規模な組織では、新しい戦略に対する個別の貢献者全員の強い賛同を得ることが重要となるため、変更管理がより困難になる可能性があります。このような移行プランニングチームを設置しない場合、効果的なコミュニケーションはリーダーシップが 100% 責任を負うことになります。移行プランニングチームを設立する際は、

すべての組織のリーダーと協力し、あらゆるレベルにおける効果的なコミュニケーションを定義して、管理するチームメンバーを割り当てます。

お客様事例

AnyCompany Retail は、AWS エンタープライズサポートにサインアップして、クラウド運用は別のサードパーティープロバイダーに依存しています。同社は、業務活動の主要なコミュニケーション媒体としてチャットと ChatOps を利用しています。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。本稼働システムへの主要な変更については、変更カレンダーを使用してスケジュールしています。

実装手順

1. 組織内の複数のレベルで発生する変化に対するコミュニケーション計画を策定し、着手する責任を担うコアチームを組織内に設立します。
2. シングルスレッドの所有権を導入して、監督体制を実現します。個別のチームが独自にイノベーションを生むことができる体制を整え、一貫性あるメカニズムをバランスよく使用できるようにすることで、適切なレベルの検査と方向性を提示するビジョンを実現できます。
3. 組織全体にわたる関係者と協力して、コミュニケーションの基準、慣行、計画への合意を取り付けます。
4. コアコミュニケーションチームが組織リーダーやプログラムリーダーと協力して、リーダーの代理として適切なスタッフへのメッセージを作成していることを確認します。
5. 告知、共有カレンダー、全員参加のミーティング、対面または 1 対 1 の方法で変更を管理するための戦略的コミュニケーションメカニズムを構築し、自身が取るべき行動についての適切な期待をチームメンバーが把握するようにします。
6. 対処が必要かを判断するために、必要となる状況、詳細、時間 (可能な場合) を提供します。アクションが必要な場合は、必要なアクションとその影響を提供します。
7. 社内チャット、E メール、ナレッジ管理など、戦術的なコミュニケーションを促進するツールを導入します。
8. すべてのコミュニケーションが期待される成果につながっているかを測定して検証するメカニズムを実装します。
9. すべてのコミュニケーションの有効性を測定するフィードバックループを確立します。特に、コミュニケーションが組織全体での変化に対する抵抗に関連する場合に、これは重要です。

10.すべての AWS アカウントについて、請求、セキュリティ、オペレーション用の代替連絡先を設定します。理想的には、各連絡先は特定の個人の連絡先ではなく、E メールの配布リストであるべきです。

11AWS サポートやその他のサードパーティープロバイダーなどの社内チームおよび社外チームと連携するために、エスカレーションとリバースエスカレーションのコミュニケーション計画を策定します。

12.各トランスフォーメーションプログラムの全期間にわたり、一貫性あるコミュニケーション戦略を開始し、実行します。

13繰り返し可能なアクションを可能な限り優先し、大規模かつ安全に自動化します。

14.アクションが自動化されているシナリオでコミュニケーションが必要な場合、コミュニケーションの目的はチームへの情報提供、監査、または変更管理プロセスの一部であるべきです。

15.アラートシステムからの通信を分析して、誤検出や絶えず発生するアラートがないかを調べます。このようなアラートを削除したり変更したりして、人の介入が必要な際に起動されるようにします。アラートが起動した場合は、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と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。

16.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. AWS Chatbot を使用すると、アラートを送信したり、組織のメッセージプラットフォーム内のイベントに応答したりできます。

17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 を使用すると、変更を実行できる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。

18脆弱性の通知とパッチ情報をモニタして、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すれば、AWS 上の脆弱性に関する通知がお手元に届きます。

19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。取り上げられることの少ないグループにコミュニケーションの機会を与えます。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。

a. 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。チームメンバーは、このようなロールから、また、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。チームメンバーはまた、自身が得た経験と視点を、やり取りをする新しいロールやチームメンバーに提供することができます。視野が広がるにつれて、新たなビジネスチャンスや新たな改善機会を見極めます。チーム内のメンバーに、その他のメンバーが通常実行している日常的なタスクを交替で担当してもらい、このようなタスクを実行する需要と影響を理解してもらいます。

b. 安全かつ安心できる環境を提供する: 組織内のチームメンバーの、精神的、身体的安全を確保するポリシーと統制を実施します。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全で安心できると、エンゲージメントと生産性が向上する可能性が高くなります。組織の多様化が進むと、お客様を含め、サポート対象への理解が深まります。チームのメンバーが安心して自由に意見を出し、話を聞いてもらえることを確信すると、貴重なインサイトを共有する可能性が高まります (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)。

c. 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、このような課題を回避するうえでのスキルを身に付けています。このように独自に開発したスキルは、組織に大きな利点をもたらします。必要な調整を行いながらチームメンバーをサポートすることで、メンバーの貢献から得られる利点が拡大します。

リソース

関連する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を提供する
- 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

関連ドキュメント:

- AWS Blog post | Accountability and empowerment are key to high-performing agile organizations
- AWS Executive Insights | 複雑さではなくイノベーションの拡大を学ぶ | シングルスレッドリーダー
- AWS セキュリティ速報
- OpenCVE
- AWS Support App in Slack でサポートケースを管理する
- AWS Chatbot を使用して Slack チャンネルで AWS リソースを管理する

関連する例:

- Well-Architected Labs: Inventory and Patch Management (Level 100)

関連サービス:

- AWS Chatbot
- AWS Systems Manager Change Calendar
- AWS Systems Manager のドキュメント

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を活用し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. 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。AWS Control Tower を使用して、これらのアカウントをプロビジョニングできます。
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 機能フラグは、機能フラグを作成する機能を提供します。
   b. Amazon CloudWatch Evidently を使用して、限られたデプロイで A/B テストを実行できます。
   c. AWS Lambda バージョンを使用して、ベータテスト用に機能の新しいバージョンをデプロイできます。

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 - インシデントから学ぶことは、実験と共にイノベーションの重要な推進要因です。

- 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。

関連ドキュメント:

- 内部から見たアマゾン文化:実験、失敗、そしてカスタマーオブセッション
- Best practices for creating and managing sandbox accounts in AWS
- Create a Culture of Experimentation Enabled by the Cloud
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros
- Experiment More, Fail Less
- 複数のアカウントで AWS 環境を構成する - サンドボックス OU
- Using AWS AppConfig Feature Flags

関連動画:

- AWS On Air ft. Amazon CloudWatch Evidently | AWS Events
- AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R)
- Programmatically Create an AWS アカウント with AWS Control Tower
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

関連する例:

- AWS Innovation Sandbox
- End-to-end Personalization 101 for E-Commerce

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。チームメンバーが磨いているスキルを検証し、認識するような業界認証を取得し、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 のイベントとウェビナー、AWS Well-Architected ラボなどのリソースを提供し、チームを教育するためのガイダンス、例、詳細なウォークスルーを提供します。

AWS Support、AWS re:Post、AWS Supportセンター、AWS ドキュメントなどのリソースは、技術的な課題を解決し、運用を改善するのに役立ちます。不明な点があれば、AWS Supportセンター経由で AWS Supportまでお問い合わせください。

AWS は、AWS のオペレーションを通じて学習したベストプラクティスとパターンを Amazon Builders' Library で公開しています。また、AWS ブログと公式 AWS ポッドキャストでさまざまな有用な教育資料も公開します。

AWS トレーニング および 認定には、セルフペースデジタルコースによる無料トレーニングと、ロールまたはドメイン別の学習プランが含まれます。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。

望ましい結果: 組織はスキルギャップを常に評価し、構造化された予算と投資でそれらを解決します。チームでは、業界をリードする認定資格の取得などのスキルアップのアクティビティを行うようにメンバーを奨励しています。チームは、ランチアンドラーン、Immersion Days、ハッカソン、ゲームデーなど、専用の相互共有ナレッジプログラムを活用します。組織のナレッジシステムを常に最新の状態に維持し、新入社員のオンボーディングトレーニングなど、クロストレーニングチームメンバーとの関連性を維持します。

一般的なアンチパターン:

- 体系的なトレーニングプログラムと予算がなく、チームはテクノロジーの進化に遅れずに対応しようとする際に不安を感じるため、離職率が増大します。
- AWS への移行の取り組みの中で、組織のチーム間でスキルギャップやクラウド知識のレベルの違いがあることが判明します。スキルアップに向けた取り組みを行わなければ、チームはレガシーで

非効率的なクラウド環境の管理に追われ、オペレーターの労力が増大することになります。このような燃え尽き症候群は従業員の不満を増大します。

このベストプラクティスを活用するメリット: 組織がチームのスキル向上に意識的に投資すると、クラウドの導入と最適化を加速し、スケールするのにも役立ちます。対象を絞った学習プログラムはイノベーションを促進し、チームがイベントを処理する準備を整えるうえでの運用能力を向上します。チームはベストプラクティスの実装と発展に意識的に投資します。チームのやる気は高く、チームメンバーはビジネスへの貢献を重要視しています。

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: 中

実装のガイダンス

新しいテクノロジーを導入し、イノベーションを促進して、需要と職務の変化に対応してワークロードをサポートするために、チームのプロフェッショナルとしての成長に継続的に投資します。

実装手順

1. 構造化されたクラウドアドボカシープログラムを使用する: AWS Skills Guild は、クラウドスキルの自信を高め、継続的な学習の社風を呼び起こすためのコンサルティングトレーニングを提供します。
2. 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。上級チームのメンバーを下級チームのメンバーのメンターとして交流できるようにしたり、上級チームのメンバーの業務を下級チームのメンバーがシャドーイングして、手法やスキルを学ぶ機会を提供したりします。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。
3. 高度な技術リソースの使用を奨励する: AWS re:Post などのリソースを活用して、厳選された知識や交流が活発なコミュニティにアクセスできます。
4. 最新のナレッジリポジトリを構築して維持する: Wiki やランブックなどのナレッジ共有プラットフォームを使用します。AWS re:Post Private を使用して独自の再利用可能なエキスパートナレッジソースを作成し、コラボレーションを合理化し、生産性を向上させ、従業員のオンボーディングを高速化します。
5. チーム教育とチーム間のエンゲージメントを実施する: 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します。

6. 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。

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:

- OPS03-BP01 エグゼクティブスポンサーシップを提供する
- OPS11-BP04 ナレッジ管理を実施する

関連ドキュメント:

- AWS Whitepaper | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS トレーニング と認定
- AWS Support
- AWS re:Post
- 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 ドキュメント
- The Official AWS Podcast
- AWS オンライン技術トーク
- AWS イベントスケジュール
- AWS Well-Architected Labs
- Amazon Builders' Library

関連動画:

- AWS re:Invent 2023 | Reskilling at the speed of cloud: Turning employees into entrepreneurs

- WS re:Invent 2023 | Building a culture of curiosity through gamification

OPS03-BP07 チームに適正なリソースを提供する

適切な数の熟練したチームメンバーを配置し、ワークロードのニーズをサポートするツールとリソースを提供します。チームメンバーの負担が過剰な場合、ヒューマンエラーのリスクが増大します。オートメーションなどのツールやリソースへの投資を通じてチームの効率を向上すると、リソースを追加する必要なく、より多くのワークロードに対応できるようになります。

期待される成果:

- 移行計画に従って AWS ワークロードを運用するために必要なスキルセットを習得できるように、チームに適切な人員を配置しています。移行プロジェクトの過程でチームがスケールアップするにつれ、チームはアプリケーション移行やモダナイズの際に企業が使用する予定の AWS のコアテクノロジーに習熟するようになりました。
- オートメーションとワークフローを活用してリソースを効率的に使用できるように、人員配置計画を慎重に調整しています。少人数のチームでも、アプリケーション開発チームの代理で、より多くのインフラストラクチャを管理できるようになりました。
- 業務上の優先順位が変化しても、ビジネスイニシアチブの成功を守るために、人員配置の制約が事前に特定されます。
- 業務上の労力 (オンコールの疲労や過剰な呼び出しなど) を報告するオペレーションメトリクスの見直しを行い、スタッフに負担がかからないことを確認します。

一般的なアンチパターン:

- 複数年にわたるクラウド移行計画が間近に迫っているのにもかかわらず、スタッフの AWS スキルは向上していません。このため、ワークロードのサポートがリスクにさらされ、従業員の士気が低下しています。
- IT 組織全体がアジャイルな働き方にシフトしています。ビジネス部門は、製品ポートフォリオに優先順位を付け、どの機能を最初に開発する必要があるかについてのメトリクスを設定しています。アジャイルプロセスでは、チームが作業計画にストーリーポイントを割り当てる必要はありません。この結果、以降の労力に必要となるキャパシティレベルを把握することも、そのタスクに適切なスキルが割り当てられているかを判断することができません。
- AWS パートナーにワークロードを移行してもらっていますが、パートナーが移行プロジェクトを完了した後のチームのサポートの移行計画が策定されていません。チームはワークロードを効率的かつ効果的にサポートするのに苦労しています。

このベストプラクティスを活用するメリット: 適切なスキルを持つ、ワークロードをサポートするチームメンバーが組織内にいます。リソースの割り当ては、パフォーマンスに影響を及ぼすことなく、優先順位の変化に適応できます。その結果、チームは顧客向けのイノベーションに集中する時間を最大限に活用しながら、ワークロードのサポートに習熟できるようになり、これが従業員の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

クラウド移行のためのリソース計画および実装する予定の運用モデルは、移行計画に沿った組織レベルで行う必要があります。新しいクラウド環境をサポートするために実装される望ましい運用モデルも同様です。これには、ビジネスチームとアプリケーション開発チームにどのクラウドテクノロジーがデプロイされるかを把握することも含まれている必要があります。インフラストラクチャと運用のリーダーシップは、クラウドの導入を主導するエンジニアのスキルギャップ分析、トレーニング、ロール定義を計画する必要があります。

実装手順

1. スタッフの生産性などの関連する運用指標 (ワークロードをサポートするためのコストやインシデント時に費やしたオペレーター時間など) を使用して、チームの成功の基準を定義します。
2. リソースキャパシティプランニングと検査のメカニズムを定義し、適切なバランスの適格なキャパシティが必要な際に利用でき、長期的に調整可能かを検証します。
3. チームに影響を及ぼす業務上の課題 (責任範囲の増大、テクノロジーの変化、人員の喪失、対応する顧客の増加など) を把握するためのメカニズムを作成します (チームに毎月アンケートを送信するなど)。
4. このようなメカニズムを使用してチームとのエンゲージメントを維持し、従業員の生産性に関する課題の一因となる可能性のある傾向を検出します。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。
5. 現在提供されているリソースが十分であるか、追加のリソースが必要かどうかを定期的に確認して、サポートチームの適切な調整を行います。

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:

- 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
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け
- 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-BP07 イベントへの対応を自動化する

関連ドキュメント:

- AWS クラウド Adoption Framework: People Perspective
- 未来に対応できる企業になる
- Prioritize your Employees' Skills to Drive Business Growth
- 高いパフォーマンスを発揮する組織 - Amazon ピザ 2 枚チーム
- How Cloud-Mature Enterprises Succeed

準備

Questions

- OPS 4. オブザーバビリティをワークロードに実装するにはどうすればよいでしょうか?
- OPS 5. どのようにして欠点を減らし、修正を簡単にし、本番環境へのフローを改善しますか?
- OPS 6. どのようにデプロイメントリスクを軽減するのですか?
- OPS 7. どのように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のですか?

OPS 4. オブザーバビリティをワークロードに実装するにはどうすればよいでしょうか?

ワークロードにオブザーバビリティを実装することで、ワークロードの状態を把握し、ビジネス要件に基づいてデータ主導の意思決定を行うことができます。

ベストプラクティス

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する
- OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する
- OPS04-BP04 依存関係のテレメトリを実装する

- OPS04-BP05 分散トレースを実装する

OPS04-BP01 主要業績評価指標を特定する

ワークロードにオブザーバビリティを実装するには、まずワークロードの状態を理解し、ビジネス要件に基づいてデータ主導の意思決定を行います。モニタリングアクティビティとビジネス目標を合致させる最も効果的な方法の 1 つは、主要業績評価指標 (KPI) を定義してモニタリングすることです。

期待される成果: ビジネス目標と緊密に連携した効率的なオブザーバビリティを実践することにより、モニタリングアクティビティが常に具体的なビジネス成果につながります。

一般的なアンチパターン:

- 未定義の KPI: 明確な KPI がないまま作業を進めると、過度なモニタリングやモニタリング不足になり、重要なシグナルを見逃してしまう可能性がある。
- 静的 KPI: ワークロードやビジネス目標が変化しても KPI を再検討したり調整したりしていない。
- ビジネスの成果と直接の相互関係がない、または実際の問題との関連性が明らかでない技術的なメトリクスに重点が置かれている。

このベストプラクティスを活用するメリット:

- 問題の特定が容易: 多くの場合、技術的なメトリクスと比較して、ビジネス KPI はより明確に問題を検出します。ビジネス KPI の低下は、多数の技術的メトリクスを細かく検証するよりも効果的に問題を特定できます。
- ビジネスとの連携: モニタリングアクティビティがビジネス目標を直接サポートしていることが確認できます。
- 効率性: モニタリングリソースに優先順位を付けて重要なメトリクスに注目できます。
- 積極性: 問題がビジネスに及ぼす影響が拡大する前に、問題を認識して対処できます。

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 KPI を効果的に定義する方法:

1. ビジネス成果を理解する: メトリクスの詳細に取り掛かる前に、望ましいビジネス成果を理解しておきます。売上の増加、ユーザーエンゲージメントの向上、または応答時間の短縮などがあります。

2. 技術メトリクスをビジネス目標と関連付ける: すべての技術メトリクスがビジネス成果に直接影響するわけではありません。直接影響するようなメトリクスを特定します。ただし、多くの場合、ビジネス KPI を使用して問題を特定する方が簡単です。
3. Amazon CloudWatch を使用する: CloudWatch を使用して、KPI と関連するメトリクスを定義およびモニタリングします。
4. KPI を定期的にレビュー、更新する: ワークロードとビジネスが進化するにつれて、KPI も関連性があるものを維持します。
5. 関係者を参画する: KPI の定義とレビューには、技術チームと業務チームの両方の関与が必要です。

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:

- the section called “OPS04-BP02 アプリケーションテレメトリを実装する”
- the section called “OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する”
- the section called “OPS04-BP04 依存関係のテレメトリを実装する”
- the section called “OPS04-BP05 分散トレースを実装する”

関連ドキュメント:

- AWS Observability Best Practices
- CloudWatch ユーザーガイド
- AWS Skill Builder - AWS Observability (Japanese) 日本語吹き替え版

関連動画:

- Developing an observability strategy

関連する例:

- 1 つのオブザーバビリティワークショップ

OPS04-BP02 アプリケーションテレメトリを実装する

アプリケーションテレメトリは、ワークロードオブザーバビリティの基盤です。アプリケーションの状態や、技術的およびビジネス上の成果の達成に関する実践的なインサイトを提供するテレメトリを送出することが重要です。トラブルシューティングから新機能の影響の測定、ビジネスの主要業績評価指標 (KPI) との整合性の確認まで、アプリケーションテレメトリを使用することで、ワークロードのビルド、運用、展開の仕方に関する情報を得ることができます。

メトリクス、ログ、トレースは、オブザーバビリティの 3 つの主要な柱となります。この 3 つの柱は、アプリケーションの状態を説明する診断ツールとして機能し、徐々にベースラインの作成や異常の特定に役立つようになります。ただし、モニタリングアクティビティとビジネス目標の整合性を確保するには、主要業績評価指標 (KPI) を定義してモニタリングすることが非常に重要です。多くの場合、ビジネス KPI を使用すると、技術的なメトリクスのみの場合よりも問題を特定しやすくなります。

リアルユーザーモニタリング (RUM) や合成トランザクションなどのその他の種類のテレメトリは、これらの主要なデータソースを補完します。RUM はリアルタイムのユーザーの操作に関するインサイトを提供します。合成トランザクションは潜在的なユーザー行動のシミュレーションを行い、実際のユーザーがボトルネックに直面する前にボトルネックを検出するのに役立ちます。

期待される成果: ワークロードのパフォーマンスに関する実践的なインサイトを導き出します。このようなインサイトを活用すると、パフォーマンスの最適化に関する積極的な意思決定、ワークロードの安定性の向上、CI/CD プロセスの合理化、リソースの効果的な活用につながります。

一般的なアンチパターン:

- 不完全なオブザーバビリティ: ワークロードのあらゆるレイヤーにオブザーバビリティを組み込むことを怠ると、死角が生じ、システムのパフォーマンスや動作に関する重要なインサイトが明らかにされない可能性があります。
- 断片化されたデータビュー: データが複数のツールやシステムに分散している場合、ワークロードの状態とパフォーマンスを包括的に把握することが困難になります。
- ユーザーから報告された問題: これは、テレメトリとビジネス KPI のモニタリングによる積極的な問題検出ができていないという兆候です。

このベストプラクティスを活用するメリット:

- 情報に基づいた意思決定: テレメトリとビジネス KPI から得られるインサイトを活用して、データ主導の意思決定を行うことができます。
- 運用効率の向上: データ主導型のリソース利用は、高いコスト効率をもたらします。

- ワークロードの安定性の向上: 問題をより迅速に検出して解決し、稼働時間を改善します。
- CI/CD プロセスの効率化: テレメトリデータから得られるインサイトにより、プロセスの改善と信頼性の高いコードの配信が容易になります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードのアプリケーションテレメトリを実装するには、Amazon CloudWatch および AWS X-Ray などの AWS サービスを使用します。Amazon CloudWatch が提供する包括的なモニタリングツールのスイートを使用して、AWS 内やオンプレミス環境のリソースやアプリケーションをモニタリングできます。メトリクスを収集、追跡、分析して、ログデータの統合とモニタリングを行い、リソースの変化に対応して、ワークロードがどのように運用されているかの詳細を明らかにします。これと連携して AWS X-Ray を使用すると、アプリケーションをトレース、分析、デバッグできるため、ワークロードの動作を詳しく把握できます。サービスマップ、レイテンシー分布、トレースタイムラインなどの機能を提供する AWS X-Ray を使用すると、ワークロードのパフォーマンスとパフォーマンスに影響を及ぼすボトルネックに関するインサイトが得られます。

実装手順

1. 収集するデータを特定する: ワークロードの状態、パフォーマンス、動作に関する重要なインサイトを提供する主要なメトリクス、ログ、トレースを確認します。
2. CloudWatch エージェントをデプロイする: CloudWatch エージェントを使用すると、ワークロードと基盤となるインフラストラクチャからシステムとアプリケーションのメトリクスとログを取得できます。CloudWatch エージェントを使用すると、OpenTelemetry や X-Ray トレースを収集して、X-Ray に送信することもできます。
3. ログとメトリクスの異常検出を実装する: CloudWatch Logs の異常検出と CloudWatch メトリクスの異常検出を使用して、アプリケーション動作における異常アクティビティを自動的に特定します。これらのツールは、機械学習アルゴリズムを使用して異常を検出して警告するのでモニタリング機能が強化され、潜在的な混乱やセキュリティ脅威への対応時間が短縮できます。これらの機能を設定すると、アプリケーションヘルスとセキュリティをプロアクティブに管理できます。
4. 機密ログデータを保護する: Amazon CloudWatch Logs のデータ保護を使用して、ログ内の機密情報をマスクします。この機能は、アクセス前に機密データを自動的に検出してマスキングするのでプライバシーとコンプライアンスの維持に役立ちます。データマスキングを実装して、個人を特定できる情報 (PII) などの機密情報を安全に処理し保護します。
5. ビジネス KPI を定義、モニタリングする: ビジネス成果に合わせたカスタムメトリクスを確立します。

6. AWS X-Ray を使用してアプリケーションを計測する: CloudWatch エージェントをデプロイするだけでなく、トレースデータを生成するためにアプリケーションを計測することが重要です。このプロセスにより、ワークロードの動作とパフォーマンスをさらに詳細に把握できます。
7. アプリケーション全体でデータ収集を標準化する: アプリケーション全体でデータ収集のプラクティスを標準化します。均一性を確立することで、データの関連付けと分析が容易になり、アプリケーションの動作を包括的に把握しやすくなります。
8. クロスアカウントオブザーバビリティを実装する: Amazon CloudWatch クロスアカウントオブザーバビリティを使用して、複数の AWS アカウントにわたってモニタリング効率を向上させます。この機能を使用すると、さまざまなアカウントのメトリクス、ログ、アラームを 1 つのビューに統合できます。これにより、組織の AWS 環境全体で特定された問題の管理が容易になり、対応時間を短縮できます。
9. データを分析して対応する: データ収集と正規化が完了したら、Amazon CloudWatch にメトリクスとログの分析、トレース分析に AWS X-Ray を使用します。このような分析を行うことで、ワークロードの状態、パフォーマンス、動作に関する重要なインサイトを入手し、意思決定プロセスの指針とすることができます。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- OPS04-BP01 ワークロード KPI の定義
- OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する
- OPS04-BP04 依存関係のテレメトリを実装する
- OPS04-BP05 トランザクショントレーサビリティを実装する

関連ドキュメント:

- AWS Observability Best Practices
- CloudWatch ユーザーガイド
- AWS X-Ray デベロッパーガイド
- 運用の可視性を高めるために分散システムに計測を実装する
- AWS Skill Builder - AWS Observability (Japanese) 日本語吹き替え版
- Amazon CloudWatch の最新情報

- AWS X-Ray の最新情報

関連動画:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2022 - Developing an observability strategy

関連する例:

- 1 つのオブザーバビリティワークショップ
- AWS ソリューションライブラリ: Amazon CloudWatch を使用したアプリケーションモニタリング

OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する

カスタマーエクスペリエンスやアプリケーション操作について詳細なインサイトを取得することは、非常に重要です。リアルユーザーモニタリング (RUM) と合成トランザクションは、この目的のための強力なツールとなります。RUM は、実際のユーザーの操作に関するデータを提供し、ユーザーの満足度を生で把握できます。一方、合成トランザクションはユーザーの操作のシミュレーションを行います。これにより、実際のユーザーに影響が及ぶ前に潜在的な問題を検出できます。

期待される成果: カスタマーエクスペリエンスの包括的な確認、積極的な問題の検出、ユーザー操作の最適化により、シームレスなデジタルエクスペリエンスを提供できます。

一般的なアンチパターン:

- リアルユーザーモニタリング (RUM) をしないアプリケーション:
  - 問題検出の遅延: RUM を行わない場合、ユーザーからの苦情があるまでパフォーマンスのボトルネックや問題に気付かない可能性があります。このような事後対応型のアプローチは、お客様の不満につながる可能性があります。
  - ユーザーエクスペリエンスに関するインサイトの欠如: RUM を採用しない場合、実際のユーザーがアプリケーションをどのように操作したかを示す重要なデータが得られず、ユーザーエクスペリエンスの最適化の面で限界があります。
- 合成トランザクションを行わないアプリケーション:
  - 細部を見逃すケース: 合成トランザクションは、一般的なユーザーには頻繁に使用されてない可能性があるにしろ、特定の業務部門にとっては重要であるパスや機能のテストに役立ちます。合成トランザクションを行わないと、このようなパスが誤動作しても検出されない場合があります。

- アプリケーション非使用時の問題の確認: 定期的に合成テストを実行して、実際のユーザーがアプリケーションを積極的に操作していない時間のシミュレーションを行うことで、システムが常に適正に機能することを確認できます。

このベストプラクティスを活用するメリット:

- 積極的な問題検出: 実際のユーザーに影響が及ぶ前に、潜在的な問題を特定して対処できます。
- ユーザーエクスペリエンスの最適化: RUM からの継続的なフィードバックを利用すると、ユーザーエクスペリエンス全体の改善と向上につながります。
- デバイスとブラウザのパフォーマンスに関するインサイト: アプリケーションがさまざまなデバイスやブラウザでどのように動作するかを把握して、さらなる最適化を実現します。
- 検証済みのビジネスワークフロー: 定期的な合成トランザクションにより、コア機能とクリティカルパスの運用と効率性を確実に維持できます。
- アプリケーションのパフォーマンスの強化: 実際のユーザーデータから収集したインサイトを活用して、アプリケーションの応答性と信頼性を向上できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS では RUM および合成トランザクションをユーザーアクティビティテレメトリに活用するため、Amazon CloudWatch RUM や Amazon CloudWatch Synthetics などのサービスを提供します。メトリクス、ログ、トレースをユーザーアクティビティデータと組み合わせることで、アプリケーションの動作のステータスとユーザーエクスペリエンスの両方を包括的に把握できます。

実装手順

1. Amazon CloudWatch RUM をデプロイする: アプリケーションを CloudWatch RUM と統合して、実際のユーザーデータを収集、分析、提示します。
   a. CloudWatch RUM JavaScript ライブラリを使用して、RUM をアプリケーションと統合します。
   b. ダッシュボードを設定して、実際のユーザーデータを可視化してモニタリングします。
2. CloudWatch Synthetics を設定する: ユーザーのアプリケーション操作をシミュレートする Canary、つまりスクリプト化したルーチンを作成します。
   a. 重要なアプリケーションのワークフローとパスを定義します。

   b. CloudWatch Synthetics スクリプトで Canary を設計し、重要なパスのユーザーインタラクションをシミュレートします。
   c. Canary を指定した間隔で実行するようにスケジュールを設定してモニタリングを実行し、着実にパフォーマンスチェックを実行します。
3. データの分析と対処を実施する: RUM と合成トランザクションからのデータを活用してインサイトを取得し、異常が検出された場合は是正措置を講じます。CloudWatch ダッシュボードとアラームを使用して常に情報を入手します。

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:

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する
- OPS04-BP04 依存関係のテレメトリを実装する
- OPS04-BP05 分散トレースを実装する

関連ドキュメント:

- CloudWatch RUM
- 合成モニタリング (canary)

関連動画:

- Optimize applications through end user insights with Amazon CloudWatch RUM
- AWS On Air ft. Real-User Monitoring for Amazon CloudWatch

関連する例:

- 1 つのオブザーバビリティワークショップ
- Git Repository for Amazon CloudWatch RUM Web Client
- Using Amazon CloudWatch Synthetics to measure page load time

OPS04-BP04 依存関係のテレメトリを実装する

依存関係のテレメトリは、ワークロードが依存する外部サービスやコンポーネントのヘルスとパフォーマンスをモニタリングするうえで不可欠です。依存関係のテレメトリにより、DNS、データベース、サードパーティー API などの依存関係に関連する到達可能性、タイムアウト、その他の重要なイベントに関する貴重なインサイトが得られます。このような依存関係に関するメトリクス、ログ、トレースを出力するようにアプリケーションをインストルメント化することで、ワークロードに影響を及ぼす可能性のある潜在的なボトルネック、パフォーマンスの問題、または障害をより明確に把握できます。

期待される成果: ワークロードを支える依存関係が期待どおりに機能し、積極的に問題に対処して、最適なワークロードパフォーマンスを確保できます。

一般的なアンチパターン:

- 外部依存の見落し: 内部アプリケーションメトリクスのみを重視し、外部の依存関係に関連するメトリクスはおろそかにしています。
- 積極的なモニタリングの不履行: 依存関係のヘルスとパフォーマンスを継続的にモニタリングするのではなく、問題が発生するまで待機しています。
- サイロ化したモニタリング: 複数の異なるモニタリングツールを使用することにより、依存関係のヘルスについての断片的かつ一貫性のないビューの生成につながっている場合があります。

このベストプラクティスを活用するメリット:

- ワークロードの信頼性の向上: 外部依存を常に利用可能にして最適なパフォーマンスを発揮できるようにすることで実現できます。
- 問題の検出と解決の迅速化: ワークロードに影響が及ぶ前に、依存関係のある問題を事前に特定して対処できます。
- 包括的なビュー: ワークロードのヘルスに影響を及ぼす内部コンポーネントと外部コンポーネントの両方を全体的に把握できます。
- ワークロードのスケーラビリティ強化: 外部依存のスケーラビリティの限界とパフォーマンス特性を把握することにより実現できます。

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

ワークロードが依存しているサービス、インフラストラクチャ、プロセスを特定することから始めて、依存関係のテレメトリを実装します。これらの依存関係が期待どおりに機能している場合の良好な状態を定量化して、測定するためにどのようなデータが必要かを判断します。その情報を使用して、依存関係のヘルスに関するインサイトを運用チームに提供するダッシュボードとアラートを作成できます。AWS ツールを使用して、依存関係が必要となるとおり機能しない場合の影響を検出して定量化します。優先事項、目標、取得したインサイトの変化に応じて、戦略を継続的に見直します。

実装手順

依存関係のテレメトリを効果的に実装する方法:

1. 外部依存関係を特定する: 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、サードパーティー API、その他の環境へのネットワーク接続ルート、DNS サービスなどのサービスが含まれます。効果的な依存関係のテレメトリを得る第一歩は、依存関係を包括的に把握することです。
2. モニタリング戦略を策定する: 外部依存を明確に把握した後、それに応じたモニタリング戦略を構築します。これには、各依存関係の重要度、予想される動作、関連するサービスレベルアグリーメントまたは目標 (SLA または SLT) を把握することなどがあります。ステータスの変化やパフォーマンスの逸脱を通知する積極的なアラートを設定します。
3. ネットワークモニタリングを使用する: Internet Monitor および Network Monitor を使用して、グローバルなインターネットとネットワークの状態を包括的に把握します。これらのツールは、外部の依存関係に影響を与える機能停止、中断、またはパフォーマンスの低下を把握し、それに対応するために役立ちます。
4. AWS Health Dashboard で状況を把握する: AWS でサービスに影響を及ぼす可能性のあるイベントが発生した場合にアラートと修正ガイダンスが得られます。
    a. Amazon EventBridge ルールを使用して AWS Health イベントをモニタリングします。または、プログラムで AWS Health API と統合して、AWS Health イベントを受信したときのアクションを自動化します。これらのアクションには、計画されたすべてのライフサイクルイベントメッセージをチャットインターフェイスに送信するなどの一般的なアクションや、IT サービス管理ツールでのワークフローの開始などの特定のアクションがあります。
    b. AWS Organizations を使用する場合、アカウント間で AWS Health イベントを集約します。
5. AWS X-Ray でアプリケーションを計測する: AWS X-Ray を使用すると、アプリケーションとアプリケーションの基盤となる依存関係のパフォーマンスに関するインサイトが得られます。リクエストを開始から終了までトレースすることにより、アプリケーションが依存している外部サービスや外部コンポーネントのボトルネックや障害を特定できます。

6. Amazon DevOps Guru を使用する: この機械学習ベースのサービスは、運用上の問題を特定し、重大な問題が発生する可能性のあるタイミングを予測して、実行すべき具体的な対応措置を推奨します。依存関係のインサイトを得て、その関係性が運用上の問題の原因ではないことを突き止めるために、非常に有益です。
7. 定期的にモニタリングする: 外部依存に関するメトリクスとログを継続的にモニタリングします。予期しない動作やパフォーマンスの低下についてのアラートを設定します。
8. 変更後の検証をする: 外部依存のいずれかが更新されたり変更されたりした場合は、そのパフォーマンスを検証して、アプリケーションの要件との整合性を確認します。

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:

- OPS04-BP01 ワークロード KPI の定義
- OPS04-BP02 アプリケーションテレメトリを実装する
- OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する
- OPS04-BP05 トランザクショントレーサビリティを実装する
- OPS08-BP04 実践的なアラートを作成する

関連ドキュメント:

- Amazon Personal AWS Health Dashboard ユーザーガイド
- AWS Internet Monitor ユーザーガイド
- AWS X-Ray デベロッパーガイド
- AWS DevOps Guru ユーザーガイド

関連動画:

- Visibility into how internet issues impact app performance
- Introduction to Amazon DevOps Guru
- Manage resource lifecycle events at scale with AWS Health

関連する例:

- Amazon DevOps Guru を使用して AIOps による運用上の洞察を得る
- AWS Health Aware
- Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale

OPS04-BP05 分散トレースを実装する

分散トレースを使用すると、分散システムのさまざまなコンポーネントを通過するリクエストをモニタリングし、可視化できます。複数のソースからトレースデータを収集して統合されたビューで分析することで、チームはリクエストの流れ、ボトルネックが発生している場所、重点的に最適化に取り組むべき個所をより正確に把握できます。

期待される成果: 分散システムを通過するリクエストを包括的に把握できるため、正確なデバッグ、パフォーマンス最適化、ユーザーエクスペリエンスの向上が実現します。

一般的なアンチパターン:

- 一貫性に欠けた計測: 分散システム内のすべてのサービスがトレースを目標に計測されているわけではない。
- レイテンシーの考慮なし: エラーのみに注目し、レイテンシーや徐々にパフォーマンスが低下していることが考慮されていない。

このベストプラクティスを活用するメリット:

- 包括的なシステムの全体像: リクエストの入力から終了まで、リクエストのパス全体にわたり可視化できます。
- デバッグの強化: 障害やパフォーマンスの問題が発生した個所を迅速に特定できます。
- ユーザーエクスペリエンスの向上: モニタリングを行い、実際のユーザーデータに基づいて最適化を行うことで、確実にシステムが実際の需要を満たせます。

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: 高

実装のガイダンス

計測が必要となるすべてのワークロードの要素を特定することから始めます。すべてのコンポーネントを把握したら、AWS X-Ray や OpenTelemetry などのツールを活用してトレースデータを収集し、X-Ray や Amazon CloudWatch ServiceLens Map などのツールを使用して分析を行います。デベロッパーとのレビューを定期的に実施し、Amazon DevOps Guru、X-Ray Analytics、X-Ray Insights

などのツールをサポートとして使用した議論により、より詳細な検出を行います。トレースデータからアラートを設定して、ワークロードのモニタリング計画で定義されている結果に対してリスクが検出された場合に通知します。

実装手順

分散トレースを効果的に実装する方法:

1. AWS X-Ray の採用: をアプリケーションに組み込むと、アプリケーションの動作に関するインサイトを取得したり、パフォーマンスを把握して、ボトルネックを特定したりできます。X-Ray Insights を自動トレース分析に活用します。
2. サービスを計測する: AWS Lambda 関数から EC2 インスタンスまですべてのサービスがトレースデータを送信していることを確認します。計測するサービスが多いほど、エンドツーエンドのビューが明確になります。
3. CloudWatch Real User Monitoring と Synthetic Monitoring を統合する: Real User Monitoring (RUM) と Synthetic Monitoring を X-Ray と統合します。これにより、実際のユーザーエクスペリエンスをキャプチャしてユーザーの操作をシミュレートし、潜在的な問題を特定できます。
4. CloudWatch エージェントを使用する: エージェントは X-Ray と OpenTelemetry のいずれかを使ってトレースを送信できるため、取得できるインサイトの奥行きがさらに深まります。
5. Amazon DevOps Guru を使用する: DevOps Guru は X-Ray、CloudWatch、AWS Config、AWS CloudTrail のデータを使用して実行可能なレコメンデーションを提供します。
6. トレースを分析する: トレースデータを定期的に確認して、アプリケーションのパフォーマンスに影響を及ぼす可能性のあるパターン、異常、またはボトルネックを特定します。
7. アラートを設定する: 異常なパターンや長時間のレイテンシー向けに CloudWatch でアラームを設定し、先を見越して問題に対処することを可能にします。
8. 継続的な改善: サービスが追加または変更されたら、関連するすべてのデータポイントが取得できるように、トレース戦略を再検討します。

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:

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する

- OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する
- OPS04-BP04 依存関係のテレメトリを実装する

関連ドキュメント:

- AWS X-Ray 開発者ガイド
- Amazon CloudWatch エージェントユーザーガイド
- Amazon DevOps Guru ユーザーガイド

関連動画:

- Use AWS X-Ray Insights
- AWS on Air ft. Observability: Amazon CloudWatch and AWS X-Ray

関連する例:

- AWS X-Ray 向けのアプリケーションのインストルメント化

OPS 5. どのようにして欠点を減らし、修正を簡単にし、本番環境へのフローを改善しますか?

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、バグ修正を有効にし、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導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 構成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 の多くのサービスは、バージョン管理機能を備えています。AWS CodeCommit などのリビジョンまたはソース管理システムを使用して、コードやその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) を管理しています。

期待される成果: チームが協力してコード作業に取り組みます。コードをマージすると、コードの一貫性が維持され、変更点が失われることはありません。エラーは、適正なバージョニングによって簡単に元に戻すことができます。

一般的なアンチパターン:

- コードを開発し、ワークステーションに保存したのに、その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、コードが失われる。
- 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、操作できなくなる。変更を元に戻すことができない。
- レポートファイルへの書き込みがロックされていて、別のユーザーが編集する必要があるとき、編集をしようとするユーザーは、ほかのユーザーに作業を停止するように求める。
- 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。誰かが誤って最終レポートを買い物リストで上書きして保存してしまう。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る。

このベストプラクティスを活用する利点: 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を活用し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。

リソース

関連するベストプラクティス:

- OPS05-BP04 構築およびデプロイ管理システムを使用する

関連ドキュメント:

- AWS CodeCommit とは

関連動画:

- AWS CodeCommitへの概論

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。デベロッパー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構築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ています。デベロッパーはテスト結果と検証にアクセスできます。組織には、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ります。

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 AWS 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ても単体テストを行わない。本稼働へのデプロイ時にソフトウェアが失敗する。

このベストプラクティスを活用する利点: ソフトウェアデプロイの変更の失敗率が減ります。ソフトウェアの品質が向上します。デベロッパーのコードの実行可能性に関する意識が向上します。確信を持ってセキュリティポリシーをロールアウトし、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を活用しない場合のリスクレベル: 高

実装のガイダンス

継続的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、デベロッパーが迅速にフィードバックを得られるように公開します。組織で、すべてのソフトウェア変更に適用されるテスト基準を施行します。

Amazon Q Developer の生成 AI の力を活用して、開発者の生産性とコード品質を高めます。Amazon Q Developer には、コード提案の生成 (大規模言語モデルに基づく)、ユニットテストの作成 (境界条件を含む)、セキュリティ脆弱性の検出と修復を通じたコードセキュリティの強化が含まれます。

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。デベロッパーは、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?

1. AWS Deployment Pipeline Reference Architecture 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、さまざまな種類のテストの、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. 生成 AI ツールである Amazon Q Developer は、ユニットテストケース (境界条件を含む) の作成、コードとコメントを使用した関数の生成、一般的なアルゴリズムの実装に役立ちます。

   b. Amazon CodeGuru Reviewer は、アプリケーションコードの欠陥を調べるときに使用します。

   c. AWS CodeBuild は、ソフトウェアアーティファクトをテストするときに使用します。

   d. AWS CodePipeline を使用すると、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
- 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-BP10 統合とデプロイを完全自動化する

関連ドキュメント:

- テスト駆動型の開発アプローチを採用
- Amazon Q でソフトウェア開発ライフサイクルを加速
- デベロッパーエクスペリエンスを再構想する新たな機能が搭載された Amazon Q Developer の一般提供開始
- IDE における Amazon Q Developer の使用に関する究極のチートシート
- テスト作成に AI を使用したシフトレフトワークロード
- Amazon Q デベロッパーセンター
- Amazon CodeWhisperer でアプリケーションをより速く構築する 10 の方法
- Amazon CodeWhisperer でコードカバレッジの先を見る

- Amazon CodeWhisperer を使ったプロンプトエンジニアリングのベストプラクティス
- TaskCat と CodePipeline を使用した自動 AWS CloudFormation テストパイプライン
- 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
- サーバーレスアプリケーションのテストの開始
- CI/CD パイプラインがリリースキャプテンに
- AWS での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:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- 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
- Testing Your Infrastructure as Code with AWS CDK

関連リソース:

- Amazon CodeWhisperer で生成 AI を使用してアプリケーションを構築する
- Amazon CodeWhisperer ワークショップ
- 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 Pipeline
- Policy as Code ワークショップ – テスト駆動型デプロイ
- AWS CodeBuild を使用して GitHub から Node.js アプリケーションのユニットテストを実施する
- インフラストラクチャコードのテスト駆動型開発に Serverspec を使用する

関連サービス:

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- AWS CodeBuild

- AWS CodePipeline

OPS05-BP03 構成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。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たりすることができます。

設定は、既知の一貫性のある状態でデプロイする必要があります。環境やリージョン全体でリソース設定を継続的にモニタリングするには、自動検査を使用する必要があります。これらの制御は、環境間でルールが一貫性をもって適用されるように、自動化されたコードおよび管理として定義する必要があります。設定の変更は、合意済みの変更管理手順に従って更新し、バージョン管理に則って、一貫性をもって適用する必要があります。アプリケーションの設定は、アプリケーションとインフラストラクチャコードとは無関係に管理する必要があります。そうすることで、複数の環境間で一貫性をもってデプロイできます。設定を変更しても、アプリケーションの再構築や再デプロイは行われません。

期待される成果: 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインの一部として設定、検証、デプロイを行います。モニタリングして、設定が正しいことを確認します。これにより、エンドユーザーや顧客への影響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。
- 検証をせずに CI/CD を使用して本番稼働前の設定を本番環境にプッシュします。ユーザーと顧客に正確でないデータやサービスを提供してしまいます。

このベストプラクティスを活用する利点: 構成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。構成管理システムを使用すると、ガバナンス、コンプライアンス、規制要件に関して保証が得られます。

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: 中

実装のガイダンス

構成管理システムは、アプリケーションと環境の設定変更を追跡して実装するために使用されます。構成管理システムは、手動プロセスを原因として発生するエラーを低減し、設定の変更を繰り返し可能かつ監査可能にして、労力を軽減するためにも使用されます。

AWS では、AWS Config を使用することで、アカウントおよびリージョンを横断して AWS リソース設定を継続的にモニタリングできます。それにより、設定履歴を追跡し、設定変更が他のリソースに与える影響を把握して、AWS Config ルール および AWS Config Conformance Packs を使用して、予測または期待される設定に照らしてそれらを監査することができます。

Amazon EC2 インスタンス、AWS Lambda、コンテナ、モバイルアプリケーション、IoT デバイスで実行されているアプリケーションの動的設定には、AWS AppConfig を使用することで、環境全体で設定、検証、デプロイ、モニタリングを行うことができます。

実装手順

1. 設定担当者を特定します。
   a. コンプライアンス、ガバナンス、または規制上のニーズを設定担当者に伝えます。
2. 設定項目と成果物を特定します。
   a. 設定項目とは、CI/CD パイプライン内のデプロイにより影響を受けるすべてのアプリケーション設定と環境設定です。
   b. 成果物には、達成基準、検証、モニタリング対象などがあります。
3. ビジネス要件とデリバリーパイプラインに基づいて、構成管理ツールを選択します。
4. 誤った構成による影響を最小限に抑えるために、構成を大幅に変更する場合は、カナリアデプロイなどの加重デプロイを検討します。
5. 構成管理を CI/CD パイプラインに統合します。
6. プッシュされたすべての変更を検証します。

リソース

関連するベストプラクティス:

- OPS06-BP01 変更の失敗に備える
- OPS06-BP02 デプロイをテストする
- OPS06-BP03 安全なデプロイ戦略を使用する
- OPS06-BP04 テストとロールバックを自動化する

関連ドキュメント:

- AWS Control Tower
- AWS Landing Zone Accelerator
- AWS Config
- AWS Config とは
- AWS AppConfig
- AWS CloudFormation とは
- AWS 開発者用ツール

関連動画:

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads
- 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
- Manage and Deploy Application Configurations with AWS AppConfig

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

期待される成果: 組織の構築およびデプロイ管理システムは、適正な設定で安全なロールアウトを自動化する機能を提供する、継続的インテグレーションと継続的デリバリー (CI/CD) システムをサポートします。

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に、実行可能ファイルを本稼働システムにコピーすると、起動に失敗する。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す。
- 開発環境でアプリケーションの新機能の構築を正常に完了し、品質保証 (QA) にコードを提供しても、静的アセットが欠如していたために、QA に合格しない。
- 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でき、これには、新しくコード化された機能も含まれるけれど、月曜日に、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ない。
- そこで、新しいリリース用に作成したテストを実行する。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する。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る。

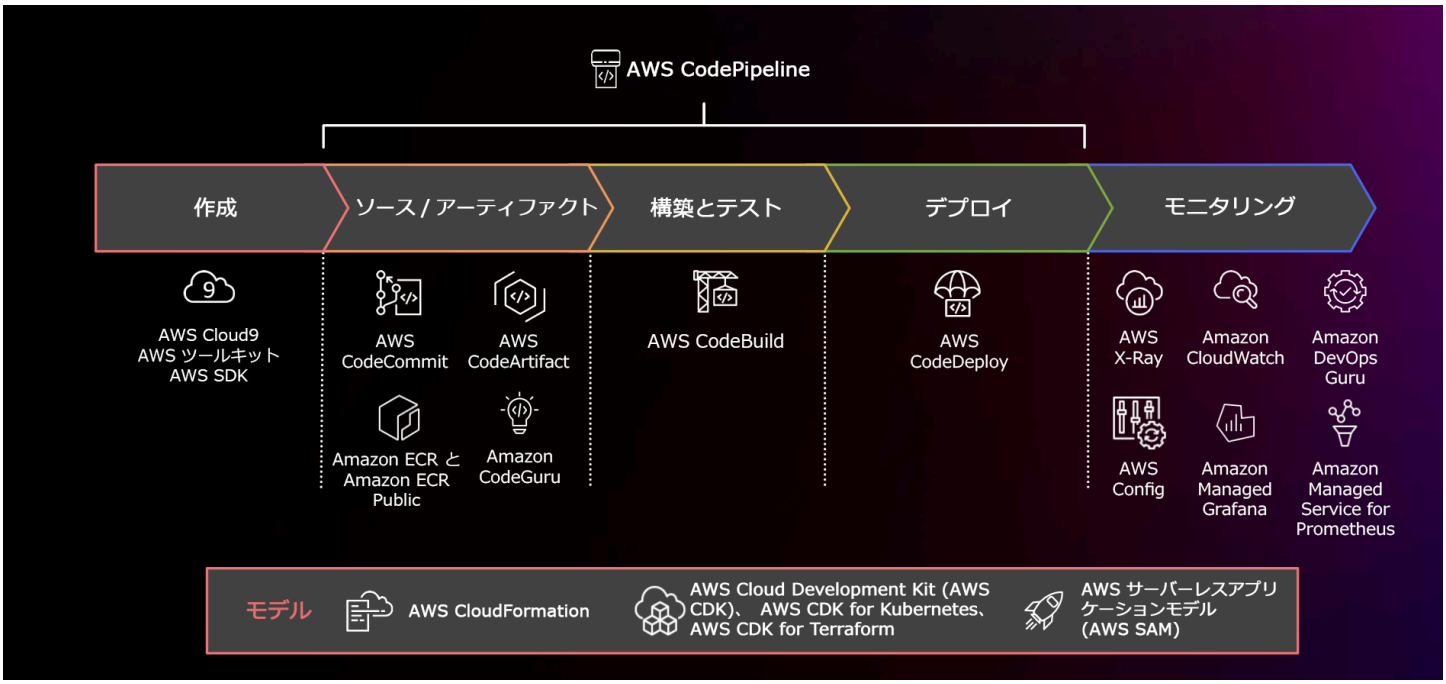
ベストプラクティスを活用する利点: ビルドとデプロイのアクティビティを管理するメカニズムを提供することで、反復的なタスクを実行するための労力の程度を減らし、チームメンバーは高価値のクリエイティブなタスクに専念し、手動の手順によるエラーの発生を抑制できます。

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: 中

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と、変更の追跡と実装、手動プロセスが原因で発生するエラーの削減、安全なデプロイに必要な労力の軽減につながります。コードのチェックインから、ビルド、テスト、デプロイ、検証を通じて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、リードタイム短縮、コスト低減、変更頻度の増加、労力の軽減、コラボレーションの強化につながります。

実装手順



AWS CodePipeline と関連サービスを使用する CI/CD を示した図

1. AWS CodeCommit を使用して、アセット (ドキュメント、ソースコード、バイナリファイルなど) のバージョン管理、保存、管理を行います。
2. CodeBuild を使用してソースコードをコンパイルし、ユニットテストを実行して、すぐにデプロイ可能なアーティファクトを作成します。
3. CodeDeploy は、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 AWS Lambda 関数、Amazon ECS へのアプリケーションのデプロイを自動化するデプロイメントサービスとして使用します。
4. 環境をモニタリングします。

リソース

関連するベストプラクティス:

- OPS06-BP04 テストとロールバックを自動化する

関連ドキュメント:

- AWS 開発者用ツール

- AWS CodeCommit とは
- What is AWS CodeBuild?
- AWS CodeBuild
- What is AWS CodeDeploy?

関連動画:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

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を低減し、スケールして、パッチに関連す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、利点とリスク管理の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。これが現実的でない場合のオプションには、パッチの適用があります。

Amazon EC2 Image Builder には、マシンイメージを更新するためのパイプラインがあります。パッチ管理の一環として、AMI イメージパイプラインを使用する Amazon マシンイメージ (AMI) または Docker イメージパイプラインを備えたコンテナイメージを検討します。一方、AWS Lambda には脆弱性を排除するためのカスタムランタイムと追加ライブラリのパターンが用意されています。

Linux 用 Amazon マシンイメージまたは Windows Server イメージの更新は、Amazon EC2 Image Builder を使用して管理します。既存のパイプラインで Amazon Elastic Container Registry (Amazon ECR) を使用することで、Amazon ECS イメージと Amazon EKS イメージを管理できます。Lambda にはバージョン管理機能があります。

パッチの本番環境のシステムへの適用は、まず安全な環境でテストした後とする必要があります。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 Patch Manager を使用して管理対象システムへのパッチ適用プロセスを自動化でき、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。

期待される成果: AMI とコンテナイメージにパッチが適用されて最新の状態であり、起動の準備が整っています。デプロイされたすべてのイメージのステータスを追跡し、パッチのコンプライアンス

を把握できます。現在のステータスを報告でき、コンプライアンスのニーズを満たすプロセスが施行されています。

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、デベロッパーに通知することなく、自動的にデベロッパー環境にパッチを適用します。あなたには、デベロッパーから、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。
- 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ない。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。
- 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が、パッチが適用されていないシステムには、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る。
- 緊急に修正が必要なゼロデイ脆弱性についての通知を受けて、すべての環境に手動でパッチを適用する必要がある。

このベストプラクティスを活用する利点: パッチ適用の基準や環境全体にわたる配布方法など、パッチ管理プロセスを確立することで、パッチレベルのスケールとレポート作成が実現します。これにより、セキュリティパッチの適用が保証され、実施されている既知の修正のステータスを明確に把握できます。これにより、必要な機能の導入、問題の迅速な解決、継続的なガバナンスへの遵守が実現します。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: 中

実装のガイダンス

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を自動化することで、パッチ適用の経過時間、手動プロセスが原因で発生するエラー、パッチに関する労力を低減できます。

実装手順

Amazon EC2 Image Builder の場合:

1. Amazon EC2 Image Builder を使用して、次のパイプラインの詳細を指定します。
   a. イメージパイプラインの作成と命名
   b. パイプラインのスケジュールとタイムゾーンの定義
   c. すべての依存関係の設定
2. 次のレシピを選択します。
   a. 既存のレシピを選択するか、新しいレシピを作成します
   b. イメージのタイプを選択します
   c. レシピに名前を付けてバージョンを付けます
   d. ベースイメージを選択します
   e. ビルドコンポーネントを追加して、ターゲットレジストリに追加します
3. オプション - インフラストラクチャの設定を定義します。
4. オプション - 設定を定義します。
5. 設定の確認
6. レシピのハイジーンを定期的に管理します。

Systems Manager Patch Manager の場合:

1. パッチベースラインの作成
2. パッチ適用オペレーションの方法を選択します。
3. コンプライアンスレポートとスキャンを有効にします。

リソース

関連するベストプラクティス:

- OPS06-BP04 テストとロールバックを自動化する

関連ドキュメント:

- What is Amazon EC2 Image Builder
- Create an image pipeline using the Amazon EC2 Image Builder
- Create a container image pipeline
- AWS Systems Manager Patch Manager
- Patch Managerの使用 (コンソール)
- パッチコンプライアンスレポートの使用
- AWS Developer Tools

関連動画:

- CI/CD for Serverless Applications on AWS
- Design with Ops in Mind

関連する例:

- Well-Architected Labs - インベントリとパッチの管理
- AWS Systems Manager Patch Manager のチュートリアル

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、それを最大化します。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応じて内容が更新されます。

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利点: 共有される標準を利用すると、ベストプラクティスの採用、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

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が台頭すると、それに応じて設計標準を最新の内容に維持するメカニズムを導入します。

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。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の責任を担う部門横断的なチームを特定します。このチームは、組織全体にわたる関係者と協力して、設計標準、業務手順、チェックリスト、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、それに応じて設計標準を最新の内容に維持するメカニズムを導入します。
3. 設計標準が一元的に施行されている場合は、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。

関連ドキュメント:

- Automate AWS Backups with AWS Service Catalog
- AWS Service Catalog を Account Factory で強化する
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog
- Maintain visibility over the use of cloud architecture patterns
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup

関連動画:

- AWS Service Catalog – Getting Started
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert

関連する例:

- AWS Service Catalog Reference Architecture
- 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ターは、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利点: 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。コード品質は、ペアプログラミング、コードレビュー、AI 生産性ツールの実装などのベストプラクティスの活用を促進します。

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

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。

Amazon Q Developer の生成 AI の力を活用して、開発者の生産性とコード品質を高めます。Amazon Q Developer には、コード提案の生成 (大規模言語モデルに基づく)、ユニットテストの作成 (境界条件を含む)、セキュリティ脆弱性の検出と修復を通じたコードセキュリティの強化が含まれます。

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 Q Developer を使用します。Amazon Q Developer は生成 AI ツールで、ユニットテストケース (境界条件を含む) の作成、コードやコメントを使った関数の生成、一般的なアルゴリズムの実装、コード内でのセキュリティポリシー違反や脆弱性の検出、シークレットの検出、Infrastructure as Code (IaC) のスキャン、コードの記録、サードパーティーのコードライブラリの迅速な学習などに役立ちます。
   b. 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。
   c. AWS Cloud9 と共有の開発環境を作成し、そこで共同でコードの開発に取り組むことができます。

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-BP02 変更をテストし、検証する
- OPS05-BP06 設計標準を共有する

関連ドキュメント:

- テスト駆動型の開発アプローチを採用
- Amazon Q でソフトウェア開発ライフサイクルを加速
- デベロッパーエクスペリエンスを再構想する新たな機能が搭載された Amazon Q Developer の一般提供開始

- IDE における Amazon Q Developer の使用に関する究極のチートシート
- テスト作成に AI を使用したシフトレフトワークロード
- Amazon Q デベロッパーセンター
- Amazon CodeWhisperer でアプリケーションをより速く構築する 10 の方法
- Amazon CodeWhisperer でコードカバレッジの先を見る
- Amazon CodeWhisperer を使ったプロンプトエンジニアリングのベストプラクティス
- Agile Software Guide
- CI/CD パイプラインがリリースキャプテンに
- Amazon CodeGuru Reviewer でのコードレビューの自動化
- テスト駆動型の開発アプローチを採用
- DevFactory による Amazon CodeGuru を使用したより良いアプリケーションの構築方法
- On Pair Programming
- 株式会社レンガにおける CodeGuru を使ったコードレビューの自動化
- The Art of Agile Development: Test-Driven Development
- コードレビューが重要である (かつ時間の節約になる) 理由

関連動画:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development

関連サービス:

- Amazon Q Developer
- 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。本番環境に近い環境ほど使用するコントロールレベルを増大し、デプロイ時にはワークロードを意図したとおりに運用できるという確信を得ます。

期待される成果: コンプライアンスとガバナンスのニーズを反映した環境が複数あります。本番環境への移行過程で、次の環境に移行する前にコードのテストを実施しています。

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、別のデベロッパー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いますが、専用環境を求める顧客のリクエストをサポートできません。
- テストは常に実行するとは限らず、テストを行う場合は本番環境でテストします。
- 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利点: デベロッパー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、本番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: 中

実装のガイダンス

複数の環境を使用して、実験を行える最小限のコントロールを備えたサンドボックス環境をデベロッパーに提供します。並行して作業が進められるように個別の開発環境を提供して、開発の俊敏性を高めます。デベロッパーがイノベーションを試せるように、本番環境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。テスト結果の有効性を向上させるためにロードテストを行う場合は、本番環境と同等の環境をデプロイします。

リソース

関連ドキュメント:

- AWS での Instance Scheduler
- What is AWS CloudFormation?

OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う

頻繁に、小規模で、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。変更管理システム、構成管理システム、ビルドおよび配信システムと組み合わせて使用して、頻繁かつ小規模で可逆的な変更を行うことは、変更の範囲と影響の低減につながります。これにより、トラブルシューティングの効果が向上し、変更をロールバックするオプションを使用すると、迅速に修復できるようになります。

一般的なアンチパターン:

- アプリケーションの新しいバージョンを変更期間を設けて四半期ごとにデプロイするが、変更期間中は、コアサービスがオフになる。
- 管理システムで変更を追跡せずに、データベーススキーマを頻繁に変更する。
- インプレースアップデートを手動で実行して、既存のインストールと設定を上書きし、明確なロールバック計画がない。

このベストプラクティスを活用するメリット: 小規模な変更を頻繁にデプロイすることで、開発作業はより迅速になります。変更が小規模である場合、意図しない結果が発生するかどうかの識別や元に戻す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。このような変更プロセスによりリスクが軽減され、変更が失敗した場合の影響も軽減されます。

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

変更の範囲と影響を低減するために、頻繁かつ小規模で可逆的な変更を行います。これにより、トラブルシューティングが容易になり、迅速に修復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。

リソース

関連するベストプラクティス:

- OPS05-BP03 構成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
- OPS06-BP04 テストとロールバックを自動化する

関連ドキュメント:

- AWS でのマイクロサービスの実装
- Microservices - Observability

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。

リソースタグと AWS Resource Groups を使用して一貫したタグ付け戦略に従ったメタデータを適用して、リソースの識別を達成します。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て、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。

期待される成果: デベロッパーはツールを使用してコードを提供し、本番環境に移行できます。デベロッパーは AWS Management Consoleにログインする必要なく、更新を提供できます。変更と設定についての完全な監査証跡があるため、ガバナンスとコンプライアンスのニーズを満たせます。プロセスは反復可能であり、複数チーム間で標準化できます。デベロッパーは開発とコードのプッシュに注力する時間ができるため、生産性が向上します。

一般的なアンチパターン:

- 金曜日に、機能ブランチ用の新しいコードの作成を完了します。月曜日になって、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に向けてコードをチェックインします。

- 本番環境の多数のお客様に影響を及ぼす重要な問題の修正のコーディング作業を担当することになります。この修正をテストした後、コードと E メールの変更管理をコミットして、本番環境でのデプロイに向けて承認をリクエストします。
- デベロッパーは、AWS Management Consoleにログインして、標準以外の方法やシステムを使用して新しい開発環境を作成します。

このベストプラクティスを活用するメリット: 自動化された構築およびデプロイ管理システムを実装することで、手動プロセスが原因で発生するエラーを削減し、変更をデプロイする労力を低減して、チームメンバーがビジネス価値の実現に注力できるようにします。本番環境への移行の提供が高速化します。

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

構築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、手動プロセスが原因で発生するエラーと労力を低減できます。コードのチェックインから、ビルド、テスト、デプロイ、検証を通じて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、リードタイムが短縮され、変更頻度が増加し、労力が軽減され、市場投入までの時間が短縮され、生産性が向上し、本番環境に移行する際のコードのセキュリティが強化されます。

リソース

関連するベストプラクティス:

- OPS05-BP03 構成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

関連ドキュメント:

- AWS CodeBuild とは
- What is AWS CodeDeploy?

関連動画:

- AWS re\:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS 6. どのようにデプロイメントリスクを軽減するのですか?

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な復旧を達成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。

ベストプラクティス

- OPS06-BP01 変更の失敗に備える
- OPS06-BP02 デプロイをテストする
- OPS06-BP03 安全なデプロイ戦略を使用する
- OPS06-BP04 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

デプロイが望ましくない結果をもたらした場合に、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。このような計画を確立するためのポリシーを用意しておくと、すべてのチームが変更の失敗から復旧する戦略を策定するうえで役立ちます。戦略の例として、デプロイとロールバック手順、ポリシーの変更、機能フラグ、トラフィックの分離、トラフィックシフトなどがあります。1 つのリリースに、関連するコンポーネントの変更が複数含まれる場合があります。この戦略は、コンポーネントの変更が失敗しても耐えうる、または復旧できる機能を備えている必要があります。

期待される成果: 変更が失敗した場合に備えて、変更に関する詳細な復旧計画を用意しています。さらに、他のワークロードコンポーネントへの潜在的な影響を最小限に抑えるために、リリースのサイズを縮小します。その結果、変更の失敗によって発生する可能性のあるダウンタイムが短縮され、復旧時間の柔軟性と効率性が向上し、ビジネスへの影響を軽減できます。

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。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、いずれにしても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。
- ルーチンを変更すると、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。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。

- システムが、より小さなリリースで更新できるように設計されていません。その結果、デプロイが失敗した際に、これらの一括変更を取り消すことが困難になります。
- Infrastructure as Code (IaC) を使用せず、インフラストラクチャを手動で更新してきた結果、望ましくない構成が生じます。手動変更を効果的に追跡して元に戻すことができません。
- デプロイ頻度の増加については測定していないため、チームには変更の規模を縮小したり、変更のたびにロールバック計画を改善したりする動機付けがなされておらず、リスクも失敗率が高まることになります。
- 変更の失敗によるシステム停止の合計時間を測定していないため、チームは、デプロイプロセスや復旧計画の効果を優先順位付けして改善することができません。

このベストプラクティスを活用するメリット: 変更の失敗からの復旧計画を立てることで、平均復旧時間 (MTTR) を最小限に抑え、ビジネスへの影響を軽減できます。

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

リリースチームが一貫性のある文書化されたポリシーとプラクティスを採用することで、組織は変更が失敗した場合の対策を計画できます。このポリシーでは、特定の状況でフィックスフォワードが許可される必要があります。いずれの場合も、変更を元に戻すためにかかる時間が最小限になるよう、本番環境へのデプロイ前にフィックスフォワードまたはロールバックの計画を適切に文書化して、十分なテストを行う必要があります。

実装手順

1. 特定の期間内に変更を元に戻すための効果的な計画を立てることをチームに要求するポリシーを文書化します。
   a. ポリシーには、フィックスフォワードが許可される状況を明記します。
   b. 関係者全員が文書化されたロールバック計画にアクセスできることを必須とします。
   c. ロールバックの要件 (許可されない変更がデプロイされたことが判明した場合など) を指定します。
2. ワークロードの各コンポーネントに関連するすべての変更の影響レベルを分析します。
   a. 反復可能な変更が変更のポリシーを実行する一貫したワークフローに従っていれば、こうした変更の標準化、テンプレート化、事前承認が許可されるようにします。
   b. 変更の規模を小さくすることで、変更による潜在的な影響を軽減し、復旧にかかる時間を短縮し、ビジネスへの影響を軽減します。

   c. 可能な限りインシデントを回避するために、ロールバック手順によってコードが確実に既知の良好な状態に戻るようにします。
3. ツールとワークフローを統合して、プログラムによってポリシーを適用します。
4. 変更に関するデータを他のワークロードオーナーにも見えるようにすることで、ロールバックができない変更の失敗の診断を迅速に行えるようにします。
   a. 目に見える変更データを使用することで、このプラクティスの成功を測定し、反復的な改善点を特定します。
5. モニタリングツールを使用してデプロイの成功または失敗を検証し、ロールバックに関する意思決定を加速します。
6. 変更の失敗時のシステム停止時間を測定して、復旧計画を継続的に改善します。

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:

- OPS06-BP04 テストとロールバックを自動化する

関連ドキュメント:

- AWS Builders Library | デプロイ時におけるロールバックの安全性の確保
- AWS ホワイトペーパー | Change Management in the Cloud

関連動画:

- re:Invent 2019 | Amazon の高可用性デプロイへのアプローチ

OPS06-BP02 デプロイをテストする

本番環境と同じデプロイ設定、セキュリティ管理、手順、プロシージャを使用して、本番稼働前にリリース手順をテストします。ファイル、設定、サービスの検査など、デプロイされたすべての手順が期待どおりに完了することを確認します。さらに、機能テスト、統合テスト、負荷テストによってすべての変更をテストして、ヘルスチェックなどのモニタリングも行います。これらのテストを行うことで、デプロイの問題を早期に特定し、本番稼働前に計画を立てて問題を軽減するよう対応できます。

すべての変更をテストするための一時的な並列環境を作成できます。Infrastructure as Code (IaC) を使用してテスト環境のデプロイを自動化することで、必要な作業量を減らし、安定性と一貫性を確保すると共に、より迅速に機能を提供できます。

期待される成果: 組織が、デプロイのテストを含むテスト駆動開発文化を採用します。これにより、チームはリリースの管理ではなくビジネス価値の提供に集中できます。チームはデプロイのリスクを早期に特定し、適切な緩和策を決定できます。

一般的なアンチパターン:

- 未テストのデプロイで、トラブルシューティングとエスカレーションを必要とする問題が頻発します。
- リリースには、既存のリソースを更新する Infrastructure as Code (IaC) が含まれています。IaC が正常に実行されるか、またはリソースに影響を及ぼすか確信がありません。
- あなたは、新しい機能をアプリケーションにデプロイします。しかし、意図したとおりに機能せず、影響を受けたユーザーからの報告を受けるまで問題を認識できません。
- あなたは、証明書を更新します。証明書を間違ったコンポーネントにインストールしてしまいますが、検出はされないままです。そのため、ウェブサイトへの安全な接続が確立されず、ウェブサイトの訪問者に影響が及びます。

このベストプラクティスを活用するメリット: デプロイ手順とデプロイによって生じる変更を本番稼働前に十分にテストすることで、デプロイ手順による本番環境への潜在的な影響を最小限に抑えることができます。これにより、変更の提供を遅らせることなく、本番リリースでの自信が高まり、運用サポートが最小限に抑えられます。

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

デプロイプロセスをテストすることは、デプロイによって生じる変更をテストすることと同じくらい重要です。そのためには、本番環境にできるだけ近い本番稼働前の環境でデプロイ手順をテストします。その結果、不完全または不正確なデプロイ手順、または設定ミスなどの一般的な問題を、本番リリース前に検出できます。さらに、復旧手順をテストすることもできます。

お客様事例

AnyCompany Retail は、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインの一環として、インフラストラクチャとソフトウェアの更新を顧客にリリースするために必要な定義済みの手順を本番環境に似た環境で実行します。このパイプラインは、デプロイ前にリソースのドリフト

を検出する (IaC 外で実行されたリソースへの変更を検出する) 事前チェックと、IaC の開始時に実行されるアクションの検証で構成されます。ロードバランサーに再登録する前に、特定のファイルや設定が整っていること、サービスが実行中の状態にあって、ローカルホストでのヘルスチェックに正しく応答していることを確認するなど、デプロイ手順が検証されます。さらに、すべての変更は、機能テスト、セキュリティテスト、リグレッションテスト、統合テスト、負荷テストなど、多くの自動テストにフラグを立てます。

実装手順

1. インストール前のチェックを実行して、本番環境をミラーリングした本番稼働前の環境を設定します。
   a. ドリフト検出を使用して、AWS CloudFormation 外でリソースが変更された場合に検出します。
   b. 変更セットを使用して、スタック更新の意図が、変更セットの開始時に AWS CloudFormation が実行するアクションと一致することを検証します。
2. これにより、AWS CodePipeline の手動承認ステップがトリガーされ、本番稼働前の環境へのデプロイが認可されます。
3. AWS CodeDeploy AppSpec ファイルなどのデプロイ設定を使用して、デプロイおよび検証ステップを定義します。
4. 該当する場合は、AWS CodeDeploy を他の AWS サービスと統合するか、AWS CodeDeploy をパートナー製品およびサービスと統合します。
5. Amazon CloudWatch、AWS CloudTrail、Amazon SNS イベント通知を使用してデプロイをモニタリングします。
6. 機能テスト、セキュリティテスト、リグレッションテスト、統合テスト、負荷テストなど、デプロイ後の自動テストを実行します。
7. デプロイ問題をトラブルシューティングします。
8. 上記の手順の検証が成功すると、本番環境へのデプロイを承認するための手動承認ワークフローが開始されます。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- OPS05-BP02 変更をテストし、検証する

関連ドキュメント:

- AWS Builders' Library | 安全なハンズオフデプロイメントの自動化 | デプロイテスト
- AWS ホワイトペーパー | Practicing Continuous Integration and Continuous Delivery on AWS
- The Story of Apollo - Amazon's Deployment Engine
- 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
- Integrating Network Connectivity Testing with Infrastructure Deployment

関連動画:

- re:Invent 2020 | Testing software and systems at Amazon

関連する例:

- Tutorial | Deploy and Amazon ECS service with a validation test

OPS06-BP03 安全なデプロイ戦略を使用する

安全な本番環境のロールアウトでは、変化による顧客への影響を最小限に抑えることを目的として、有益な変化の流れを管理します。安全管理は、期待される結果を検証し、変更やデプロイの失敗によって生じた不具合による影響の範囲を制限するための検査メカニズムを提供します。安全なロールアウトには、機能フラグ、ワンボックス、ローリング (canary リリース)、イミュータブル、トラフィック分割、ブルー/グリーンデプロイなどの戦略が含まれる場合があります。

期待される成果: 組織は、安全なロールアウトを自動化する機能を備えた継続的インテグレーションと継続的デリバリー (CI/CD) システムを使用します。チームは適切な安全なロールアウト戦略を使用する必要があります。

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。その結果、すべての顧客に一斉に影響が及びます。
- 全システムへの同時デプロイで生じた不具合により、緊急リリースが必要となります。すべての顧客への影響を修正するには数日かかります。
- 本番リリースを管理するために、複数のチームの計画と参加が必要です。これにより、顧客のために頻繁に機能を更新する能力が制限されます。

- 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。変更の失敗が判明した後、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ことを強いられ、これにより復旧にかかる時間が長くなります。

このベストプラクティスを活用するメリット: 自動デプロイにより、ロールアウトの速度と、顧客に一貫して有益な変更を提供することのバランスを取ることができます。影響を制限することで、コストのかかるデプロイの失敗を防ぎ、チームが失敗に効率的に対応する能力を最大限に高めることができます。

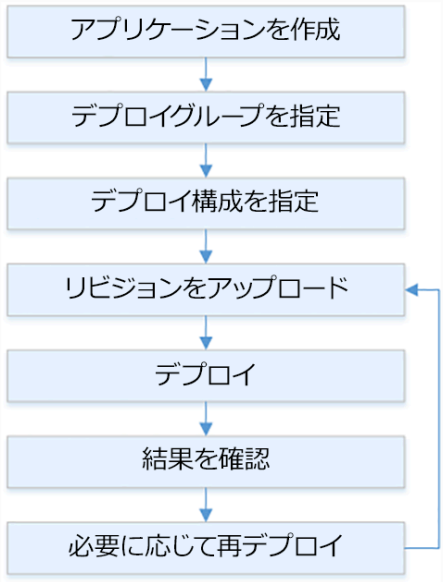
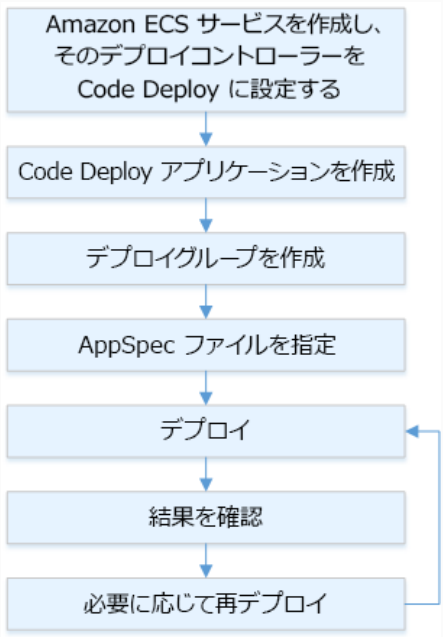
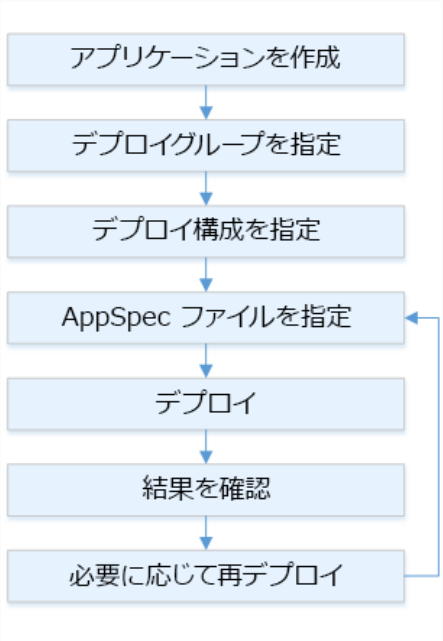
このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

継続的デリバリーの失敗は、サービス可用性の低下と、カスタマーエクスペリエンスの低下につながる可能性があります。デプロイの成功率を最大化するには、デプロイの失敗ゼロを目標に、エンドツーエンドのリリースプロセスに安全管理を実装してデプロイエラーを最小限に抑えます。

お客様事例

AnyCompany Retail は、ダウンタイムを最小限またはゼロにすることを目指しています。これは、デプロイ中に認識されるユーザーへの影響がまったくないことを意味します。これを実現するために、同社はローリングデプロイやブルー/グリーンデプロイなどのデプロイパターン (次のワークフロー図を参照) を確立しました。すべてのチームが、CI/CD パイプラインでこれらのパターンを 1 つ以上採用しています。

Amazon EC2 の CodeDeploy ワークフロー	Amazon ECS の CodeDeploy ワークフロー	Lambda の CodeDeploy ワークフロー
		

実装手順

1. 承認ワークフローを使用して、本番環境に移行する際に、一連の本番環境のロールアウト手順を開始します。
2. AWS CodeDeploy などの自動デプロイシステムを使用します。AWS CodeDeploy デプロイオプションには、EC2/オンプレミス向けのインプレースデプロイと、EC2/オンプレミス向けのブルー/グリーンデプロイ、AWS Lambda、Amazon ECS が含まれています (上のワークフロー図を参照)。
   a. 該当する場合は、AWS CodeDeploy を他の AWS サービスと統合するか、AWS CodeDeploy をパートナー製品およびサービスと統合します。
3. Amazon Aurora や Amazon RDS などのデータベースには、ブルー/グリーンデプロイを使用します。
4. Amazon CloudWatch、AWS CloudTrail、Amazon Simple Notification Service (Amazon SNS) イベント通知を使用してデプロイをモニタリングします。
5. 機能テスト、セキュリティテスト、リグレッションテスト、統合テスト、負荷テストといった、デプロイ後の自動テストを実行します。
6. デプロイ問題をトラブルシューティングします。

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:

- OPS05-BP02 変更をテストし、検証する
- 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

関連ドキュメント:

- AWS Builders' Library | 安全なハンズオフデプロイメントの自動化 | 本番デプロイメント
- AWS Builders Library | CI/CD パイプラインがリリースキャプテンに | 安全かつ自動の本番リリース
- AWS ホワイトペーパー | AWS での継続的インテグレーションと継続的デリバリーの実践 | デプロイ方法
- AWS CodeDeploy ユーザーガイド
- AWS CodeDeploy でのデプロイ設定の使用
- API Gateway の Canary リリースデプロイの設定
- Amazon ECS デプロイタイプ
- 新機能 – Amazon Aurora と Amazon RDS でのフルマネージド型 Blue/Green Deployments
- AWS Elastic Beanstalk を使用したブルー/グリーンデプロイ

関連動画:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

関連する例:

- AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す
- Workshop | Building CI/CD pipelines for Lambda canary deployments using AWS CDK
- ワークショップ | EKS と ECS のブルー/グリーンデプロイとカナリアデプロイ
- ワークショップ | クロスアカウント CI/CD パイプラインの構築

OPS06-BP04 テストとロールバックを自動化する

デプロイプロセスの速度、信頼性、自信を高めるには、本番稼働前環境と本番環境でテストとロールバック機能を自動化する戦略を立てます。本番環境にデプロイする際のテストを自動化して、デプロイされる変更を検証する人間とシステムの操作をシミュレートします。ロールバックを自動化して、迅速に以前の既知の正常な状態に戻します。ロールバックは、変更によって望ましい結果が得られなかった場合や、自動テストが失敗した場合など、事前に定義された条件に基づいて自動的に開始する必要があります。これら 2 つのアクティビティを自動化することで、デプロイの成功率が向上し、復旧時間を最小限に抑え、ビジネスへの潜在的な影響を軽減できます。

期待される成果: 自動テストとロールバック戦略は、継続的インテグレーション、継続的デリバリー (CI/CD) パイプラインに統合されます。モニタリングによって、成功基準に照らして検証を行い、失敗の発生時に自動ロールバックを開始できます。これにより、エンドユーザーや顧客への影響を最小限に抑えることができます。例えば、すべてのテスト結果が期待を満たす場合は、同じテストケースを活用して、自動リグレッションテストが開始される本番環境にコードを昇格させます。リグレッションテストの結果が期待を満たさない場合、パイプラインワークフローで自動ロールバックが開始されます。

一般的なアンチパターン:

- システムが、より小さなリリースで更新できるように設計されていません。その結果、デプロイが失敗した際に、これらの一括変更を取り消すことが困難になります。
- デプロイプロセスが一連の手動のステップで構成されています。ワークロードに変更をデプロイした後に、デプロイ後のテストを開始します。テスト後、ワークロードが操作できず、顧客の接続が切断されたことに気付きます。あなたはその後、以前のバージョンへのロールバックを開始します。こうした手動の手順すべてが、システム復旧を遅らせるだけでなく、顧客への影響も長引く原因となります。
- アプリケーションで使用頻度の低い機能に対する自動テストケースを時間をかけて構築したことで、自動テスト機能の投資利益率が最小化されています。
- リリースには、アプリケーション、インフラストラクチャ、パッチ、および設定の更新が含まれ、これらは互いに独立しています。ただし、単一の CI/CD パイプラインを使用して、すべての変更を一度に提供しています。1 つのコンポーネントで失敗が発生すると、すべての変更を元に戻すことを強いられることになり、ロールバックが複雑で非効率なものになります。
- あなたのチームは、スプリント 1 でコード作業を完了し、スプリント 2 の作業を開始しますが、計画にはスプリント 3 まではテストが含まれていません。その結果、自動テストによって、スプリント 2 の成果物のテストを開始する前に解決が必要な障害がスプリント 1 で検出されたため、リリース全体が遅れ、あなたの自動テストの評価が下がってしまいます。

- 本番リリースに対する自動リグレッションテストケースは完了していますが、ワークロードの状態はモニタリングしていません。サービスが再起動したかどうかを確認できないため、あなたはロールバックが必要なのか、ロールバックが実行済みなのかがわかりません。

このベストプラクティスを活用するメリット: 自動テストにより、テストプロセスの透明性が高まり、さらに短い期間でより多くの機能をカバーできるようになります。本番環境での変更をテストして検証することで、即座に問題を特定できます。自動テストツールとの整合性が向上すると、不具合の検出も向上します。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。自動ロールバックによってビジネスへの影響が軽減し、デプロイ機能の信頼性が高まります。全体的に、これらの機能によって品質を確保しながら納期を短縮できます。

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: 中

実装のガイダンス

デプロイした環境のテストを自動化し、望ましい結果をよりすばやく確認します。事前に定義された結果が達成されない場合に以前の既知の正常な状態に自動的にロールバックすることで、復旧時間を最小限に抑えると共に、手動プロセスによるエラーを減らします。テストツールをパイプラインワークフローと統合することで、一貫したテストを行い、手動入力を最小限に抑えます。最大のリスクを軽減し、変更のたびに頻繁にテストする必要があるようなテストケースの自動化を優先します。さらに、テスト計画で事前に定義されている特定の条件に基づいてロールバックを自動化します。

実装手順

1. 要件の計画から、テストケースの作成、ツールの設定、自動テスト、テストケースの完了に至る、テストプロセスのあらゆる段階を定義する、開発ライフサイクルのテストライフサイクルを確立します。
   a. 全体的なテスト戦略から、ワークロード固有のテストアプローチを作成します。
   b. 開発ライフサイクル全体を通じて、必要に応じて継続的なテスト戦略を検討します。
2. ビジネス要件とパイプラインへの投資に基づいて、テストとロールバック向けの自動ツールを選択します。
3. どのテストケースを自動化し、どのテストケースを手動で実行するかを決めます。これは、テスト対象の機能に対するビジネス価値の優先順位に基づいて定義できます。チームメンバー全員にこの計画を浸透させて、手動テストを実施する責任を確認します。
   a. 反復可能なケースや頻繁に実行されるケース、反復的なタスクが必要なケース、複数の構成で必要なケースなど、自動化に適した特定のテストケースに自動テスト機能を適用します。

   b. 自動化ツールでテスト自動化スクリプトと成功基準を定義して、特定のケースが失敗した場合に継続的なワークフローの自動化が開始されるようにします。

   c. 自動ロールバックの具体的な失敗基準を定義します。

4. テスト自動化を優先させ、複雑さと人間の操作によって失敗のリスクが高まる部分で、綿密なテストケースにより一貫した結果が達成されるようにします。

5. 自動テストツールとロールバックツールを CI/CD パイプラインに統合します。

   a. 変更の明確な成功基準を策定します。

   b. モニタリングと観察によってこうした基準を検出し、特定のロールバック基準を満たす場合は自動的に変更を元に戻します。

6. 次のようなさまざまなタイプの自動の本番環境テストを実施します。

   a. 2 つのユーザーテストグループ間の現在のバージョンとの比較結果を示す A/B テスト。

   b. すべてのユーザーにリリースする前に、変更を一部のユーザーにロールアウトできる canary テスト。

   c. アプリケーションの外部から新しいバージョンの機能に一度に 1 つずつフラグを付け/外し、新しい機能を 1 つずつ検証することが可能な機能フラグテスト。

   d. 相互に関連する既存のコンポーネントで新機能を検証するリグレッションテスト。

7. アプリケーションの運用、トランザクション、他のアプリケーションやコンポーネントとのやり取りをモニタリングします。ワークロードごとに変更の成功を示すレポートを作成して、自動化とワークフローでさらに最適化の余地がある部分を特定できるようにします。

   a. ロールバック手順を呼び出すべきかどうかについて迅速な判断を可能にする、テスト結果レポートを作成します。

   b. 1 つまたは複数のテスト方法を基に事前定義された失敗条件に基づいて自動ロールバックを許可する戦略を実装します。

8. 将来の反復可能な変更で再利用が可能な自動テストケースを作成します。

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:

- OPS06-BP01 変更の失敗に備える
- OPS06-BP02 デプロイをテストする

関連ドキュメント:

- AWS Builders Library | デプロイ時におけるロールバックの安全性の確保
- AWS CodeDeploy によるデプロイの再デプロイとロールバック
- AWS CloudFormation でデプロイを自動化する際の 8 つのベストプラクティス

関連する例:

- Selenium, AWS Lambda, AWS Fargate, AWS Developer Tools を使ったサーバーレスな UI テスト

関連動画:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

OPS 7. どのように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のですか?

ワークロード、プロセスと手順、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。

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作成する

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は、ワークロードの効果的なサポートに役立ちます。
- 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 に関するコースのライブラリがあります。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 は、AWS エキスパートから学ぶイベントやウェビナーを主催します。
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。

関連ドキュメント:

- AWS のイベントとウェビナー
- 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従って、組織のアーキテクチャのデフォルトの要件として ORR を組み込むことができます。

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

お客様事例

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 を使用していくつかの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、「Operational Readiness Reviews (ORR) ホワイトペーパー」を参照してください。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。
   - 「Operational Readiness Reviews (ORR) ホワイトペーパー」の「Appendix B: Example ORR questions」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。
   - AWS Well-Architected Tool でカスタムレンズを使用して ORR を開発し、アカウントと AWS 組織全体で共有できます。

4. ORR を実施するワークロードを 1 つ選びます。ローンチ前のワークロード、または内部ワークロードが理想的です。
5. ORR チェックリストを実施し、発見した事柄をメモします。定められた緩和がある場合、発見は問題になる可能性があります。緩和が定められていない発見については、対応予定の項目に追加して、ローンチ前に対応を実施します。
6. 時間の経過とともに、ベストプラクティスや要件を ORR に継続的に追加します。

エンタープライズサポートのある AWS Supportのお客様は、テクニカルアカウントマネージャーに運用準備の確認に関する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、独自の ORR チェックリストを作成するためのインタラクティブなバックワードセッションです。

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 — セキュリティ統制目標により、優れた ORR 要件が設定されます。
- 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計画は ORR 要件として適切です。
- 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 チェックリストに含めることをお勧めします。

関連ドキュメント:

- AWS Control Tower - AWS Control Tower のガードレール
- AWS Well-Architected Tool - カスタムレンズ
- Operational Readiness Review Template by Adrian Hornsby
- Operational Readiness Reviews (ORR) ホワイトペーパー

関連動画:

- AWS Supports You | Building an Effective Operational Readiness Review (ORR)

関連する例:

- Sample Operational Readiness Review (ORR) Lens

関連サービス:

- AWS Config
- AWS Control Tower
- AWS Security Hub
- AWS Well-Architected Tool

OPS07-BP03 ランブックを使用して手順を実行する

ランブックは、特定の成果を達成するための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。端的に言えば、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、ワークロードの運用に不可欠な部分です。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、ランブックはユーザーに関係なく、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、ランブックは一元的な場所で公開し、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。短期的、かつ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費やします。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。一元的な場所 (バージョン管理システム) に保管され、頻繁に更新されています。例えば、ランブックは、アプリケーションアラーム、運用上の問題、計画されたライフサイクルイベントの発生時に、重要なアカウントの AWS Health イベントをモニタリング、通知、対応するための機能をチームに提供します。

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する際に、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できなくなる。

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をすぐに実行できます。
- ランブックの自動化により、労力を軽減できます。

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: 中

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示されている必要があります。必要な特殊なアクセス許可やツールを明確に文書化します。問題発生時に、エラー処理とエスカレーションに関する詳細なガイダンスを提供します。ランブックの所有者をリストアップし、一元的な場所で公開します。ランブックが文書化されたら、チームの別のメンバーに使用してもらい、検証します。手順の進化につれて、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。AWS Systems Manager Automations などのサービスを使用すると、ワークロードに対してフラットなテキストを自動化に変換できます。これらの自動化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。AWSSystems Manager Automation は、ローコードのビジュアルデザインエクスペリエンスも提供し、自動化ランブックをより簡単に作成できます。

お客様事例

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、他のランブックと共に社内 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。ランブック作成を始める際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。

```
# Runbook Title
## Runbook Info
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last
 Updated | Escalation POC |
|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions
 | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
2. Step two
```

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。[ランブック情報] にある必須フィールドに [ランブックのタイトル] を入力します。
4. 最初のステップから開始して、ランブックのステップ部分を入力します。

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い、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. 経時的に、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、実装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 プロセスと手順に特定の所有者が存在する
- OPS07-BP04 プレイブックを使用して問題を調査する
- 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
- OPS10-BP02 アラートごとにプロセスを用意する
- OPS11-BP04 ナレッジ管理を実施する

関連ドキュメント:

- AWS Well-Architected フレームワーク: 概念: ランブックの作成
- 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: ランブックの使用
- 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
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

関連する例:

- 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
- AWS ブログ記事: クラウドオートメーションプラクティスを構築して運用上の優秀性を実現する: AWS Managed Services 提供のベストプラクティス
- AWS Systems Manager: オートメーションのチュートリアル
- 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
- ドキュメントビルダーを使用したカスタムランブックの作成

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がない。
- 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が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を活用し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。AWS Systems Manager Automations などのサービスを使用すると、フラットなテキストを自動化に変換できます。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。

お客様は 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、インシデントを通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。

お客様事例

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。

```
# Playbook Title
## Playbook Info
| Playbook ID | Description | Tools Used | Special Permissions | Playbook Author | Last
 Updated | Escalation POC | Stakeholders | Communication Plan |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this playbook for? What incident is it used for? | Tools |
 Permissions | Your Name | 2022-09-21 | Escalation Name | Stakeholder Name | How will
 updates be communicated during the investigation? |
## Steps
1. Step one
2. Step two
```

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、[プレイブック名] セクションと [プレイブック情報] の下のフィールドに入力します。
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
- OPS07-BP03 ランブックを使用して手順を実行する
- 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
- OPS10-BP02 アラートごとにプロセスを用意する
- OPS11-BP04 ナレッジ管理を実施する

関連ドキュメント:

- AWS Well-Architected Framework: Concepts: Playbook development
- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: ランブックの使用
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

関連動画:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS Virtual Workshops
- Integrate Scripts into AWS Systems Manager

関連する例:

- AWS Customer Playbook Framework
- AWS Systems Manager: オートメーションのチュートリアル
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix - Jupyter Notebook でランブックを作成するための Python ライブラリ
- カスタムランブック作成のためのドキュメントビルダーの使用
- Well-Architected Labs: 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 Labs: Incident response playbook with Jupyter

関連サービス:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

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、チームが緩和戦略の策定に失敗した場合にシミュレーションを行う演習です。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ことの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、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:

- 失敗したデプロイを処理するプロセスを使用せずに、ワークロードに変更をデプロイする。

- 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る。
- 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:

- ワークロードへの変更が失敗した場合の準備が整います。
- ワークロードへの変更が、ガバナンスポリシーに準拠します。

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

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準拠であ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要件に準拠しています。

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開始するシナリオまたは基準を策定します。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、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、組織全体にわたる関係者の調整と作業が必要になります。

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 - 失敗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 デプロイをテストする - 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 - システム変更のデプロイ時に、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。

関連ドキュメント:

- Amazon Web Services: リスクとコンプライアンス
- AWS 責任共有モデル
- 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作成する

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を活用し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任意のアカウントで AWS ビジネスサポート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 ビジネスサポート以上を契約すると、AWS Support 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 Supportからの支援が必要です。AWS Supportは、さまざまなツール、テクノロジー、人、プログラムの組み合わせを提供します。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ビジネスサポートには AWS Trusted Advisor や AWS Personal Health Dashboard へのアクセス、応答時間の短縮などの追加の利点があります。
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

関連ドキュメント:

- AWS Support Plans

関連サービス:

- AWS ビジネスサポート
- AWS エンタープライズサポート

運用

Questions

- OPS 8. 組織でワークロードのオブザーバビリティを活用するにはどうすればよいでしょうか?
- OPS 9. 運用の正常性はどのように理解するのですか?
- 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. 組織でワークロードのオブザーバビリティを活用するにはどうすればよいでしょうか?

オブザーバビリティを活用して、ワークロードの最適な状態を確保します。関連するメトリクス、ログ、トレースを活用して、ワークロードのパフォーマンスを包括的に把握し、問題に効率的に対処します。

ベストプラクティス

- OPS08-BP01 ワークロードメトリクスを分析する
- OPS08-BP02 ワークロードログを分析する
- OPS08-BP03 ワークロードのトレースを分析する
- OPS08-BP04 実践的なアラートを作成する
- OPS08-BP05 ダッシュボードを作成する

OPS08-BP01 ワークロードメトリクスを分析する

アプリケーションテレメトリを実装したら、収集したメトリクスを定期的に分析します。レイテンシー、リクエスト、エラー、容量 (またはクォータ) はシステムパフォーマンスに関するインサイトを提供しますが、ビジネス成果メトリクスの確認を優先することが不可欠です。これにより、ビジネス目標に沿ったデータ主導の意思決定を確実に行うことができます。

期待される成果: ワークロードのパフォーマンスを正確に把握することで、データに基づいた意思決定ができるようになり、ビジネス目標と合致させることができます。

一般的なアンチパターン:

- ビジネス成果への影響を考慮せずに、メトリクスを個別に分析している。
- ビジネス上のメトリクスは重視せず、過度に技術メトリクスに頼っている。
- メトリクスを見直す頻度が低く、リアルタイムの意思決定を行う機会を逃している。

このベストプラクティスを活用するメリット:

- 技術的なパフォーマンスとビジネス成果の相関関係についてより詳しく把握できます。
- リアルタイムのデータに基づいて意思決定プロセスが改善されます。
- ビジネス成果に影響が及ぶ前に、問題を事前に特定して軽減できます。

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: 中

実装のガイダンス

Amazon CloudWatch などのツールを活用してメトリクス分析を行います。特に静的なしきい値が不明な場合や動作パターンが異常検出に適している場合、CloudWatch 異常検出や Amazon DevOps Guru などの AWS サービスを異常検出に使用できます。

実装手順

1. 分析とレビュー: ワークロードメトリクスを定期的に見直して分析します。
   a. 純粋に技術的なメトリクスよりもビジネス成果メトリクスを優先します。
   b. データ内のスパイク、ドロップ、パターンの重要性を理解します。
2. Amazon CloudWatch を利用する: Amazon CloudWatch を使用して、一元化されたビューと詳細な分析を行います。
   a. メトリクスを可視化して時系列で比較できるように、CloudWatch ダッシュボードを設定します。
   b. CloudWatch でパーセンタイルを使用すると、メトリクスの分布を明確に把握できます。これは、SLA の定義や外れ値の理解に役立ちます。
   c. CloudWatch 異常検出を設定して、静的なしきい値に依存せずに異常なパターンを特定します。
   d. CloudWatch クロスアカウントオブザーバビリティを実装して、リージョン内の複数のアカウントにまたがるアプリケーションをモニタリングおよびトラブルシューティングします。
   e. CloudWatch Metric Insights を使用して、アカウントやリージョンのメトリクスデータをクエリして分析し、傾向や異常を特定します。
   f. CloudWatch Metric Math を適用すると、メトリクスの変換、集計、または計算を実行して、より深いインサイトを得られます。
3. Amazon DevOps Guru の導入: 機械学習で強化された異常検出に Amazon DevOps Guru を組み込み、サーバーレスアプリケーションの運用上の問題の初期兆候を特定し、顧客に影響を与える前に修正します。

4. インサイトに基づく最適化: メトリクス分析を基盤に情報に基づいた意思決定を行い、ワークロードを調整して改善します。

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:

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する

関連ドキュメント:

- The Wheel ブログ - メトリクスの継続的なレビューの重要性
- パーセンタイルは重要
- AWS Cost Anomaly Detection の使用
- CloudWatch クロスアカウントオブザーバビリティ
- CloudWatch Metrics Insights を使用してメトリクスをクエリする

関連動画:

- Enable Cross-Account Observability in Amazon CloudWatch
- Amazon DevOps Guru の概要
- AWS Cost Anomaly Detection を使用してメトリクスを継続的に分析する

関連する例:

- 1 つのオブザーバビリティワークショップ
- Amazon DevOps Guru を使用して AIOps による運用上の洞察を得る

OPS08-BP02 ワークロードログを分析する

アプリケーションの運用面をより詳細に把握するには、ワークロードログを定期的に分析することが不可欠です。ログデータを効率的にふるい分け、可視化し、解釈することで、アプリケーションのパフォーマンスとセキュリティを継続的に最適化できます。

期待できる成果: 詳細なログ分析から得られるアプリケーションの動作と運用に関する豊富なインサイトを利用することで、積極的な問題の検出と軽減が実現します。

一般的なアンチパターン:

- 重大な問題が発生するまでログの分析を怠っている。
- ログ分析に利用できるツールをフルセットで使用していないため、重要なインサイトを見逃してしまう。
- 自動化やクエリ機能を活用せずに、ログの手動確認のみに依存している。

このベストプラクティスを活用するメリット:

- 運用上のボトルネック、セキュリティ上の脅威、その他の潜在的な問題を事前に特定できます。
- ログデータを効率的に利用して、アプリケーションを継続的に最適化できます。
- アプリケーションの動作に関してより詳細に把握できるようになり、デバッグ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を活用しない場合のリスクレベル: 中

実装のガイダンス

Amazon CloudWatch Logs は、ログ分析のための強力なツールです。CloudWatch Logs Insights や Contributor Insights などの統合された機能を使用すると、ログから意義ある情報を導き出すプロセスが直感的かつ効率的になります。

実装手順

1. CloudWatch Logs の設定: CloudWatch Logs にログを送信するようにアプリケーションとサービスを設定します。
2. ログ異常検出を使用する: Amazon CloudWatch Logs の異常検出を使用して、異常なログパターンを自動的に識別し、警告します。このツールを使用すると、ログの異常を積極的に管理し、潜在的な問題を早期に検出できます。
3. CloudWatch Logs Insights のセットアップ: CloudWatch Logs Insights を使用すると、ログデータをインタラクティブに検索し、分析することができます。
    a. クエリを作成してパターンを抽出し、ログデータを可視化して、実践的なインサイトを導き出します。

   b. CloudWatch Logs Insights パターン分析を使用して、頻繁なログパターンを分析および視覚化します。この機能は、ログデータの一般的な運用傾向と潜在的な外れ値を理解するのに役立ちます。
   c. CloudWatch Logs 比較 (差分) を使用して、異なる期間の間または異なるロググループの間で差分分析を実行します。この機能を使用すると、変更点を特定し、システムのパフォーマンスや動作への影響を評価できます。
4. Live Tail を使用してログをリアルタイムでモニタリングする: Amazon CloudWatch Logs Live Tail を使用して、ログデータをリアルタイムで表示します。アプリケーションの運用アクティビティを発生時に積極的にモニタリングできるため、システムパフォーマンスと潜在的な問題を即座に把握できます。
5. Contributor Insights の活用: CloudWatch Contributor Insights を使用して、IP アドレスやユーザーエージェントなど、カーディナリティの高い次元でトップトーカーを特定します。
6. CloudWatch Logs メトリクスフィルターの実装: CloudWatch Logs メトリクスフィルターを設定して、ログデータを実用的なメトリクスに変換します。これにより、アラームを設定したり、パターンをさらに詳細に分析したりできます。
7. CloudWatch クロスアカウントオブザーバビリティを実装する: リージョン内の複数のアカウントにまたがるアプリケーションをモニタリングおよびトラブルシューティングできます。
8. 定期的なレビューと改善: ログ分析戦略を定期的に確認して、すべての関連情報を収集し、アプリケーションのパフォーマンスを継続的に最適化します。

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:

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する
- OPS08-BP01 ワークロードメトリクスを分析する

関連ドキュメント:

- CloudWatch Logs Insights を使用したログデータの分析
- CloudWatch Contributor Insights の使用
- CloudWatch ログメトリクスフィルターの作成と管理

関連動画:

- Analyze Log Data with CloudWatch Logs Insights
- Use CloudWatch Contributor Insights to Analyze High-Cardinality Data

関連する例:

- CloudWatch Logs のサンプルクエリ
- 1 つのオブザーバビリティワークショップ

OPS08-BP03 ワークロードのトレースを分析する

トレースデータの分析は、アプリケーションの運用過程を包括的に把握するために不可欠です。さまざまなコンポーネント間の相互作用を可視化して把握することで、パフォーマンスを微調整し、ボトルネックを特定し、ユーザーエクスペリエンスを向上させることができます。

期待される成果: アプリケーションの分散された運用を明確に可視化することで、より迅速な問題解決とユーザーエクスペリエンスの向上につながります。

一般的なアンチパターン:

- トレースデータを見落とし、ログとメトリクスのみに依存している。
- トレースデータが関連するログと関連付けられていない。
- レイテンシーや障害率など、トレースから導き出されたメトリクスを考慮していない。

このベストプラクティスを活用するメリット:

- トラブルシューティングを改善し、平均解決時間 (MTTR) を短縮します。
- 依存関係とその影響についてのインサイトが得られます。
- パフォーマンスの問題を迅速に特定して修正できます。
- トレースから導き出されたメトリクスを活用して、情報に基づいた意思決定を行うことができます。
- コンポーネントのインタラクションが最適化され、ユーザーエクスペリエンス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

AWS X-Ray は、トレースデータ分析のための包括的なスイートを提供し、サービスインタラクションの全体像の把握、ユーザーアクティビティのモニタリング、パフォーマンスに関する問題の検出を可能にします。ServiceLens、X-Ray Insights、X-Ray Analytics、Amazon DevOps Guru などの機能により、トレースデータから導き出される実践的なインサイトが向上します。

実装手順

次の手順は、AWS サービスを使用してトレースデータ分析を効果的に実装するための構造化されたアプローチを提供します。

1. AWS X-Ray を統合する: トレースデータをキャプチャするために、X-Ray をアプリケーションと統合します。
2. X-Ray メトリクスの分析: サービスマップを使用してアプリケーションのヘルスをモニタリングし、レイテンシー、リクエスト率、障害率、応答時間の分布など、X-Ray トレースから派生したメトリクスを詳しく調べます。
3. ServiceLens を使用する: ServiceLens マップを活用して、サービスとアプリケーションのオブザーバビリティを強化します。これにより、トレース、メトリクス、ログ、アラーム、その他のヘルス情報を総合的に確認できます。
4. X-Ray Insights を有効にする:
   a. X-Ray Insights をオンにして、トレース内の異常を自動検出します。
   b. インサイトを調べてパターンを特定し、障害率の増加やレイテンシーの増大などの根本原因を突き止めます。
   c. 検出された問題を時系列で分析するには、インサイトタイムラインを参照します。
5. X-Ray Analytics を使用する: X-Ray Analytics を使用すると、トレースデータを徹底的に調べたり、パターンを特定したり、インサイトを抽出したりできます。
6. X-Ray でループを使用する: X-Ray でグループを作成して、高レイテンシーなどの条件に基づいてトレースをフィルタリングすると、より的を絞った分析につながります。
7. Amazon DevOps Guru を組み込む: Amazon DevOps Guru をエンゲージして、機械学習モデルが運用上の異常をトレースで特定する利点を活用します。
8. CloudWatch Synthetics を使用する: CloudWatch Synthetics を使用して Canary を作成し、エンドポイントとワークフローを継続的にモニタリングします。Canary を X-Ray と統合することで、テスト対象のアプリケーションを詳細に分析するためのトレースデータを提供できます。
9. Real User Monitoring (RUM) を使用する: AWS X-Ray および CloudWatch RUM を使用すると、アプリケーションのエンドユーザーからダウンストリームの AWS マネージドサービスまでのリク

エストパスを分析およびデバッグできます。これにより、エンドユーザーに影響を与えるレイテンシーの傾向やエラーを特定できます。

10. ログとの相関: トレースデータを X-Ray トレースビュー内の関連ログと相関させて、アプリケーションの動作を詳細に把握します。これにより、トレース対象のトランザクションに直接関連するログイベントを確認できます。

11. CloudWatch クロスアカウントオブザーバビリティを実装する: リージョン内の複数のアカウントにまたがるアプリケーションをモニタリングおよびトラブルシューティングできます。

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:

- OPS08-BP01 ワークロードメトリクスを分析する
- OPS08-BP02 ワークロードログを分析する

関連ドキュメント:

- ServiceLens を使用したアプリケーションのヘルスのモニタリング
- X-Ray Analytics を使用したトレースデータの検索
- X-Ray Insights を使用したトレースの異常検出
- CloudWatch Synthetics を使用した継続的なモニタリング

関連動画:

- Amazon CloudWatch Synthetics と AWS X-Ray を使用してアプリケーションを分析しデバッグする
- AWS X-Ray Insights を使用する

関連する例:

- 1 つのオブザーバビリティワークショップ
- AWS Lambda を使用した X-Ray の実装
- CloudWatch Synthetics Canary テンプレート

OPS08-BP04 実践的なアラートを作成する

アプリケーションの動作の逸脱を迅速に検出して対応することが重要です。特に重要なのは、主要業績評価指標 (KPI) に基づく成果がリスクにさらされている場合や、予期しない異常が発生した場合を認識することです。KPI に基づいてアラートを送信することで、受信される警告が直接的に業務や運用上の影響と関連付けられるようになります。実践的なアラートに関するこのようなアプローチを採用すると、積極的な対応の促進とシステムのパフォーマンスと信頼性の維持につながります。

期待される成果: 特に KPI の結果がリスクにさらされている場合に、潜在的な問題を迅速に特定して軽減するための、タイムリーで関連性のある実用的なアラートを受け取ることができます。

一般的なアンチパターン:

- 重大ではないアラートを多数設定しすぎて、アラート疲れを引き起こしている。
- アラートに KPI に基づく優先順位付けを行っていないため、問題が業務に及ぼす影響を把握できにくくなっている。
- 根本原因への対処を怠っているため、同じ問題について繰り返しアラートが送信される。

このベストプラクティスを活用するメリット:

- 実践的で関連性の高いアラートに重点を置くことで、アラート疲労を軽減します。
- 問題を事前に検出して軽減することで、システムの稼働時間と信頼性が向上します。
- 一般的なアラートツールやコミュニケーションツールと統合することで、チームのコラボレーションを強化し、問題を迅速に解決できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

効果的なアラートメカニズムを構築するには、KPI に基づく結果がリスクにさらされている場合や異常が検出された場合にフラグを立てるメトリクス、ログ、トレースデータを使用することが重要です。

実装手順

1. 主要業績評価指標 (KPI) を決定する: アプリケーションの KPI を特定します。正確に業務への影響を反映するには、アラートをこのような KPI に関連付ける必要があります。
2. 異常検出の実装:

- Amazon CloudWatch 異常検出を使用する: Amazon CloudWatch 異常検出を設定して、異常なパターンを自動的に検出します。これにより、真の異常に関するアラートのみが生成されます。
- AWS X-Ray Insights の使用:
  a. X-Ray Insights を設定して、トレースデータの異常を検出します。
  b. 検出された問題について警告するように、X-Ray Insights の通知を設定します。
- Amazon DevOps Guru との統合:
  a. Amazon DevOps Guru の機械学習機能を活用して、既存データの運用上の異常を検出します。
  b. DevOps Guru の [通知設定] に移動して、異常アラートを設定します。

3. 実践的なアラートを実装する: 迅速なアクションに必要な、適切な情報を提供するアラートを設計します。
   1. Amazon EventBridge ルールで AWS Health イベントをモニタリングするか、プログラムで AWS Health API と統合して、AWS Health イベント受信時のアクションを自動化します。これらのアクションには、計画されたすべてのライフサイクルイベントメッセージをチャットインターフェイスに送信するなどの一般的なアクションや、IT サービス管理ツールでのワークフローの開始などの特定のアクションがあります。
4. アラート疲れを軽減する: 重要でないアラートを最小限に抑えます。多数の重要でないアラートによりチームに負担がかかると、重大な問題の見落としにつながり、アラートメカニズムの全体的な有効性が低下する場合があります。
5. 複合アラームを設定する: Amazon CloudWatch 複合アラームを使用して、複数のアラームを統合します。
6. アラートツールと統合する: Ops Genie や PagerDuty などのツールを組み込みます。
7. AWS Chatbot と連結させる: AWS Chatbot と統合して、Amazon Chime、Microsoft Teams、Slack にアラートを中継します。
8. ログに基づくアラート: CloudWatch のログメトリクスフィルターを使用して、特定のログイベントに基づいてアラームを作成します。
9. レビューと反復: アラート設定を定期的に見直して調整します。

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:

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する
- OPS04-BP03 ユーザーエクスペリエンステレメトリを実装する
- OPS04-BP04 依存関係のテレメトリを実装する
- OPS04-BP05 分散トレースを実装する
- OPS08-BP01 ワークロードメトリクスを分析する
- OPS08-BP02 ワークロードログを分析する
- OPS08-BP03 ワークロードのトレースを分析する

関連ドキュメント:

- Amazon CloudWatch でのアラームの使用
- アラームの組み合わせ
- 異常検出に基づいて CloudWatch アラームを作成する
- DevOps Guru 通知
- X-Ray インサイト通知
- インタラクティブな ChatOps による AWS リソースのモニタリング、運用、トラブルシューティング
- Amazon CloudWatch 統合ガイド | PagerDuty
- OpsGenie を Amazon CloudWatch と統合する

関連動画:

- Create Composite Alarms in Amazon CloudWatch
- AWS Chatbot の概要
- AWS On Air ft. Mutative Commands in AWS Chatbot

関連する例:

- Amazon CloudWatch を使用したクラウドでのアラーム、インシデント管理、修復
- チュートリアル: AWS Chatbot に通知を送信する Amazon EventBridge ルールの作成
- 1 つのオブザーバビリティワークショップ

OPS08-BP05 ダッシュボードを作成する

ダッシュボードは、ワークロードのテレメトリデータを理解しやすいように表示します。ダッシュボードは重要な視覚的インターフェイスを提供するとはいえ、アラートメカニズムに取って代わるものではなく、補完となるべきものです。考慮して作成することにより、システムのヘルスとパフォーマンスに関する迅速なインサイトが得られるのみでなく、ビジネス成果や問題の影響に関するリアルタイムの情報をステークホルダーに提供できます。

期待される成果:

視覚的な表示を使用して、システムとビジネスのヘルスに関する明確かつ実践的なインサイトが得られます。

一般的なアンチパターン:

- メトリクスが多すぎてダッシュボードが必要以上に複雑化する。
- 以上を検出するアラートを設定せずにダッシュボードに依存している。
- ワークロードが進化してもダッシュボードが更新されない。

このベストプラクティスを活用するメリット:

- 重要なシステムメトリクスと KPI を即座に可視化します。
- 関係者のコミュニケーションと理解が強化されます。
- 運用上の問題の影響についてのインサイトを迅速に把握できます。

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: 中

実装のガイダンス

ビジネス視点のダッシュボード

ビジネス KPI に応じてカスタマイズしたダッシュボードは、幅広いステークホルダーのエンゲージメントを向上させます。関係者はシステムメトリクスに関心を持つとは限りませんが、このような数値のビジネスへの影響を把握することには熱心です。ビジネス視点のダッシュボードにより、モニタリングおよび分析されるすべての技術的および運用上のメトリクスが、包括的なビジネス目標に沿っていることを確認できます。このような調整により、透明性が実現し、重要な事項とそうでない事項について、組織全体のコンセンサスが得られます。さらに、ビジネス KPI を強調表示するダッシュボードは、より実践的となる傾向があります。関係者は、業務の状態、注意が必要な領域、ビジネス成果への潜在的な影響を迅速に把握できます。

これらの点を考慮に入れて、ダッシュボード作成の際は、技術的なメトリクスとビジネス KPI のバランスが取れていることを確認します。どちらも不可欠であるとはいえ、対象者は異なります。理想的には、システムのヘルスとパフォーマンスを包括的に把握すると同時に、主要なビジネス成果とその影響を強調表示するダッシュボードが求められます。

Amazon CloudWatch ダッシュボードは、CloudWatch コンソールにあるカスタマイズ可能なホームページであり、ダッシュボードを使用すれば、異なる AWS リージョンにまたがっているリソースでも、1 つのビューでモニタリングできます。

実装手順

1. 基本的なダッシュボードを作成する: CloudWatch で新しいダッシュボードを作成し、わかりやすい名前を付けます。
2. マークダウンウィジェットを使用する: メトリクスに絞り込む前に、マークダウンウィジェットを使用してダッシュボードの上部にテキストコンテキストを追加します。これにより、ダッシュボードの内容、表示されるメトリクスの重要性を説明できます。説明には、その他のダッシュボードやトラブルシューティングツールへのリンクも記載できます。
3. ダッシュボード変数を作成する: 必要に応じてダッシュボード変数を組み込み、動的で柔軟なダッシュボードビューを許可します。
4. メトリクスウィジェットを作成する: メトリクスウィジェットを作成して、アプリケーションが出力するさまざまなメトリクスを可視化し、ウィジェットを調整してシステムのヘルスとビジネス成果を効果的に表示します。
5. Log Insights クエリを活用する: CloudWatch Log Insights を使用してログから実用的なメトリクスを取得し、ダッシュボードにこれらのインサイトを表示します。
6. アラームを設定する: CloudWatch アラームをダッシュボードに統合して、しきい値を超えているメトリクスを簡単に確認できるビューを提供します。
7. Contributor Insights を使用する: CloudWatch Contributor Insights を組み込み、高カーディナリティフィールドを分析し、リソースの上位コントリビューターをより明確に理解します。
8. カスタムウィジェットを設計する: 標準ウィジェットでは満たされない特定のニーズについては、カスタムウィジェットの作成を検討してください。カスタムウィジェットを使用すると、さまざまなデータソースからデータを引き出したり、独自の方法でデータを表示したりできます。
9. AWS Health Dashboard を使用する: AWS Health Dashboard を使用して、サービスとリソースに影響を与える可能性のあるアカウントのヘルス、イベント、今後の変更に関するより深いインサイトを取得します。また、AWS Organizations でヘルスイベントを一元表示したり、独自のカスタムダッシュボードを作成したりすることもできます (詳細については、以下の「関連する例」を参照してください)。

10反復と改良を実施する: アプリケーションの進化に応じて、定期的にダッシュボードを見直し、関連性を確認します。

リソース

関連するベストプラクティス:

- OPS04-BP01 主要業績評価指標を特定する
- OPS08-BP01 ワークロードメトリクスを分析する
- OPS08-BP02 ワークロードログを分析する
- OPS08-BP03 ワークロードのトレースを分析する
- OPS08-BP04 実践的なアラートを作成する

関連ドキュメント:

- 運用を可視化するためのダッシュボードの構築
- Amazon CloudWatch ダッシュボードの使用

関連動画:

- 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
- AWS re:Invent 2021 - AWS クラウドオペレーションダッシュボードを使用してエンタープライズレベルの可視化を実現する

関連する例:

- 1 つのオブザーバビリティワークショップ
- Amazon CloudWatch Application Insights を使用したアプリケーションモニタリング
- AWS Health イベントインテリジェンスダッシュボードとインサイト
- Amazon Managed Grafana を使用して AWS Health イベントを視覚化する

OPS 9. 運用の正常性はどのように理解するのですか?

運用メトリクスを定義し、キャプチャして、分析することによって運用イベントへの可視性が得られ、適切な措置を講じることができます。

ベストプラクティス

- OPS09-BP01 メトリクスを使用して業務目標と KPI を測定する
- OPS09-BP02 ステータスと傾向を伝達して運用の可視性を確保する
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け

OPS09-BP01 メトリクスを使用して業務目標と KPI を測定する

業務の成功を定義する目標と KPI を組織から取得し、それを反映するメトリクスを決定します。基準点としてベースラインを設定し、定期的に再評価します。このようなメトリクスをチームから収集して評価するメカニズムを開発します。

期待される成果:

- 組織の業務チームの目標と KPI が公開され、共有されています。
- このような KPI を反映したメトリクスが確立されています。以下はその例です。
  - チケットキューの長さ、またはチケットの平均経過時間
  - 問題の種類別のチケット数
  - 標準業務手順書 (SOP) の有無を問わず、問題の処理に費やした時間
  - 失敗したコードプッシュからの回復に費やされた時間
  - 通話ボリューム

一般的なアンチパターン:

- デベロッパーがトラブルシューティングタスクに追われてしまうため、デプロイの期限が守れない。開発チームは追加の人員を求めていますが、開発作業に取り組めなかった時間を測定できないため、必要な人数がわからない。
- Tier 1 デスクが、ユーザーからの電話に対応するために設置され、時間が経つにつれて、ワークロードは増えましたが、Tier 1 デスクへの人員は追加されない。通話時間が長くなり、問題が解決されないまま問題が長引くと、顧客満足度は低下するが、経営陣にはそのような兆候が明らかでないため、対策がとられていない。
- 問題のあるワークロードは、メンテナンスのために別の運用チームに引き継がる。その他のワークロードとは異なり、この新しいワークロードには適切なドキュメントとランブックが付属していないため、チームはトラブルシューティングや障害への対処に時間を費やすが、これを文書化するメトリクスがないため、説明責任が困難となる。

このベストプラクティスを活用するメリット: ワークロードのモニタリングではアプリケーションとサービスのステータスを明らかにするのに対し、モニタリングする運用チームは、ビジネスニーズの変化など、ワークロードのコンシューマー間の変化についてオーナーにインサイトを提供します。運用状況を反映するメトリクスを作成することで、チームの有効性を測定し、ビジネス目標に照らして評価できます。メトリクスでは、サポート上の問題を浮き彫りにしたり、サービスレベル目標から逸脱した時期を特定したりできます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

業務部門のリーダーおよび関係者とサービスの全体的な目標を決定します。さまざまな業務チームのタスクがどうあるべきか、またどのような課題に取り組むことができるかを判断します。これらを使用して、これらの業務目標を反映すると思われる主要業績評価指標 (KPI) についてブレインストーミングを行います。これには、顧客満足度、機能の構想からデプロイまでの時間、平均問題解決時間などが含まれます。

KPI に基づいて、このような目標を最もよく反映すると思われるメトリクスとデータソースを特定します。顧客満足度は、通話の待ち時間や応答時間、満足度スコア、発生した問題の種類など、さまざまなメトリクスを組み合わせたものです。デプロイ時間は、テストとデプロイに必要な時間に加えて、デプロイ後に追加する必要のある修正を加算したものである場合があります。さまざまな種類の課題に費やされた時間 (またはそれらの課題の数) を示す統計値から、集中的に取り組む必要がある個所を把握できます。

リソース

関連ドキュメント:

- Amazon QuickSight - KPI の使用
- Amazon CloudWatch メトリクスの使用
- ダッシュボードの構築
- KPI ダッシュボードでコスト最適化 KPI を追跡する方法

OPS09-BP02 ステータスと傾向を伝達して運用の可視性を確保する

運用のステータスと傾向の方向性を把握することとは、その結果がリスクにさらされる可能性がある時期、追加の作業をサポートできるかどうか、または変更がチームに及ぼす影響を特定するために必要です。運用イベント中に、ユーザーや運用チームが情報を参照できるステータスページを提供することにより、コミュニケーションチャネルの負担を軽減し、情報を積極的に広めることができます。

期待される成果:

- 運用リーダーは、チームがどのような種類のコール数に対応して業務を行っているのか、デプロイなど、どのような取り組みが進行中であるかをひとめで把握できます。
- 通常の運用に影響が及ぶ場合、アラートが関係者やユーザーコミュニティに配信されます。
- 組織のリーダーや関係者は、アラートや影響に応じてステータスページを確認したり、連絡先、チケット情報、推定復旧時間など、運用上のイベントに関する情報を取得したりすることができます。
- 経営陣やその他の関係者には、特定期間のコール数、ユーザー満足度スコア、未処理のチケット数、チケットの経過時間などの運用に関する統計値を表示するレポートが提供されます。

一般的なアンチパターン:

- ワークロードがダウンして、サービスが利用できなくなります。ユーザーは何が起こっているのかを問い合わせるため、コール数が急増します。マネージャーは、問題に対処している担当者を突き止めるために問い合わせをするため、さらに負荷が増大します。さまざまな運用チームが個別に調査を行うため、作業が重複します。
- 新しい機能が必要になると、そのエンジニアリング業務に数人の担当者が再配置されます。運用への補完人員が提供されないため、問題解決に要する時間が急増します。このような情報はキャプチャされていないため、数週間経って不満を抱くユーザーからのフィードバックが寄せられるようになってからやっと経営陣は問題に気づきます。

このベストプラクティスを活用するメリット: 業務に影響が及ぶ運用上のイベントの場合、状況を理解しようとするさまざまなチームからの情報請求の問い合わせに、多くの時間と労力が浪費される可能性があります。広範囲にステータスを伝えるステータスページとダッシュボードを提供することで、関係者は、問題が検出されているか、問題解決のリーダーは誰か、通常の運用に戻る予想時間はいつか、などの情報を迅速に入手できます。これにより、チームメンバーはその他のメンバーへのステータスの伝達に多くの時間を費やす必要がなくなり、問題の対処により多くの時間を割くことができます。

さらに、ダッシュボードとレポートを使用すると、意思決定者やステークホルダーにインサイトを提供し、運用チームがビジネスニーズにどのように対応できるか、およびそのリソースがどのように配分されているかを確認できます。これは、ビジネスをサポートするのに十分なリソースが存在するかどうかを判断するために重要です。

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: 中

実装のガイダンス

運用チームの現在の主要メトリクスを表示するダッシュボードを構築して、運用リーダーと経営陣の両方が簡単にアクセスできるようにします。

迅速に更新できるステータスページを作成して、インシデントやイベントの発生時、担当者、対応の調整担当者などを表示できます。ユーザーが考慮すべき手順や回避策をこのページで共有し、このページの場所を広範囲に周知させます。未知の問題に直面した場合は、まずこの場所を確認するようにユーザーに勧めます。

長期にわたる運用のヘルスを説明するレポートを収集して提供し、リーダーや意思決定者に配布して、運用の作業状況を課題やニーズと共に説明します。

目標と KPI を最適な方法で反映し、変化を推進するうえで影響を及ぼした点を示すメトリクスとレポートをチーム間で共有します。このような取り組みに時間を割いて、チーム内とチーム間での運用の重要性を強化します。

リソース

関連ドキュメント:

- 進捗状況を測定する
- 運用を可視化するためのダッシュボードの構築

関連するソリューション:

- データオペレーション

OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け

運用のステータスをレビューする時間とリソースを確保することで、日常業務への対応が優先事項として維持されていることを確認できます。運用リーダーと関係者を集めて、定期的にメトリクスのレビューを行い、目標と目的を再確認したり変更したりして、改善の優先順位を決めます。

期待される成果:

- 運用リーダーとスタッフは定期的にミーティングを開き、特定の報告期間におけるメトリクスのレビューを行います。課題が伝達され、成功が認知され、学んだ教訓が共有されます。

- 関係者と業務部門のリーダーは定期的に運用状況について説明を受け、目標、KPI、将来のイニシアチブに関する意見を求められます。サービスの提供、運用、メンテナンスの間のトレードオフが議論され、考慮されます。

一般的なアンチパターン:

- 新製品が発売されたのに、Tier 1 と Tier 2 の運用チームがサポートを提供できるだけのトレーニングを受けていなかったり、追加のスタッフが割り当てられたりしていません。チケットの解決時間の短縮やインシデント件数の増加を示すメトリクスは、リーダーに確認されていません。数週間後、不満を抱いているユーザーがプラットフォームの利用を止めてサブスクリプション数が減少し始めてから対策が講じられます。
- 長い間、ワークロードのメンテナンスを手動で実行するプロセスが実行されていました。自動化を望む声はありましたが、システムの重要性が低いため、低い優先順位が付けられていました。しかし、時間が経つにつれて、システムの重要性が高まり、現在ではこのような手動プロセスに運用時間の大半を費やしています。運用に追加のツールを提供するためのリソース計画がないため、作業負荷が増加するにつれてスタッフが燃え尽き症候群に陥ります。スタッフが離職してその他の競合他社に転職している報告を受けて、やっと経営陣が事態を把握します。

このベストプラクティスを活用するメリット: 組織によっては、サービスの提供や新しい製品や新しいサービスに費やされるのと同様の時間と注意を費やすことが難しい場合があります。この場合、期待されるサービスのレベルが徐々に低下し、業務部門が損害を受ける可能性があります。これは、事業の成長に伴って運用が変化したり進化したりせず、すぐに遅れをとったままになる可能性があるためです。運用部門が収集したインサイトを定期的に確認しなければ、事業に関するリスクは手遅れになるまで明らかにならない可能性があります。運用スタッフと経営陣の両方にメトリクスと手順をレビューする時間を割り当てることで、運用が果たす重要な役割の可視性が維持され、リスクが重大なレベルとなるよりもかなり前もってリスクを特定できます。運用チームは、今後起こる事業上の変化やイニシアチブについてより的確なインサイトを取得できるため、積極的な対処ができるようになります。経営陣への運用メトリクスの可視化により、チームが顧客満足度において内外の両方で果たす役割が示されるため、優先順位の選択の検討がより適切になり、新しいビジネスやワークロードの取り組みに応じて変更したり進化したりするための時間とリソースを確実に運用に対して確保できます。

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: 中

実装のガイダンス

関係者と運用チーム間の運用メトリクスのレビューを行う時間を割いて、レポートのデータを確認します。このようなレポートを組織の目標と目的の文脈内で考察し、目標や目的が達成されているかどうかを判断します。目標が明確でない場合や、需要と提供されている内容の間に矛盾が生じる可能性がある場合は、あいまいさの原因を特定します。

時間、人材、ツールが運用の成果に貢献している個所を特定します。これがどの KPI に影響し、どのような目標を成功に導くべきかを判断します。定期的に見直して、事業部門をサポートするうえで十分なリソースが運用にあることを確認します。

リソース

関連ドキュメント:

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

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手順を準備して検証し、それらによるワークロードの中断を最小限に抑えます。

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 サービスに影響するイベント発生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

イベント、インシデント、問題を効率的に管理する能力は、ワークロードの正常性とパフォーマンスを維持するために不可欠です。これらの要素の違いを認識し、理解することが、対応と解決の効果的な戦略を策定するうえで極めて重要です。各側面に対して明確に定義されたプロセスを確立し、それに従うことで、チームは運用面で生じる課題に迅速かつ効果的に対処できます。

期待される成果: 組織は、適切に文書化され、一元的に保存されたプロセスを介して、運用上のイベント、インシデント、問題を効果的に管理します。これらのプロセスは随時見直され、変更を反映させることで、処理を効率化し、サービスの信頼性とワークロードのパフォーマンスを高く維持します。

一般的なアンチパターン:

- イベントに先回りして対応するのではなく、事後対応になる。
- さまざまなタイプのイベントやインシデントに対するアプローチに一貫性がない。
- 組織が、再発防止のためのインシデントの分析や学習を行わない。

このベストプラクティスを活用するメリット:

- 対応プロセスが合理化され、標準化されます。
- インシデントがサービスや顧客に与える影響を軽減します。
- 問題解決を早めます。
- 運用プロセスが継続的に改善されます。

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

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。

イベント、インシデント、問題の理解

- イベント: イベントの例として、アクションの観察、発生、状態の変化があります。イベントは計画的な場合も計画外の場合もあり、ワークロードの内部または外部から発生する可能性があります。

- インシデント: インシデントとは、予定外の中断やサービス品質の低下など、対応が必要なイベントのことです。これらは、ワークロードを通常運用に復旧するために早急な対応を迫られる障害です。
- 問題: 問題は、1 つ以上のインシデントの根本原因です。問題を特定して解決するには、再発防止のため、インシデントを掘り下げて調査することなどが必要です。

実装手順

イベント

1. イベントのモニタリング:
   - オブザーバビリティを実装し、ワークロードオブザーバビリティを活用します。
   - ユーザー、ロール、AWS サービスによって実行されたアクションを監視します。これらのアクションは、AWS CloudTrail イベントとして記録されます。
   - Amazon EventBridge を使用して、アプリケーションで運用上の変更にリアルタイムに対応します。
   - AWS Config を使用して、リソース構成の変更を継続的に評価、監視、記録します。
2. プロセスを作成する:
   - どのイベントが重要でモニタリングが必要かを評価するプロセスを考案します。正常なアクティビティと異常なアクティビティのしきい値やパラメータの設定などを行います。
   - イベントをインシデントにエスカレートする基準を決定します。これは、重大度やユーザーへの影響、想定される動作から逸脱しているかどうかなどに基づいて行います。
   - イベントの監視と対応のプロセスを定期的に見直します。例えば、過去のインシデントの分析、しきい値の調整、警告メカニズムの改善などを行います。

インシデント

1. インシデントに対応する:
   - オブザーバビリティツールから得たインサイトを活用して、インシデントを迅速に特定し、対応します。
   - AWS Systems Manager Ops Center を実装して、運用上の問題とインシデントを集約して整理し、優先順位を付けます。
   - Amazon CloudWatch および AWS X-Ray などのサービスを使用して、より詳細な分析とトラブルシューティングを行います。

- インシデント管理を強化するには AWS Managed Services (AMS) を検討して、そのプロアクティブ、予防、検出機能を活用します。AMS は、モニタリング、インシデントの検出および対応、セキュリティ管理などのサービスで運用サポートを拡張します。
- エンタープライズサポートのお客様は、AWS Incident Detection and Response を使用できます。これにより、本番ワークロードの継続的なプロアクティブモニタリングとインシデント管理が可能になります。

2. インシデント管理プロセスを作成する:
   - 役割、コミュニケーションプロトコル、解決手順などを明確に定義した、構造化されたインシデント管理プロセスを確立します。
   - インシデント管理を AWS Chatbot などのツールと統合して、効率的な対応と連携を実現します。
   - インシデントを重大度別に分類し、事前定義されたインシデント対応計画を各カテゴリに設定します。
3. 学習して改善する:
   - インシデント後の分析を実施して、根本原因と解決の有効性を理解します。
   - 見直しと変化する慣行に基づいて、対応計画を継続的に更新および改善します。
   - 学んだ教訓を文書化し、チーム全体で共有することで、業務のレジリエンスを強化します。
   - エンタープライズサポートのお客様は、テクニカルアカウントマネージャーにインシデント管理ワークショップを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。

問題点

1. 問題を特定する:
   - 過去のインシデントからのデータを活用して、システム上の深層の問題を示唆している可能性のある、反復的なパターンを洗い出します。
   - AWS CloudTrail や Amazon CloudWatch などのツールを活用して傾向を分析し、根本的な問題を発見します。
   - 運用、開発、ビジネスユニットなど、部門横断的なチームを組織し、多様な視点から根本原因を探ります。
2. 問題管理プロセスを作成する:
   - 構造化された問題管理プロセスを開発し、その場しのぎの修正ではなく長期的な解決策に焦点を当てます。

- 根本原因分析 (RCA) 手法を取り入れて、インシデントの根本原因を調査し、理解します。
- 検出結果に基づいて運用ポリシー、手順、インフラストラクチャを更新し、再発を防ぎます。

3. 継続的に改善する:
   - 絶え間ない学習と改善の文化を育み、潜在的な問題を先回りして特定し、対処することをチームに奨励します。
   - ビジネスとテクノロジーにおける環境の変化に応じて、問題管理のプロセスとツールを定期的に見直し、改訂します。
   - 組織全体でインサイトとベストプラクティスを共有して、よりレジリエントで効率的な運用環境を構築します。
4. AWS Supportと連携する:
   - AWS Trusted Advisor などの AWS サポートリソースを使用して、プロアクティブなガイダンスや最適化のレコメンデーションを行います。
   - Enterprise Support のお客様は、AWS Countdown などの専用プログラムを利用して、重要なイベント中のサポートを受けられます。

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:

- OPS04-BP01 主要業績評価指標を特定する
- OPS04-BP02 アプリケーションテレメトリを実装する
- 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
- OPS08-BP01 ワークロードメトリクスを分析する
- OPS11-BP02 インシデント後の分析を実行する

関連ドキュメント:

- AWS セキュリティインシデント対応ガイド
- AWS Incident Detection and Response
- AWS クラウド導入フレームワーク: オペレーションのパースペクティブ - インシデントと問題管理
- 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 の推奨インシデント対応ヒント
- AWS re:Invent 2022 - Amazon Builders' Library: Amazon での 25 年間の運用上の優秀性
- AWS re:Invent 2022 - AWS インシデント対応とレスポンス (SUP201)
- AWS Systems Manager の Incident Manager の紹介

関連する例:

- 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
- PagerDuty と AWS Systems Manager Incident Manager でのインシデント対応の自動化
- AWS Systems Manager Incident Manager のオンコールスケジュールでのインシデント対応担当者のエンゲージメント
- AWS Systems Manager Incident Manager でのインシデント対応の可視性とコラボレーションを改善
- AMS でのインシデントレポートとサービスリクエスト

関連サービス:

- Amazon EventBridge

OPS10-BP02 アラートごとにプロセスを用意する

効果的かつ効率的なインシデント管理においては、システム内のアラートごとに明確なプロセスを定義しておくことが重要です。そうすることで、すべてのアラートに対して具体的な対応をすぐに行動に移すことができ、運用の信頼性と応答性が向上します。

期待される成果: すべてのアラートに対して、明確に定義された具体的な対応計画が実践に移されます。可能な場合は、所有権を明確にし、エスカレーション経路を定義して、対応を自動化します。アラートは最新のナレッジベースにリンクされているため、どのオペレーターでも一貫して効果的に対応できます。対応が全体的に迅速で一貫しており、運用の効率と信頼性が向上します。

一般的なアンチパターン:

- アラートに対応プロセスが事前定義されていないため、その場しのぎの対応や解決の遅れにつながる。
- アラート過多になり、重要なアラートが見過ごされる。
- アラートの所有権と責任が明確でないため、アラートの処理に一貫性がない。

このベストプラクティスを活用するメリット:

- 対処可能なアラートのみを発生させることで、アラート疲労が軽減されます。
- 運用上の問題の平均解決時間 (MTTR) が短縮されます。
- 平均調査時間 (MTTI) が短縮され、MTTR の短縮につながります。
- 運用上の対応のスケーラビリティが向上します。
- 運用イベント処理の一貫性と信頼性が向上します。

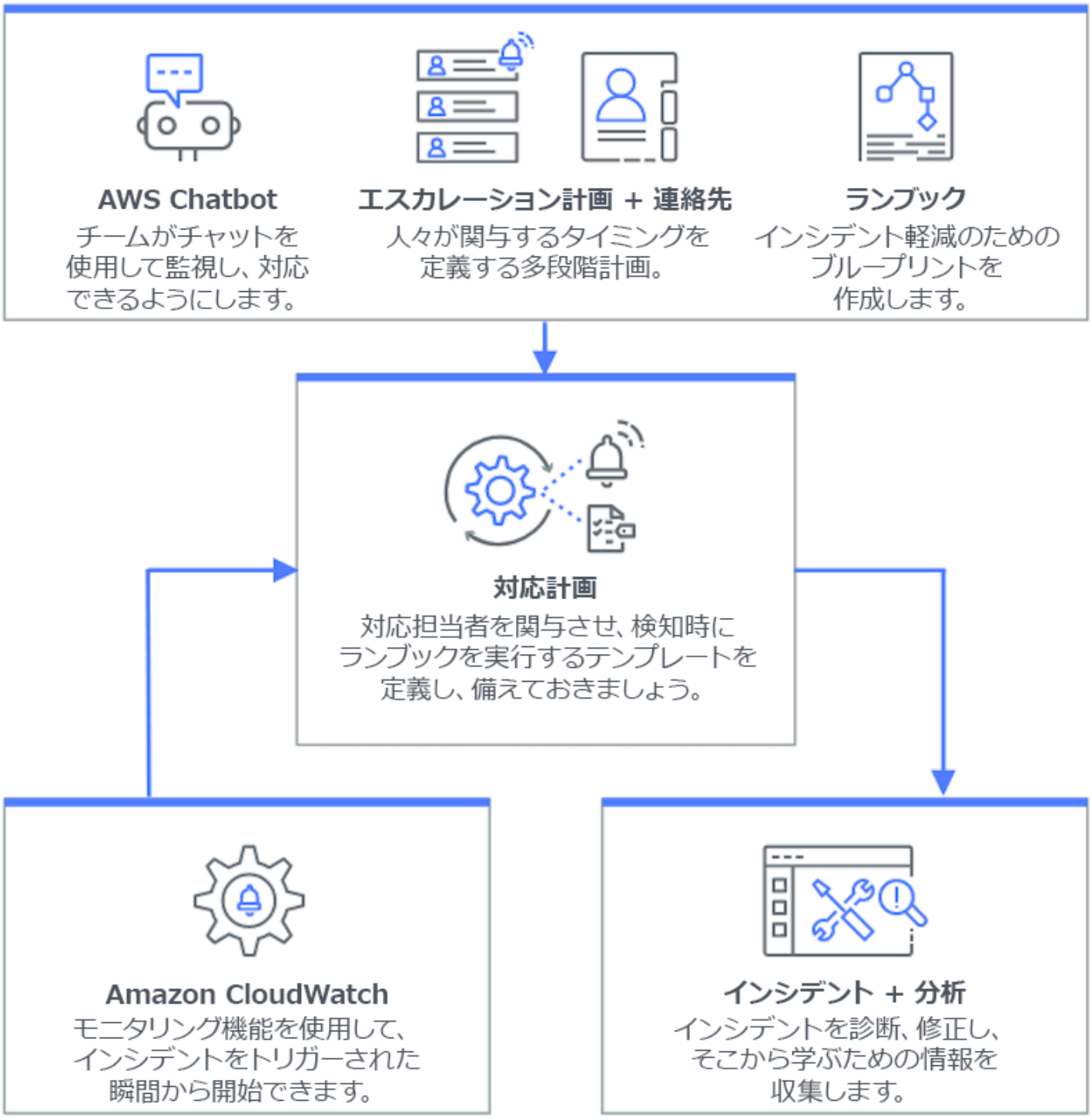
このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

アラートごとにプロセスを用意するには、各アラートに対して明確な対応計画を策定し、可能な場合は対応を自動化します。また、運用上のフィードバックや変化する要件に基づいて、これらのプロセスを継続的に改善していきます。

実装手順

次の図は、AWS Systems Manager Incident Manager 内のインシデント管理ワークフローです。これは、Amazon CloudWatch または Amazon EventBridge からの特定イベントに対してインシデントを自動的に作成して、運用上の課題に迅速に対応するよう設計されています。インシデントが自動または手動で作成されると、Incident Manager がインシデントの管理を一元化し、関連する AWS リソース情報を整理し、事前定義されている対応計画を実践に移します。例えば、即時対応のために Systems Manager Automation ランブックを実行したり、関連するタスクや分析を追跡するための親の運用作業項目を OpsCenter で作成したりします。この合理化されたプロセスにより、AWS 環境全体でインシデント対応が迅速化され、調整されます。



1. 複合アラームを使用する: CloudWatch で複合アラームを作成して、関連するアラームをグループ化し、ノイズを減らし、より意味のある応答を可能にします。
2. Amazon CloudWatch アラームを Incident Manager と統合する: CloudWatch アラームを設定して、AWS Systems Manager Incident Manager でインシデントを自動的に作成します。

3. Amazon EventBridge を Incident Manager と統合する: EventBridge ルールを作成してイベントに対応し、定義された対応計画を使用してインシデントを作成します。
4. Incident Manager でのインシデントへの準備:
    - Incident Manager で、アラートのタイプごとに詳細な対応計画を作成します。
    - Incident Manager の対応計画に接続された AWS Chatbot を通じてチャットチャネルを確立します。インシデント発生時に Slack、Microsoft Teams、Amazon Chime などのプラットフォーム間でのリアルタイムコミュニケーションを促進します。
    - Incident Manager 内に Systems Manager Automation ランブックを組み込み、インシデントへの自動応答を促進します。

リソース

関連するベストプラクティス:

- OPS04-BP01 主要業績評価指標を特定する
- OPS08-BP04 実践的なアラートを作成する

関連ドキュメント:

- AWS クラウド導入フレームワーク: オペレーションのパースペクティブ - インシデントと問題管理
- Amazon CloudWatch アラームの使用
- AWS Systems Manager Incident Manager のセットアップ
- Incident Manager でのインシデントへの準備

関連動画:

- AWS の推奨インシデント対応ヒント

関連する例:

- AWS ワークショップ - AWS Systems Manager Incident Manager - セキュリティイベント対応の自動化

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

運用上のイベントに迅速に対応することは重要ですが、すべてのイベントが同じというわけではありません。ビジネスへの影響に基づいて優先順位を付けて、安全性、財務上の損失、規制違反、評判の低下など、重大な結果を招く可能性のあるイベントも優先的に対処します。

期待される成果: 運用上のイベントへの対応に、ビジネスの運用や目標への潜在的な影響に応じて優先順位が付けられます。これにより、効率的かつ効果的に対応できます。

一般的なアンチパターン:

- すべてのイベントが同じ緊急度で扱われるため、混乱が生じ、重大な問題への対処が遅れる。
- 影響の大きいイベントと小さいイベントの区別がつかず、リソースの誤配分につながる。
- 組織に明確な優先順位付けのフレームワークがないため、運用上のイベントへの対応に一貫性がなくなる。
- イベントの優先順位が、ビジネス成果への影響ではなく、報告された順序で決まる。

このベストプラクティスを活用するメリット:

- 重要なビジネス機能が最初に注目されるようにし、潜在的な損害を最小限に抑えます。
- 複数のイベントが同時に発生した際のリソース配分が改善されます。
- 組織の信頼を維持し、規制要件を満たす能力を高めます。

この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

複数の運用上のイベントに直面した際には、影響と緊急性に基づいて優先順位を決める体系的なアプローチが重要です。このアプローチは、情報に基づいた意思決定を行い、最も必要なところに努力を振り向け、事業継続に対するリスクを軽減するのに役立ちます。

実装手順

1. 影響を評価する: ビジネスの運用や目標への潜在的な影響の観点からイベントの重大度を評価するための分類システムを開発します。次の例は、影響のカテゴリを示しています。

影響度	説明
高い	多くのスタッフや顧客に影響を及ぼす、財務上の影響が大きい、評判への悪影響が大きい、または怪我につながる。
中程度	スタッフや顧客のグループに影響を及ぼす。財務上の影響が中程度、または評判への悪影響が中程度である。
低	個々のスタッフまたは顧客に影響を及ぼす、財務上の影響が小さい、または評判への悪影響が小さい。

2. 緊急度を評価する: 安全性、財務上の影響、サービスレベル契約 (SLA) などの要素を考慮して、イベントにどれだけ迅速に対応する必要があるかを示す緊急度を定義します。次の例は、緊急度のカテゴリを示しています。

緊急度	説明
高い	被害が指数関数的に大きくなる、時間的制約のある作業に影響が出ている、差し迫ったエスカレーションが発生している、VIP ユーザーやグループに影響が出ている。
中程度	被害が時間の経過とともに大きくなる、または 1 人の VIP ユーザーまたは 1 つのグループが影響を受けている。
低	時間の経過とともにわずかながら被害が大きくなる、または時間的制約のない作業に影響が出ている。

3. 優先順位付けのマトリクスを作成する:

   - マトリクスを使用して影響と緊急性を相互参照し、さまざまな組み合わせに優先度を割り当てます。

- 運用上のイベント対応を担当するチームメンバー全員がマトリクスにアクセスし、理解できるようにしてください。
- 次のマトリクスの例は、緊急性と影響に応じたインシデントの重大度を示しています。

緊急性と影響	高い	中程度	低
高い	[非常事態]	緊急	高い
中程度	緊急	高い	[普通]
低	高い	[普通]	低

4. トレーニングとコミュニケーションを行う: 優先順位付けのマトリクスと、イベント発生時にそれに従うことの重要性について、対応チームにトレーニングを行います。優先順位付けのプロセスをすべてのステークホルダーに伝え、明確な期待値を設定します。
5. インシデント対応に統合する:
   - 優先順位付けのマトリクスをインシデント対応計画とツールに組み込みます。
   - 可能な場合は、イベントの分類と優先順位付けを自動化して、対応時間を短縮します。
   - エンタープライズサポートのお客様は、AWS Incident Detection and Response を使用できます。これにより、24 時間 365 日の本番ワークロードのプロアクティブモニタリングとインシデント管理が可能になります。
6. 見直して適応させる: 優先順位付けプロセスの有効性を定期的に見直し、フィードバックやビジネス環境の変化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:

- OPS03-BP03 エスカレーションが推奨されている
- OPS08-BP04 実践的なアラートを作成する
- OPS09-BP01 メトリクスを使用して業務目標と KPI を測定する

関連ドキュメント:

- Atlassian - インシデントの重大度レベルの把握
- IT プロセスマップ - インシデント優先度のチェックリスト

OPS10-BP04 エスカレーション経路を決定する

インシデント対応プロトコル内に明確なエスカレーション経路を確立して、タイムリーかつ効果的に対応できるようにします。そのためには、エスカレーションのプロンプトを指定し、エスカレーションプロセスを詳述し、意思決定を早めて解決までの平均時間 (MTTR) を短縮するためにアクションを事前承認します。

期待される成果: インシデントを適切な担当者にエスカレーションし、対応時間と影響を最小限に抑えるための、構造化された効率的なプロセス。

一般的なアンチパターン:

- 復旧手順が明確でないため、重大なインシデントが発生した際の対応がその場しのぎになる。
- 権限と所有権が定義されていないため、緊急の対応が必要な状況で対応が遅れる。
- ステークホルダーや顧客への情報提供が期待にそっていない。
- 重要な決断が遅れる。

このベストプラクティスを活用するメリット:

- 事前定義されたエスカレーション手順により、インシデント対応が合理化されます。
- 事前に承認されたアクションと明確な所有権により、ダウンタイムを短縮できます。
- インシデントの重大度に応じて、リソース配分とサポートレベルの調整を改善できます。
- ステークホルダーや顧客とのコミュニケーションが改善されます。

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: 中

実装のガイダンス

迅速なインシデント対応には、適切に定義されたエスカレーション経路が不可欠です。AWS Systems Manager Incident Manager は、構造化されたエスカレーション計画とオンコールスケジュールの設定をサポートします。これにより、適切な担当者にアラートが送信され、インシデントが発生したときにすぐに対応できるようになります。

実装手順

1. エスカレーションプロンプトを設定する: CloudWatch アラームを設定して、AWS Systems Manager Incident Manager でインシデントを作成します。

2. オンコールスケジュールを設定する: Incident Manager でエスカレーションパスに沿ったオンコールスケジュールを作成します。オンコール担当者が即座に行動できるように、必要な権限とツールを提供します。
3. エスカレーション手順を詳述する:
    - インシデントをエスカレーションすべき具体的な条件を決定します。
    - Incident Manager でエスカレーション経路を作成します。
    - エスカレーションチャネルは、連絡先またはオンコールスケジュールで構成する必要があります。
    - 各エスカレーションレベルにおけるチームの役割と責任を定義します。
4. 軽減アクションを事前承認する: 意思決定者と協力して、予想されるシナリオに対するアクションを事前に承認しておきます。Incident Manager と統合された Systems Manager Automation ランブックを使用して、インシデント解決を高速化します。
5. 所有権を指定する: エスカレーション経路の各ステップにおける内部の所有者を明確に指定します。
6. サードパーティーエスカレーションについて詳述する:
    - サードパーティーのサービスレベルアグリーメント (SLA) を文書化し、社内の目標とすり合わせます。
    - インシデント発生時のベンダーとのコミュニケーションに対し、明確なプロトコルを設定します。
    - ベンダーの連絡先をインシデント管理ツールに統合し、直接アクセスできるようにします。
    - サードパーティーによる対応シナリオを含む定期的な訓練を実施します。
    - ベンダーのエスカレーション情報を明確に文書化し、簡単にアクセスできるようにします。
7. エスカレーション計画のトレーニングとリハーサルを行う: エスカレーションプロセスについてチームをトレーニングし、インシデント対応訓練やゲームデーを定期的に実施します。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リクエストできます。
8. 継続的に改善する: エスカレーション経路の有効性を定期的に見直します。インシデントの事後分析と継続的なフィードバックから学んだ教訓に基づいてプロセスを更新します。

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:

- OPS08-BP04 実践的なアラートを作成する
- OPS10-BP02 アラートごとに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

関連ドキュメント:

- AWS Systems Manager Incident Manager エスカレーション計画
- Incident Manager でのオンコールスケジュールの操作
- ランブックの作成と管理
- AWS IAM Identity Center での一時的な昇格アクセス管理
- Atlassian - 効果的なインシデント管理のためのエスカレーションポリシー

OPS10-BP05 サービスに影響するイベント発生時の顧客コミュニケーション計画を定義する

顧客との信頼関係を維持し、透明性を確保するためには、サービスに影響を及ぼすイベントが発生した際の効果的なコミュニケーションが不可欠です。コミュニケーション計画が明確に定義されていれば、インシデントの発生時に組織内外で迅速かつ明確に情報を共有することができます。

期待される成果:

- サービスに影響を及ぼすイベントが発生した際に顧客やステークホルダーに効果的に情報を伝えるための、確固たるコミュニケーション計画。
- 透明性が高いコミュニケーションを通じて、信頼を築き、顧客の不安を解消する。
- サービスに影響を及ぼすイベントがカスタマーエクスペリエンスや事業運営に与える影響を最小限に抑える。

一般的なアンチパターン:

- コミュニケーションの不足や遅延が、顧客の混乱や不満につながる。
- メッセージが技術的すぎる、またはあいまいなせいで、ユーザーへの実際の影響を伝えることができない。
- コミュニケーション戦略が事前に定義されていないため、メッセージが一貫性を欠き、事後対応的になる。

このベストプラクティスを活用するメリット:

- 予防的かつ明確なコミュニケーションを通じて、顧客の信頼と満足度が高まります。
- 顧客の不安に先回りして対応することで、サポートチームの負担が軽減します。
- インシデントを効果的に管理し、復旧する能力が向上します。

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: 中

実装のガイダンス

サービスに影響を及ぼすイベントに備えた包括的なコミュニケーション計画の策定には、適切なチャネルの選択からメッセージやトーンの作成まで、さまざまな側面が関与します。適応性と拡張性に優れ、さまざまな障害シナリオに対応できる計画を用意する必要があります。

実装手順

1. 役割と責任を定義する:
    - インシデント対応活動を監督する重大インシデントマネージャーを任命します。
    - 外部および内部のすべてのコミュニケーションの調整を担当するコミュニケーションマネージャーを指名します。
    - サポートマネージャーを関与させ、サポートチケットを通じて一貫したコミュニケーションを実現します。
2. コミュニケーションチャネルを特定する: 職場のチャット、E メール、SMS、ソーシャルメディア、アプリ内通知、ステータスページなどのチャネルを選択します。これらのチャネルには、耐障害性があること、サービスに影響を及ぼすイベントが発生した場合でも独立して動作できることが求められます。
3. 顧客に迅速、明確、定期的に伝える:
    - 重要な詳細情報を簡潔に伝えることに重点を置いて、さまざまなサービス障害シナリオ用のテンプレートを作成します。サービスの障害、想定される解決時間、影響に関する情報を含めてください。
    - Amazon Pinpoint を使用して、プッシュ通知、アプリ内通知、E メール、テキストメッセージ、音声メッセージ、カスタムチャネル経由のメッセージで顧客に警告します。
    - Amazon Simple Notification Service (Amazon SNS) を使用して、プログラムによって、または E メール、モバイルプッシュ通知、テキストメッセージで、サブスクライバーに警告します。
    - Amazon CloudWatch ダッシュボードをパブリックに共有して、ダッシュボードを通じて状況を伝えます。
    - ソーシャルメディアでのエンゲージメントを促す:

- ソーシャルメディアを積極的に監視して、顧客の感情を把握します。
- ソーシャルメディアプラットフォームに投稿して、最新情報を公開し、コミュニティに参加します。
- 一貫性のある明確なソーシャルメディアコミュニケーションのためのテンプレートを用意します。

4. 内部コミュニケーションを調整する: AWS Chatbot などのツールを使用して、チームの調整やコミュニケーションのための内部プロトコルを実装します。CloudWatch ダッシュボードでステータスを知らせます。
5. 専用のツールとサービスでコミュニケーションを調整する:
   - AWS Systems Manager Incident Manager と AWS Chatbot を使用して、インシデントの発生時にリアルタイムで内部コミュニケーションと調整を行うための専用チャットチャネルを設置します。
   - AWS Systems Manager Incident Manager ランブックを使用して、インシデントの発生時に Amazon Pinpoint、Amazon SNS、またはソーシャルメディアプラットフォームなどのサードパーティーツールを通じて顧客への通知を自動化します。
   - ランブックに承認ワークフローを組み込んで、すべての外部コミュニケーションを送信前に任意で確認し、承認できます。
6. 実践して改善する:
   - コミュニケーションツールと戦略の利用に関するトレーニングを実施します。インシデントの発生時にチームがタイムリーな意思決定を行えるようにします。
   - 定期的な訓練やゲームデーを設けて、コミュニケーションプランをテストします。これらのテストを基にメッセージを改良し、チャネルの有効性を評価してください。
   - インシデント発生時のコミュニケーションの有効性を評価するためのフィードバックメカニズムを実装します。フィードバックと変化するニーズに応じて、コミュニケーションプランを継続的に進化させます。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- OPS07-BP03 ランブックを使用して手順を実行する
- OPS10-BP06 ダッシュボードでステータスを知らせる
- OPS11-BP02 インシデント後の分析を実行する

関連ドキュメント:

- Atlassian - インシデントコミュニケーションのベストプラクティス
- Atlassian - 効果的なステータスアップデートの記述方法
- PagerDuty - インシデントコミュニケーションガイド

関連動画:

- Atlassian - 独自のインシデントコミュニケーションプランの作成: インシデントテンプレート

関連する例:

- AWS Health ダッシュボード
- AWS のステータスの最新情報の例

OPS10-BP06 ダッシュボードでステータスを知らせる

ダッシュボードを戦略的なツールとして使用して、内部の技術チーム、経営陣、顧客など、さまざまな対象者にリアルタイムの運用状況と主要なメトリクスを伝えます。これらのダッシュボードでは、システムの状態とビジネスパフォーマンスを一元的に視覚化できるため、透明性と意思決定の効率が向上します。

期待される成果:

- ダッシュボードには、さまざまなステークホルダーに関連するシステムとビジネスのメトリクスが包括的に表示されます。
- ステークホルダーは運用情報に積極的にアクセスできるため、状況確認のリクエストを頻繁に行う必要がなくなります。
- 通常運用中やインシデント発生時には、リアルタイムの意思決定が強化されます。

一般的なアンチパターン:

- インシデント管理の会議に参加するエンジニアが、最新状況を把握するために、状況確認のリクエストをしなければならない。
- 管理面は手作業による報告に頼っているため、遅延が起きたり正確さを欠いたりする可能性がある。

- インシデント発生時に、運用チームが最新の状況確認のために頻繁に中断される。

このベストプラクティスを活用するメリット:

- ステークホルダーが重要な情報にすぐにアクセスできるようになり、情報に基づいた意思決定が促されます。
- 手作業による報告や頻繁なステータス照会を最小限に抑えることで、運用上の非効率性が軽減されます。
- システムのパフォーマンスとビジネスのメトリクスをリアルタイムで可視化し、透明性と信頼性を高めます。

この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

ダッシュボードはシステムの状況やビジネスメトリクスを効果的に伝え、さまざまな対象者グループのニーズに合わせてカスタマイズできます。Amazon CloudWatch ダッシュボードや Amazon QuickSight などのツールを使用すれば、システムモニタリングやビジネスインテリジェンスを目的としたインタラクティブなリアルタイムダッシュボードを作成できます。

実装手順

1. ステークホルダーのニーズを特定する: 技術チーム、経営陣、顧客など、さまざまな対象者グループの特定の情報ニーズを判断します。
2. 適切なツールを選択する: システムモニタリング用の Amazon CloudWatch ダッシュボードや、インタラクティブなビジネスインテリジェンス用の Amazon QuickSight などの適切なツールを選択します。
3. 効果的なダッシュボードを設計する:
    - 関連するメトリクスと KPI をわかりやすく提示するダッシュボードを設計し、それらの情報が理解しやすく、すぐに行動に結び付くようにします。
    - 必要に応じて、システムレベルとビジネスレベルのビューを組み込みます。
    - 高レベル (大まかな概要用) と低レベル (詳細な分析用) のダッシュボードの両方を含めます。
    - 重大な問題を強調するため、自動アラームをダッシュボードに統合します。
    - ダッシュボードに重要なメトリクスのしきい値と目標を示す注釈を付け、すぐに視認できるようにします。
4. データソースを統合する:

- Amazon CloudWatch を使用すると、さまざまな AWS サービスのメトリクスを集約して表示したり他のデータソースのメトリクスをクエリしたりして、システムの正常性とビジネスメトリクスを一元的に把握できます。
- CloudWatch Logs Insights のような機能を使用して、さまざまなアプリケーションやサービスのログデータをクエリしたり可視化したりすることを可能にします。

5. セルフサービスアクセスを可能にする:
   - 関連するステークホルダーと CloudWatch ダッシュボードを共有し、ダッシュボード共有機能を使ってセルフサービスで情報にアクセスできるようにします。
   - ダッシュボードに簡単にアクセスできるようにし、リアルタイムで最新情報が提供されるようにします。
6. 定期的に更新して改良する:
   - 進化するビジネスニーズとステークホルダーのフィードバックに応じて、ダッシュボードを継続的に更新し、改良していきます。
   - ダッシュボードを定期的に見直し、必要な情報を伝えるために適切かつ効果的であり続けるようにします。

リソース

関連するベストプラクティス:

- OPS08-BP05 ダッシュボードを作成する

関連ドキュメント:

- 運用を可視化するためのダッシュボードの構築
- Amazon CloudWatch ダッシュボードの使用
- ダッシュボード変数を使用して柔軟なダッシュボードを作成する
- CloudWatch ダッシュボードの共有
- 他のデータソースにあるメトリクスへのクエリ
- CloudWatch ダッシュボードにカスタムウィジェットを追加する

関連する例:

- 1 つのオブザーバビリティワークショップ - Dashboards

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することは、迅速で一貫性があり、ミスのない運用処理を実現するために不可欠です。プロセスを合理化し、ツールを使用してイベントを自動的に管理および対応することで、手作業による介入を極力なくし、運用効率を高めます。

期待される成果:

- 自動化を通じて、ヒューマンエラーを抑制し、解決所要時間を短縮できる。
- 一貫性があり信頼できる運用上のイベント処理。
- 運用効率とシステムの信頼性が向上する。

一般的なアンチパターン:

- 手作業によるイベント処理は、遅延やミスにつながりやすい。
- 反復的でありながら重要なタスクに対し、自動化が見過ごされる。
- 繰り返しのタスクを手作業で行うと、アラート疲労が起きやすく、重大な問題を見逃しかねない。

このベストプラクティスを活用するメリット:

- イベントへの対応を迅速化し、システムのダウンタイムを短縮する。
- 自動化された一貫したイベント処理による、信頼性の高い運用。

この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

自動化を組み込んで運用ワークフローを効率化し、手作業による介入を極力抑えます。

実装手順

1. 自動化の機会を特定する: 問題の修正、チケットの強化、容量管理、スケーリング、デプロイ、テストなど、自動化の余地がある反復的なタスクを判断します。
2. 自動化のプロンプトを特定する:
   - 自動応答の契機となる特定の条件やメトリクスを Amazon CloudWatch アラームアクションを使用して評価し、定義します。
   - Amazon EventBridge を使用して、AWS サービス、カスタムワークロード、SaaS アプリケーションでイベントに対応します。

- AWS リソースでの特定のログエントリ、パフォーマンスメトリクスのしきい値、状態の変化など、契機となるイベントを検討します。

3. イベント駆動型の自動化を実装する:
   - AWS Systems Manager オートメーションランブックを使用して、メンテナンス、デプロイ、修正のタスクを簡素化します。
   - Incident Manager でインシデントを作成して、関連する AWS リソースに関する情報を自動的に収集し、インシデントに追加します。
   - AWS のクォータモニタを使用してクォータをプロアクティブにモニタリングします。
   - AWS Auto Scaling を使用して容量を自動的に調整し、可用性とパフォーマンスを維持します。
   - Amazon CodeCatalyst を使用して開発パイプラインを自動化します。
   - 合成モニタリングを使用して、エンドポイントと API をスモークテストするか継続的にモニタリングします。
4. 自動化を通じてリスクを軽減する:
   - リスクに迅速に対処するため自動化されたセキュリティ対応を実施します。
   - AWS Systems Manager State Manager を使用して設定のドリフトを減らします。
   - AWS Config ルール を使用して非準拠のリソースを修復します。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- OPS08-BP04 実践的なアラートを作成する
- OPS10-BP02 アラートごとにプロセスを用意する

関連ドキュメント:

- Incident Manager での Systems Manager Automation ランブックの使用
- Incident Manager でのインシデントの作成
- AWS サービスクォータ
- リソース使用状況のモニタリングとクォータ間近の通知の送信
- AWS Auto Scaling
- Amazon CodeCatalyst とは

- Amazon CloudWatch アラームの使用
- Amazon CloudWatch でのアラームの使用
- AWS Config ルール による非準拠リソースの是正
- 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
- AWS Systems Managerステートマネージャー

関連動画:

- Create Automation Runbooks with AWS Systems Manager
- How to automate IT Operations on AWS
- AWS Security Hub automation rules
- Start your software project fast with Amazon CodeCatalyst blueprints

関連する例:

- Amazon CodeCatalyst Tutorial: Creating a project with the Modern three-tier web application blueprint
- 1 つのオブザーバビリティワークショップ
- Respond to incidents using Incident Manager

進化

質問

- OPS 11. 運用はどのように進化させるのですか?

OPS 11. 運用はどのように進化させるのですか?

ほぼ継続的で漸進的な改善に時間とリソースを費やすことで、オペレーションの効果と効率を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。頻繁かつ意図的なワークロード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:

- 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頻繁に分析します。
- ソフトウェア開発プロセスにおいて、新機能の開発と改善の機会に同程度の優先順位を与えます。

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。
- 改善の機会の優先順位が低い。新機能と比較して、これらの機会は未処理のままである。
- 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。
- ワークロードを意図を持って進化させることができます。
- 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。
- わずかなメリットが累積的な影響をもたらし、効率性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: 高

実装のガイダンス

ワークロードの構造的レビューを頻繁に実施します。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。

実装手順

1. 合意された頻度で、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、社内で定義された基準を使用します。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します。
   c. AWS ソリューションアーキテクトまたはテクニカルアカウントマネージャーに連絡して、ワークロードのガイド付き Well-Architected Framework レビューを実施します。
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
- OPS11-BP08 教訓を文書化して共有する
- OPS04 オブザーバビリティを実装する

関連ドキュメント:

- 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
- カスタムレンズと AWS Well-Architected Tool で Well-Architected レビューをカスタマイズする
- AWS Well-Architected カスタムレンズライフサイクルを組織に実装する

関連動画:

- Well-Architected ラボ - レベル 100: AWS Well-Architected Tool のカスタムレンズ
- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

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。

期待される成果:

- インシデント後の分析を含むインシデント管理プロセスが確立されます。
- イベントに関するデータを収集するためのオブザーバビリティ計画が整います。
- このデータから、インシデント後の分析プロセスを支えるメトリクスを理解し、収集できます。
- インシデントから学び、その後の成果の向上につなげることができます。

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。
- インシデント後の分析のデータを改善に役立てます。

この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します。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。インシデントの根本原因を適宜伝達し、伝える相手に合わせて伝え方を調整します。教訓を組織内で広く共有します。

実装手順

1. デプロイの変更、構成変更、インシデントの開始時刻、アラーム時刻、エンゲージメント時間、緩和開始時刻、インシデント解決時刻などのメトリクスを収集します。
2. タイムライン上で重要な時点を特定し、インシデントの該当時点のイベントを把握します。
3. 次の質問について検討します。
    a. 検出までの時間を短縮できますか?
    b. インシデントを早く検出するメトリクスとアラームの更新はありますか。
    c. 診断までの時間を短縮できますか?
    d. 対応計画またはエスカレーション計画の更新があり、正しい応答者をより早くエンゲージすることはありますか。
    e. 緩和までの時間を短縮できますか?
    f. ランブックやプレイブックに追加または改善できる手順はありますか?
    g. 今後のインシデントの発生を防止できますか?
4. チェックリストとアクションを作成します。すべてのアクションを追跡し、実行します。

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:

- OPS11-BP01 継続的改善のプロセスを用意する

- OPS 4 - オブザーバビリティを実装する

関連ドキュメント:

- Incident Manager でのインシデント後分析の実行
- 運用準備状況レビュー

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と遡及分析の 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。

お客様事例

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 を使用することで、これらの改善を OpsItems として作成し追跡することができます。
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 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。

関連ドキュメント:

- CCOE を構築するときに避けるべき 7 つの落とし穴
- Atlassian Team Playbook - Retrospectives
- E メール定義: 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いたフィードバックループの確立
- IBM Garage Methodology - Hold a retrospective

- Investopedia – The PDCS Cycle
- Maximizing Developer Effectiveness by Tim Cochran
- 運用準備状況レビュー (ORR) ホワイトペーパー - イテレーション
- ITIL CSI - Continual Service Improvement
- When Toyota met e-commerce: Lean at Amazon

関連動画:

- Building Effective Customer Feedback Loops

関連する例:

- Astuto - Open source customer feedback tool
- AWS ソリューション - QnABot on AWS
- Fider - A platform to organize customer feedback

関連サービス:

- AWS Systems Manager OpsCenter

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を活用しない場合のリスクレベル: 高

実装のガイダンス

ナレッジ管理は、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。

関連ドキュメント:

- Atlassian - Knowledge Management

関連する例:

- DokuWiki
- gollum
- MediaWiki
- Wiki.js

OPS11-BP05 改善の推進要因を定義する

データとフィードバックループに基づいて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。システムやプロセスの改善機会を探り、適切な場合は自動化します。

期待される成果:

- 環境全体のデータを追跡します。

- イベントやアクティビティをビジネスの成果に関連付けます。
- 環境とシステムを比較対照できます。
- デプロイと結果の詳細なアクティビティ履歴を管理できます。
- セキュリティ体制をサポートするためのデータを収集します。

一般的なアンチパターン:

- 環境全体からデータを収集していますが、イベントとアクティビティの関連付けは行っていません。
- 資産全体から詳細なデータを収集しているため、Amazon CloudWatch および AWS CloudTrail のアクティビティとコストの増加につながっています。ただし、このデータを有意義に使用することはできていません。
- 改善の推進要因を定義する際、ビジネス成果を考慮していません。
- 新機能の効果は測定していません。

このベストプラクティスを活用するメリット:

- 改善の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。
- 技術的なイベントだけでなく、ビジネスイベントにも対応できます。
- 環境を測定して、改善すべき領域を特定します。

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: 中

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。適切なサイジングオプションを追跡し、最適化の機会を探します。
    - AWS セキュリティ速報
    - AWS Trusted Advisor

    - クラウドインテリジェンスダッシュボード
- 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 Compliance Programs
    - AWS コンプライアンスの最新情報

リソース

関連するベストプラクティス:

- OPS01 組織の優先順位
- OPS02 関係性と所有権
- OPS04-BP01 主要業績評価指標を特定する
- OPS08 ワークロードのオブザーバビリティの活用
- OPS09 運用状態の把握
- OPS11-BP03 フィードバックループを実装する

関連ドキュメント:

- Amazon Athena
- Amazon QuickSight
- AWS コンプライアンス
- AWS コンプライアンスの最新情報
- AWS Compliance Programs
- AWS Glue
- AWS セキュリティ速報
- AWS Trusted Advisor
- Export your log data to Amazon S3
- AWS の最新情報
- 顧客中心のイノベーションの必要性
- デジタルトランスフォーメーション: 誇大広告? それとも戦略的必然?

関連動画

- AWS re:Invent 2023 - Improve operational efficiency and resilience with AWS Support (SUP310)

OPS11-BP06 インサイトを検証する

分析結果を確認して部門横断的な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。

期待される成果:

- ビジネスオーナーと定期的にインサイトを見直します。ビジネスオーナーは、新たに得たインサイトに追加のコンテキストを提供します。
- インサイトを確認して技術者にフィードバックを求め、学んだことをチーム間で共有します。
- 他の技術チームやビジネスチームが確認できるようにデータやインサイトを公開します。学んだことを他の部署の新しい実践に取り入れます。
- シニアリーダーと共に新しいインサイトをまとめ、レビューします。シニアリーダーは、新しいインサイトを活用して戦略を定義します。

一般的なアンチパターン:

- 新しい機能をリリースします。この機能により、顧客の行動の一部が変わります。オブザーバビリティではこうした変更が考慮に入れられておらず、こうした変更のメリットの定量化も行われていません。
- 新しいアップデートをプッシュしますが、CDN は更新されません。CDN キャッシュは最新リリースとの互換性がなくなります。エラーのあるリクエストの割合を測定します。バックエンドサーバーとの通信時に、すべてのユーザーが HTTP 400 エラーを報告します。クライアントのエラーを調査したところ、誤ったディメンションを測定したために時間を無駄にしていたことがわかりました。
- サービスレベル契約では 99.9% のアップタイムが規定されており、目標復旧時間は 4 時間です。サービスオーナーは、システムのダウンタイムはゼロだと主張しています。高価で複雑なレプリケーションソリューションを実装すると、時間と費用が無駄になります。

このベストプラクティスを活用するメリット:

- 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。
- 隠れた問題を発見し、それを将来の意思決定に取り入れることができます。
- 技術的な成果からビジネスの成果にフォーカスを移します。

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

- 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。

リソース

関連するベストプラクティス:

- OPS01-BP06 メリットとリスクを管理しながらトレードオフを評価する
- OPS02-BP06 チーム間の責任は事前定義済みまたは交渉済みである
- OPS11-BP03 フィードバックループを実装する

関連ドキュメント:

- Cloud Center of Excellence (CCOE) の設計

関連動画:

- Building observability to increase resiliency

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運用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、本番など) で改善する機会を探します。

期待される成果:

- ビジネスに影響するメトリクスを頻繁に確認する

- オブザーバビリティ機能を通じて異常を検出し確認する
- データをビジネスの成果と目標の裏付けに使用する

一般的なアンチパターン:

- 大規模な販促活動によってメンテナンスウィンドウが中断されます。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。
- 組織で古いライブラリを頻繁に使用していたため、長い時間システムが停止しました。その後、サポートされている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。
- 顧客の SLA の達成状況を定期的に確認していません。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、イベント、インシデントを定期的に確認することで、チーム間の共通理解を維持します。
- チームは定期的にミーティングを行い、メトリクスやインシデントを確認します。これにより、リスクに対処し、顧客の SLA を確認できます。
- 学んだ教訓を共有することで、ビジネス成果の優先順位付けや目標とする改善のためのデータが得られます。

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: 中

実装のガイダンス

- ビジネスのさまざまな分野のチームメンバー間で運用メトリクスの遡及分析を定期的に実施します。
- ビジネス、開発、オペレーションチームを含むステークホルダーを参加させて、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。
- それらのインサイトに基づいて、改善の機会と取り得る一連のアクションを特定します。

リソース

関連するベストプラクティス:

- OPS08-BP05 ダッシュボードを作成する
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け
- 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

関連ドキュメント:

- Amazon CloudWatch
- CloudWatch メトリクスを発行する AWS のサービス
- 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
- CloudWatch を使用したダッシュボードとビジュアライゼーション

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、開発作業を容易にして、期待される機能の提供にフォーカスします。

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。

期待される成果:

- バージョン管理された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します。
- インフラストラクチャ標準は、バージョン管理された AWS CloudFormation テンプレートとして共有します。
- チーム全体で学んだ教訓を確認します。

一般的なアンチパターン:

- 組織でバグが含まれているライブラリを頻繁に使用していたため、長い時間システムが停止しました。その後、チーム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ことを知りません。このライブラリでの経験が文書化や共有されていないため、誰もリスクに気づいていません。

- あるユーザーが、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。そのユーザーは、このエッジケースを回避するために、サービスへの自分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。
- 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が見つかりました。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。例えば、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。
  - Delegating access to your AWS environment
  - AWS CodeCommit リポジトリの共有

リソース

関連するベストプラクティス:

- OPS02-BP06 チーム間の責任は事前定義済みまたは交渉済みである
- OPS05-BP01 バージョン管理を使用する
- OPS05-BP06 設計標準を共有する
- OPS11-BP03 フィードバックループを実装する
- OPS11-BP07 オペレーションメトリクスのレビューを実行する

関連ドキュメント:

- Docs-as-Code ソリューションによるプロジェクト遅延の軽減

関連動画:

- Delegating access to your AWS environment
- AWS Supports You | Exploring the Incident Management Tabletop Exercise

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。

期待される成果:

- 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。
- ゲームデーを実施し、Fault Injection Service (FIS) を使用して、チームが本番環境に似た環境で実験を行うために必要な制御とガードレールを提供します。

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。当該問題は、すべての計画された機能実装の背後にあるバックログに追加されます。計画された機能が一定の割合で追加され続ければ、パフォーマンスの問題は解決しません。
- 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。
- 運用上の承認が完了した後は、運用プラクティスの再テストを行っていません。

このベストプラクティスを活用するメリット: 時間とリソースをプロセス内に設けることで、漸進的な改善を継続的に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

- 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のために、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検討します。
- ゲームデーを通して本番環境のワークロードをシミュレートし、これらのシミュレーションから学んだことを改善に生かします。

リソース

関連するベストプラクティス:

- OPS05-BP08 複数の環境を使用する

関連動画:

- AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service

セキュリティ

セキュリティの柱では、データ、システム、資産を保護し、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

- セキュリティ基盤
- ID およびアクセス管理
- 検出
- インフラストラクチャの保護
- データ保護
- インシデントへの対応
- アプリケーションのセキュリティ

セキュリティ基盤

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ためには、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。「運用上の優秀性」で定義された要件とプロセスを組織やワークロードのレベルで作成し、あらゆる領域に適用します。AWS や業界の推奨事項と脅威インテリジェンスを最新の状態に保つことで、脅威モデルを進化させ、目標を制御できます。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スケールできます。

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 セキュリティの脅威と推奨事項の最新情報を入手する
- SEC01-BP05 セキュリティ管理のスコープを縮小する
- SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する
- 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、関連しないワークロード、環境を別々のアカウントに分離し、クラウドインフラストラクチャ全体のセキュリティを強化するアカウント構造。

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を軽減。

- 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 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概念、パターン、実装に関するガイダンスについては、「複数のアカウントで AWS 環境を構成する」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。優れた設計では継承を利用して、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 と AWS Control Tower は、AWS 環境でマルチアカウント構造を実装および管理するサービスです。AWS Organizations では、アカウントを OU の 1 つ以上のレイヤーで定義された階層に整理できます。各 OU には複数のメンバーアカウントが含まれます。サービスコントロールポリシー (SCP) を使用すると、組織管理者はメンバーアカウントに対してきめ細かな予防的コントロールを確立できます。AWS Config を使用してメンバーアカウントに対してプロアクティブコントロールと検出コントロールを確立できます。多くの AWS サービスは AWS Organizations と統合して、委任された管理コントロールを提供し、組織内のすべてのメンバーアカウントでサービス固有のタスクを実行します。

AWS Organizations の上にレイヤー化される AWS Control Tower は、ランディングゾーンを持つマルチアカウント AWS 環境のワンクリックのベストプラクティス設定を提供します。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 よりも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コントロール。
- 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コントロール。
- AWS Control Tower Account Factory は、事前承認されたベースラインと設定オプションを含むアカウントを組織内に自動デプロイします。

実装手順

1. 組織単位構造の設計: 適切に設計された組織単位構造により、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維持に必要な管理負担が軽減されます。組織単位の構造は、ビジネスニーズ、データ機密性、ワークロード構造と整合させる必要があります。
2. マルチアカウント環境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、組織がワークロードを迅速に開発、起動、デプロイできる一貫したセキュリティとインフラストラクチャの基盤を提供します。カスタム構築のランディングゾーンまたは AWS Control Tower を使用して、環境をオーケストレーションできます。
3. ガードレールの確立: ランディングゾーンを通して環境に一貫したセキュリティガードレールを実装します。AWS Control Tower には、デプロイできる一連の必須およびオプションのコントロールが用意されています。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、有効に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 を使用する場合はコンソールから、または AWS Organizations の IAM アクセス権限ポリシーを調整することで実行できます。
5. AWS CloudFormation 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できます。

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 を利用した複数の AWS アカウントやリージョンを横断したリソース展開
- Organizations に関するよくある質問
- AWS Organizations の用語と概念
- マルチアカウント環境における AWS Organizations サービスコントロールポリシー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
- 複数のアカウントで AWS 環境を構成する

関連動画:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- AWS Control Tower を使用した複数アカウントのビルドと管理
- 既存の組織で Control Tower を有効にする

関連するワークショップ:

- 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: ルートユーザーのセキュリティを保護することで、ルートユーザーの認証情報が不正利用された場合に付帯的または意図的な損害が発生する可能性を抑えることができます。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で、アカウントで行われるアクションは制御され監査されているという安心感が得られます。

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

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。アカウントの作成に使用したメールアドレスとパスワードを使用して、ルートユーザーとしてサインインできます。AWS ルートユーザーに付与されるアクセス権が昇格しているため、特にそれが必要とな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厳重に保護し、AWS アカウントルートユーザーには必ず多要素認証 (MFA) を使用します。

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 benchmark version 1.4.0 に対応します。AWS アカウントとリソースのセキュリティを保護するときは、これらのステップに加えて、AWS のベストプラクティスの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. アカウントの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既にある場合は、そのキーを使用するプロセスを移行させて、AWS Identity and Access Management (IAM) ロールの一時的なアクセスキーを使用した後、ルートユーザーのアクセスキーを削除する必要があります。
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。必要に応じてメンバーアカウントへのアクセス権を取得するときは、AWS Organization 管理アカウントのパスワードリセットフローを使用することを検討します。

   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。ルートユーザーに MFA を使用します。

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は、予防的コントロールを使用します。

   a. メンバーアカウントには、ルートユーザー向けのルートアクセスキーの作成を拒否する予防的ガードレールの使用を検討します。

   b. メンバーアカウントには、ルートユーザーとしてのアクションを拒否する予防的ガードレールの使用を検討します。

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 を設定するときは、仮想 MFA またはハードウェア MFA デバイスのいずれかを作成する手順に従います。

   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ます。MFA デバイスは 1 アカウントにつき 8 台まで登録できます。

      - 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自動的に無効になります。

   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。
  - ルートユーザーのパスワードを定期的に変更することで、意図せず漏洩したパスワードが不正利用されるリスクを減らせます。

「発見的コントロール」

- 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 を使用すると、RootCredentialUsage の検出結果を使ってルートユーザー API 認証情報の使用状況をモニタリングしたり、アラートを発したりすることができます。
- 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管理を、評価し実装します。AWS Control Tower を使用している場合は、Control Tower 内で提供されている強く推奨されるコントロールを評価し実装します。

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 タスクを実行するときは、必ずルートユーザーとしてログインします。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ている E メールアドレスと代理連絡先に記載されている 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、セキュリティに関する通知が <abuse@amazon.com> から届いていないか監視します。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策定に関する詳細は、「AWS セキュリティインシデント対応

ガイド」と、ホワイトペーパー「セキュリティの柱」の「インシデント対応」セクションにある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する
- 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 – root credential usage alert
- CloudTrail を使用してルート認証情報の使用状況をモニタリングするためのステップバイステップガイド
- AWS での使用が承認された MFA トークン
- AWS に Break Glass アクセスを実装する
- AWS アカウントのセキュリティを改善するための 10 個の項目
- 自分の AWS アカウントで不正なアクティビティに気づいた場合はどうすればよいですか?

関連動画:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Limiting use of AWS root credentials from AWS re:inforce 2022 – Security best practices with AWS IAM

関連する例とラボ:

- Lab: AWS アカウント setup and root user

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。

期待される成果: ビジネスのセキュリティコントロール上の目標が、コンプライアンス要件に一致するように明確に定義されます。自動化とポリシーを通じて統制が実装および実施され、目標を達成するために有効かどうかが継続的に評価されています。ある時点および一定期間の双方における有効性を示す証拠を、監査担当者にすぐに報告可能です。

一般的なアンチパターン:

- セキュリティを保証するために守るべき規制要件、市場の期待、業界標準についての理解が、ビジネスにとって不十分である
- サイバーセキュリティフレームワークと統制目標が、ビジネスの要件とかみ合っていない
- 実施されている統制が、測定可能な方法で統制目標にしっかりと適合していない
- 統制の有効性の報告に自動化を使っていない

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

セキュリティ統制目標の基礎として活用できる、一般的なサイバーセキュリティフレームワークは多数あります。ビジネスの規制要件、市場の期待、業界標準を考慮し、どのフレームワークがニーズに最も適しているかを判断してください。例えば、AICPA SOC 2、HITRUST、PCI-DSS、ISO 27001、NIST SP 800-53 などがあります。

特定した統制目標について、利用する AWS サービスがそれらの目標の達成にどのように役立つかを理解してください。目標とするフレームワークに沿ったドキュメントやレポートを探すときは AWS Artifact を使用します。これらの文書には、AWSが引き受ける責任の範囲と、顧客が引き受けるそれ以外の範囲についてのガイダンスが記されています。さまざまなフレームワークのコントロールステートメントに沿った、サービス固有のガイダンスの詳細については、「AWS Customer Compliance Guides」を参照してください。

目標達成を目指して統制を定義する際は、予防的統制を用いて実施について明文化し、発見的統制を用いて緩和策を自動化します。サービスコントロールポリシー (SCP) を使用すると、AWS

Organizations 全体で非準拠のリソース構成やアクションを予防することができます。非準拠のリソースを監視および報告するときは AWS Config にルールを適用し、動作を確認できたらルールを強制モデルに切り替えます。自社のサイバーセキュリティのフレームワークに合わせて、事前定義されたマネージドルール一式を導入するときは、最初の選択肢として AWS Security Hub の標準の使用を検討します。AWS Foundational Service Best Practices (FSBP) 標準と CIS AWS Foundations Benchmark は、複数の標準フレームワークに共通の多数の目標に沿った統制を実現できるため、出発点として優れています。Security Hub に希望する統制検出の機能が組み込まれていない場合は、AWS Config コンフォーマンスパックで補完できます。

AWS Global Security and Compliance Acceleration (GSCA) チームが推奨する APN パートナーバンドルを使用すると、セキュリティアドバイザー、コンサルティング機関、証拠収集および報告のシステム、監査人、その他補完サービスによる支援を必要に応じて利用できます。

実装手順

1. 一般的なサイバーセキュリティフレームワークを評価し、選択したフレームワークに合わせて統制目標を定めます。
2. AWS Artifact を使用して、フレームワークのガイダンスと責任に関する関連文書を入手します。責任共有モデルにおいて、コンプライアンスのどの部分が AWS 側の責任で、どの部分が貴社の責任であるかを理解します。
3. SCP、リソースポリシー、ロール信頼ポリシー、その他のガードレールを使用して、非準拠のリソース構成やアクションを防止します。
4. 自社の統制目標に沿った Security Hub の標準と AWS Config コンフォーマンスパックの導入を評価します。

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
- 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
- SEC07-BP01 データ分類スキームを理解する
- 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
- PERF01-BP05 ポリシーとリファレンスアーキテクチャを使用する
- COST02-BP01 組織の要件に基づいてポリシーを策定する

関連ドキュメント:

- AWS Customer Compliance Guides

関連ツール:

- AWS Artifact

SEC01-BP04 セキュリティの脅威と推奨事項の最新情報を入手する

業界の脅威インテリジェンスの公開情報やデータフィードが更新されていないか監視して、脅威や緩和策の最新情報を常に把握します。最新の脅威データに基づいて自動更新されるマネージドサービスを評価してください。

期待される成果: 業界で最新の脅威や推奨事項が公表されると同時にそれらを把握できます。 自動化を活用して、新たな脅威を特定した時点で、潜在的な脆弱性や工クスポージャを検出します。これらの脅威に対して緩和措置を講じます。 最新の脅威インテリジェンスで自動的に更新される AWS サービスを採用します。

一般的なアンチパターン:

- 最新の脅威インテリジェンスを常に把握するための、信頼性と再現性が高いメカニズムがない。
- テクノロジーポートフォリオ、ワークロード、依存関係の手動インベントリを保持していて、潜在的な脆弱性や工クスポージャについて人間がレビューする必要がある。
- ワークロードと依存関係を、既知の脅威に対する緩和策が盛り込まれた最新バージョンに更新するメカニズムが導入されていない。

このベストプラクティスを活用するメリット: 脅威インテリジェンスの情報源から最新の情報を入手することで、脅威の分野における重要な変化を見落としてビジネスに影響を及ぼすリスクを避けることができます。 ワークロードやその依存関係をスキャンして、脆弱性や工クスポージャが潜んでいる箇所を検出および修正するための自動化体制が整い、手動での代替手段と比較して、リスクを迅速かつ予測どおりに軽減できます。 これにより、脆弱性の緩和に関連する時間やコスト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

信頼できる脅威インテリジェンスの公開情報を確認して、脅威の状況を常に把握してください。 既に周知の脅威への対抗手段、テクニック、手順 (TTP) に関するドキュメントは、MITRE ATT&CK のナレッジベースでご覧いただけます。MITRE の「Common Vulnerabilities and Exposures (CVE)」リストでは、ご利用の製品に関する既知の脆弱性情報を入手できます。Open Worldwide Application Security Project (OWASP) の中でも人気の高い OWASP Top 10 プロジェクトでは、ウェブアプリケーションの重大なリスクを把握できます。

AWS のセキュリティイベントと推奨される修復手順に関する最新情報は、CVE の AWS セキュリティ速報で入手できます。

最新情報を入手するための負担やオーバーヘッドを全体的に減らすために、新しい脅威インテリジェンスを自動で随時取り込む AWS サービスの使用を検討してください。 例えば、Amazon GuardDuty は業界の脅威インテリジェンスの最新情報を常に把握し、アカウント内の異常動作や脅威の兆候の検出に役立てています。 Amazon Inspector は、継続スキャンに使用している CVE のデータベースを自動更新しています。 AWS WAF と AWS Shield Advanced はどちらも、新しい脅威が登場すると自動的に更新されるマネージドルールグループを利用しています。

フリートの自動管理とパッチ適用については、「運用上の優秀性の柱 – AWS Well-Architected フレームワーク」を参照してください。

実装手順

- ビジネスや業界に関連する脅威インテリジェンスの最新公開情報を購読します。AWS セキュリティ速報を購読します。
- Amazon GuardDuty や Amazon Inspector など、新しい脅威インテリジェンスを自動的に組み込むサービスの採用を検討してください。
- Well-Architected 運用上の優秀性の柱のベストプラクティスに沿って、フリート管理とパッチ適用の戦略をデプロイします。

リソース

関連する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
- OPS01-BP05 脅威の状況を評価する
- OPS11-BP01 継続的改善のプロセスを用意する

SEC01-BP05 セキュリティ管理のスコープを縮小する

特定の統制の管理を AWS に移行する AWS サービス (マネージドサービス) を利用することで、セキュリティの範囲を縮小できるか判断します。そうしたサービスを導入することで、インフラストラクチャのプロビジョニング、ソフトウェアのセットアップ、パッチ適用、バックアップなどのセキュリティメンテナンスのタスクを軽減できます。

期待される成果: ワークロードに適した AWS サービスを選ぶときに、セキュリティ管理の範囲を検討します。管理オーバーヘッドとメンテナンスタスクのコスト (総保有コスト (TCO)) が、Well-Architected の他の考慮事項に加えて、選択したサービスのコストと比較検討されます。統制の評価と検証の手順に、AWS の統制とコンプライアンスの文書が組み込まれています。

一般的なアンチパターン:

- 選択したサービスの責任共有モデルをしっかりと理解しないまま、ワークロードをデプロイする。
- データベースやその他のテクノロジーを、同等のマネージドサービスを評価することなく仮想マシンでホストする。
- マネージドサービスオプションと比較する際に、仮想マシンでテクノロジーをホストする場合の総保有コスト (TCO) にセキュリティ管理タスクを考慮していない。

このベストプラクティスを活用するメリット: マネージドサービスを使用すると、運用上のセキュリティ統制を管理する全体的な負担を軽減でき、セキュリティリスクと総保有コストを減らすことができます。本来なら特定のセキュリティタスクに費やしていた時間を、ビジネスに付加価値をもたらすタスクに使うことができます。マネージドサービスを利用すれば、一部の統制要件を AWS に移し、コンプライアンス要件のスコープを縮小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: 中

実装のガイダンス

AWS では、多数の方法でワークロードのコンポーネントを統合できます。多くの場合、Amazon EC2 インスタンスにテクノロジーをインストールして実行するには、セキュリティの責任の大部分を自社で引き受けなければなりません。特定の統制の運用負担を軽減するには、責任共有モデルにおける自社の責任範囲が狭くなる AWS マネージドサービスを特定し、それらを既存のアーキテクチャでどのように使用できるかを理解してください。例えば、データベースのデプロイに Amazon Relational Database Service (Amazon RDS) を使用する、コンテナのオーケストレーションに Amazon Elastic Kubernetes Service (Amazon EKS) または Amazon Elastic Container Service (Amazon ECS) を使用する、もしくはサーバーレスのオプションを使用する、といったことが挙げら

れます。新しいアプリケーションを構築するときは、セキュリティ統制の実装と管理に関して、どのサービスが時間とコストの削減に役立つかを考えてください。

コンプライアンス要件も、サービス選択時の検討材料となり得ます。マネージドサービスでは、一部の要件のコンプライアンスを AWS に移すことができます。サービスの自社で運用管理する側面の監査や、関連する AWS 監査報告書の統制に関するステートメントの受け入れがどの程度容易かをコンプライアンスチームと話し合ってください。AWS Artifact で検出した監査アーティファクトは、AWS セキュリティ統制の証拠として監査人または規制当局に提出することができます。また、AWS の監査アーティファクトの一部によって提供された責任ガイダンスを、「AWS Customer Compliance Guides」と併せて使用することで、アーキテクチャを設計することもできます。このガイダンスは、システムの特定のユースケースをサポートするために導入すべき追加のセキュリティ統制を決定するうえで役立ちます。

マネージドサービスを使用するときは、リソースを新しいバージョンに更新するプロセス (Amazon RDS で管理されるデータベースのバージョンや、AWS Lambda 関数のプログラミング言語ランタイムの更新など) をよく理解しておきましょう。そうした操作はマネージドサービスで実行される場合もありますが、更新のタイミングを設定し、運用への影響を理解することは依然としてお客様の責任です。AWS Health のようなツールを使用すると、こうした更新を自社の環境全体で追跡し管理することができます。

実装手順

1. マネージドサービスで置き換え可能なワークロードのコンポーネントを評価します。
    a. ワークロードを AWS に移行する場合は、ワークロードのリホスト、リファクタリング、リプラットフォーム、再構築、または交換が必要かどうかを評価する際に、管理 (時間と費用) の削減とリスクの軽減を考慮してください。移行の開始時に追加投資を行うことで、長期的には大幅な節約になる場合があります。
2. 独自のテクノロジーデプロイをインストールして管理する代わりに、Amazon RDS などのマネージドサービスを導入することを検討します。
3. AWS Artifact の責任ガイダンスを参考にして、ワークロードに対して導入すべきセキュリティ統制を決定します。
4. 使用中のリソースのインベントリを保管し、新しいサービスやアプローチに関する最新情報を入手して、スコープを縮小する新たな機会を特定します。

リソース

関連するベストプラクティス:

- PERF02-BP01 ワークロードに最適なコンピューティングオプションを選択する
- PERF03-BP01 データアクセスとストレージの要件に最適な専用データストアを使用する
- SUS05-BP03 マネージドサービスを使用する

関連ドキュメント:

- Planned lifecycle events for AWS Health

関連ツール:

- AWS Health
- AWS Artifact
- AWS Customer Compliance Guides

関連動画:

- How do I migrate to an Amazon RDS or Aurora MySQL DB instance using AWS DMS?
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health

SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する

あらゆる AWS 環境で標準とするセキュリティ統制の開発とデプロイに際しては、最新の DevOps プラクティスを適用してください。 標準的なセキュリティ統制と構成を Infrastructure as Code (IaC) テンプレートに定義し、バージョン管理システムで変更を取り込み、CI/CD パイプラインの一環として変更をテストし、AWS 環境への変更のデプロイを自動化します。

期待される成果: IaC テンプレートで標準化されたセキュリティ統制が定義され、バージョン管理システムにコミットされます。 変更を検出し、テストと AWS 環境へのデプロイを自動化する CI/CD パイプラインが整備されています。 ガードレールが効いていて、テンプレート内の設定ミスをデプロイ前に検出し、警告します。 標準の統制が効いている環境にワークロードがデプロイされます。 チームには、承認済みのサービス構成をセルフサービスメカニズムを通じてデプロイする権限が与えられています。 統制の構成、スクリプト、関連データに対して、安全なバックアップと復旧の戦略が実施されています。

一般的なアンチパターン:

- 標準のセキュリティ統制に対する変更をウェブコンソールやコマンドラインインターフェイスを使用して手作業で行っている。
- 中央のチームが定義した統制の実装は、個々のワークロードチームによる手作業に頼っている。
- ワークロードチームの要求に応じてワークロードレベルの統制をデプロイするのは、中央のセキュリティチームに一任されている。
- セキュリティ統制の自動化スクリプトの開発、テスト、デプロイを同じ個人またはチームが担当でき、職務分離やチェックアンドバランス (抑制と均衡) が適切に機能していない。

このベストプラクティスを活用するメリット: 標準のセキュリティ統制をテンプレートに定義しておくことで、バージョン管理システムを使って変化を経時的に追跡し、比較することができます。 変更のテストとデプロイを自動化することで、プロセスが標準化されて予測可能性が高まり、デプロイの成功率が上がり、繰り返しの手作業を省くことができます。 承認済みのサービスと構成をワークロードチームがデプロイできるセルフサービスのメカニズムが用意されているため、構成ミスや誤用のリスクが軽減されます。また、開発プロセスの早い段階で統制を組み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: 中

実装のガイダンス

「SEC01-BP01 アカウントを使用してワークロードを分ける」に記載のベストプラクティスに従うと、AWS Organizations を使って管理している異なる環境ごとに、複数の AWS アカウントアカウントを抱えることになります。 これらの環境やワークロードで個別のセキュリティ統制が必要になる場合がある一方で、一部のセキュリティ統制は標準化して組織全体に適用できます。 例えば、一元管理の ID プロバイダーの統合、ネットワークとファイアウォールの定義、ログの保管と分析のための標準の場所の設定などが該当します。 Infrastructure as Code (IaC) を使用すると、アプリケーションコードの開発と同等の厳格さをインフラストラクチャのプロビジョニングに適用できるように、IaC を使用すると、標準のセキュリティ統制を同様に定義してデプロイすることができます。

セキュリティ統制は、可能な限り宣言的な方法で定義し (AWS CloudFormation で定義する場合と同じ)、ソース管理システムに保存します。 DevOps のプラクティスを使用して、統制のデプロイを自動化してリリースの予測可能性を高め、AWS CloudFormation Guard などのツールを使ってテストを自動化し、デプロイした統制の、目的とする構成からの逸脱を検出します。 CI/CD パイプラインの構築には、AWS CodePipeline、AWS CodeBuild、AWS CodeDeploy などのサービスを使用できます。これらのサービスを、他のデプロイパイプラインから分離された専用のアカウントで設定するときは、「複数のアカウントで AWS 環境を構成する」のガイダンスを考慮します。

テンプレートを定義して、AWS アカウント、サービス、構成の定義とデプロイを標準化することもできます。 この方法なら、それらの定義を中央のセキュリティチームが管理し、セルフサービスア

プローチでワークロードチームに提供できます。 これを実現する方法の 1 つが Service Catalog を使用した方法です。この方法では、ワークロードチームが独自のパイプラインデプロイに組み込みこむことのできる製品として、テンプレートをパブリッシュできます。 AWS Control Tower を使用している場合、出発点として使用できるテンプレートや統制がいくつか用意されています。 Control Tower には、ワークロードチームが、ユーザーが定義した標準を使って新しい AWS アカウントアカウントを作成できる、Account Factory という機能もあります。 新しいアカウントが必要だとワークロードチームが判断した際に、その承認と作成を中央のチームに依存する必要がなくなります。 これらのアカウントは、さまざまなワークロードコンポーネントを、それぞれの機能や動作、処理対象のデータの機密性といった理由で分離する場合に必要になることがあります。

実装手順

1. テンプレートをバージョン管理システムに保存し、管理する方法を決定します。
2. テンプレートをテストおよびデプロイする CI/CD パイプラインを作成します。 設定ミスがないかチェックし、テンプレートが会社の標準に準拠していることを確認するテストを定義します。
3. ワークロードチームが要件に従って AWS アカウントやサービスをデプロイできるように、標準化されたテンプレートのカタログを作成しておきます。
4. 統制の構成、スクリプト、関連データに対して、安全なバックアップと復旧の戦略を実装します。

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
- OPS05-BP04 構築およびデプロイ管理システムを使用する
- REL08-BP05 オートメーションを使用して変更をデプロイする
- 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

関連ドキュメント:

- 複数のアカウントで AWS 環境を構成する

関連する例:

- Automate account creation, and resource provisioning using Service Catalog, AWS Organizations, and AWS Lambda

- Strengthen the DevOps pipeline and protect data with AWS Secrets Manager, AWS KMS, and AWS Certificate Manager

関連ツール:

- AWS CloudFormation Guard
- Landing Zone Accelerator on AWS

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

脅威モデリングを実行し、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を活用しない場合のリスクレベル: 高

実装のガイダンス

脅威モデリングとは?

「脅威モデリングとは、価値あるものを保護する状況において、脅威とその緩和策を特定し、伝達して理解する取り組みをいう」。– The Open Web Application Security Project (OWASP) Application Threat Modeling

なぜ脅威モデリングが必要なのか?

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。

脅威モデリングは、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、業界の原則であるセキュリティ戦略「シフトレフト」と一致しています。最終的に脅威モデリング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。

脅威モデリングを実行するタイミング

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モデリング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

脅威モデリングの実行方法

脅威モデリング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。そのうちの 1 つが「Shostack's 4 Question Frame for Threat Modeling」から始める方法です。ここでは、自由形式の質問をたずねることで脅威モデリングに枠組みを与えます。

1. 現在取り組んでいることは何か?

   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。この質問に答える最も一般的な方法は、モデルや図を作成することです。それにより、現在構築しているものをデータフロー図などを使って視覚化することができます。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、脅威モデルに取り組む担当者全員の目指す方向が合致し、対象範囲外のトピック (システムの古いバージョンなど) に脱線して時間を浪費する事態を回避できます。例えば、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. 問題化する可能性があるものは何か?

   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、チームのメンバーと脅威モデリングに参加する関係者との全員による、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、脅威を特定するモデルを使用するとスムーズに行うことができます。STRIDE は、評価すべきさまざまなカテゴ

リ (なりすまし、改ざん、否認、情報漏洩、サービス妨害、権限昇格) を提案するものです。また、OWASP Top 10、HiTrust Threat Catalog、組織独自の脅威カタログなど、既存のリストを見直して調査することもブレインストーミングの刺激となり役に立ちます。

3. どのように対処すべきか?

   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 とユーザー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?」ということもたずねていると理解することが重要です。ユーザーと AWS との間の、責任のバランスを理解することにより、ユーザーのコントロール下にある脅威モデリングの範囲を理解することができます。こちらは通常、AWS サービスの設定オプションと、ユーザー独自のシステムごとの緩和策とを組み合わせたものになります。

   共有責任の AWS の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対象範囲内であることがわかるはずです。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の監査レポートは、AWS の顧客は 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 のサービス自体に対するセキュリティコントロール緩和策として、ドメイン間にセキュリティコントロールを実装することを検討してください。対象となるドメインには、アイデンティティとアクセス管理 (認証と認可)、データ保護 (保管中と転送中)、インフラストラクチャのセキュリティ、ログ記録、モニタリングのドメインが含まれます。各 AWS サービスのドキュメントにはそれぞれセキュリティに関する項目があり、緩和策として利用できるセキュリティコントロールについてのガイダンスが記されています。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。こうしたコントロールには、入力の検証、セッションの処理、境界の処理などが含まれます。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. 十分に優れた仕事をしたか?

   狙いは、チームと組織が脅威モデルの質と、脅威モデリング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。十分に理解して実践的な経験を積むには、お客様とお客様のチームメンバー全員が「Threat modeling the right way for builders training course」またはワークショップを受講することが推奨

されます。さらに、組織のアプリケーション開発のライフサイクルに、脅威モデリングを組み込む方法について知りたい方は、AWS セキュリティブログの「脅威モデリングのアプローチ方法」を参照してください。

Threat Composer

脅威モデリングをサポートし実行を導くには、Threat Composer ツールの使用を検討します。このツールは、脅威モデリングの価値を実感できるまでにかかる時間を、短縮するためのツールです。このツールは、以下の用途で役立ちます。

- 自然な非線形のワークフローに該当する、Threat grammar に沿った有益な脅威のステートメントを記述する
- 人間が読める脅威モデルを生成する
- 機械可読な脅威モデルを生成して、脅威モデルをコードとして扱えるようにする
- Insights Dashboard を使用して、品質や対象範囲を改善できる分野をすばやく特定する

詳細については、Threat Composer にアクセスした後、システムで定義されたExample Workspace に切り替えます。

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:
- SEC01-BP04 セキュリティの脅威と推奨事項の最新情報を入手する
- SEC01-BP05 セキュリティ管理のスコープを縮小する
- 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

関連ドキュメント:

- 脅威モデリングのアプローチ方法 (AWS セキュリティブログ)
- NIST: Guide to Data-Centric System Threat Modelling

関連動画:

- AWS Summit ANZ 2021 - How to approach threat modelling

- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery

関連するトレーニング:

- Threat modeling the right way for builders – AWS Skill Builder virtual self-paced training
- Threat modeling the right way for builders – AWS Workshop

関連ツール:

- Threat Composer

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ために役立つ、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。

期待される成果: AWS や AWS パートナーがリリースした新しい機能やサービスについて知らせるための標準的な実践方法が確立されています。これらの新機能が、環境とワークロードに対する既存および新規の統制の設計にどのように影響するかを評価します。

一般的なアンチパターン:

- 関連する新しい機能やサービスの情報を迅速に入手するために、AWS のブログや RSS フィードを購読していない
- セキュリティサービスや機能に関するニュースや最新情報を二次情報源から入手している
- 組織内の AWS ユーザーに、常に最新情報に触れるよう推奨していない

このベストプラクティスを活用するメリット: 新しいセキュリティサービスや機能を常に把握していれば、クラウド環境やワークロードへの統制の実装について、情報に基づいて決断を下せます。進化するセキュリティ環境や、新たに出現した脅威への対策として AWS サービスを活用する方法を把握するうえで、それらの情報源が役に立ちます。

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: 低

実装のガイダンス

AWS では、新しいセキュリティサービスや機能について、複数のチャネルを通じてお客様にご案内しています。

- AWS の最新情報
- AWS ニュースブログ
- AWS セキュリティブログ
- AWS セキュリティ速報
- AWS ドキュメントの概要

Amazon Simple Notification Service (Amazon SNS) を使用して AWS Daily Feature Updates トピックを購読し、最新情報の包括的なサマリーを確認できます。Amazon GuardDuty や AWS Security Hub などの一部のセキュリティサービスは、独自の SNS トピックで各サービスに関する新しい標準、検出結果、その他の最新情報を配信しています。

また、毎年世界中で開催されるカンファレンス、イベント、ウェビナーでも、新しいサービスや機能が発表され、詳しく説明されています。中でも注目すべきは、毎年恒例の AWS re:Inforce セキュリティカンファレンスと、より一般的な AWS re:Invent カンファレンスです。前述の AWS のニュースチャネルでは、これらのカンファレンスでセキュリティやその他のサービスに関して発表した内容を共有しています。また、YouTube の AWS Events チャンネルでは、深く掘り下げて学ぶブレイクアウトセッションをオンラインでご視聴いただけます。

セキュリティサービスの最新情報や新しい推奨事項について、AWS アカウントチームにお問い合わせいただくこともできます。セールスサポートチーム直通の連絡先情報がお手元にない場合は、セールスサポートフォームからお問い合わせください。同様に、AWS エンタープライズサポートにご登録いただいている場合は、Technical Account Manager (TAM) が毎週最新情報をお知らせします。また、TAM との定期的なレビューミーティングを設定できます。

実装手順

1. お気に入りの RSS リーダーでさまざまなブログや速報を購読するか、「Daily Feature Updates」SNS トピックを購読します。
2. 新しい機能やサービスについて直に学ぶため、どの AWS イベントに参加すべきかを評価します。
3. セキュリティサービスや機能の更新について質問がある場合は、AWS アカウントチームとのミーティングを設定します。
4. エンタープライズサポートへの登録を検討します。登録すると、Technical Account Manager (TAM) に定期的に相談できます。

リソース

関連するベストプラクティス:

- PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する
- 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

ID およびアクセス管理

Questions

- 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をどのように管理するのですか。

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 の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。管理し、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と、適切な ID が適切な条件下で適切なリソースにアクセスしていることの検証に役立ちます。

人の ID: 管理者、デベロッパー、オペレーター、エンドユーザーには、AWS 環境とアプリケーションにアクセスするための ID が必要です。これらは、組織のメンバーやコラボレーションする外部のユーザーであり、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 リソースとやり取りします。

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。これらの ID には、Amazon EC2 インスタンスや AWS Lambda 関数などの、AWS 環境で実行中のマシンが含まれます。また、アクセスを必要とする外部の関係者のマシン ID を管理することもできます。さらに、AWS 環境へのアクセスを必要とする AWS 外部のマシンが含まれる場合もあります。

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採用する

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 Identity Center (AWS Single Sign-On の後継)、およびサードパーティー ID プロバイダーに強力なサインインメカニズムを使用することで、AWS の認証情報への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を活用し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 Identity Center 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。アカウントのルートユーザーは、ユーザー (およびルートユーザーを必要とするタスク) のアクセスを設定するときのみに使用することを推奨します。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 Identity Center 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。コンシューマーアイデンティティについては、Amazon Cognito ユーザープー

ルを使用して、そのサービスで、または Amazon Cognito ユーザープールユーザープールがサポートする 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とおりです。実際に行う設定は、組織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。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。
- 自分のパスワードの変更をユーザーに許可します。
- 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 アイデンティティセンターレコメンデーション:

- IAM アイデンティティセンターは、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 ユーザープール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 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 ユーザープール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 アイデンティティセンター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

関連ドキュメント:

- AWS IAM Identity Center 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 Identity Center
- Mastering identity at every layer of the cake

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。例えば、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 ユーザー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を活用しない場合のリスクレベル: 高

実装のガイダンス

すべての 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 Management Console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 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 ユーザープールを使用して、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、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

関連ドキュメント:

- 一時的な認証情報
- 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
- AWS IAM アイデンティティセンター
- ID プロバイダーとフェデレーション
- アクセスキーの更新
- AWS セキュリティコンピテンシーパートナー
- 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
- Mastering identity at every layer of the cake

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します。

- 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して捉える) 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を活用し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。例えば、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。

認証情報のタイプ	説明	推奨される戦略
IAM アクセスキー	ワークロード内で IAM ロールを引き受けるために使用される AWS IAM アクセスとシークレットキー	置き換え: 代わりに、コンピューティングインスタンス (Amazon EC2 や AWS Lambda など) に割り当てられた IAM ロールを使用します。AWS アカウント内のリソースへのアクセスを必要と

認証情報のタイプ	説明	推奨される戦略
		するサードパーティーとの相互運用性については、AWS クロスアカウントアクセスをサポートしているかどうかを確認してください。モバイルアプリの場合は、Amazon Cognito アイデンティティプール (フェデレーティッド ID) を介して一時的な認証情報を使用することを検討してください。AWS の外部で実行されているワークロードについては、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。
SSH キー	Linux EC2 インスタンスへの手動または自動プロセスの一環としてのログインに使用されるセキュアシェルプライベートキー	置き換え: AWS Systems Manager または EC2 Instance Connect を使用して、IAM ロールを使用して EC2 インスタンスへのプログラムおよび人間によるアクセスを提供します。
アプリケーションとデータベースの認証情報	パスワード – プレーンテキスト文字列	ローテーション: 認証情報を AWS Secrets Manager に保存し、可能であれば自動ローテーションを確立します。

認証情報のタイプ	説明	推奨される戦略
Amazon RDS と Aurora 管理データベースの認証情報	パスワード – プレーンテキスト文字列	置き換え: Secrets Manager と Amazon RDS または Amazon Aurora の統合を使用します。さらに、一部の RDS データベースタイプでは、一部のユースケースでパスワードの代わりに IAM ロールを使用できます (詳細については、「IAM データベース認証」を参照してください)。
OAuth トークン	シークレットトークン – プレーンテキスト文字列	ローテーション: トークンを AWS Secrets Manager に保存し、自動ローテーションを設定します。
API トークンとキー	シークレットトークン – プレーンテキスト文字列	ローテーション: AWS Secrets Manager に保存し、可能であれば自動ローテーションを確立します。

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。AWS サービスと通信するために IAM アクセスキーが必要な場合は、一時的な (短期的な) セキュリティ認証情報を使用します。これらの短期認証情報は、EC2 インスタンスの 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用の Cognito IAM ロール、および IoT デバイス用の IoT Core ポリシーを通じて提供できます。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 には、機密認証情報の安全なストレージと処理を確保するための 5 つの主要な機能があります。保管時の暗号化、転送中の暗号化、包括的な監査、きめ細かなアクセスコントロール、および拡張可能な認証情報ローテーションです。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 などの自動ツールを使用して、ハードコードされた認証情報を含むコードパスを特定します。
   a. 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` をフィルタリングして、問題のあるコード行を見つけます。
2. 削除または置換できる認証情報を特定します。
   a. 既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 にシークレットを作成できます。
   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。
      i. サーバーレス Lambda 関数では、言語に依存しない Lambda 拡張機能を使用できます。
      ii. 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。
   a. 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

関連ドキュメント:

- AWS Secrets Manager の開始方法
- ID プロバイダーとフェデレーション
- Amazon CodeGuru Introduces Secrets Detector
- AWS Secrets Manager での AWS Key Management Service の使用方法
- Secret encryption and decryption in Secrets Manager
- 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
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector
- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager

関連するワークショップ:

- Store, retrieve, and manage sensitive credentials in AWS Secrets Manager
- AWS Systems Manager Hybrid Activations

SEC02-BP04 一元化された ID プロバイダーを利用する

ワークフォースユーザー ID (従業員と契約社員) 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、割り当て、管理、取り消し、監査を行うため、複数のアプリケーションおよびシステムにまたがる権限を効率的に管理できます。

期待される成果: 一元化された ID プロバイダーがあり、ワークフォースユーザー、認証ポリシー (多要素認証 (MFA) を要求するなど)、システムやアプリケーションへの認可 (ユーザーのグループメンバーシップや属性に基づいてアクセスを割り当てるなど) を一元的に管理できます。ワークフォースユーザーは一元化された ID プロバイダーにサインインし、内部アプリケーションと外部アプリケーションにフェデレーション (シングルサインオン) します。これにより、ユーザーは複数の認証情報を覚えておく必要がなくなります。ID プロバイダーは人事 (HR) システムと統合されているため、人事上の変更は ID プロバイダーと自動的に同期されます。例えば、誰かが組織を離れた場合、フェデレーションされたアプリケーションやシステム (AWS を含む) へのアクセスを自動的に取り消すことができます。ID プロバイダーで詳細な監査ログを有効にし、これらのログでユーザーの異常な行動がないか監視します。

一般的なアンチパターン:

- フェデレーションとシングルサインオンを使用しない。ワークフォースユーザーが、複数のアプリケーションやシステムで個別のユーザーアカウントと認証情報を作成する。
- ID プロバイダーを人事システムに統合するなど、ワークフォースユーザーのアイデンティティのライフサイクルを自動化していない。ユーザーが組織を離れたり、役割を変更したりした場合に、複数のアプリケーションやシステムのレコードを手動のプロセスで削除または更新する。

このベストプラクティスを活用するメリット: 一元化された ID プロバイダーを使用することで、ワークフォースユーザーのアイデンティティとポリシーを 1 か所で管理でき、ユーザーやグループにアプリケーションへのアクセス権を割り当てたり、ユーザーのサインインアクティビティを監視したりできます。人事 (HR) システムと統合することで、ユーザーの役割が変更された場合は、これらの変更が ID プロバイダーと同期され、ユーザーに割り当てられたアプリケーションと権限が自動的に更新されます。ユーザーが組織を離れると、そのユーザーのアイデンティティは ID プロバイダーで自動的に無効になり、フェデレーションアプリケーションおよびシステムへのアクセス権が取り消されます。

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

AWS にアクセスするワークフォースユーザー向けのガイダンス

組織内の従業員や契約社員などのワークフォースユーザーは、AWS Management ConsoleまたはAWS Command Line Interface (AWS CLI) を使って職務を遂行するため、AWS へのアクセス権を必要とする場合があります。一元化された ID プロバイダーから 2 つのレベルで AWS にフェデレーションすることで、ワークフォースユーザーに AWS へのアクセス権を付与できます。1 つは各 AWS アカウントへの直接フェデレーション、もう 1 つは AWS 組織内の複数のアカウントへのフェデレーションです。

- ワークフォースユーザーをそれぞれの AWS アカウントと直接フェデレーションするには、一元化された ID プロバイダーを使用して、そのアカウントの AWS Identity and Access Management にフェデレーションできます。IAM の柔軟性により、SAML 2.0 または Open ID Connect (OIDC) という別々の ID プロバイダーを各 AWS アカウントで有効にして、アクセスコントロールにはフェデレーションユーザー属性を使用することができます。ワークフォースユーザーはウェブブラウザを使用し、認証情報 (パスワードや MFA トークンコードなど) を入力して ID プロバイダーにサインインします。ID プロバイダーは、AWS Management Consoleのサインイン URL に送信される SAML アサーションをユーザーのブラウザに発行して、IAM ロールを引き受けることで、ユーザーが AWS Management Consoleにシングルサインオンできるようにします。ユーザーは、ID プロバイダーからの SAML アサーションを使用して、IAM ロールを引き受けることで、AWS CLI や AWS STS の AWS SDK で使用する 一時的な AWS API 認証情報を 取得することもできます。
- ワークフォースユーザーを AWS 組織内の複数のアカウントにフェデレーションするには、AWS IAM Identity Center を使用して、AWS アカウントやアプリケーションへのワークフォースユーザーのアクセスを一元管理できます。組織のアイデンティティセンターを有効にし、ID ソースを設定します。IAM Identity Center は、ユーザーやグループの管理に使用できるデフォルトの ID ソースディレクトリを提供します。または、SAML 2.0 を使用して外部 ID プロバイダーに接続し、 SCIM を使用してユーザーとグループを自動的にプロビジョニングするか、または AWS Directory Service を使用して Microsoft AD Directory に接続することで、外部 ID ソースを選択することもできます。ID ソースを設定したら、アクセス許可セットで最小権限ポリシーを定義して、ユーザーとグループに AWS アカウントへのアクセス権を割り当てることができます。ワークフォースユーザーは一元化された ID プロバイダーを通じて認証を行い、AWS アクセスポータルにサインインして、自分に割り当てられた AWS アカウントとクラウドアプリケーションにシングルサインオンします。ユーザーは AWS CLI v2 を設定して、アイデンティティセンターで認証を行い、AWS CLI コマンドを実行するための認証情報を取得できます アイデンティティセンターでは、AWS アプリケーション (Amazon SageMaker Studio や AWS IoT IoT Sitewise Monitor ポータル) へのアクセスにシングルサインオンも使用できます。

前述のガイダンスに従うと、ワークフォースユーザーは AWS でワークロードを管理する際、通常の操作で IAM ユーザーおよびグループを使用する必要がなくなります。管理するのではなく、ユー

ザーとグループは AWS の外部で管理され、ユーザーはフェデレーション ID として AWS リソースにアクセスできます。フェデレーション ID では、一元化された ID プロバイダーで定義されたグループを使用します。AWS アカウントで不要になった IAM グループ、IAM ユーザー、および永続的なユーザー認証情報 (パスワードとアクセスキー) を特定して削除する必要があります。また、IAM 認証情報レポートを使用して、未使用の認証情報を検索し、該当する IAM ユーザーや IAM グループを削除できます。組織にサービスコントロールポリシー (SCP) を適用して、新しい IAM ユーザーやグループが作成されないようにし、フェデレーション ID を介した AWS へのアクセスを強制できます。

アプリケーションのユーザー向けガイダンス

モバイルアプリなどのアプリケーションのユーザーの ID を管理するには、一元化された ID プロバイダーとして Amazon Cognito を使用できます。Amazon Cognito は、ウェブおよびモバイルアプリに認証、認可、ユーザー管理の機能を提供します。Amazon Cognito は数百万人のユーザーにスケール可能な ID ストアを備え、ソーシャル ID フェデレーションとエンタープライズ ID フェデレーションをサポートし、ユーザーとビジネスの保護に役立つ高度なセキュリティ機能を提供します。カスタムのウェブまたはモバイルアプリケーションを Amazon Cognito と統合すると、アプリケーションへのユーザー認証とアクセスコントロールを数分で追加できます。SAML や Open ID Connect (OIDC) などのオープン ID 標準に基づいて構築された Amazon Cognito は、さまざまなコンプライアンス規制に対応し、フロントエンドおよびバックエンドの開発リソースと統合します。

実装手順

ワークフォースユーザーの AWS へのアクセス手順

- 以下のいずれかの方法を使用し、一元化された ID プロバイダーを使用して、ワークフォースユーザーを AWS にフェデレーションします。
  - IAM Identity Center を使用し、ID プロバイダーとフェデレーションすることで、AWS 組織内の複数の AWS アカウントへのシングルサインオンを有効にします。
  - IAM を使用して、ID プロバイダーを各 AWS アカウントに直接接続し、フェデレーションによるきめ細かいアクセスを可能にします。
- フェデレーション ID で置き換えられた IAM ユーザーとグループを特定して削除します。

アプリケーションのユーザー向けの手順

- アプリケーション用の一元化された ID プロバイダーとして Amazon Cognito を使用します。
- OpenID Connect と OAuth を使用して、カスタムアプリケーションを Amazon Cognito と統合します。認証のための Amazon Cognito など、さまざまな AWS サービスと統合するためのシンプル

なインターフェイスを提供する Amplify ライブラリを使用して、カスタムアプリケーションを開発できます。

リソース

関連する Well-Architected のベストプラクティス:

- SEC02-BP06 ユーザーグループと属性を採用する
- SEC03-BP02 最小特権のアクセスを付与する
- SEC03-BP06 ライフサイクルに基づいてアクセスを管理する

関連ドキュメント:

- AWS での ID フェデレーション
- IAM でのセキュリティのベストプラクティス
- AWS Identity and Access Management のベストプラクティス
- Getting started with IAM Identity Center delegated administration
- How to use customer managed policies in IAM Identity Center for advanced use cases
- AWS CLI v2: IAM Identity Center credential provider

関連動画:

- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2018: Mastering Identity at Every Layer of the Cake

関連する例:

- Workshop: Using AWS IAM Identity Center to achieve strong identity management
- Workshop: Serverless identity

関連ツール:

- AWS セキュリティコンピテンシーパートナー: ID およびアクセスの管理
- AWS IAM Identity Center

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ますが、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で軽減できます。

期待される成果: 長期的認証情報の使用に関連するリスクを軽減するために、認証情報のローテーションを実装します。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用する。
- 長期的認証情報を使用しているが、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活用しない場合のリスクレベル: 高

実装のガイダンス

一時的な認証情報に頼れず、長期的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て、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。

正しい管理方法が実施されていることを確認するには、定期的な検証、できれば自動化ツールによる検証が必要です。人的 ID の場合は、パスワードを定期的に変更し、一時的な認証情報を優先してアクセスキーを廃止するように、ユーザーに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。

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もお勧めします。AWS Config ルール を設定するか、AWS Security Hub セキュリティ標準を使用して、ユーザーが MFA を設定しているかどうかをモニタリングできます。IAM Roles Anywhere を使用して、マシン ID の一時的な認証情報を付与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を使用できないときは、アクセスキーの監査とローテーション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、認証された 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。これらの ID は、IAM ユー

ザー、AWS IAM Identity Center 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、これらに限定されるものではありません。例えば、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 を使用して、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できます。Amazon GuardDuty は、IAM 認証情報へのアクセスが過度に許可されているか、意図しないアクセスを示している可能性のある、予期しないアクティビティを警告することもできます。

- 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は、IAM アクセスキーを定期的に (最大 90 日ごとに) ローテーションします。知らない間にアクセスキーが開示された場合でも、ローテーションを行うことで、該当する認証情報を使用してリソースにアクセスされる期間を制限できます。アクセスキーのローテーションの詳細については、IAM ユーザーの「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従っていることを確認します。
- IAM リソースの作成と更新の自動化を検討する: IAM Identity Center は、ロールやポリシーの管理など、多くの IAM タスクを自動化します。または、AWS CloudFormation を使用することで、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軽減できます。
- IAM Roles Anywhere を使用してマシン ID の IAM ユーザーを置き換える: IAM Roles Anywhere を使用すると、オンプレミスサーバーなど、従来は使用できなかった領域でロールを使用できます。IAM Roles Anywhere は、信頼された X.509 証明書を使用して AWS を認証し、一時的な認証情報を受け取ります。IAM Roles Anywhere を使用することで、長期的認証情報がオンプレミス環境に保存されなくなるため、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

関連ドキュメント:

- AWS Secrets Manager 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
- 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
- 一時的な認証情報
- AWS アカウントの認証情報レポートの取得

関連動画:

- シークレットを大規模に管理、取得、変更するためのベストプラクティス
- AWS IAM Identity Center を使用した大規模なユーザー権限の管理
- すべての層での ID の把握

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 グループとロールの自動デプロイ

SEC02-BP06 ユーザーグループと属性を採用する

ユーザーグループと属性に従ってアクセス許可を定義すると、ポリシーの数と複雑度が軽減され、最小特権の原則を簡単に遵守できます。 ユーザーグループを使用して、多数のユーザーのアクセス許可をそれぞれが組織内で果たす職務に基づいて 1 か所で管理できます。 部門や場所などの属性は、ユーザーが同じような職務を果たすが、対象となるリソースのサブセットが異なる場合に、アクセス許可の範囲をさらに限定することができます。

期待される成果: 権限の変更を、職務に基づき、その職務を実行するすべてのユーザーに適用できます。 グループのメンバーシップと属性によってユーザーのアクセス許可が管理されるため、個々のユーザーレベルでアクセス許可を管理する必要がなくなります。 ID プロバイダー (IdP) で定義したグループと属性が、AWS 環境に自動的に反映されます。

一般的なアンチパターン:

- 個々のユーザーのアクセス許可を管理し、複数のユーザーで重複作業をしている。

- グループの定義が大まか過ぎるため、アクセス許可の付与範囲が広過ぎる。
- グループの定義が細か過ぎるため、メンバーシップに関する重複や混乱が生じている。
- 代わりに属性を使用できる場面でグループを使用し、リソースの複数のサブセットに対してグループが持つアクセス許可が重複している。
- 標準に準拠した ID プロバイダーの AWS 環境への統合によるグループ、属性、メンバーシップの管理を行っていない。

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: 中

実装のガイダンス

AWS アクセス許可は、ユーザー、グループ、ロール、リソースなどのプリンシパルに関連付けられている、ポリシーと呼ばれるドキュメントで定義されます。 従業員に対しては、アクセス対象のリソースではなく、ユーザーが組織で果たす職務に基づいてグループを定義できます。例えば、`WebAppDeveloper` グループには、開発アカウント内で Amazon CloudFront などのサービスを設定するためのポリシーがアタッチされている場合があります。 `AutomationDeveloper` グループには、`WebAppDeveloper` グループと共通の CloudFront アクセス許可が複数存在する場合があります。これらのアクセス許可は、両方の職務のユーザーを `CloudFrontAccess` グループに所属させるのではなく、別々のポリシーで取得して両方のグループに関連付けることができます。

グループに加え、属性を使用してアクセスの範囲をさらに絞り込むことができます。例えば、`WebAppDeveloper` グループ内のユーザーに対して、プロジェクト固有のリソースへのアクセス範囲を限定するための `Project` 属性を設定できます。 この手法を使用すれば、異なるプロジェクトで作業するアプリケーション開発者に対して、それ以外の点ではアクセス許可が同じ場合、別々のグループを用意する必要がなくなります。 アクセス許可ポリシーで属性を参照する方法は、そのソースがフェデレーションプロトコル (SAML、OIDC、SCIM など) の一部として定義されているか、カスタム SAML アサーションとして定義されているか、IAM Identity Center 内で設定されているかに応じて異なります。

実装手順

1. グループと属性を定義する場所を確保します。
   a. SEC02-BP04 一元化された ID プロバイダーを利用する のガイダンスに従って、ID プロバイダー内、IAM Identity Center 内、または特定のアカウント内の IAM ユーザーグループを使用し、グループと属性を定義する必要があるかどうかを判断します。
2. グループを定義します。
   a. 必要な職務とアクセス範囲に基づいてグループを決定します。

b. IAM Identity Center 内で定義する場合は、グループを作成し、アクセス許可セットを使用して、目的とするレベルのアクセス許可を関連付けます。

c. 外部の ID プロバイダー内で定義する場合は、プロバイダーが SCIM プロトコルをサポートしているかどうかを確認し、IAM Identity Center 内で自動プロビジョニングを有効にすることを検討してください。 この機能は、プロバイダーと IAM Identity Center との間で、メンバーシップの作成、およびグループの削除を同期します。

3. 属性を定義します。

a. 外部の ID プロバイダーを使用する場合、SCIM プロトコルと SAML 2.0 プロトコルは両方とも一部の属性をデフォルトで提供します。 追加の属性は、`https://aws.amazon.com/SAML/Attributes/PrincipalTag` 属性名および SAML アサーションを使用して定義し、渡すことができます。

b. IAM Identity Center 内で定義する場合は、属性ベースのアクセス制御 (ABAC) 機能を有効にし、必要に応じて属性を定義します。

4. グループと属性に基づいてアクセス許可の範囲を設定します。

a. プリンシパルの属性と、アクセス対象のリソースの属性を比較する条件をアクセス許可ポリシーに含めることを検討してください。 例えば、`PrincipalTag` 条件キーの値が同じ名前の `ResourceTag` キーの値と一致する場合にのみ、リソースへのアクセスを許可する条件を定義できます。

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
- SEC03-BP02 最小特権のアクセスを付与する
- COST02-BP04 グループとロールを実装する

関連ドキュメント:

- IAM ベストプラクティス
- IAM Identity Center で ID を管理する
- AWS の ABAC とは
- IAM Identity Center - ABAC

関連動画:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC 3. 人とマシンのアクセス許可を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。アクセス許可は、誰が何に、どのような条件で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する
- 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やリソースには、管理者、エンドユーザー、またはその他のコンポーネントによるアクセスが必要です。各コンポーネントへのアクセス権のある人とマシンを明確に定義し、適切な ID のタイプと、認証および認可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

ワークロードの各コンポーネントやリソースには、管理者、エンドユーザー、またはその他のコンポーネントによるアクセスが必要です。各コンポーネントへのアクセス権のある人とマシンを明確に定義し、適切な ID のタイプと、認証および認可の方法を選択します。

組織内の AWS アカウントへの定期的なアクセスは、フェデレーションアクセスか一元化された 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。推奨されるアプローチは、最小特権アクセスモデルで構築された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、事前定義済みの IAM ポリシーを提供します。

AWS Secrets Manager や AWS Systems Manager Parameter Store などの AWS のサービスは、IAM ロールを使用することが不可能な場合に、アプリケーションまたはワークロードからシークレットを安全に分離するのに役立ちます。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。Secrets 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、自動化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、AWS の外部で実行されるワークロードの一時的なセキュリティ認証情報を IAM で取得できます。ワークロードでは、AWS アプリケーションが AWS リソースにアクセスする際に使用するのと同じ IAM ポリシーと IAM ロールを使用できます。

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。プログラムによるアクセスと長期的認証情報を持つユーザーが必要なシナリオでは、アクセスキーが最後に使用した情報を使用して、アクセスキーをローテーションおよび削除します。

AWS Management Console の外部で AWS を操作するには、ユーザーはプログラムによるアクセスが必要です。プログラマチックアクセス権を付与する方法は、AWS にアクセスしているユーザーのタイプによって異なります。

ユーザーにプログラマチックアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラマチックアクセス権を必要とするユーザー	目的	方法
ワークフォースアイデンティティ (IAM アイデンティティセンターで管理されているユーザー)	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラマチックリクエストに署名します。	使用するインターフェイス用の手引きに従ってください。 • AWS CLI については、AWS Command Line Interface ユーザーガイドの「AWS IAM Identity Center を使用するための AWS CLI の設定」を参照してください。 • AWS SDK、ツール、および AWS API については、AWS SDK とツールリファレンスガイドの「IAM Identity Center 認証」を参照してください。
IAM	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	「IAM ユーザーガイド」の「AWS リソースでの一時的な認証情報の使用」の指示に従ってください。
IAM	(非推奨) 長期的な認証情報を使用して、AWS CLI、AWS SDK、AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイス用の手順に従ってください。 • AWS CLI については、AWS Command Line Interface ユーザーガイドの「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDK とツールについては、AWS SDK とツールリファレンスガイドの「長

プログラマチックアクセス権を必要とするユーザー	目的	方法
		期認証情報を使用して認証する」を参照してください。 • AWS API については、IAM ユーザーガイドの「IAM ユーザーのアクセスキーの管理」を参照してください。

リソース

関連ドキュメント:

- 属性ベースのアクセスコントロール (ABAC)
- AWS IAM Identity Center
- IAM Roles Anywhere
- AWS Managed policies for IAM Identity Center
- AWS IAM ポリシー条件
- IAM のユースケース
- 不要な認証情報の削除
- 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
- Identify, arrange, and manage secrets easily using enhanced search in AWS Secrets Manager

関連動画:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- Streamlining identity and access management for innovation

SEC03-BP02 最小特権のアクセスを付与する

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ことが、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、スケールに応じてアクセス許可を動的に設定します。例えば、デベロッパーのグループに、プロジェクトのリソースのみを管理するためのアクセスを許可することができます。これにより、開発者がプロジェクトを離れると、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: ユーザーに、ジョブの実行に必要なアクセス許可のみが付与されます。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を付与し、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に、アクセス許可は定期的に見直す必要があります。マシンまたはシステムのアカウントには、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。
- 最小特権のアクセスを付与するかどうかを把握するためにアクセス許可を見直していない。

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

最小特権の原則では、特定のタスクを実行するために必要な、最小限のアクションの実行のみを ID に許可する必要があると規定されています。これにより、使いやすさ、効率性、セキュリティのバランスが取れます。この原則の下に運用すると、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 ユーザーやロールに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ますが、厳密に制御、モニタリングされ、ルートアクセスを必要とする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します。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 ポリシーの作成時に、サービスアクション、リソース、および AWS がアクセスを許可または拒否するために true でなければならない条件を指定できます。AWS は、アクセスの範囲を絞り込むのに役立つさまざまな条件をサポートしています。例えば、`PrincipalOrgID` 条件キー を使用すると、リクエスタが AWS Organization の一部でない場合にアクションを拒否できます。

`CalledVia` 条件キーを使用して、AWS Lambda 関数を作成する AWS CloudFormation など、AWS サービスがユーザーに代わって行うリクエストを制御することもできます。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。どのアクセス許可をどのような条件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に対して、構築するシステムに独自の IAM ポリシーを作成することを許可できますが、システムが受け取ることができる最大権限を制限するために、アクセス許可の境界を適用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させます。
  - API の使用状況に基づくポリシー: 必要なアクセス許可を決定する 1 つの方法は、AWS CloudTrail ログを確認することです。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、カスタマイズし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。組織またはアカウントレベルで IAM Access Advisor を使用し、特定のポリシーの最終アクセス情報を追跡できます。
- ジョブ機能の AWS マネージドポリシーを使用すること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、請求、データベース管理者、データサイエンティストなど、一般的なジョブロール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する際に、ユーザーの持つアクセスを絞り込むことができます。
- 不要なアクセス許可を削除する: 不要なアクセス許可と過度に許可されたポリシーを削除します。IAM Access Analyzer ポリシーの生成は、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ていることを確認する: ユーザーは、有効なユースケースを持つ本番環境にのみアクセスする必要があります。ユーザーが本番環境へのアクセスが必要な特定のタスクを実行した後で、アクセス許可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、本番に影響する想定外の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、マネージド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により、エンティティは、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を使用すると、リソースの目的、所有者、環境、その他の基準に基づいてアクセスを許可できます。例えば、リソースタグを使用して、開発環境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実現することができ、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する。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することもできます。
- 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 にオンボーディングされたとき、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をユーザー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する必要があります。
- アクセス許可を見直して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見直して、ユーザーに過剰なアクセス許可がないことを確認する必要があります。AWS Config および IAM Access Analyzer は、ユーザーのアクセス許可を監査する際に役立ちます。
- ジョブロールマトリックスを確立する: ジョブロールマトリックスは、AWS フットプリント内に必要なさまざまなロールとアクセスレベルを視覚化します。ジョブロール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のではなく、グループを使用します。

リソース

関連ドキュメント:

- 最小特権を認める
- IAM エンティティのアクセス許可の境界
- 最小特権の IAM ポリシーを作成するテクニック

- 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
- Delegate permission management to developers by using IAM permissions boundaries
- 最終アクセス時間情報を使用したアクセス許可の調整
- IAM policy types and when to use them
- 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
- ゼロトラストアーキテクチャ: AWS の視点
- CloudFormation StackSets を使用して最小特権の原則を実装する方法
- 属性ベースのアクセス制御 (ABAC)
- ユーザーアクティビティを表示してポリシー範囲を狭める
- ロールのアクセスの表示
- Use Tagging to Organize Your Environment and Drive Accountability
- AWS タグ付け戦略
- AWS リソースのタグ付け

関連動画:

- 次世代のアクセス許可管理
- ゼロトラスト: AWS の視点

関連する例:

- ラボ: IAM permissions boundaries delegating role creation
- ラボ: IAM tag based access control for EC2

SEC03-BP03 緊急アクセスのプロセスを確立する

一元化された ID プロバイダーで万一障害が発生した場合に備え、ワークロードへの緊急アクセスを許可するプロセスを作成します。

緊急事態につながる可能性のある、さまざまな障害モードに対応するプロセスを設計する必要があります。例えば、通常の状況では、従業員ユーザーは一元化された ID プロバイダー (SEC02-BP04) を

使用してクラウドにフェデレーションし、ワークロードを管理します。しかし、一元化された ID プロバイダーに障害が発生した場合や、クラウドのフェデレーションの設定が変更された場合、従業員ユーザーはクラウドにフェデレーションできなくなる可能性があります。緊急アクセスのプロセスでは、権限を持つ管理者がフェデレーション設定やワークロードの問題を解決するために、代替手段 (代替フェデレーションやユーザーの直接アクセスなど) を通じてクラウドリソースにアクセスすることを許可します。緊急アクセスのプロセスは、通常のフェデレーションメカニズムが復旧するまで使用されます。

期待される成果:

- 緊急事態とみなされる障害モードを定義して文書化します。通常の状況と、ユーザーがワークロードの管理に使用するシステムを考慮してください。それぞれの依存関係でどのように障害が発生する可能性があるか、またその障害がどのように緊急事態を引き起こすかを検討します。信頼性の柱の質問とベストプラクティスは、障害モードを特定し、障害の可能性を最小限に抑えるための、より回復力のあるシステムの構築に役立ちます。
- 障害が緊急事態であると確認する際に従うべき手順を文書化します。例えば、ID 管理者がプライマリ ID プロバイダーとスタンバイ ID プロバイダーのステータスを確認すること、両方とも使用できない場合は、ID プロバイダーに障害発生時の緊急事態を宣言することを要求できます。
- 各種の緊急モードまたは障害モードに固有の緊急アクセスのプロセスを定義します。具体的に定義することで、ユーザーが緊急事態の種類にかかわらず一般的なプロセスを使いすぎる状況を軽減することができます。緊急アクセスのプロセスで、各プロセスを使用すべき状況、逆にそのプロセスを使用すべきでない状況、適用される可能性のある代替プロセスを説明します。
- 詳細な指示とプレイブックを含めてプロセスを十分に文書化することで、迅速かつ効率的に実行できます。緊急事態はユーザーにとってストレスの多い時間であり、ユーザーは極度の時間的プレッシャーにさらされる可能性があるため、プロセスはできるだけシンプルに設計してください。

一般的なアンチパターン:

- 緊急アクセスのプロセスの文書化およびテストが不十分である。ユーザーは緊急事態への備えができておらず、緊急事態が発生しても即席のプロセスに従う。
- 緊急アクセスのプロセスで、通常のアクセスメカニズムと同じシステム (一元化された ID プロバイダーなど) を使用する。これにより、このようなシステムの障害が、通常および緊急の両方のアクセスメカニズムに影響を及ぼし、障害からの回復能力が損なわれる可能性があります。
- 緊急アクセスのプロセスが、緊急ではない状況で使用される。例えば、ユーザーは、パイプラインを通じて変更を送信するよりも直接変更する方が簡単だと感じるため、頻繁に緊急アクセスのプロセスを誤用します。

- 緊急アクセスのプロセスで、プロセスを監査するための十分なログが生成されていない、またはログが監視されておらず、プロセスの誤用の可能性について警告されない。

このベストプラクティスを活用するメリット:

- 緊急アクセスのプロセスを十分に文書化し、十分にテストすることで、ユーザーが緊急事態に対応して解決するための時間を短縮できます。これにより、ダウンタイムが減少し、顧客に提供するサービスの可用性が高まります。
- 緊急アクセスのリクエストをそれぞれ追跡し、緊急事態以外の場合にプロセスを誤用しようとする不正な試みを検出して警告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: 中

実装のガイダンス

このセクションでは、AWS 上にデプロイされたワークロードに関連する複数の障害モードに対する緊急アクセスのプロセス作成のためのガイダンスを提供します。すべての障害モードに適用される共通のガイダンスから始めて、次に障害モードの種類に基づいた具体的なガイダンスを示します。

すべての障害モードに共通のガイダンス

障害モードに対する緊急アクセスのプロセスを設計する際は、次の点を考慮してください。

- プロセスの前提条件と仮定事項 (プロセスを使用すべき場合と使用すべきでない場合) を文書化します。障害モードを詳しく説明し、他の関連システムの状態などの仮定事項を文書化しておくと役立ちます。例えば、障害モード 2 に対するプロセスでは、ID プロバイダーは使用可能ですが、AWS の設定が変更されているか、有効期限が切れていると仮定しています。
- 緊急アクセスのプロセスで必要となるリソースを事前に作成しておきます (SEC10-BP05)。例えば、IAM ユーザーとロールを持つ緊急アクセス用の AWS アカウントと、すべてのワークロードアカウントでのクロスアカウントの IAM ロールを事前に作成します。これにより、緊急事態の発生時にリソースが準備され、使用可能な状態を確保できます。リソースを事前に作成することで、緊急時に使用できない可能性のある AWS コントロールプレーン API (AWS リソースの作成と変更に使用) への依存関係がなくなります。さらに、IAM リソースを事前に作成することで、結果整合性による潜在的な遅延を考慮する必要がなくなります。
- インシデント管理計画に緊急アクセスのプロセスを含めます (SEC10-BP02)。緊急事態の追跡方法、同僚チームやリーダーシップなど組織内の他のメンバーへの伝達方法、該当する場合は外部の顧客やビジネスパートナーへの伝達方法を文書化します。

- 既存のサービスリクエストワークフローシステム (ある場合) で緊急アクセスリクエストプロセスを定義します。通常、このようなワークフローシステムでは、リクエストに関する情報を収集する受付フォームを作成したり、ワークフローの各段階でリクエストを追跡したり、自動および手動の承認ステップを追加したりできます。各リクエストを、インシデント管理システムで追跡される、対応する緊急イベントに関連付けます。緊急アクセス用の統一されたシステムがあると、このようなリクエストを単一のシステムで追跡し、使用傾向を分析して、プロセスを改善できます。
- 緊急アクセスのプロセスは権限を持つユーザーのみが開始できることと、必要に応じてそのユーザーの同僚または管理者の承認が必要であることを確認します。承認プロセスは、営業時間内、時間外を問わず、効果的に実施される必要があります。承認リクエストについて、一次承認者が不在の場合はどのように二次承認者を許可するのか、承認を得るまでに一連の管理層にリクエストをどのようにエスカレーションするのかを定義します。
- 緊急アクセスに成功した場合と失敗した場合の両方について、詳細な監査ログとイベントが生成されることを確認します。リクエストプロセスと緊急アクセスメカニズムの両方を監視して、誤用や不正アクセスを検出します。インシデント管理システムで、進行中の緊急事態とアクティビティを関連付けて、想定時間外に障害が発生した場合に警告を発します。例えば、緊急アクセス AWS アカウントでのアクティビティを監視して警告する必要がありますが、こうしたアクティビティは通常の運用では使用されるべきではありません。
- 緊急アクセスのプロセスを定期的にテストして、手順が明確であること、適切なレベルのアクセス権を迅速かつ効率的に付与できることを確認します。緊急アクセスのプロセスは、インシデント対応シミュレーション (SEC10-BP07) とディザスタリカバリテスト (REL13-BP03) の一部としてテストする必要があります。

障害モード 1: AWS へのフェデレーションに使用する ID プロバイダーが使用できない

「SEC02-BP04 一元化されたプロバイダーを使用する」で説明したとおり、ワークフォースユーザーをフェデレーションして AWS アカウントへのアクセス権を付与するには、一元化された ID プロバイダーを使用することを推奨します。IAM Identity Center を使用して AWS 組織内の複数の AWS アカウントにフェデレーションするか、IAM を使用して個別の AWS アカウントにフェデレーションすることができます。いずれの場合も、従業員ユーザーは、シングルサインオンのために AWS へのサインインエンドポイントにリダイレクトされる前に、一元化された ID プロバイダーで認証されます。

万一、一元化された ID プロバイダーが使用できない場合、従業員ユーザーは AWS アカウントにフェデレーションすることも、ワークロードを管理することもできなくなります。こうした緊急事態では、AWS アカウントにアクセスするための緊急アクセスのプロセスを少数の管理者に提供し、一元化された ID プロバイダーのオンライン復帰を待つ余裕のない重要なタスクを実行できるようにし

ます。例えば、ID プロバイダーを 4 時間利用できない間に、顧客トラフィックの想定外の急増に対応するために、本番稼働用アカウントの Amazon EC2 Auto Scaling グループの上限を変更する必要が生じたとします。この場合、緊急管理者は、緊急アクセスのプロセスに従って、特定の本番稼働用 AWS アカウントへのアクセス権を取得し、必要な変更を加える必要があります。

緊急アクセスのプロセスでは、事前に作成された緊急アクセス用 AWS アカウントを使用します。このアカウントは緊急アクセスの目的でのみ使用され、緊急アクセスのプロセスに対応するための AWS リソース (IAM ロールや IAM ユーザーなど) が設定されています。通常の運用中は、緊急アクセスアカウントへのアクセスを全面的に禁止し、このアカウントの誤用を監視し、警告する必要があります (詳細については、前述の「共通のガイダンス」セクションを参照してください)。

緊急アクセスアカウントには、緊急アクセスを必要とする AWS アカウントでクロスアカウントロールを引き受ける権限を持つ、緊急アクセス IAM ロールがあります。これらの IAM ロールは事前に作成され、緊急アカウントの IAM ロールを信頼する信頼ポリシーで設定されます。

緊急アクセスのプロセスでは、次のいずれかの方法を使用できます。

- 緊急アクセスアカウントでは、関連する強力なパスワードと MFA トークンを使用して、緊急管理者用の一連の IAM ユーザーを事前に作成します。これらの IAM ユーザーには、IAM ロールを引き受け、緊急アクセスが必要な AWS アカウントへのクロスアカウントアクセスを許可する権限があります。このようなユーザーはできるだけ少人数にし、各ユーザーを 1 人の緊急管理者に割り当てることをお勧めします。緊急時には、緊急管理者ユーザーがパスワードと MFA トークンコードを使用して緊急アクセスアカウントにサインインし、緊急アカウントで緊急アクセス IAM ロールに切り替え、最後にワークロードアカウントで緊急アクセス IAM ロールに切り替えて、緊急の変更アクションを実行します。この方法の利点は、それぞれの IAM ユーザーが 1 人の緊急管理者によって割り当てられるため、CloudTrail イベントを確認することで、どのユーザーがサインインしたかを把握できることです。欠点は、複数の IAM ユーザーと、それぞれに関連付けられた永続的なパスワードおよび MFA トークンを管理する必要があることです。
- 緊急アクセス AWS アカウントルートユーザーを使用して緊急アクセスアカウントにサインインし、緊急アクセスの IAM ロールを引き受け、ワークロードアカウントでクロスアカウントロールを引き受けることができます。ルートユーザーには、強力なパスワードと複数の MFA トークンを設定することをお勧めします。また、パスワードと MFA トークンは、強力な認証と認可を強制する安全なエンタープライズ認証情報ボールトに格納することをお勧めします。パスワードと MFA トークンのリセット要因を確保する必要があります。アカウントの E メールアドレスをクラウドセキュリティ管理者が監視するメール配布リストに設定し、アカウントの電話番号は、セキュリティ管理者が同様に監視する共有電話番号に設定します。この方法の利点は、管理するルートユーザーの認証情報が 1 セットだけであることです。欠点は、これが共有ユーザーであるため、複数の管理者がルートユーザーとしてサインインできてしまうことです。エンタープライズボールトの

ログイベントを監査して、どの管理者がルートユーザーのパスワードをチェックアウトしたかを特定する必要があります。

障害モード 2: AWS の ID プロバイダー設定が変更された、または有効期限が切れている

従業員ユーザーが AWS アカウントにフェデレーションできるようにするには、外部 ID プロバイダーを使用して IAM Identity Center を設定するか、IAM ID プロバイダーを作成します (SEC02-BP04)。通常、これらを設定するには、ID プロバイダーが提供する SAML メタデータ XML ドキュメントをインポートします。メタデータ XML ドキュメントには、ID プロバイダーが SAML アサーションの署名に使用する、プライベートキーに対応する X.509 証明書が含まれています。

AWS 側でのこれらの設定は、管理者が誤って変更または削除する可能性があります。もう 1 つのシナリオとして、AWS にインポートされた X.509 証明書の有効期限が切れ、新しい証明書を含む新しいメタデータ XML が AWS にインポートされていない場合があります。いずれの場合も、従業員ユーザーの AWS へのフェデレーションが失敗し、緊急事態が発生する可能性があります。

このような緊急事態が発生した場合は、フェデレーションの問題を解決するために、ID 管理者に AWS へのアクセスを提供します。例えば、ID 管理者は緊急アクセスのプロセスを使用して緊急アクセス用 AWS アカウントにサインインし、Identity Center 管理者アカウントのロールに切り替え、ID プロバイダーから提供された最新の SAML メタデータ XML ドキュメントをインポートして外部 ID プロバイダーの設定を更新することで、フェデレーションを再有効化することができます。フェデレーションが修正されたら、従業員ユーザーは引き続き通常の操作プロセスに従って、ワークロードアカウントにフェデレーションできます。

前述の「障害モード 1」で説明した方法に従って、緊急アクセスのプロセスを作成します。Identity Center 管理者アカウントだけにアクセスし、そのアカウントで Identity Center 上でのアクションを実行するための、最小特権のアクセス許可を ID 管理者に付与できます。

障害モード 3: ID センターの中断

万一、IAM Identity Center または AWS リージョンが中断した場合に備えて、AWS Management Consoleへの一時的なアクセスを提供するための構成を設定しておくことをお勧めします。

緊急アクセスのプロセスでは、ID プロバイダーから緊急アカウントの IAM への、直接フェデレーションを使用します。プロセスと設計上の考慮事項の詳細については、「AWS Management Consoleへの緊急アクセスを設定する」を参照してください。

実装手順

すべての障害モードで共通の手順

- 緊急アクセスのプロセス専用の AWS アカウントを作成します。IAM ロール、IAM ユーザー、IAM ID プロバイダー (オプション) など、アカウントで必要となる IAM リソースを事前に作成しておきます。さらに、緊急アクセスアカウントで対応する IAM ロールとの信頼関係を持つクロスアカウントの IAM ロールを、ワークロード AWS アカウントで事前に作成します。AWS CloudFormation StackSets with AWS Organizations を使用して、組織内のメンバーアカウントでこうしたリソースを作成できます。

- メンバー AWS アカウント内のクロスアカウント IAM ロールの削除と変更を拒否する、AWS Organizations サービスコントロールポリシー (SCP) を作成します。
- 緊急アクセス AWS アカウントの CloudTrail を有効にし、ログ収集 AWS アカウントの中央の S3 バケットに証跡イベントを送信します。AWS Control Tower を使用して AWS マルチアカウント環境を設定・管理している場合は、AWS Control Tower を使用して作成した、または AWS Control Tower に登録したすべてのアカウントで CloudTrail がデフォルトで有効になっており、専用のログアーカイブ AWS アカウントの S3 バケットに送信されます。
- 緊急 IAM ロールごとのコンソールログインと API アクティビティに一致する EventBridge ルールを作成して、緊急アクセスアカウントのアクティビティを監視します。インシデント管理システムで追跡中の進行中の緊急事態以外でアクティビティが発生した場合は、セキュリティオペレーションセンターに通知を送信します。

「障害モード 1 : AWS へのフェデレーションに使用する ID プロバイダーが使用できない」、および「障害モード 2: AWS の ID プロバイダー設定が変更された、または有効期限が切れている」での追加手順

- 緊急アクセス用に選択したメカニズムに応じて、リソースを事前に作成します。
  - IAM ユーザーを使用する: 強力なパスワードおよび関連付けられた MFA デバイスを持つ IAM ユーザーを事前に作成します。
  - 緊急アカウントのルートユーザーを使用する: ルートユーザーに強力なパスワードを設定し、そのパスワードをエンタープライズ認証情報ボールトに保存します。複数の物理 MFA デバイスをルートユーザーに関連付け、緊急管理チームのメンバーがすぐにアクセスできる場所に保存します。

「障害モード 3: ID センターの中断」での追加手順

- 「AWS Management Consoleへの緊急アクセスを設定する」で説明しているとおり、緊急アクセス AWS アカウントで IAM ID プロバイダーを作成し、ID プロバイダーからの直接 SAML フェデレーションを有効にします。
- ID プロバイダーで、メンバーのいない緊急オペレーショングループを作成します。
- 緊急アクセスアカウントで、緊急オペレーショングループに対応する IAM ロールを作成します。

リソース

関連する Well-Architected のベストプラクティス:

- SEC02-BP04 一元化された ID プロバイダーを利用する
- SEC03-BP02 最小特権のアクセスを付与する
- SEC10-BP02 インシデント管理計画を作成する
- SEC10-BP07 ゲームデーを実施する

関連ドキュメント:

- AWS Management Consoleへの緊急アクセスを設定する
- SAML 2.0 フェデレーションユーザーの AWS Management Consoleへのアクセスを可能にする
- Break glass access

関連動画:

- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

関連する例:

- AWS Break Glass Role
- AWS customer playbook framework
- AWS incident response playbook samples

SEC03-BP04 アクセス許可を継続的に削減する

必要とするアクセスをチームで決定したら、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、最小特権の原則に従う必要があります。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、認証情報が誤って公開された場合や、許可なくアクセスされた場合の影響範囲が小さく抑えられます。

一般的なアンチパターン:

- デフォルトでユーザーに管理者アクセス許可を付与する。
- 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活用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。例えば、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。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。属性ベースのアクセスコントロール (ABAC) は、アクセス許可ポリシーを各ユーザーに直接アタッチするのではなく、属性を使用してユーザーにアクセス許可を付与できるため、アクセス許可の管理を簡素化するのに役立ちます。

実装手順

- AWS Identity and Access Management Access Analyzer を使用する: IAM Access Analyzer の機能は、外部エンティティと共有されている Amazon Simple Storage Service (Amazon S3) バケットや IAM ロールなど、組織とアカウントのリソースを識別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 ユーザーおよびロールの許容時間枠と使用ポリシーを決定する: 最後にアクセスされたタイムスタンプを使用して、未使用のユーザーとロールを特定し、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て、特定のユーザーとロールのアクセス許可を特定して範囲を設定します。例えば、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 Management Consoleおよびプログラムで使用でき、インフラストラクチャワークフローや自動化ツールに組み込むことができます。
- AWS CloudTrail でのデータイベントのログ記録を検討する: デフォルトでは、CloudTrail は Amazon S3 オブジェクトレベルのアクティビティ (`GetObject` や など `DeleteObject`) や Amazon DynamoDB テーブルアクティビティ (`PutItem` や `DeleteItem` など) などのデータイベントをログに記録しません。これらのイベントのロ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

関連ドキュメント:

- 最小特権アクセス許可を適用する
- 不要な認証情報の削除
- AWS CloudTrail とは
- IAM ポリシーを管理する
- DynamoDB での口グ記録とモニタリング
- S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化
- AWS 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
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

SEC03-BP05 組織のアクセス許可ガードレールを定義する

アクセス許可ガードレールを使用して、プリンシパルに付与できるアクセス許可の範囲を縮小します。アクセス許可ポリシーの評価チェーンには、承認の決定を行う際のプリンシパルの有効なアクセス許可を決定するガードレールが含まれています。 ガードレールは階層型のアプローチで定義できます。特定のガードレールは組織全体に広く適用し、他のガードレールはきめ細かく一時的なアクセスセッションに適用します。

期待される成果: 個別の AWS アカウントを使用して環境を明確に分離できます。 サービスコントロールポリシー (SCP) を使用して、組織全体のアクセス許可ガードレールを定義します。比較的広範なガードレールは組織のルートに近い階層に設定し、比較的厳格なガードレールは各アカウントのレベルの近くで設定します。 リソースポリシー (サポートされている場合) で、プリンシパルがリソースにアクセスするために満たす必要がある条件を定義します。リソースポリシーは、許可される一連のアクションも適宜限定します。 ワークロードのアクセス許可を管理するプリンシパルにアクセス許可境界を設けたうえで、アクセス許可管理が個々のワークロード所有者に委任されています。

一般的なアンチパターン:

- AWS Organization 内に AWS アカウントメンバーを作成しているが、SCP を使用してルート認証情報で利用できる使用とアクセス許可を制限していない。
- 最小特権に基づいてアクセス許可を割り当てているが、付与できるアクセス許可一式に上限を設けるガードレールが敷かれていない。
- AWS IAM の暗黙的な拒否基盤に依存してアクセス許可を制限し、ポリシーが望ましくない明示的な許可のアクセス許可を付与しないことに頼る。
- 同じ AWS アカウント内で複数のワークロード環境を実行していて、アクセス許可境界の設定は VPC、タグ、リソースポリシーなどのメカニズム頼みである。

このベストプラクティスを活用するメリット: アクセス許可ガードレールは、アクセス許可ポリシーが付与しようとしても、望ましくないアクセス許可を付与できないという確信を持つために役立ちます。 考慮する必要があるアクセス許可の最大範囲が縮小されるため、アクセス許可の定義と管理が簡単になります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

組織のアクセス許可ガードレールは、階層型のアプローチで定義することを推奨します。このアプローチでは、層を重ねるに従い、付与できるアクセス許可一式の上限が体系的に引き下げられます。最小特権の原則に基づいてアクセス権を付与できるため、ポリシーの設定ミスによる意図しないアクセスが起きるリスクが軽減されます。

アクセス許可ガードレールを敷くには、まず、ワークロードと環境を個別の AWS アカウントに分離します。 1 つのアカウントのプリンシパルは、両方のアカウントが同じ AWS 組織または同じ組織単位 (OU) にある場合でも、明示的なアクセス許可なしに別のアカウントのリソースにアクセスすることはできません。 OU を使用して、管理対象の複数のアカウントを 1 つのユニットとしてグループ化できます。

次に、組織のメンバーアカウント内のプリンシパルに付与できるアクセス許可一式の上限を引き下げます。 これにはサービスコントロールポリシー (SCP) を使用できます。SCP は OU またはアカウントに適用できます。 SCP では、特定の AWS リージョンへのアクセスを制限する、リソースが削除されないように防ぐ、リスクが懸念されるサービスアクションを無効にするなど、一般的なアクセス制御を適用できます。 組織のルートに適用する SCP は、そのメンバーアカウントにのみ影響し、管理アカウントには影響しません。 SCP は組織内のプリンシパルにのみ適用されます。組織の外部からリソースにアクセスするプリンシパルは対象外です。

さらに、IAM リソースポリシーを使用して、管理対象のリソースに対して実行できるアクションと、動作するプリンシパルが満たす必要がある条件の範囲を絞り込むことができます。 これは、プリンシパルが組織の一部である限り (PrincipalOrgId 条件キーを使用) すべてのアクションを許可するのと同じくらい広範囲にすることも、特定の IAM ロールによる特定のアクションのみを許可するのと同じくらい詳細にすることもできます。 IAM ロールの信頼ポリシーの条件でも、同様のアプローチをとることができます。 リソースまたはロールの信頼ポリシーで、適用対象のロールまたはリソースと同じアカウントのプリンシパルの名前が明示的に指定されている場合、そのプリンシパルには同じアクセス許可を付与する IAM ポリシーをアタッチする必要はありません。 プリンシパルがリソースとは異なるアカウントにある場合は、該当するアクセス許可を付与する IAM ポリシーをプリンシパルにアタッチする必要があります。

多くの場合、ワークロードチームが担当ワークロードに必要なアクセス許可を管理することを望みます。 その場合は、そのチームが新しい IAM ロールとアクセス許可のポリシーを適宜作成する必要があります。 チームが IAM アクセス許可の境界で付与できるアクセス許可の最大範囲をキャプチャし、このドキュメントをチームが IAM ロールとアクセス許可の管理に使用できる IAM ロールに関連付けることができます。 このアプローチにより、業務の完遂に必要な自由をチームに与えつつ、IAM の管理権限を持つことのリスクを軽減できます。

よりきめ細かいステップは、特権アクセス管理 (PAM) と一時的な昇格アクセス管理 (TEAM) の手法を実装することです。 PAM の一例としては、特権が必要なアクションの実行前にプリンシパルに多要素認証の実行を要求します。 詳細については、「MFA 保護 API アクセスの設定」を参照してください。TEAM では、プリンシパルへの昇格アクセスの承認とそのアクセス権を持っていられる期間を管理するソリューションが必要です。 1 つの方法は、昇格アクセス権を持つ IAM ロールのロール信頼ポリシーにプリンシパルを一時的に追加することです。 もう 1 つの方法は、通常のオペレーションでは、セッションポリシーを使用して IAM ロールによってプリンシパルに付与されたアクセス許可の範囲を絞り込み、承認された時間枠内にこの制限を一時的に解除することです。AWS と特定のパートナーが検証したソリューションの詳細については、「一時的な昇格アクセス」を参照してください。

実装手順

1. ワークロードと環境を個別の AWS アカウントに分離します。
2. SCP を使用して、組織のメンバーアカウント内のプリンシパルに付与できるアクセス許可一式の上限を引き下げます。
   a. SCP の作成には、許可リスト方式を採用することをお勧めします。許可したアクションを所定の条件下でのみ認め、それ以外のアクションはすべて拒否します。まず、制御したいリソース (Resources) を定義し、Effect を Deny に設定します。NotAction 要素を使用して、指定したアクション以外のすべてのアクションを拒否します。これを NotLike 条件と組み合わせて、指定したアクションを許可する状況を適宜定義します (StringNotLike や ArnNotLike など)。
   b. 「サービスコントロールポリシーの例」を参照します。
3. IAM リソースポリシーを使用して、リソースに対して許可されるアクションの範囲を限定し、その条件を指定します。 IAM ロールの信頼ポリシーで条件を指定して、ロールを引き受けた場合の制限を設けます。
4. IAM アクセス許可境界を IAM ロールに割り当てます。ワークロードチームがこのロールを使用して、各自のワークロードの IAM ロールとアクセス許可を管理できるようになります。
5. ニーズに基づいて PAM ソリューションや TEAM ソリューションを評価します。

リソース

関連ドキュメント:

- AWS データ境界
- データ境界を使用してアクセス許可のガードレールを確立する
- ポリシーの評価論理

関連する例:

- サービスコントロールポリシーの例

関連ツール:

- AWS 解決策: 一時的な昇格アクセス管理
- TEAM 用の検証済みセキュリティパートナーソリューション

SEC03-BP06 ライフサイクルに基づいてアクセスを管理する

プリンシパル (ユーザー、ロール、グループ) に付与されるアクセス許可を、組織内での各々の全ライフサイクルにわたり監視し、調整します。ユーザーの役割の変更に応じてグループメンバーシップを調整し、ユーザーが組織を離れた時点でアクセス権を取り消します。

期待される成果: 組織内のプリンシパルのライフサイクルを通じてアクセス許可をモニタリングおよび調整し、不要な権限のリスクを減らします。ユーザーの作成時に適切なアクセス権が付与されます。ユーザーの責任が変わった時点でアクセス権を変更し、ユーザーが業務を遂行していないときや組織を離れたときはアクセス権を取り消します。ユーザー、ロール、グループに対する変更を一元管理します。自動化を使用して、AWS 環境に変更を反映します。

一般的なアンチパターン:

- 初期段階で必要とされる以上に過剰または広範なアクセス権限をアイデンティティに事前に付与している。
- アイデンティティの役割と責任が経時的に変化しても、アクセス権限を見直したり調整したりしない。
- 非アクティブまたは終了したアイデンティティに、アクティブなアクセス権限を与えたままにしている。これにより、不正アクセスのリスクが高まります。
- アイデンティティライフサイクルの管理が自動化されていない。

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: 中

実装のガイダンス

アイデンティティ (ユーザー、ロール、グループなど) に付与するアクセス権限は、それらのライフサイクル全体にわたって慎重に管理および調整してください。このライフサイクルには、初期のオン

ボーディングフェーズ、役割と責任の継続的な変更、そして最終的なオフボーディングまたは終了が含まれます。ライフサイクルの段階に基づいてアクセス権をプロアクティブに管理し、適切なアクセスレベルを維持します。最小特権の原則に従い、過剰または不必要なアクセス権限が付与されるリスクを軽減してください。

IAM ユーザーのライフサイクルは AWS アカウント内で直接管理することも、ワークフォース ID プロバイダーと AWS IAM Identity Center とのフェデレーションを通じて管理することもできます。IAM ユーザーの場合は、ユーザーと関連するアクセス許可を AWS アカウント内で作成、変更、削除できます。フェデレーションユーザーについては、IAM Identity Center を使用してライフサイクルを管理できます。その場合は、System for Cross-domain Identity Management (SCIM) プロトコルを使用して、組織の ID プロバイダーからのユーザーとグループの情報を同期します。

SCIM は、さまざまなシステム間でユーザー ID のプロビジョニングとプロビジョニング解除を自動化するためのオープンスタンダードのプロトコルです。SCIM を使用して ID プロバイダーを IAM Identity Center に統合することで、ユーザーとグループの情報を自動的に同期し、組織の信頼できるアイデンティティソースにおける変更に基づいてアクセス権限が付与、変更、または取り消されているか検証できます。

組織内での従業員の役割と責任が変化したら、その従業員のアクセス権限を適宜調整してください。IAM Identity Center のアクセス許可セットを使用して、さまざまな職務または責任を定義し、それらに適切な IAM ポリシーやアクセス許可を関連付けることができます。従業員の役割が変更されたら、割り当てられているアクセス許可セットを更新して、その従業員の新しい責任を反映させることができます。最小特権の原則に従いながら、従業員に必要なアクセス権が与えられていることを確認してください。

実装手順

1. 初期アクセス権の付与、定期的なレビュー、オフボーディングの手順を含む、アクセス管理ライフサイクルのプロセスを定義し、文書化します。
2. IAM のロール、グループ、アクセス許可の境界を実装して、アクセス許可を一元管理し、最大許容アクセスレベルを適用します。
3. フェデレーテッド ID プロバイダー (Microsoft Active Directory、Okta、Ping Identity など) を、ユーザーおよびグループ情報の信頼できるソースとして IAM Identity Center を使用して統合します。
4. SCIM プロトコルを使用して、ID プロバイダーからのユーザー情報やグループ情報を IAM Identity Center の ID ストアに同期します。
5. 組織内のさまざまな職務や責任を表すアクセス許可セットを IAM Identity Center で作成します。アクセス許可セットごとに適切な IAM ポリシーとアクセス許可を定義します。

6. 定期的なアクセスレビュー、迅速なアクセス取り消し、アクセス管理ライフサイクルプロセスの継続的な改善を実施します。
7. アクセス管理のベストプラクティスについて、従業員にトレーニングを行い、周知徹底させます。

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

関連ドキュメント:

- ID ソースを管理する
- IAM Identity Center で ID を管理する
- AWS Identity and Access Management Access Analyzer の使用
- IAM Access Analyzer ポリシーの生成

関連動画:

- AWS re:Inforce 2023 - Manage temporary elevated access with AWS IAM Identity Center
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2022 - Harness power of IAM policies & rein in permissions w/Access Analyzer

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: どの AWS リソースが誰と共有されているかを把握します。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。リソースベースのポリシー (例: Amazon Simple Storage Service (Amazon S3) バケットポリシー) を使用して、または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することで、クロスアカウントアクセスを直接付与できます。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。すべてのアカウントが表示可能であることを確認するために、IAM Access Analyzer で AWS Organizations を設定することを検討します。IAM Access Analyzer では、リソースのアクセス許可をデプロイする前に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用に設計する場合、信頼ポリシーを使用して、ロールを引き受けることができるケースを制御できます。例えば、PrincipalOrgId 条件キーを使用して、AWS Organizations の外部からロールを割り当てる試みを拒否できます。

AWS Config は、誤って設定されたリソースをレポートでき、AWS Config ポリシーチェックを通じて、パブリックアクセスが設定されているリソースを検出できます。AWS Control Tower および AWS Security Hub などのサービスでは、AWS Organizations 全体でチェックとガードレールのデプロイが簡素化され、パブリックに公開されているリソースを特定、修復できます。例えば、AWS Control Tower には、Amazon EBS スナップショットが AWS アカウントによって復元可能かどうかを検出できるマネージドガードレールがあります。

実装手順

- AWS Config に AWS Organizations を使用することを検討する: AWS Config では、AWS Organizations 内の複数のアカウントから委任された管理者アカウントに検出結果を集約できま

す。これにより、包括的なビューが提供され、アカウント間で 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検出できます。

- AWS Identity and Access Management Access Analyzer を設定する: IAM Access Analyzer は、外部エンティティと共有されている Amazon S3 バケットや IAM ロールなど、組織とアカウントのリソースを識別するのに役立ちます。
- AWS Config で自動修復を使用して、Amazon S3 バケットのパブリックアクセス設定の変更に対応する: Amazon S3 バケットのパブリックアクセスブロック設定を自動的にオンにできます。
- モニタリングとアラートを実装して、Amazon S3 バケットがパブリックになったかどうかを確認する: Amazon S3 パブリックアクセスブロックがオフになって、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モニタリングとアラートを設定する必要があります。さらに、AWS Organizations を使用している場合は、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できます。AWS Trusted Advisor は、オープンアクセス許可を持つ 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、「パブリックアクセスを許可する Amazon S3 バケットを AWS Config でモニタリングおよび応答する方法」を参照してください。アクセスを確認するときは、Amazon S3 バケットに含まれるデータの種類を考慮することが重要です。Amazon Macie は、PII、PHI、プライベートや AWS キーなどの認証情報などの機密データを検出して保護するのに役立ちます。

リソース

関連ドキュメント:

- AWS Identity and Access Management Access Analyzer の使用
- 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 を使用した AWS Trusted Advisor チェック結果のモニタリング
- 組織のすべてのアカウント全体で AWS Config ルールを管理する
- AWS Config および AWS Organizations

- Amazon EC2 で使用するために AMI を公開する

関連動画:

- Best Practices for securing your multi-account environment
- Dive Deep into IAM Access Analyzer

SEC03-BP08 組織内でリソースを安全に共有する

ワークロードが増えるにつれて、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されるわけではありません。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でリソースを共有し、データ損失防止イニシアチブを支援することで、意図しないアクセスを最小限に抑えます。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とき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活用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の分析」を確認し、ガバナンスを確立して、外部アクセスを必要なリソースのみに減らします。また、継続的かつ自動的にアラートをモニタリングするプロセスを確立します。

AWS Organizations 内のクロスアカウント共有は、AWS Security Hub、Amazon GuardDuty、AWS Backup など、多数の AWS サービスで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 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) を使用することで、VPC サブネットや Transit Gateway アタッチメント、AWS Network Firewall、Amazon SageMaker Pipelines など、他の一般的なリソースを共有することができます。

アカウントが組織内のリソースのみを共有するように制限するには、サービスコントロールポリシー (SCP) を使用して、外部プリンシパルへのアクセスを防止します。リソースを共有するときは、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し、意図しないアクセスから保護します。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` 条件を使用して、Amazon S3 バケットにアクセスする ID が組織に属していることを確認できます。SCP はサービスにリンクされたロールまたは AWS サービスプリンシパルには適用されないことにご注意ください。

Amazon S3 を使用する場合は、Amazon S3 バケット ACL をオフにし、IAM ポリシーを使用してアクセスコントロールを定義します。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。OAC では AWS Key Management Service によるサーバー側暗号化などの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。外部でリソースを共有するアクセス許可を管理するための規範ガイダンスについては、「アクセス許可の管理」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、作成して一元管理する組織に、複数の AWS アカウントを統合するために使用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で自動的にタスクを実行するために AWS サービスを使用すると、AWS Organizations はそのサービス用のサービスにリンクされた IAM ロール (SLR) を各メンバーアカウントに作成します。AWS Management Console、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有効にするための規範ガイダンスについては、「AWS Organizations を AWS の他のサービスと併用する」および「Organizations と併用できる AWS サービス」を参照してください。

3. データ境界を確立する。

   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。

      認可条件ごとに、「AWS での境界の構築」ホワイトペーパーの「境界の構築」で説明されている手順に従います。ネットワークレイヤーの保護に関する規範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、外部エンティティと共有されている組織とアカウントのリソースを識別するのに役立ちます。IAM Access Analyzer を AWS Security Hub と統合して、IAM Access Analyzer から Security Hub にリソースの検出結果を送信して集約し、環境のセキュリティ体制の分析に役立てることができます。統合するには、アカウントの各リージョンで IAM Access Analyzer と Security Hub の両方を有効にします。AWS Config ルール を使用して設定を監査し、AWS Chatbot で AWS Security Hub を使用して適切な関係者にアラートすることもできます。その後、AWS Systems Manager オートメーションドキュメントを使用して、非準拠のリソースを修正できます。

c. 外部で共有されているリソースを継続的にモニタリング、アラートするための規範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の分析」を参照してください。

4. AWS サービスでリソース共有を使用し、適切に制限します。

   多くの AWS サービスでは、リソースを別のアカウントと共有できます。また、Amazon マシンイメージ (AMI) および AWS Resource Access Manager (AWS RAM) など別のアカウントのリソースをターゲットにできます。`ModifyImageAttribute` API を制限して、AMI を共有する信頼されたアカウントを指定します。AWS RAM を使用して組織への共有のみを制限する場合は、信頼できない ID からのアクセスを防ぐために `ram:RequestedAllowsExternalPrincipals` 条件を指定します。規範ガイダンスと考慮事項については、「リソース共有と外部ターゲット」を参照してください。

5. AWS RAM を使用して、アカウントまたは他の AWS アカウントと安全に共有します。

   AWS RAM は、作成したリソースをアカウント内のロールやユーザー、および他の AWS アカウントとともに安全に共有することができます。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。

   リソースベースのポリシーを使用して過去に共有したリソースが存在する場合、`PromoteResourceShareCreatedFromPolicy` API または同等機能を使用して、リソース共有を完全な AWS RAM リソース共有に昇格できます。

   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。例えば、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
- IAM ロールと信頼ポリシーを使用する方法
- 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 で使用できる AWS のサービス
- AWS でのデータ境界の確立: 信頼できる ID のみ企業データへのアクセスを許可させる

関連動画:

- Granular Access with AWS Resource Access Manager
- Securing your data perimeter with VPC endpoints
- Establishing a data perimeter on AWS

関連ツール:

- 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: ユーザーに関連付けられた長期的な AWS Identity and Access Management (IAM) 認証情報、IAM アクセスキー、シークレットキーは、認証情報が有効かつアクティブな間は誰でも使用することができます。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 ポリシーを制御下に置くことにより、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 を再利用する。

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: 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。例えば、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、外部の ID 条件を使用し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 API を使用している AWS の外部のアプリケーションの 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。例えば、AWS Well-Architected Tool を使用すると、他の AWS アカウントとワークロードを共有でき、AWS Resource Access Manager を使用すると、自分が所有する AWS リソースを他のアカウントと安全に共有することができます。

実装手順

1. クロスアカウントロールを使用して外部アカウントにアクセスできるようにします。

   クロスアカウントロールを使用すると、顧客にサービスを提供する目的で外部アカウントやサードパーティーが保存している機密情報の量を、減らすことができます。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は AWS ワークロードと安全に通信でき、長期的な認証情報の必要性も減らすことができます。

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 を使用すると、IAM 信頼ポリシーでロールを引き受ける人を指定することができます。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、アカウント所有者が、特定の環境においてのみロールを引き受けることができるようにする方法の 1 つでもあります。外部 ID の最も重要な機能は、混乱した代理問題の防止と対処です。

   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と協力して IAM 信頼ポリシーに含める外部 ID 条件を規定します。

3. 外部の汎用一意 ID 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。キー管理の詳細については、「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への対応とその軽減に関する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、または規範的ガイダンスが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、最小特権の原則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。AWS CloudFormation テンプレートを使用してクロスアカウントロールを作成する方法の詳細については、「Cross-Account Roles」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。AWS 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. 変更点を説明します。

   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。

   検知方法に関する規範的なガイダンスについては、「検知」内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
- 検知

関連ドキュメント:

- 例 4 - 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 roles
- IAM チュートリアル: IAM ロールを使用した AWS アカウント間でのアクセスの委任
- AWS 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
- Securely Using External ID for Accessing AWS Accounts Owned by Others
- Extend IAM roles to workloads outside of IAM with IAM Roles Anywhere

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?
- AWS re:Invent 2018: Become an IAM Policy Master in 60 Minutes or Less
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions

関連する例:

- Well-Architected Lab - Lambda cross account IAM role assumption (Level 300)
- Configure cross-account access to Amazon DynamoDB

- AWS STS Network Query Tool

検出

質問

- SEC 4. セキュリティイベントを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をどのように検出して調査するのですか。

ログとメトリクスからイベントをキャプチャして分析し、可視性を得ることができます。ワークロードを保護するため、セキュリティイベントと潜在的な脅威に対するアクションを実行します。

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 標準化した場所にログ、検出結果、メトリクスを取り込む
- SEC04-BP03 セキュリティアラートを相関付けて充実させる
- SEC04-BP04 非準拠リソースの修復を開始する

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のプロセスまたは義務を遂行する必要があるときは、AWS サービスや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にかつ一貫性をもって、タイムリーに取得できるようにしておく必要があります。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) のメカニズムを導入し、ガバナンス、リスク、コンプライアンスの義務における証拠の源とします。

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

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。クエリと取得のメカニズムを選択、有効化、保存、設定してアラートを発するのに不可欠です。

実装手順

- ログのソースを選択して使用します。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。各 AWS アカウントに AWS CloudTrail または AWS Organizations 証跡を使って証跡を作成し、それらの 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 Management Console、AWS CLI、AWS のいずれかを使用して 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し、確認できるようにするには、CloudTrail 証跡を作成して、これを Amazon S3 バケットと Amazon CloudWatch ロググループ (任意) に関連付けます。あるいは、CloudTrail Lake を作成する方法もあります。この方法では、CloudTrail ログを最長 7 年間保持でき、SQL ベースのクエリ機能を利用できます。

  AWS では、VPC を使用しているお客様には、VPC フローログと Amazon Route 53 Resolver 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 ログを有効にし、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、Route 53 Resolver のクエリログは、AWS でのセキュリティ調査をサポートする基本的なログ記録ソースです。Amazon Security Lake を使用して、こ

のログデータを収集、正規化し、Apache Parquet フォーマットと Open Cybersecurity Schema Framework (OCSF) で保存することもできます。これらはクエリに使用できます。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 の検出結果、Amazon Elastic Kubernetes Service (Amazon EKS) 監査ログ、Amazon EC2 インスタンスのオペレーティングシステムとアプリケーションログなど、基本のログソースでは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一覧については、「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 – ログ記録とイベント」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にはログストレージのさまざまなオプションが用意されており、それぞれ独自の保持機能とライフサイクル機能を備えています。最も一般的な 2 つの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。ログ記録の主要な方法が Amazon CloudWatch Logs である場合、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ぶ: どのログストレージを選ぶかは、使用しているクエリツール、保持機能、使いやすさ、コストなどが関わってきます。ログストレージの一般的な選択肢は、Amazon S3 バケットまたは CloudWatch Log 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。
- 適切なログ保持を特定する: 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ときは、各ログソースに対して適切なライフサイクルを選び、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 か月 ~ 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使用する: 組織の各 AWS サービスまたはアプリケーションには、特定のログ記録の設定ガイダンスを確認します。
  - Configure AWS CloudTrail Trail

  - VPC フローログの設定
  - Configure Amazon GuardDuty Finding Export
  - 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 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 Logs Insights 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。

  例えば、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でアラート機能を提供しています。
  - AWS Config 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、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て AWS アカウントとワークロードを保護する脅威検知サービスです。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、Amazon EKS 監査ログなどのソースから情報を取得して、集計、分析します。GuardDuty は、CloudTrail、VPC フローログ、DNS クエリログ、Amazon EKS から、独立したデータストリームを直接プルします。Amazon S3 バケットポリシーを管理したり、ログの収集と保存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 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検知の設定に関する詳細は、「AWS セキュリティインシデント対応ガイド」の「検知」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 標準化した場所にログ、検出結果、メトリクスを取り込む
- SEC07-BP04 スケーラブルな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

関連ドキュメント:

- 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
- Getting started: 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:

- AWS re:Invent 2022 - Introducing Amazon Security Lake

関連する例:

- Assisted Log Enabler for AWS
- AWS Security Hub Findings Historical Export

関連ツール:

- Snowflake for Cybersecurity

SEC04-BP02 標準化した場所にログ、検出結果、メトリクスを取り込む

セキュリティチームは、ログと検出結果に基づいて、不正なアクティビティや意図しない変更を示唆する可能性があるイベントを分析します。この分析の効率を高めるため、標準化した場所にセキュリティログや検出結果を集めてください。 重要なデータポイントを相関のために使えるようになり、ツールの統合を簡素化できます。

期待される成果: ログデータ、検出結果、メトリクスの収集、分析、視覚化に、標準化された方法を使用できます。セキュリティチームは、さまざまなシステムから集めたセキュリティデータを効率的に相関付けて分析し、視覚化して、潜在的なセキュリティイベントを発見し、異常を検知できます。セキュリティ情報とイベント管理 (SIEM) システムやその他のメカニズムを統合してログデータを照会および分析し、セキュリティイベントに対し適時の対応、追跡、エスカレーションを行います。

一般的なアンチパターン:

- チームがそれぞれ独自にログやメトリクスのコレクションを所有および管理していて、それが組織のログ記録の戦略と矛盾している。
- チームが収集したデータの表示と変更を制限するための十分なアクセス制御を実施していない。
- チームがセキュリティログ、検出結果、メトリクスをデータ分類ポリシーに則って管理していない。
- チームがデータ収集の設定時に、データ主権とローカリゼーションの要件を無視している。

このベストプラクティスを活用するメリット: 標準化されたログ記録ソリューションを使用してログデータやイベントを収集しクエリすることで、ログに含まれる情報からより良いインサイトを引き出すことができます。収集したログデータに対して自動ライフサイクルを設定することで、ログの保管にかかるコストを削減できます。収集されたログ情報に対して、データの機密性とチームが求めるアクセスパターンに応じて、きめ細かくアクセスを制御できます。ツールを統合して、データを相関付け、視覚化し、データからインサイトを引き出すことができます。

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

組織内で AWS の使用量が増えれば、分散したワークロードと環境の数が増えます。これらのワークロードと環境が各々、その中のアクティビティに関するデータを生成するため、そのデータを取り込んでローカルに保存することが、セキュリティ運用にとって課題となります。セキュリティチームは、セキュリティ情報とイベント管理 (SIEM) システムなどのツールを使用して、分散したソースからデータを収集し、相関付け、分析、対応のワークフローを実行します。そのためには、さまざまなデータソースにアクセスするための複雑な許可一式を管理する必要があり、抽出、変換、ロード (ETL) プロセスの運用における追加のオーバーヘッドも生じます。

こうした課題を解消するには、セキュリティログデータの関連ソースをすべて、ログアーカイブアカウントに集約することを検討します。詳細については「複数のアカウントで AWS 環境を構成する」を参照してください。これには、ワークロードのすべてのセキュリティ関連データと、AWS CloudTrail、AWS WAF、Elastic Load Balancing、Amazon Route 53 などの AWS サービスによって生成されるログが含まれます。このデータを個別の AWS アカウントの標準化した場所に保存し、適切なクロスアカウントアクセス許可を設定しておくことには、利点がいくつかあります。ワークロードや環境が侵害された場合にその内部でのログの改ざん防止に役立ち、追加のツールの統合先を一元化できるだけでなく、データ保持とライフサイクルの設定モデルを簡素化できます。 データ主権、コンプライアンス範囲、その他の規制の影響を評価して、セキュリティデータの保管場所と保持期間を複数設ける必要があるかどうかを判断します。

ログと検出結果をすぐに取り込んで標準化できるようにするため、ログアーカイブアカウントの Amazon Security Lake を評価します。Security Lake で、CloudTrail、Route 53、Amazon EKS、VPC フローログなどの一般的なソースから自動的にデータを取り込むように設定できます。AWS Security Hub を Security Lake のデータソースとして設定することで、Amazon GuardDuty や Amazon Inspector など他の AWS サービスの検出結果をログデータに関連付けることもできます。 サードパーティーのデータソース統合を利用したり、カスタムのデータソースを設定することもできます。すべての統合により、データが Open Cybersecurity Schema Framework (OCSF) フォーマットに標準化され、Amazon S3 バケットに Parquet ファイルとして保存されるため、ETL 処理は不要になります。

セキュリティデータを標準化された場所に保存することで、高度な分析機能を使用できるようになります。AWS は、AWS 環境で動作するセキュリティ分析ツールを、ログアーカイブのアカウントとは別のセキュリティツール用アカウントにデプロイすることを推奨しています。 このアプローチなら、細かい統制を実装し、ログとログ管理プロセスの整合性と可用性を、ログにアクセスするツールとは別個に保護できます。 Amazon Athena などのサービスを使用して、複数のデータソースを関連付けるオンデマンドクエリを実行することを検討します。Amazon QuickSight などの視覚化ツール

を組み込むこともできます。AI を活用したソリューションがますます普及し、検出結果を人間が読める要約や自然言語での対話に変換するなどの機能を提供しています。 これらのソリューションは多くの場合、標準化したデータ保存場所をクエリ用に用意することで統合しやすくなります。

実装手順

1. ログアーカイブアカウントとセキュリティツール用アカウントを作成する
   a. AWS Organizations を使用して、セキュリティ組織単位の下にログアーカイブアカウントとセキュリティツール用アカウントを作成します。AWS Control Tower を使用して組織を管理している場合、ログアーカイブアカウントとセキュリティツール用アカウントが自動的に作成されます。必要に応じて、これらのアカウントにアクセスして管理するためのロールとアクセス許可を設定します。
2. セキュリティデータの標準化した保存場所を設定する
   a. セキュリティデータの標準化した保存場所の作成戦略を決定します。 これは、一般的なデータレイクアーキテクチャのアプローチ、サードパーティーのデータ製品、Amazon Security Lake などのオプションを使用して実行できます。AWS では、アカウント用にオプトインしている AWS リージョンからも、積極的に使用していない場合でもセキュリティデータを取得することを推奨しています。
3. 標準化した保存場所へのデータソースの公開を設定する
   a. セキュリティデータのソースを特定し、標準化された場所に公開するように設定します。ETL プロセスの開発が必要な形式ではなく、希望する形式でデータを自動的にエクスポートする方法を検討します。Amazon Security Lake を使用すると、サポートされている AWS ソースや統合されたサードパーティーシステムからデータを収集できます。
4. 標準化した場所にアクセスするためのツールを設定する
   a. Amazon Athena、Amazon QuickSight、サードパーティーソリューションなどのツールを設定して、標準化した場所にアクセスできるようにします。 これらのツールをセキュリティツール用アカウント外で動作するように設定し、ログアーカイブアカウントへのクロスアカウントの読み取りアクセス権を適宜設定します。Amazon Security Lake にサブスクライバーを作成し、これらのツールからデータにアクセス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:

- SEC01-BP01 アカウントを使用してワークロードを分ける
- SEC07-BP04 データのライフサイクル管理を定義する

- SEC08-BP04 アクセスコントロールを適用する
- OPS08-BP02 ワークロードログを分析する

関連ドキュメント:

- AWS ホワイトペーパー: 複数のアカウントで AWS 環境を構成する
- AWS Prescriptive Guidance: AWS Security Reference Architecture (AWS SRA)
- AWS Prescriptive Guidance: Logging and monitoring guide for application owners

関連する例:

- Aggregating, searching, and visualizing log data from distributed sources with Amazon Athena and Amazon QuickSight
- How to visualize Amazon Security Lake findings with Amazon QuickSight
- Generate AI powered insights for Amazon Security Lake using Amazon SageMaker Studio and Amazon Bedrock
- Identify cybersecurity anomalies in your Amazon Security Lake data using Amazon SageMaker
- Ingest, transform, and deliver events published by Amazon Security Lake to Amazon OpenSearch Service
- How to use AWS Security Hub and Amazon OpenSearch Service for SIEM

関連ツール:

- Amazon Security Lake
- Amazon Security Lake のパートナー
- Open Cybersecurity Schema Framework (OCSF)
- Amazon Athena
- Amazon QuickSight
- Amazon Bedrock

SEC04-BP03 セキュリティアラートを相関付けて充実させる

予期しないアクティビティが検知されると、さまざまなソースがそれぞれのセキュリティアラートを発します。状況を完全に理解するには、それらをさらに関連付けて情報を強化 (エンリッチメント)

しなくてはなりません。セキュリティアラートの関連付けとエンリッチメントを自動化すると、インシデントの特定と対応をより正確に行えるようになります。

期待される成果: アクティビティによってワークロードや環境内でさまざまなアラートが生成されると、自動メカニズムがデータを関連付け、補足情報によってそのデータを強化します。この前処理のおかげで、イベントについて詳しく把握できるようになり、調査員はイベントの重要度や、正式な対応を必要とするインシデントであるかどうかを判断できます。これにより、監視チームと調査チームの負担が軽減します。

一般的なアンチパターン:

- 職務分掌の要件で別途義務付けられているわけでもないのに、さまざまなグループの人員がさまざまなシステムによって生成された検出結果やアラートを調査している。
- 組織はセキュリティ検出結果と警告データをすべて標準の保存先に集めているが、相関付けやエンリッチメントは調査員が手作業で行う必要がある。
- 検出結果の報告と重要度の決定は、脅威検出システムのインテリジェンスのみに頼っている。

このベストプラクティスを活用するメリット: アラートの関連付けや強化を自動化すると、調査担当者に求められる認知的な負担や手作業によるデータ準備を、全体的に減らすことができます。これにより、イベントがインシデントを示唆しているかどうかを判断し、正式な対応に着手するまでにかかる時間を短縮できます。また、文脈が補足されることで、イベントの実際の重大度を正確に評価できます。実際の重大度は、1 つのアラートが示す重大度よりも高い場合や低い場合が考えられます。

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: 低

実装のガイダンス

セキュリティアラートは、次のような AWS 内のさまざまなソースが生成する可能性があります。

- Amazon GuardDuty、AWS Security Hub、Amazon Macie、Amazon Inspector、AWS Config、AWS Identity and Access Management Access Analyzer、Network Access Analyzer などのサービス。
- Amazon OpenSearch Service のセキュリティ分析など、AWS サービス、インフラストラクチャ、アプリケーションログの自動分析から発せられたアラート。
- Amazon CloudWatch、Amazon EventBridge、AWS Budgets などのソースからの、請求アクティビティの変化に反応したアラート。
- 脅威インテリジェンスのフィードや AWS Partner Networkのセキュリティパートナーソリューションなど、サードパーティーのソース。

- AWS Trust & Safety、または顧客や従業員などその他のソースからの連絡。

アラートには、最も基本的な形式で、誰が (プリンシパルまたはアイデンティティ)、何を (実行されたアクション)、何に (影響を受けるリソース) 対して行っているか、という情報が含まれます。これらのソースごとに、相関付けの土台として、アイデンティティ、アクション、リソースの識別子間のマッピングを作成する方法があるかどうかを確認してください。例えば、アラートソースをセキュリティ情報とイベント管理 (SIEM) ツールと統合して相関付けを自動で行う、独自のデータパイプラインを構築して処理する、またはその両方を組み合わせるという形が考えられます。

ユーザーに代わって関連付けを行うサービスの代表的な例が Amazon Detective です。Detective は、AWS やサードパーティーのさまざまなソースからのアラートを継続的に取り込み、さまざまな形式のインテリジェンスを使用してそれらの関係を視覚的にグラフ化して調査に役立てます。

アラートの初期の重要度は優先順位付けに役立ちますが、実際の重要度はアラートが発生した文脈によって決まります。例えば、Amazon GuardDuty が、ワークロード内の Amazon EC2 インスタンスが想定されていないドメイン名をクエリしたとしてアラートを発したとします。GuardDuty は、このアラートそれ自体には、重要度を「低」と判定しています。ただし、アラートの発生前後の他のアクティビティと自動で相関付けた結果、数百の EC2 インスタンスが同じアイデンティティによってデプロイされていて、全体的な運用コストが増加していることが判明しました。この場合、GuardDuty は、関連付けられたイベントの文脈を新しいセキュリティアラートとして公開し、重要度を「高」に調整する可能性があります。これを受けて、その後のアクションが迅速化します。

実装手順

1. セキュリティアラート情報のソースを特定します。これらのシステムからのアラートがアイデンティティ、アクション、リソースをどのように表すかを理解して、どこで相関付けることができるかを判断してください。
2. さまざまなソースからのアラートを取りまとめるメカニズムを確立します。この用途では、Security Hub、EventBridge、CloudWatch などのサービスを利用することを検討します。
3. データの相関付けとエンリッチメントのためのソースを特定します。ソースの例には、CloudTrail、VPC フローログ、Amazon Security Lake、インフラストラクチャログとアプリケーションログなどがあります。
4. アラートをデータの相関付けやエンリッチメントのソースと統合して、セキュリティイベントのより詳細な文脈を作成し、重要度を設定します。
    a. Amazon Detective、SIEM ツール、その他のサードパーティーソリューションでは、一定レベルの取り込み、相関付け、エンリッチメントを自動的に実行できます。

b. AWS サービスを使用して独自に構築することもできます。例えば、AWS Lambda 関数を呼び出して Amazon Athena クエリを AWS CloudTrail や Amazon Security Lake に対して実行し、結果を EventBridge にパブリッシュできます。

リソース

関連するベストプラクティス:

- SEC10-BP03 フォレンジック機能を備える
- OPS08-BP04 実践的なアラートを作成する
- REL06-BP03 通知を送信する (リアルタイム処理とアラーム)

関連ドキュメント:

- AWS セキュリティインシデント対応ガイド

関連する例:

- How to enrich AWS Security Hub findings with account metadata
- How to use AWS Security Hub and Amazon OpenSearch Service for SIEM

関連ツール:

- Amazon Detective
- Amazon EventBridge
- AWS Lambda
- Amazon Athena

SEC04-BP04 非準拠リソースの修復を開始する

発見的統制は、構成要件に準拠していないリソースについて警告する場合があります。プログラムで定義された修復を手動または自動で開始して、該当するリソースを修正し、潜在的な影響を最小限に抑えることができます。プログラムで修復手順を定義しておくと、迅速に一貫した対応をすることができます。

自動化によってセキュリティの運用を強化できますが、自動化の実装と管理は慎重に行う必要があります。 適切な監視と統制のメカニズムを導入して、自動対応が効果的かつ正確であり、組織の方針やリスクアペタイトに合致していることを検証します。

期待される成果: リソース構成の標準を定義し、リソースが非準拠であることが検出された場合の修復手順も定義します。可能な場合は、修復手順をプログラムで定義して、手動または自動で開始できるようにしておきます。検知システムが導入されていて、このシステムが非準拠リソースを検知して、セキュリティ担当者が監視している一元管理ツールにアラートを発行します。これらのツールは、プログラムによる修復の手動実行または自動実行に対応しています。自動修復については、適切な監視と統制のメカニズムが導入され、その使用が管理されています。

一般的なアンチパターン:

- 自動化を実装しているが、修復アクションを徹底的にテストおよび検証できていない。正当な事業運営が中断されたり、システムが不安定になったりといった、意図しない結果が生じる可能性があります。
- 自動化によって応答時間と手続きは改善されたが、適切な監視や、必要に応じて人間が介入して判断できるメカニズムが欠如している。
- 修復だけに頼り、インシデント対応および復旧プログラムという広い枠組みの中に修復を組み込んでいない。

このベストプラクティスを活用するメリット: 自動修復は、手動プロセスよりも迅速に構成ミスに対応できるため、潜在的なビジネスへの影響が最小限に抑えられ、意図しない使用の可能性が低くなります。修復をプログラムで定義しておけば、一貫して適用されるため、人為的ミスのリスクが軽減されます。また、自動化により大量のアラートを同時に処理することもできます。これは、大規模な運用環境では特に重要です。

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: 中

実装のガイダンス

「SEC01-BP03 管理目標を特定および検証する:」で説明したように、AWS Config などのサービスは、アカウント内のリソースの構成が要件に準拠しているかどうかを監視するのに役立ちます。 非準拠のリソースが検出された場合は、クラウドセキュリティ体制の管理 (CSPM) ソリューション (AWS Security Hub など) にアラートを送信するように設定し、修正に役立てることをお勧めします。これらのソリューションは、セキュリティ調査員が問題を監視して是正措置を講じるための中心的な場所となります。

非準拠リソースの中には、状況が独特で修復には人間の判断が必要となる場合がありますが、プログラムで定義できる標準的な対応で間に合う状況もあります。例えば、VPC セキュリティグループが誤って設定されている場合、標準的な対応として、許可されていないルールを削除して所有者に通知することができます。応答は、AWS Lambda 関数や AWS Systems Manager Automation ドキュメントで定義するか、任意の他のコード環境で定義できます。是正措置を行うために必要最低限のアクセス許可しか持たない IAM ロールを使用して、その環境を AWS に対して認証できるようにしてください。

必要な修復手順を定義したら、その修復を開始する希望の方法を決定できます。AWS Config は修復を自動で開始できます。Security Hub を使用している場合は、この目的でカスタムアクションを使用し、検出結果の情報を Amazon EventBridge に公開できます。それを受けて、EventBridge ルールで修復を開始できます。Security Hub でカスタムアクションを自動または手動で実行するように設定できます。

プログラムによる修復では、実行されたアクションとその結果について包括的なログ記録と監査を行うことをお勧めします。 これらのログを確認および分析して、自動化プロセスの有効性を評価し、改善すべき部分を特定します。ログは Amazon CloudWatch Logs に記録し、修復結果は Security Hub に検出結果メモとして記録します。

手始めに、AWS での自動化されたセキュリティ対応を検討してください。よくあるセキュリティの構成ミスを解決する修復機能が事前に組み込まれています。

実装手順

1. アラートを分析して優先順位を付けます。
   a. さまざまな AWS サービスから届くセキュリティアラートを Security Hub に統合して、可視化、優先順位付け、修復を一元的に行います。
2. 修復手順を考案します。
   a. Systems Manager や AWS Lambda などのサービスを使用して、プログラムによる修復を実行します。
3. 修復の開始方法を設定します。
   a. Systems Manager を使用して、検出結果を EventBridge に公開するカスタムアクションを定義します。これらのアクションを手動または自動で開始するように設定します。
   b. また、Amazon Simple Notification Service (SNS) を使用して、関連するステークホルダー (セキュリティチームやインシデント対応チームなど) に通知やアラートを送信し、必要に応じて手動による介入やエスカレーションを行うこともできます。
4. 修復ログを確認して分析し、有効性と改善点を検討します。

a. ログ出力を CloudWatch Logs に送信します。結果を Security Hub に検出結果メモとして記録します。

リソース

関連するベストプラクティス:

- SEC06-BP03 手動管理とインタラクティブアクセスを削減する

関連ドキュメント:

- AWS Security Incident Response Guide - Detection

関連する例:

- AWS での自動化されたセキュリティ対応
- Monitor EC2 instance key pairs using AWS Config
- Create AWS Config custom rules by using AWS CloudFormation Guard policies
- Automatically remediate unencrypted Amazon RDS DB instances and clusters

関連ツール:

- AWS Systems Manager Automation
- AWS での自動化されたセキュリティ対応

インフラストラクチャの保護

Questions

- 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。
- 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。

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。

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 ネットワークレイヤー内のトラフィックフローを制御する
- SEC05-BP03 検査に基づく保護を実装する
- SEC05-BP04 ネットワーク保護を自動化する

SEC05-BP01 ネットワークレイヤーを作成する

ワークロードコンポーネントをそのデータの機密度とアクセス要件に応じて論理的にグループ化し、そのグループに基づいてネットワークトポロジをさまざまなレイヤーに分割します。インターネットからのインバウンドアクセスを必要とするコンポーネント (公開ウェブエンドポイントなど) と、内部アクセスのみを必要とするコンポーネント (データベースなど) は区別します。

期待される成果: ネットワークのレイヤーは、ワークロードにおける ID の認証と認可の戦略を補完する多層防御セキュリティアプローチに不可欠な部分です。データの機密度とアクセス要件に応じてレイヤーが配置され、トラフィックフローと統制のメカニズムが適切に機能しています。

一般的なアンチパターン:

- 単一の VPC またはサブネットですべてのリソースを作成する。
- データの機密度の要件、コンポーネントの動作、機能を考慮せずにネットワークレイヤーを構築する。
- ネットワークレイヤーのすべての考慮事項について、デフォルトとして VPC とサブネットを使用し、AWS マネージドサービスがトポロジに与える影響を考慮していない。

このベストプラクティスを活用するメリット: ネットワークレイヤーを確立することは、ネットワーク内の不要な経路、特に重要なシステムやデータにつながる経路を制限するための第一歩です。これにより、権限のない攻撃者がネットワークにアクセスしたり、ネットワーク内の他のリソースを操作したりしづらくなります。ネットワークレイヤーが分離されていると、侵入検知やマルウェア防止などの検査システムの分析範囲が狭くなるという利点があります。そのおかげで、誤検出や不要な処理に伴うオーバーヘッドの発生確率が下がります。

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: 高

実装のガイダンス

ワークロードのアーキテクチャを設計する場合、一般的には、コンポーネントをそれぞれの役割に応じて異なるレイヤーに分けます。例えば、ウェブアプリケーションは、プレゼンテーションレイヤー、アプリケーションレイヤー、データレイヤーで構成できます。ネットワークトポロジを設計する際も似たようなアプローチを採用できます。基盤にあるネットワーク統制が、ワークロードのデータアクセス要件を適用するのに役立つことがあります。例えば、3 層のウェブアプリケーションのアーキテクチャでは、プレゼンテーションレイヤーの静的ファイルを Amazon S3 に保存し、それらのファイルを Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) から提供できます。アプリケーションレイヤーには、Application Load Balancer (ALB) が Amazon VPC のパブリックサブネット (非武装地帯 (DMZ) と類似) で提供するパブリックエンドポイントを設け、バックエンドのサービスをプライベートサブネットにデプロイできます。データレイヤーは、データベースや共有ファイルシステムなどのリソースをホストします。アプリケーションレイヤーのリソースとは異なるプライベートサブネットに配置できます。これらのレイヤー境界 (CDN、パブリックサブネット、プライベートサブネット) のそれぞれで統制を効かせて、承認されたトラフィックしかそれらの境界を超えられないようにすることができます。

ワークロードのコンポーネントの機能面の目的に基づいてネットワークレイヤーをモデル化するのと同様に、処理対象のデータの機密度も考慮してください。ウェブアプリケーションの例で考えてみると、ワークロードサービスはすべてアプリケーションレイヤー内にある一方で、それぞれのサービスが処理するデータの機密度は異なる場合があります。この場合、統制の要件によっては、複数のプライベートサブネット、同じ AWS アカウントの異なる VPC、または異なる AWS アカウントの異なる VPC を使用して、データの機密度ごとにアプリケーションレイヤーを分割した方が適切なこともあります。

ネットワークレイヤーについてさらに考慮すべき点は、ワークロードのコンポーネントの動作の一貫性です。引き続き同じ例で言えば、アプリケーションレイヤーにエンドユーザーや統合先の外部システムからの入力を受け入れるサービスがあり、その入力のリスクが他のサービスへの入力と比べて本質的に高いという場合が考えられます。例えば、ファイルのアップロード、実行するコードスクリプト、メールのスキャンなどが該当します。こうしたサービスを独自のネットワークレイヤーに配置すれば、より強固な分離境界を張り巡らせることができ、それらの固有の動作のせいで検査システムで誤検知アラートが発生するのを防止できます。

設計の過程で、AWS マネージドサービスを使用することで、ネットワークトポロジにどのような影響があるかを検討してください。Amazon VPC Lattice などのサービスが、ネットワークレイヤー間のワークロードコンポーネントの相互運用性の向上にいかに役立つかをご確認ください。AWS Lambda を使用する場合は、特に理由がない限り、VPC サブネットにデプロイしてください。イン

ターネットゲートウェイへのアクセスを制限するセキュリティポリシーの遵守を VPC エンドポイントや AWS PrivateLink で簡素化できるのはどこか、判断してください。

実装手順

1. ワークロードのアーキテクチャを見直します。コンポーネントやサービスを、それらが提供する機能、処理対象のデータの機密度、動作に基づいて論理的にグループ化します。
2. インターネットからのリクエストに応答するコンポーネントについては、ロードバランサーやその他のプロキシを使用してパブリックエンドポイントを設けることを検討します。CloudFront、Amazon API Gateway、Elastic Load Balancing、AWS Amplify などのマネージドサービスを利用してパブリックエンドポイントをホストすることで、セキュリティ統制を移行することを検討してください。
3. Amazon EC2 インスタンス、AWS Fargate コンテナ、Lambda 関数など、コンピューティング環境で実行されるコンポーネントの場合、手順 1 で決めたグループに基づいて、これらをプライベートサブネットにデプロイします。
4. Amazon DynamoDB、Amazon Kinesis、Amazon SQS などのフルマネージド型の AWS サービスについては、プライベート IP アドレス経由のアクセスにはデフォルトで VPC エンドポイントを使用することを検討します。

リソース

関連するベストプラクティス:

- REL02 ネットワークトポロジを計画する
- PERF04-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

関連動画:

- AWS re:Invent 2023 - AWS networking foundations

関連する例:

- VPC の例
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront

SEC05-BP02 ネットワークレイヤー内のトラフィックフローを制御する

ネットワークのレイヤー内でさらにセグメンテーションを行い、各ワークロードに必要なフローのみにトラフィックを制限します。まず、インターネットや他の外部システムからワークロードや環境へのトラフィック (North-South トラフィック) の制御に着目します。その後、さまざまなコンポーネントとシステム間のフロー (East-West トラフィック) を確認します。

期待される成果: ワークロードのコンポーネントが相互通信や、クライアントや依存先のその他のサービスとの通信に必要とするネットワークフローのみを許可します。設計では、パブリックとプライベートの送受信、データ分類、地域の規制、プロトコル要件などが考慮されます。可能な限り、最小特権の原則に基づく設計の一環として、ネットワークピアリングよりもポイントツーポイントフローを優先します。

一般的なアンチパターン:

- ネットワークセキュリティに境界ベースのアプローチを採用し、ネットワークレイヤーの境界でのみトラフィックフローを制御する。
- ネットワークレイヤー内のすべてのトラフィックが認証済み、承認済みだと仮定している。
- 受信トラフィックと送信トラフィックのいずれかに制御を適用し、両方には適用していない。
- トラフィックの認証と認可を、ワークロードコンポーネントとネットワーク制御にのみ依存している。

このベストプラクティスを活用するメリット: このプラクティスを実践することで、ネットワーク内の不正な移動のリスクを軽減し、ワークロードに承認のレイヤーを追加できます。トラフィックフロー制御を行うことで、セキュリティインシデントによる影響の範囲を制限し、検出と対応をスピードアップできます。

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

ネットワークレイヤーによって、機能、データの機密レベル、動作が似ているワークロードのコンポーネント群の周りに境界を確立できますが、最小特権の原則に従ってこれらのレイヤー内のコンポーネントをさらにセグメント化する手法を用いて、よりきめ細かくトラフィックを制御できます。AWS 内では、ネットワークレイヤーは主に、Amazon VPC 内の IP アドレス範囲に応じたサブネットを使用して定義されます。また、マイクロサービス環境をビジネスドメインごとにグループ化するなど、さまざまな VPC を使用してレイヤーを定義することもできます。複数の VPC を使用する場合は、AWS Transit Gateway を使用してルーティングを行います。この場合、セキュリティ

グループとルートテーブルを使用してレイヤー 4 レベル (IP アドレスとポートの範囲) でトラフィックを制御できますが、AWS PrivateLink、Amazon Route 53 Resolver DNS Firewall、AWS Network Firewall、AWS WAF などの追加サービスを使用して制御を強化することもできます。

ワークロードのデータフローと通信の要件を接続の開始側、ポート、プロトコル、ネットワークレイヤーの観点から把握し、インベントリを作成します。接続の確立とデータ転送に使用できるプロトコルを評価して、保護要件を満たすプロトコル (例えば、HTTP ではなく HTTPS) を選択します。ネットワークの境界と各レイヤー内の両方で、これらの要件を把握してください。これらの要件を特定できたら、オプションを検討し、必要なトラフィックのみが各接続ポイントを流れるようにします。まず、VPC 内でセキュリティグループを使用することをお勧めします。セキュリティグループは、Amazon EC2 インスタンス、Amazon ECS タスク、Amazon EKS ポッド、Amazon RDS データベースなど、Elastic Network Interface (ENI) を使用するリソースにアタッチできます。レイヤー 4 のファイアウォールとは異なり、セキュリティグループには別のセキュリティグループからのトラフィックを識別子ごとに許可するルールを設定できるため、グループ内のリソースが時間の経過とともに変化しても更新を最小限に抑えることができます。セキュリティグループを使用し、インバウンドルールとアウトバウンドルールの両方を用いて、トラフィックをフィルタリングすることもできます。

トラフィックが VPC 間を移動する場合、シンプルルーティングには VPC ピアリングを使用し、複雑なルーティングには AWS Transit Gateway を使用するのが一般的です。これらのアプローチにより、送信元ネットワークと宛先ネットワークの両方の IP アドレス範囲間のトラフィックフローが円滑になります。ただし、異なる VPC にある特定のコンポーネント間のトラフィックフローのみをワークロードが必要とする場合は、AWS PrivateLink を使用してポイントツーポイント接続を確立することを検討してください。その場合は、プロデューサーとして機能するサービスと、コンシューマーとして機能するサービスを特定します。プロデューサーには互換性のあるロードバランサーをデプロイし、PrivateLink を適宜有効にしてから、コンシューマーからの接続リクエストを受け入れます。その後、プロデューサーサービスには、コンシューマーの VPC からプライベート IP アドレスが割り当てられます。コンシューマーはこれを使用して以降のリクエストを行うことができます。この方法だと、ネットワークのピアリングはほとんど必要なくなります。PrivateLink の評価の中で、データ処理と負荷分散のコストも考慮してください。

セキュリティグループや PrivateLink は、ワークロードのコンポーネント間のフロー制御に役立ちますが、もう 1 つの重要な考慮事項は、リソースがアクセスできる DNS ドメイン (存在する場合) を制御する方法です。VPC の DHCP 構成に応じて、この目的で 2 つの異なる AWS サービスを検討できます。大半のお客様は、VPC で CIDR 範囲 +2 のアドレスを利用できるデフォルトの Route 53 Resolver DNS サービス (別称 Amazon DNS サーバーまたは AmazonProvidedDNS) を使用します。この方法では、DNS ファイアウォールルールを作成して VPC に関連付け、指定したドメインリストに対して実行するアクションを決定できます。

Route 53 Resolver を使用していない場合や、ドメインフィルタリング以外の詳細な検査やフロー制御の機能で Resolver を補完したい場合は、AWS Network Firewall の デプロイを検討してください。このサービスは、ステートレスルールまたはステートフルルールを使用して個々のパケットを検査し、トラフィックを拒否するか許可するかを決定します。AWS WAF を使用してパブリックエンドポイントへのインバウンドウェブトラフィックをフィルタリングする場合も、同様のアプローチを採用できます。これらのサービスの詳細については、「SEC05-BP03 検査に基づく保護を実装する」を参照してください。

実装手順

1. ワークロードのコンポーネント間で必要なデータフローを特定します。
2. セキュリティグループやルートテーブルの使用など、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、多層防御のアプローチで複数の統制を適用します。
3. Route 53 Resolver DNS Firewall、AWS Network Firewall、AWS WAF など、ファイアウォールを使用して、VPC に出入りするネットワークトラフィックに対する制御をきめ細かく定義します。組織全体でファイアウォールルールを一元的に設定および管理するには、AWS Firewall Manager の使用を検討してください。

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
- SEC09-BP02 伝送中に暗号化を適用する

関連ドキュメント:

- VPC のセキュリティのベストプラクティス
- AWS Network Optimization Tips
- Guidance for Network Security on AWS
- Secure your VPC's outbound network traffic in the AWS クラウド

関連ツール:

- AWS Firewall Manager

関連動画:

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield
- AWS re:Inforce 2023: Firewalls and where to put them

関連する例:

- ラボ: CloudFront for Web Application

SEC05-BP03 検査に基づく保護を実装する

ネットワークレイヤー間にトラフィックの検査ポイントを設定して、転送中のデータが、予想されるカテゴリやパターンと一致していることを確認します。 トラフィックフロー、メタデータ、パターンを分析して、イベントの識別、検出、対応をより効果的に行えるようにします。

期待される成果: ネットワークレイヤー間を通過するトラフィックが検査され、承認されます。 許可と拒否の決定は、明示的なルールや脅威インテリジェンス、ベースラインの動作から逸脱しているかどうかに基づいて行われます。 トラフィックが機密データに近づくにつれて、保護は厳格化されます。

一般的なアンチパターン:

- ポートとプロトコルに基づくファイアウォールルールのみに依存している。インテリジェントシステムを利用していない。
- 特定の最新の脅威パターンに基づいてファイアウォールルールを作成しているが、このパターンは変更される可能性がある。
- トラフィックがプライベートサブネットからパブリックサブネットに、またはパブリックサブネットからインターネットに転送される箇所のみを検査している。
- 動作の異常の比較基準となるネットワークトラフィックのベースラインビューがない。

このベストプラクティスを活用するメリット: 検査システムでは、トラフィックデータが特定の条件に該当する場合にのみトラフィックを許可または拒否するなど、インテリジェントなルールを作成できます。脅威の状況は時間とともに変化するので、AWS やパートナーが最新の脅威インテリジェンスに基づいて提供するマネージドルールセットが役立ちます。 これにより、ルールの維持や侵害の兆候の調査にかかるオーバーヘッドが減り、誤検出率が下がります。

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

AWS Network Firewall や、Gateway Load Balancer (GWLB) の背後にデプロイできる AWS Marketplace の他のファイアウォールおよび侵入防止システム (IPS) を使用して、ステートフルネットワークトラフィックとステートレスネットワークトラフィックの両方をきめ細かく制御します。AWS Network Firewall は、Suricata 互換のオープンソースの IPS 仕様に対応し、ワークロードの保護に役立ちます。

AWS Network Firewall と、GWLB を使用するベンダーソリューションはどちらも、さまざまなインライン検査デプロイモデルをサポートしています。 例えば、VPC 単位で検査を実行することや、検査用 VPC に一元化することができます。また、ハイブリッドモデルでデプロイし、East-West トラフィックは検査用 VPC に流し、インターネットの受信トラフィックは VPC 単位で検査することもできます。 もう 1 つ考慮すべき点は、そのソリューションが Transport Layer Security (TLS) のラップ解除に対応しているかどうかです。これにより、どちらの方向から開始されたトラフィックフローでもディープパケット検査が可能になります。これらの構成の詳細については、AWS Network Firewall のベストプラクティスガイドを参照してください。

プロミスキャスモードで動作しているネットワークインターフェイスからのパケットデータの pcap 分析など、アウトオブバンド検査を実行するソリューションを使用している場合は、VPC トラフィックミラーリングを設定できます。ミラーリングされたトラフィックは、インターフェイスで使用可能な帯域幅にカウントされ、ミラーリングされていないトラフィックと同じデータ転送料金が課されます。これらのアプライアンスの仮想バージョンが AWS Marketplace で提供されているかどうかを確認できます。GWLB の背後のインラインデプロイに対応している場合があります。

HTTP ベースのプロトコルを介してトランザクションを行うコンポーネントの場合は、ウェブアプリケーションファイアウォール (WAF) で一般的な脅威からアプリケーションを保護します。AWS WAF は、HTTP(S) リクエストを監視し、設定可能なルールに一致するものを Amazon API Gateway、Amazon CloudFront、AWS AppSync、または Application Load Balancer に送信する前に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。ウェブアプリケーションファイアウォールのデプロイを評価するときは、ディープパケット検査を検討してください。一部、トラフィック検査の前に TLS を終了しなければならないものがあるためです。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 Firewall Manager を使用して、AWS Organization 全体で AWS WAF、AWS Shield Advanced、AWS Network Firewall、Amazon VPC セキュリティグループを一元管理できます。

実装手順

1. 検査ルールの範囲を広く設定できるか (検査用 VPC を使用するなど)、または VPC 単位のよりきめ細かいアプローチが必要かどうかを判断します。
2. インライン検査ソリューションの場合:
   a. AWS Network Firewall を使用する場合は、ルール、ファイアウォールポリシー、ファイアウォール自体を作成します。これらを設定したら、トラフィックをファイアウォールエンドポイントにルーティングして検査を有効にすることができます。
   b. Gateway Load Balancer (GWLB) とサードパーティー製アプライアンスを使用する場合は、アプライアンスを 1 つ以上のアベイラビリティーゾーンにデプロイして構成します。次に、GWLB、エンドポイントサービス、エンドポイントを作成し、トラフィックのルーティングを設定します。
3. アウトオブバンド検査ソリューションの場合:
   1. 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をミラーリングする必要があるインターフェイスで VPC トラフィックミラーリングを有効にします。Amazon EventBridge ルールを使用して、新しいリソースが作成されたときに AWS Lambda 関数を呼び出してインターフェイスでトラフィックミラーリングを有効にすることができます。トラフィックミラーリングセッションを、トラフィックを処理するアプライアンスの前にある Network Load Balancer に送ります。
4. インバウンドのウェブトラフィックソリューションの場合:
   a. AWS WAF を設定するには、まずウェブアクセスコントロールリスト (ウェブ ACL) を設定します。ウェブ ACL は、WAF がトラフィックを処理する方法を定義する、逐次処理されるデフォルトアクション (ALLOW または DENY) を指定したルールのコレクションです。独自のルールとグループを作成することも、ウェブ ACL で AWS マネージドルールグループを使用することもできます。
   b. ウェブ ACL を設定したら、ウェブ ACL を AWS リソース (Application Load Balancer、API Gateway REST API、CloudFront ディストリビューションなど) に関連付けて、ウェブトラフィックの保護を開始します。

リソース

関連ドキュメント:

- What is Traffic Mirroring?
- Implementing inline traffic inspection using third-party security appliances

- AWS Network Firewall example architectures with routing
- Centralized inspection architecture with AWS Gateway Load Balancer and AWS Transit Gateway

関連する例:

- ゲートウェイロードバランサーをデプロイするためのベストプラクティス
- TLS inspection configuration for encrypted egress traffic and AWS Network Firewall

関連ツール:

- AWS Marketplace IDS/IPS

SEC05-BP04 ネットワーク保護を自動化する

Infrastructure as Code (IaC) や CI/CD パイプラインなどの DevOps のプラクティスを活用して、ネットワーク保護のデプロイを自動化します。 これらのプラクティスに則って、ネットワーク保護の変更をバージョン管理システムを通じて追跡し、変更のデプロイにかかる時間を短縮できます。また、ネットワーク保護が目的の設定から逸脱していないかどうかを検知できます。

期待される成果: ネットワーク保護をテンプレートに定義して、バージョン管理システムにコミットします。 新しい変更が加えられると、自動パイプラインが開始して、そのテストとデプロイを調整します。 変更をデプロイ前に検証するための、ポリシーチェックやその他の静的テストが整備されています。 変更をステージング環境にデプロイして、統制が予期したとおりに機能していることを検証します。 統制が承認されると、本番環境へのデプロイも自動的に実行されます。

一般的なアンチパターン:

- 個々のワークロードチームに、ネットワークスタック、保護、自動化の完全な定義を任せている。 ネットワークスタックと保護の標準的な側面が、ワークロードチームが利用できるように中央で公開されていない。
- ネットワーク、保護、自動化のあらゆる側面の定義を中央のネットワークチームに一任している。 ネットワークスタックと保護のワークロード固有の側面を、そのワークロードのチームに委任していない。
- ネットワークチームとワークロードチームの間で一元管理と委任の適切なバランスを取っているが、一貫したテストとデプロイの標準がすべての IaC テンプレートと CI/CD パイプラインにわたっては適用されていない。 テンプレートの準拠状況をチェックするために必要な設定が、ツールに取り込まれていない。

このベストプラクティスを活用するメリット: テンプレートにネットワーク保護を定義しておくと、経時的な変更をバージョン管理システムで追跡し、比較できます。 変更のテストとデプロイを自動化することで、プロセスが標準化されて予測可能性が高まり、デプロイの成功率が上がり、繰り返しの手動設定を省くことができます。

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

「SEC05-BP02 ネットワークレイヤー内のトラフィックフローを制御する」や「SEC05-BP03 検査に基づく保護を実装する」で説明されている多くのネットワーク保護統制では、最新の脅威インテリジェンスを反映して自動更新できるマネージドルールシステムを採用します。 ウェブエンドポイントを保護する例としては、AWS WAF マネージドルールや AWS Shield Advanced アプリケーションレイヤー DDoS の自動緩和などがあります。AWS Network Firewall マネージドルールグループを使用すれば、レピュテーションの低いドメインリストや脅威シグネチャの最新情報が随時反映されます。

マネージドルール以外にも、DevOps のプラクティスを使用して、指定したネットワークリソース、保護、ルールのデプロイを自動化することをお勧めします。 これらを AWS CloudFormation や任意の Infrastructure as Code (IaC) ツールに定義してバージョン管理システムにコミットし、CI/CD パイプラインを使用してデプロイできます。 この方法なら、リリースの予測可能性の向上、AWS CloudFormation Guard などのツールを使用した自動テスト、デプロイした環境が目的の構成とずれている場合の検知など、ネットワーク統制管理に関する DevOps の従来のメリットが得られます。

「SEC05-BP01 ネットワークレイヤーを作成する」の過程で行った決断に伴い、一元管理のアプローチで受信フロー、送信フロー、検査フロー専用の VPC を作成する場合があります。「AWS Security Reference Architecture (AWS SRA)」で説明されているとおり、これらの VPC は専用のネットワークインフラストラクチャアカウントで定義できます。 同様の手法を用いて、他のアカウントのワークロード、そのセキュリティグループ、AWS Network Firewall デプロイ、Route 53 Resolver ルールと DNS ファイアウォール構成、その他のネットワークリソースが使用する VPC を一元的に定義できます。 これらのリソースは、AWS Resource Access Manager を使用して他のアカウントと共有できます。 このアプローチなら、管理先が 1 つになり、ネットワーク統制の自動テストとネットワークアカウントへの自動デプロイを簡素化できます。 これは、ハイブリッドモデルで実現できます。つまり、特定の統制を一元的にデプロイして共有し、他の統制を個々のワークロードチームとそれぞれ該当するアカウントに委任します。

実装手順

1. ネットワークと保護のどの側面を一元的に定義し、どの側面をワークロードチームで管理できるのか、所有権を明確にします。

2. ネットワークとその保護に対する変更をテストし、デプロイする環境を作成します。 例えば、ネットワークテストアカウントとネットワーク本稼働アカウントを用意します。
3. テンプレートをバージョン管理システムに保存し、管理する方法を決定します。 中央テンプレートはワークロードリポジトリとは別のリポジトリに保存し、ワークロードテンプレートはそのワークロード固有のリポジトリに保存できます。
4. テンプレートをテストしてデプロイするための CI/CD パイプラインを作成します。 設定ミスがないかチェックし、テンプレートが会社の標準に準拠していることを確認するテストを定義します。

リソース

関連するベストプラクティス:

- SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する

関連ドキュメント:

- AWS セキュリティリファレンスアーキテクチャ - Network account

関連する例:

- AWS Deployment Pipeline Reference Architecture
- NetDevSecOps to modernize AWS networking deployments
- Integrating AWS CloudFormation security tests with AWS Security Hub and AWS CodeBuild reports

関連ツール:

- AWS CloudFormation
- AWS CloudFormation Guard
- cfn_nag

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は、外部や内部のネットワークベースの脅威から保護するための複数の防御層を必要とします。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 強化されたイメージからコンピューティングをプロビジョニングする
- SEC06-BP03 手動管理とインタラクティブアクセスを削減する
- SEC06-BP04 ソフトウェアの整合性を検証する
- SEC06-BP05 コンピューティング保護を自動化する

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します。Amazon EC2 インスタンス、Amazon Elastic Container Service (Amazon ECS) コンテナ、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します。Amazon Relational Database Service (Amazon RDS) データベースなどの 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します。静的コードスキャンを使用して、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を検査します。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を調べるために、Web アプリケーションペンテストを検討してください。

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し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。自動化は、ワークロードの問題や想定外の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、時間を節約し、さらなる問題を発生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。開発中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します。

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とどのように関連しているかを十分に理解する必要があります。責任共有モデルでは、AWS は 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、施設で構成されます。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のクラウド内のセキュリティ、および 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応じて異なります。例えば、AWS はマネージド型リレーショナルデータベースサービスである 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に対する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さまざまなサービスがあります。Amazon Inspector は、ソフトウェアの問題や想定外のネットワークアクセスについて、AWS ワークロードを継続的にスキャンします。AWS Systems ManagerPatch Manager は、Amazon EC2 インスタンス全体のパッチ適用を管理するのに役立ちます。Amazon Inspector と Systems Manager は、AWS セキュリティチェックの自動化とセキュリティアラートの一元化に役立つクラウドセキュリティ体制管理サービスである、AWS Security Hub で表示できます。

Amazon CodeGuru は、静的コード分析を使用して Java および Python アプリケーションの潜在的な問題を特定するのに役立ちます。

実装手順

- Amazon Inspector を設定する: Amazon Inspector は、新しく起動された Amazon EC2 インスタンス、Lambda 関数、および Amazon ECR にプッシュされた対象となるコンテナイメージを自動的に検出し、ソフトウェアの問題、潜在的な欠陥、およびs想定外のネットワークへの露出がないかを直ちにスキャンします。
- 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、問題や欠陥がないかを調べます。Amazon CodeGuru は、Java アプリケーションと Python アプリケーションの両方で、一般

的なセキュリティ問題を修正するためのレコメンデーションをスキャンして提供します。OWASP Foundation は、ソースコード分析ツール (SAST ツールとも呼ばれます) のリストを発行します。

- CI/CD パイプライン構築プロセスの一環として、既存の環境をスキャンしてパッチを適用するメカニズムを実装する: 依存関係やオペレーティングシステムの問題をスキャンしてパッチを適用し、新しい脅威から保護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。ソフトウェアの脆弱性管理は、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把握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 サービスに応じて異なります。Amazon EC2 インスタンスで実行されているソフトウェアで潜在的な問題を確認するには、Amazon Inspector をパイプラインに追加して警告します。問題または潜在的な欠陥が検出された場合は、ビルドプロセスを中止します。Amazon Inspector は リソースを継続的に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、AWS Partnerツールなどのオープンソース製品を使用することもできます。
- 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 ManagerPatch Manager は、セキュリティ関連のアップグレードと他の種類のアップグレードの両方で、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 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。Patch Manager は、Amazon EC2 に加えて、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。

  サポートされているオペレーティングシステムのリストについては、「Systems Manager ユーザーガイド」の「サポートされているオペレーティングシステム」を参照してください。インスタンスをスキャンし、見つからないパッチのレポートのみを表示できます。または、すべての見つからないパッチをスキャンして自動的にインストールできます。
- AWS Security Hub を使用する: セキュリティハブは、AWS のセキュリティ状態の包括的ビューを提供します。複数の AWSのサービス全体でセキュリティデータを収集し、標準化された形式でこれらの検出結果を提供するために、AWS のサービス全体でセキュリティ検出結果を優先できます。
- AWS CloudFormation を使用する: AWS CloudFormation は、リソースのデプロイを自動化し、複数のアカウントと環境でリソースアーキテクチャを標準化することで脆弱性管理に役立つ、Infrastructure as Code (IaC) サービスです。

リソース

関連ドキュメント:

- AWS Systems Manager
- AWS Lambda のセキュリティの概要
- Amazon CodeGuru
- Improved, Automated Vulnerability Management for Cloud Workloads with a New Amazon Inspector
- Automate vulnerability management and remediation in AWS using Amazon Inspector and AWS Systems Manager – Part 1

関連動画:

- 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

SEC06-BP02 強化されたイメージからコンピューティングをプロビジョニングする

強化されたイメージからランタイム環境をデプロイすることで、ランタイム環境への意図しないアクセスの機会を減らします。コンテナイメージやアプリケーションライブラリなどのランタイム依存関係は、信頼できるレジストリからのみ取得し、その署名を検証します。独自のプライベートレジストリを作成して、ビルドおよびデプロイのプロセスで使用する、信頼できるイメージとライブラリを保存します。

期待される成果: コンピューティングリソースは、強化されたベースラインイメージからプロビジョニングされます。コンテナイメージやアプリケーションライブラリなどの外部依存関係は、信頼できるレジストリからのみ取得し、その署名を検証します。これらはプライベートレジストリに保存され、ビルドおよびデプロイのプロセスで参照できます。新たに発見された脆弱性に対応できるように、イメージと依存関係を定期的にスキャンして更新します。

一般的なアンチパターン:

- 信頼できるレジストリからイメージとライブラリを取得しているが、使用前に署名の検証や脆弱性スキャンを行っていない。
- イメージを強化しているが、新たな脆弱性がないか定期的にテストしたり、最新バージョンに更新したりしていない。

- イメージの想定されるライフサイクル中に、不要なソフトウェアパッケージをインストールしている、または削除していない。
- 本番環境のコンピューティングリソースを最新の状態に保つために、パッチの適用しか行っていない。パッチを適用するだけでは、経時的に、コンピューティングリソースが強化された標準に準拠しなくなる可能性があります。また、パッチを適用しても、セキュリティイベントの発生時に脅威アクターによってインストールされた可能性のあるマルウェアを削除できない場合があります。

このベストプラクティスを活用するメリット: イメージを強化すると、権限のないユーザーやサービスへの意図しないアクセスを許可する可能性がある、ランタイム環境で利用可能なパスの数を減らすことができます。また、万一、不正アクセスが発生した場合でも、影響の範囲を低減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

システムを強化するには、まず、オペレーティングシステム、コンテナイメージ、アプリケーションライブラリを最新バージョンにします。既知の問題にはパッチを適用します。不要なアプリケーション、サービス、デバイスドライバー、デフォルトユーザー、その他の認証情報を削除し、システムを最小限に抑えます。ポートを無効にしてワークロードに必要なリソースと機能のみを備えた環境を作成するなど、その他の必要な対策を行ってください。その状態をベースラインとして、ワークロードの監視や脆弱性管理などの目的に必要なソフトウェア、エージェント、その他のプロセスをインストールします。

Center for Internet Security (CIS) や Defense Information Systems Agency (DISA) の Security Technical Implementation Guide (STIG) など、信頼できるソースが提供するガイダンスを利用することで、システム強化の負担を軽減できます。AWS または APN パートナーによって公開された Amazon マシンイメージ (AMI) から開始して、AWS EC2 Image Builder を使用し、CIS コントロールと STIG コントロールの適切な組み合わせに従って構成を自動化することをお勧めします。

CIS または DISA STIG のレコメンデーションを適用する強化済みのイメージや EC2 Image Builder レシピが用意されていますが、それらの構成によって、ご利用のソフトウェアの正常な実行が妨げられる場合があります。このような状況では、まず、強化されていないベースイメージを用意してソフトウェアをインストールし、CIS の統制を段階的に適用してその影響をテストします。ソフトウェアの実行を妨げている CIS 統制がある場合は、代わりに DISA でよりきめ細かな強化のレコメンデーションを実装できるかをテストしてください。正常に適用できる各種の CIS 統制とDISA STIG 構成を記録しておきましょう。これらを使用して、イメージ強化のレシピを EC2 Image Builder で適宜、定義します。

コンテナ化されたワークロードの場合、Docker の強化されたイメージは、Amazon Elastic Container Registry (ECR) パブリックリポジトリで利用できます。EC2 Image Builder を使用して、AMI と共にコンテナイメージを強化できます。

オペレーティングシステムやコンテナイメージと同様に、pip、npm、Maven、NuGet などのツールを通じて、パブリックリポジトリからコードパッケージ (またはライブラリ) を取得できます。AWS CodeArtifact 内などのプライベートリポジトリを信頼できるパブリックリポジトリと統合して、コードパッケージを管理することをお勧めします。この統合により、パッケージを取得、保存し、最新の状態に保つことができます。その後、アプリケーションビルドプロセスで、Software Composition Analysis (SCA)、Static Application Security Testing (SAST)、および Dynamic Application Security Testing (DAST) などの手法を使用して、これらのパッケージの最新バージョンをアプリケーションと一緒に取得し,テストできます。

AWS Lambda を使用するサーバーレスワークロードの場合、Lambda レイヤーを使用してパッケージの依存関係を簡単に管理できます。Lambda レイヤーを使用して、さまざまな関数間で共有される一連の標準依存関係をスタンドアロンアーカイブに構成します。独自のビルドプロセスでレイヤーを作成して管理できるため、関数を一元的に最新の状態に保つことができます。

実装手順

- オペレーティングシステムを強化する。信頼できるソースから取得したベースイメージを、強化した AMI を構築するための基盤として使用します。EC2 Image Builder を使用すると、イメージにインストールされたソフトウェアをカスタマイズできます。
- 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に、コンテナ化されたリソースを設定します。コンテナを使用する場合は、ビルドパイプラインに ECR イメージスキャンを実装し、イメージリポジトリに対して定期的にコンテナ内の CVE を探します。
- AWS Lambda でサーバーレス実装を使用する場合は、Lambda レイヤーを使用して、アプリケーション関数コードと共有依存ライブラリを分離します。Lambda でコード署名を設定すると、信頼できるコードのみを Lambda 関数で実行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:

- OPS05-BP05 パッチ管理を実行する

関連動画:

- Deep dive into AWS Lambda security

関連する例:

- Quickly build STIG-compliant AMI using EC2 Image Builder
- Building better container images
- Using Lambda layers to simplify your development process
- Develop & Deploy AWS Lambda Layers using Serverless Framework
- 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

SEC06-BP03 手動管理とインタラクティブアクセスを削減する

デプロイ、構成、保守、調査のタスクは、可能な限り自動化します。自動化を利用できない場合は、緊急対応が必要な事態や安全な (サンドボックス) 環境でのコンピューティングリソースへの手動アクセスを検討してください。

期待される成果: プログラムスクリプトと自動化ドキュメント (ランブック) は、コンピューティングリソースに対する承認されたアクションをキャプチャします。これらのランブックは、変更検出システムによって自動的に開始されるか、人間による判断が必要な場合は手動で開始されます。コンピューティングリソースへの直接アクセスは、自動化を利用できない緊急時にのみ可能です。手動のアクティビティはすべてログに記録され、自動化機能を継続的に改善していくため、レビュープロセスに組み込まれます。

一般的なアンチパターン:

- SSH や RDP などのプロトコルを使用して、Amazon EC2 インスタンスにインタラクティブにアクセスする。
- `/etc/passwd` や Windows ローカルユーザーなどの個々のユーザーログインを維持する。
- インスタンスにアクセスするためのパスワードまたはプライベートキーを複数のユーザー間で共有している。
- 手動でソフトウェアをインストールし、構成ファイルを作成または更新している。
- 手動でソフトウェアを更新するかパッチを適用する。
- インスタンスにログインして問題をトラブルシューティングする。

このベストプラクティスを活用するメリット: 自動化を使用してアクションを実行すると、意図しない変更や設定ミスの運用リスクを軽減できます。インタラクティブアクセスに Secure Shell (SSH) や Remote Desktop Protocol (RDP) を使用しなくなれば、コンピューティングリソースへのアクセスの範囲が限定されます。これにより、不正行為に対して一般的な経路を遮断できます。コンピューティングリソースの管理タスクを自動化ドキュメントやプログラムスクリプトに定義しておくことで、承認されるアクティビティの全範囲をきめ細かく定義および監査するメカニズムを確立できます。

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

インスタンスへのログインは、システム管理における従来のアプローチです。サーバーのオペレーティングシステムをインストールした後、ユーザーは通常手動でログインしてシステムを設定し、必要なソフトウェアをインストールします。サーバーの存続期間中、ユーザーはログインしてソフトウェアの更新、パッチの適用、構成の変更、問題のトラブルシューティングを行うことができます。

ただし、手動アクセスは多くのリスクを伴います。サーバーは SSH や RDP サービスなどのリクエストをリスニング (待ち受け) しなければならず、それが不正アクセスに対して経路を開くことになりかねません。また、手作業による人為的ミスのリスクも高まります。その結果、ワークロードインシデント、データの破損や破壊、その他のセキュリティ問題が発生するおそれがあります。また、人為的アクセスには認証情報の共有に対する保護も必須となり、管理オーバーヘッドが増えます。

これらのリスクを軽減するために、AWS Systems Manager などのエージェントベースのリモートアクセスソリューションを実装できます。AWS Systems Managerエージェント (SSM エージェント) は、暗号化されたチャネルを自ら確立するため、外部発信のリクエストをリッスンする必要がありません。VPC エンドポイントを介してこのチャネルを確立するように、SSM エージェントを設定することを検討してください。

Systems Manager では、マネージドインスタンスとの対話方法をきめ細かく制御できます。実行する自動化、実行できるユーザー、実行できるタイミングを定義します。Systems Manager は、インスタンスへのインタラクティブなアクセスなしで、パッチの適用、ソフトウェアのインストール、および設定変更を行うことができます。Systems Manager は、リモートシェルへのアクセスを提供し、セッション中に呼び出されたすべてのコマンドとその出力を、ログと Amazon S3に記録することもできます。AWS CloudTrail は、検査のために Systems Manager API の呼び出しを記録します。

実装手順

1. Amazon EC2 インスタンスに、AWS Systems Manager Agent (SSM エージェント) をインストールします。SSM Agent が基本の AMI 構成に組み込まれていて、自動的に起動されるかどうかを確認してください。
2. EC2 インスタンスプロファイルに関連付けられた IAM ロールに、`AmazonSSMManagedInstanceCore` マネージド IAM ポリシーが含まれていることを確認します。
3. インスタンスで実行されている SSH、RDP、その他のリモートアクセスサービスを無効にします。このためには、起動テンプレートのユーザーデータセクションで設定したスクリプトを実行するか、EC2 Image Builder などのツールを使用してカスタムの AMI を構築します。
4. EC2 インスタンスに適用されるセキュリティグループの受信 (イングレス) ルールで、ポート 22/tcp (SSH) またはポート 3389/tcp (RDP) へのアクセスが許可されていないことを確認します。AWS Config などのサービスを使用して、セキュリティグループの構成ミスに対する検出とアラートを実装します。
5. Systems Manager で適切な自動化、ランブックを定義し、コマンドを実行します。IAM ポリシーを使用して、これらのアクションを実行できるユーザーと、アクションが許可される条件を定義します。これらの自動化を本稼働環境以外で徹底的にテストしてください。インスタンスにインタラクティブにアクセスする代わりに、必要に応じてこれらの自動化を呼び出します。
6. AWS Systems Manager Session Manager を使用し、必要に応じてインスタンスへのインタラクティブなアクセスを提供します。セッションアクティビティのログ記録を有効にして、Amazon CloudWatch Logs または Amazon S3 で監査証跡を維持します。

リソース

関連するベストプラクティス:

- 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

関連する例:

- Replacing SSH access to reduce management and security overhead with AWS Systems Manager

関連ツール:

- AWS Systems Manager

関連動画:

- Controlling User Session Access to Instances in AWS Systems Manager Session Manager

SEC06-BP04 ソフトウェアの整合性を検証する

暗号化技術による検証を使用して、ワークロードが使用するソフトウェアアーティファクト (イメージを含む) の整合性を検証します。 コンピューティング環境内で行われる不正変更への対策として、暗号化技術によりソフトウェアに署名します。

期待される成果: すべてのアーティファクトが信頼できるソースから取得されます。ベンダーのウェブサイトの証明書が検証されます。 ダウンロードしたアーティファクトは、署名により暗号化技術を使用して検証されます。独自のソフトウェアは暗号化技術を使用して署名され、コンピューティング環境によって検証されます。

一般的なアンチパターン:

- 定評あるベンダーのウェブサイトを信頼してソフトウェアアーティファクトを入手しているが、証明書の有効期限の通知を無視している。 証明書が有効であることを確認せずにダウンロードを続行する。
- ベンダーウェブサイトの証明書を検証するが、これらのウェブサイトからダウンロードしたアーティファクトについては、暗号化技術による検証を行わない。
- ソフトウェアの整合性の検証をダイジェストまたはハッシュのみに頼っている。 ハッシュでは、アーティファクトが元のバージョンから変更されていないことは確認できますが、ソースは検証されません。
- 独自のソフトウェア、コード、またはライブラリに署名しない (独自のデプロイでしか使用しない場でも、署名は必要です)。

このベストプラクティスを活用するメリット: ワークロードが依存するアーティファクトの整合性を検証すると、マルウェアがコンピューティング環境に侵入するのを防ぐことができます。 ソフトウェアに署名することで、コンピューティング環境での不正実行を防ぐことができます。 コードに署名して検証することで、ソフトウェアサプライチェーンが保護されます。

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: 中

実装のガイダンス

オペレーティングシステムイメージ、コンテナイメージ、コードアーティファクトは、多くの場合、ダイジェストやハッシュなどによる整合性チェックが可能な状態で配布されます。 その場合、クラ

イアントはペイロードのハッシュを独自に計算し、それが公開されたものと同じであることを検証することで、整合性を検証できます。 これらのチェックはペイロードが改ざんされていないことを確認するのに役立ちますが、ペイロードが元のソース (データの来歴) から送信されたかどうかは検証しません。 データの来歴を確認するには、信頼できる機関がアーティファクトにデジタル署名するために発行した証明書が必要です。

ダウンロードしたソフトウェアまたはアーティファクトをワークロードで使用している場合は、プロバイダーがデジタル署名検証用のパブリックキーを提供しているかどうかを確認してください。 AWS では、公開しているソフトウェアのパブリックキーと検証手順を提供しています。以下に例を示します。

- EC2 Image Builder: AWS TOE インストールダウンロードの署名を検証します
- AWS Systems Manager: SSM エージェントの署名を検証します
- Amazon CloudWatch: CloudWatch エージェントパッケージの署名を検証します

「SEC06-BP02 強化されたイメージからコンピューティングをプロビジョニングする」で説明しているように、イメージの取得と強化に使用するプロセスにデジタル署名検証を組み込みます。

AWS Signer を使用して、署名の検証と、独自のソフトウェアとアーティファクトの独自のコード署名ライフサイクルを管理できます。 AWS Lambda と Amazon Elastic Container Registry はどちらも Signer との統合が可能で、コードとイメージの署名を検証します。 「リソース」セクションの例を参考にして、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインに Signer を組み込み、署名の検証と独自のコードやイメージへの署名を自動化できます。

リソース

関連ドキュメント:

- Cryptographic Signing for Containers
- Best Practices to help secure your container image build pipeline by using AWS Signer
- Announcing Container Image Signing with AWS Signer and Amazon EKS
- AWS Lambda でのコード署名の設定
- Best practices and advanced patterns for Lambda code signing
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys

関連する例:

- Automate Lambda code signing with Amazon CodeCatalyst and AWS Signer
- Signing and Validating OCI Artifacts with AWS Signer

関連ツール:

- AWS Lambda
- AWS Signer
- AWS Certificate Manager
- AWS Key Management Service
- AWS CodeArtifact

SEC06-BP05 コンピューティング保護を自動化する

コンピューティング保護操作を自動化して、人的介入の必要性を減らします。自動スキャンを使用してコンピューティングリソース内の潜在的な問題を検知し、プログラムによる自動応答またはフリート管理操作で修正します。 CI/CD プロセスに自動化を組み込むことで、最新の依存関係を反映した信頼できるワークロードをデプロイできます。

期待される成果: 自動化システムは、コンピューティングリソースのすべてのスキャンとパッチ適用を実行します。自動検証を使用して、ソフトウェアイメージと依存関係が信頼できるソースから取得され、改ざんされていないことを確認します。ワークロードは自動的に最新の依存関係をチェックし、AWS コンピューティング環境での信頼度を確立するために署名されます。 非準拠リソースが検出されると、自動修復が開始します。

一般的なアンチパターン:

- イミュータブルインフラストラクチャの慣習に従っているが、本稼働システムの緊急時のパッチ適用や交換に備えたソリューションが不在である。
- 誤った構成のリソースを自動修正しているが、手動によるオーバーライドメカニズムが導入されていない。 要件の調整が必要となる事態が発生し、そうした変更を行うまで自動化を中断しなければならない場合が考えられます。

このベストプラクティスを活用するメリット: 自動化は、コンピューティングリソースへの不正アクセスと不正使用のリスクを軽減します。 本番環境に構成ミスが波及しないよう防ぎ、構成ミスが生じた場合は検知して修正できます。 コンピューティングリソースへの不正アクセスや不正使用を検

知し、対応にかかる時間を短縮するうえでも、自動化が役に立ちます。 その結果、問題による全体的な影響範囲を縮小できます。

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

コンピューティングリソースを保護するために、セキュリティの柱のプラクティスで説明されている自動化を適用できます。「SEC06-BP01 脆弱性管理を実行する」では、CI/CD パイプラインの両方で Amazon Inspector を使用し、ランタイム環境を継続的にスキャンして既知の共通脆弱性識別子 (CVE) を見つける方法を説明しています。 AWS Systems Manager を使用してパッチを適用したり、自動ランブックを通じて新しいイメージから再デプロイしたりして、コンピューティングフリートを最新のソフトウェアやライブラリで更新できます。 これらの手法を使用すれば、手作業によるプロセスやコンピューティングリソースへのインタラクティブアクセスの必要性を低減できます。 詳細については、「SEC06-BP03 手動管理とインタラクティブアクセスを削減する」を参照してください。

自動化は、「SEC06-BP02 強化されたイメージからコンピューティングをプロビジョニングする」と「SEC06-BP04 ソフトウェアの整合性を検証する」で説明している、信頼できるワークロードのデプロイでも役割を果たします。 EC2 Image Builder、AWS Signer、AWS CodeArtifact、Amazon Elastic Container Registry (ECR) などのサービスを使用することで、強化され、承認されたイメージとコードの依存関係をダウンロード、検証、コンストラクト、保存できます。 Inspector と同様、これらのサービスがそれぞれ CI/CD プロセスで役割を果たし、依存関係が最新であり、出所が信頼できるソースであることが確認された場合にのみ、ワークロードが本番環境に投入されるようにします。 ワークロードも署名されるため、AWS Lambda や Amazon Elastic Kubernetes Service (EKS) などの AWS コンピューティング環境は、実行を許可する前に改ざんされていないことを確認できます。

このような予防的統制のほかに、コンピューティングリソースの発見的統制でも自動化を活用できます。 一例として、AWS Security Hub は、[EC2.8] EC2 インスタンスはインスタンスメタデータサービスバージョン 2 (IMDSv2) を使用する必要があるか、などのチェックを含む、NIST 800-53 Rev.5標準を提供しています IMDSv2 はセッション認証の手法を使用して、X-Forwarded-For HTTP ヘッダーを含むリクエストをブロックし、ネットワーク TTL を 1 に設定して、EC2 インスタンスに関する情報を取得する外部ソース発信のトラフィックを停止します。セキュリティハブのこのチェックでは、EC2 インスタンスが IMDSv1 を使用しているタイミングを検出し、自動修復を開始できます。。自動検出と修復の詳細については、「SEC04-BP04 非準拠リソースの修復を開始する」を参照してください。

実装手順

1. EC2 Image Builder を使用して、安全で規制に準拠し、強化された AMI の作成を自動化します。 基本の AWS イメージや APN パートナーイメージから、Center for Internet Security (CIS) ベンチマークまたは Security Technical Implementation Guide (STIG) 標準の統制を組み込んだイメージを作成できます。
2. 構成管理を自動化します。構成管理のサービスやツールを使うことで、コンピューティングリソースで安全性の高い構成を自動的に適用および検証します。
   a. AWS Config を使用した自動構成管理
   b. AWS Security Hub を使用したセキュリティとコンプライアンス体制の自動管理
3. Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのパッチ適用または置き換えを自動化する。AWSSystems Manager Patch Manager は、セキュリティ関連および他の種類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することができます。
   a. AWS Systems Manager Patch Manager
4. 共通脆弱性識別子 (CVE) を検知するためのコンピューティングリソースのスキャンを自動化し、セキュリティスキャンソリューションをビルドパイプラインに埋め込みます。
   a. Amazon Inspector
   b. ECR イメージスキャン
5. コンピューティングリソースを保護するために、マルウェアと脅威を自動検出する Amazon GuardDuty を検討してください。GuardDuty は、AWS 環境で AWS Lambda 関数が呼び出されたときに発生する可能性のある問題を特定することもできます。
   a. Amazon GuardDuty
6. AWS パートナー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統制に匹敵するか同等の、またはそれらと統合できる、業界をリードする製品を提供しています。これらの製品で AWS の既存のサービスを補完して、包括的なセキュリティアーキテクチャをデプロイし、クラウド環境とオンプレミス環境の全体でよりシームレスなエクスペリエンスを実現できます。
   a. インフラストラクチャセキュリティ

リソース

関連するベストプラクティス:

- SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する

関連ドキュメント:

- Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure

関連動画:

- Security best practices for the Amazon EC2 instance metadata service

データ保護

Questions

- SEC 7. どのようにデータを分類していますか?
- SEC 8. 保管中のデータはどのように保護するのですか。
- SEC 9. 転送中のデータはどのように保護するのですか。

SEC 7. どのようにデータを分類していますか?

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 データ分類スキームを理解する
- SEC07-BP02 データの機密性に基づいてデータ保護統制を適用する
- SEC07-BP03 識別および分類を自動化する
- SEC07-BP04 スケーラブルなデータのライフサイクル管理を定義する

SEC07-BP01 データ分類スキームを理解する

ワークロードが処理するデータの分類、その取り扱い要件、関連するビジネスプロセス、データの保存場所、データの所有者について理解します。 データの分類および取り扱いのスキームでは、ワークロードに適用される法的要件とコンプライアンス要件、必要なデータ統制を考慮する必要があります。データを理解することが、データ分類作業の第一歩です。

期待される成果: ワークロードに存在するデータの種類が十分に理解され、文書化されます。 適切な統制が効き、機密データがその分類に基づいて保護されます。 こうした統制では、データへのアクセス許可を持つユーザーとアクセスの目的、データの保存場所、データの暗号化ポリシーと暗号化キーの管理方法、データのライフサイクルとその保持要件、適切な破棄プロセス、実施されているバックアップと復旧のプロセス、アクセスの監査などの考慮事項が規定されます。

一般的なアンチパターン:

- データの機密レベルと取り扱い要件を定義する正式なデータ分類ポリシーが導入されていない。
- ワークロード内のデータの機密レベルが十分に理解されておらず、その情報がアーキテクチャや運用のドキュメントに記録されていない。
- データに対し、その機密性と要件に基づいた適切な統制を、データの分類と取り扱いに関するポリシーに従って適用できていない。
- データの分類と取り扱い要件に関するフィードバックを、ポリシーの所有者と共有できていない。

このベストプラクティスを活用するメリット: このプラクティスは、ワークロード内のデータの適切な処理に関するあいまいさを排除します。 組織内のデータの機密レベルと各レベルで求められる保護を定義した正式なポリシーを適用することで、法的規制やサイバーセキュリティに関する証明書、認証に準拠しやすくなります。 ワークロードの所有者は、機密データがどこに保存されているか、どのような保護統制が実施されているかを確実に把握できます。 こうした情報を文書に記録することで、新しいチームメンバーが職務の早い段階でそれらを理解し、統制を維持できるようになります。これらを実践すると、データの種類ごとに統制の適切なサイジングを行い、コストも削減できます。

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: 高

実装のガイダンス

ワークロードを設計する際、機密データを保護する方法を直感的に考慮しているかもしれません。 例えば、マルチテナントアプリケーションの場合、各テナントのデータを機密とみなし、任意のテナントが別のテナントのデータにはアクセスできないように保護するのは、直感的な対応です。 同様に、データを変更できるのは管理者だけで、他のユーザーは読み取り専用のアクセス権しかないか、一切アクセス権がないというように、直感的にアクセス制御を設計する場合もあるでしょう。

こうしたデータ機密レベルをデータ保護要件と併せて定義し、ポリシーに取り込むことで、ワークロードに含まれるデータを正式に特定できます。その上で、適切な統制が実施されているか、統制を監査できるか、データの取り扱いミスが判明した場合はどのような対応が適切かを判断できます。

ワークロード内の機密データが存在する場所を分類するには、利用可能な場合、リソースタグの使用を検討してください。 例えば、保護されたヘルス情報 (PHI) PHI のタグキー `Classification` とタグ値を持つタグ、およびタグキー `Sensitivity` とタグ値 `High` を持つ別のタグを適用できます。 その後、AWS Config などのサービスを使用して、これらのリソースの変更をモニタリングし、保護要件に準拠していない方法で変更された場合 (暗号化設定を変更するなど) に警告できます。 AWS Organizations の機能であるタグポリシーを使用して、タグキーと許容値の標準定義をキャプチャできます。タグのキーや値にプライベートデータや機密データを含めることはお勧めできません。

実装手順

1. 組織のデータ分類スキームと保護要件を理解します。
2. ワークロードによって処理される機密データの種類を特定します。
3. 機密データがポリシーに従ってワークロード内に保存され、保護されていることを確認します。 自動テストなどの手法を使用して、統制の有効性を監査します。
4. 可能であれば、リソースレベルとデータレベルのタグ付けを行い、監視やインシデント対応に役立つ可能性のある機密レベルやその他の運用メタデータをデータにタグ付けすることを検討してください。
   a. AWS Organizations タグポリシーを使用して、タグ付けの標準を適用できます。

リソース

関連する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

関連ドキュメント:

- データ分類に関するホワイトペーパー
- AWS リソースのタグ付けのベストプラクティス

関連する例:

- AWS Organizations タグポリシーの構文と例

関連ツール

- AWS タグエディタ

SEC07-BP02 データの機密性に基づいてデータ保護統制を適用する

データ保護統制を適用し、分類ポリシーで定義されている各クラスのデータに適切なレベルの統制を行います。 これにより、データの可用性と使用を確保しながら、機密データを不正アクセスや不正使用から守ることができます。

期待される成果: 組織内のデータのさまざまな機密性レベルを定義する分類ポリシーがあります。 機密レベルごとに、承認された保管や取り扱いのサービスと場所、それらに必要な構成に関する明確なガイドラインが公開されています。 必要とされる保護のレベルと付随するコストに応じて、レベルごとに統制を実施します。 データが許可されていない場所に存在する場合、許可されていない環境で処理されている場合、権限のないアクターによってアクセスされている場合、または関連サービスの構成がコンプライアンス違反になった場合に検知し、警告する監視体制が整っています。

一般的なアンチパターン:

- すべてのデータに同じレベルの保護統制が適用されている。機密性の低いデータに対してセキュリティ統制が行き過ぎたり、機密性の高いデータの保護が不十分になったりするおそれがあります。
- データ保護統制を定義する際に、セキュリティチーム、コンプライアンスチーム、ビジネスチームの関係者が関与していない。
- データ保護統制の導入と維持に関連する運用上のオーバーヘッドとコストを見落としている。
- 分類ポリシーとの整合性を維持するための、データ保護統制の定期的なレビューを実施していない。

このベストプラクティスを活用するメリット: コントロールをデータの分類レベルに合わせることで、必要に応じてより高いレベルのコントロールに投資できます。例えば、保護、監視、測定、修復、報告に関するリソースの増強が該当します。 統制を緩めた方が適切な場面では、従業員、顧客、または関係者にとってデータのアクセシビリティと完全性を向上させることができます。 このアプローチにより、組織はデータ保護要件を順守しながら、データを最大限柔軟に活用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

データの機密レベルに基づいてデータ保護統制を実施するには、いくつかの重要なステップが必要です。まず、ワークロードアーキテクチャ内のさまざまなデータ機密レベル (公開、内部、機密、制

限など) を特定し、該当データを保存して処理する場所を評価します。次に、機密レベルに基づいてデータの周囲の分離境界を定義します。サービスコントロールポリシー (SCP) を使用して、データ機密性レベルごとに許可されるサービスとアクションを制限し、データを異なる AWS アカウントに分けることをお勧めします。これにより、強固な分離境界を確立し、最小特権の原則を適用できます。

分離境界を定義したら、データの機密レベルに基づいて適切な保護統制を実装します。暗号化、アクセスコントロール、監査などの関連するコントロールを実装するには、保管中のデータを保護するベストプラクティスと転送中のデータを保護するベストプラクティスを参照してください。データの機密レベルを下げるには、トークン化や匿名化などの手法を検討してください。トークン化とトークン化解除の一元化システムにより、ビジネス全体に一貫したデータポリシーを簡単に適用できます。

実装した統制の有効性を継続的に監視し、テストします。組織のデータ環境や脅威が進化するにつれて、データ分類スキーム、リスク評価、保護統制を定期的に見直し、更新してください。実装されているデータ保護統制を、関連する業界規制、基準、法的要件に適応させてください。さらに、従業員がデータ分類スキームと、機密データの取り扱いと保護における各自の責任を把握できるように、セキュリティについて周知徹底し、トレーニングを実施してください。

実装手順

1. ワークロード内のデータの分類と機密レベルを特定します。
2. 各レベルの分離境界を定義し、実施戦略を決定します。
3. データ分類ポリシーで必要とされるアクセス、暗号化、監査、保持などについて定義した統制を評価します。
4. 必要に応じて、トークン化や匿名化の使用など、データの機密レベルを下げるオプションを評価してください。
5. 構成されたリソースのテストと監視を自動化し、統制を検証します。

リソース

関連するベストプラクティス:

- PERF03-BP01 データアクセスとストレージ要件に最適な専用データストアを使用する
- COST04-BP05 データ保持ポリシーを適用する

関連ドキュメント:

- データ分類に関するホワイトペーパー

- セキュリティ、アイデンティティ、コンプライアンスに関するベストプラクティス
- AWS KMS のベストプラクティス
- Encryption best practices and features for AWS services

関連する例:

- Building a serverless tokenization solution to mask sensitive data
- How to use tokenization to improve data security and reduce audit scope

関連ツール:

- AWS Key Management Service (AWS KMS)
- AWS CloudHSM
- AWS Organizations

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。手動での判断を自動化によって補強することで、人為的ミスやエクスポージャのリスクが軽減されます。

期待される成果: 分類と処理ポリシーに基づいて、適切なコントロールが設定されているかどうかを確認できます。自動化されたツールとサービスは、データの機密レベルを特定して分類するのに役立ちます。 また、自動化によって環境を継続的に監視して、データが不正な方法で保存または処理されているかどうかを検出して警告できるため、是正措置を迅速に講じることができます。

一般的なアンチパターン:

- データの識別と分類を手動プロセスでしか行っていないため、ミスが起こりやすく、時間もかかる。 特にデータ量が増えてくると、データ分類が非効率になり、一貫性を欠くことにつながりかねません。
- 組織全体のデータ資産を追跡および管理するメカニズムがない。
- 組織内でデータが移動したり進化したりする過程で、データを継続的に監視および分類する必要性を見落としている。

このベストプラクティスを活用するメリット: データ識別と分類を自動化すると、より一貫性のある正確なデータ保護コントロールの適用が可能になり、人為的ミスのリスクが軽減されます。 自動化

により、機密データへのアクセスと移動を可視化できるため、不正処理を検出して是正措置を講じることができます。

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

ワークロードの初期設計段階では、人間の判断でデータが分類されることが少なくありませんが、予防的統制として、テストデータの識別と分類を自動化するシステムの導入を検討してください。例えば、代表的なデータをスキャンして機密性を判断するためのツールやサービスを開発者に提供できます。 AWS では、データセットを Amazon S3 にアップロードし、Amazon Macie、Amazon Comprehend、または Amazon Comprehend Medical を使用してスキャンできます。 同様に、ユニットテストや統合テストの一環としてデータをスキャンして、予期しない場所に機密データが存在していないか検知することも検討してください。この段階で機密データに関する警告を行うことで、本番環境へのデプロイ前にセキュリティ保護のギャップを浮き彫りにすることができます。AWS Glue、Amazon SNS、Amazon CloudWatch での機密データ検出などの機能を使用して、PII を検出し、修正アクションを実行することもできます。どの自動化ツールやサービスでも、機密データがどのように定義されているかを理解し、他の人間によるソリューションや自動化されたソリューションで適宜補強することで、ギャップを埋めることができます。

発見的統制として、環境を継続的に監視し、機密データがコンプライアンスに違反する方法で保存されていないかを検出してください。 これにより、機密データが適切な匿名化や伏字化を行わないまま口グファイルに出力されたり、データ分析環境にコピーされたりする事態を検知できます。 Amazon S3 に保存されているデータは、Amazon Macie を使用して、機密データがないか継続的に監視できます。

実装手順

1. 環境の初期スキャンを実行して、自動で識別と分類を行います。
    a. データを最初にフルスキャンすることで、環境内のどこに機密データが存在するかを包括的に把握できます。フルスキャンが初期段階では不要な場合や、コスト面で事前に完了するのが難しい場合は、データサンプリング手法で成果が得られるか評価してください。例えば、S3 バケット全体で広範にわたって機密データを自動検出するように Amazon Macie を設定できます。 この機能では、サンプリング手法を使用して、機密データの所在の事前分析をコスト効率よく実行します。 その後、機密データの検出ジョブを用いて、S3 バケットの詳細な分析を実行できます。他のデータストアも S3 にエクスポートして Macie でスキャンできます。
2. 環境の継続的なスキャンを設定します。

   a. Macie の自動機密データ検出機能を使用して、環境を継続的にスキャンできます。 機密データの保存が許可されている既知の S3 バケットは、Macie の許可リストを使用して除外できます。

3. 識別と分類をビルドとテストのプロセスに組み込みます。

   a. ワークロードの開発中に、開発者がデータの機密性をスキャンするために使用できるツールを特定します。 これらのツールを統合テストの一環として使用して、予期しない場所に機密データが存在する場合に警告し、その後のデプロイを防ぎます。

4. 許可されていない場所で機密データが見つかったときに対処するためのシステムまたはランブックを実装します。

リソース

関連ドキュメント:

- AWS Glue: Detect and process sensitive data
- Using managed data identifiers in Amazon SNS
- Amazon CloudWatch Logs: 機密性の高いログデータをマスキングで保護する

関連する例:

- Enabling data classification for Amazon RDS database with Macie
- Detecting sensitive data in DynamoDB with Macie

関連ツール:

- Amazon Macie
- Amazon Comprehend
- Amazon Comprehend Medical
- AWS Glue

SEC07-BP04 スケーラブルなデータのライフサイクル管理を定義する

データのライフサイクルの要件を、関連するさまざまなレベルのデータ分類や取り扱いに応じて把握してください。 例えば、データを初めて環境に取り込んだときの取り扱い方法や、データの変換方

法、破棄のルールなどが該当します。保存期間、アクセス、監査、出所追跡などの要素を考慮してください。

期待される成果: 取り込みに近いポイントおよび時点でデータを分類します。データの分類にマスキングやトークン化、機密情報を保護するその他の対策が必要な場合は、そうした作業をできるだけ取り込みポイントおよび時点に近いポイントおよび時点で行います。

保管しておくことが適切でなくなったデータは、その分類に基づいて、ポリシーに従って削除します。

一般的なアンチパターン:

- データのライフサイクル管理に画一的なアプローチを実装し、さまざまな機密度やアクセス要件が考慮されていない。
- 利用可能なデータとバックアップされているデータの両方ではなく、いずれか一方の視点でのみライフサイクル管理を検討している。
- データがその価値や出所を確認することなく、ワークロードに入力された時点で有効だと仮定されている。
- データのバックアップや保護を行う代わりに、データの耐久性に頼り切っている。
- データが有用でなくなり、必要な保持期間が過ぎても保持し続けている。

このベストプラクティスを活用するメリット: 明確に定義されスケーラブルなデータライフサイクル管理戦略は、適切なコントロールを維持しながら、規制コンプライアンスの維持、データセキュリティの向上、ストレージコストの最適化、効率的なデータアクセスと共有を可能にします。

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

ワークロード内のデータは多くの場合、動的です。 ワークロード環境に入ってくるときの形式は、ビジネスロジック、レポート、分析、機械学習で保存または使用されるときの形式とは異なる場合があります。 さらに、データの価値は時間とともに変化する可能性があります。一部のデータは時間に依存する性質があり、古くなるにつれて価値を失います。 データのこうした変更が、データ分類スキームや関連する統制の下での評価にどのように影響するかを検討してください。 可能な場合は、Amazon S3 ライフサイクルポリシーや Amazon Data Lifecycle Manager などの自動ライフサイクルメカニズムを使用して、データ保持、アーカイブ、有効期限のプロセスを設定します。

使用可能なデータと、バックアップとして保存されているデータは区別します。 AWS Backup を使用して、AWS サービス間のデータのバックアップを自動化することを検討してください。 Amazon

EBS スナップショットは、EBS ボリュームをコピーし、ライフサイクル、データ保護、保護メカニズムへのアクセスなどの S3 機能を使用して保存する方法を提供します。これらの機能のうち 2 つは S3 Object Lock と AWS Backup ボールトロックです。これにより、バックアップのセキュリティと制御を強化できます。バックアップに関して、職務とアクセス権を明確に分離して管理します。バックアップはアカウントレベルで分離し、イベントの発生時に影響を受ける環境から分離した状態を維持できるようにします。

ライフサイクル管理のもう 1 つの要素は、データ出所追跡と呼ばれるワークロードの進行状況に応じたデータの履歴を記録することです。これにより、データがどこから来たのか、変換されている場合はどのような変換か、どの所有者やプロセスがいつそれらの変更を行ったのかを確実に把握できます。 こうした履歴は、潜在的なセキュリティイベントが発生した際の問題のトラブルシューティングや調査に役立ちます。 例えば、Amazon DynamoDB テーブルの変換に関するメタデータをログに記録できます。 データレイク内では、変換後のデータのコピーをデータパイプラインのステージごとに異なる S3 バケットに保存できます。スキーマとタイムスタンプ情報を AWS Glue Data Catalog に保存します。 どのソリューションを使用する場合でも、エンドユーザーの要件を考慮して、データの出所に関するレポートに必要な適切なツールを判断してください。 そうすることで、出所を最も適切に追跡する方法を決定できます。

実装手順

1. ワークロードのデータタイプ、機密レベル、アクセス要件を分析してデータを分類し、適切なライフサイクル管理戦略を定義します。
2. 法律、規制、組織の要件に沿ったデータ保持ポリシーと自動破棄プロセスを設計し、実装します。
3. ワークロードの要件や規制の変化に応じて、データライフサイクル管理の戦略、統制、ポリシーを継続的に監視、監査、調整するためのプロセスと自動化を確立します。

リソース

関連するベストプラクティス:

- COST04-BP05 データ保持ポリシーを適用する
- 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

関連ドキュメント:

- データ分類に関するホワイトペーパー

- AWS Blueprint for Ransomware Defense
- DevOps Guidance: Improve traceability with data provenance tracking

関連する例:

- How to protect sensitive data for its entire lifecycle in AWS
- Build data lineage for data lakes using AWS Glue, Amazon Neptune, and Spline

関連ツール:

- AWS Backup
- Amazon Data Lifecycle Manager
- AWS Identity and Access Management Access Analyzer

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することで保管中のデータを保護し、不正アクセスや誤操作のリスクを軽減します。

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 保管中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-BP01 安全なキー管理を実装する

安全なキー管理には、ワークロード用に保管中のデータを保護するために必要な、キーマテリアルの保管、ローテーション、アクセス制御、監視が含まれます。

期待される成果: スケーラブルで反復可能な、自動化されたキー管理メカニズム。このメカニズムは、キーマテリアルへの最小特権アクセス権を強制する能力を提供し、キーの可用性、機密性、完全性の適切なバランスを実現できるものでなければなりません。キーへのアクセスは監視され、キーマテリアルは自動化されたプロセスでローテーションされる必要があります。キーの内容は、決して人的 ID にアクセス可能なものであってはなりません。

一般的なアンチパターン:

- 暗号化されていないキーマテリアルに人間がアクセスする。
- カスタム暗号化アルゴリズムを作成する。
- キーマテリアルへのアクセス許可の範囲が広すぎる。

このベストプラクティスを活用するメリット: ワークロード用の安全なキー管理メカニズムを確立することで、不正アクセスからコンテンツを保護することができます。さらに、データの暗号化を要求する規制要件の対象となる場合があります。効果的なキー管理ソリューションがあれば、それらの規制に合わせた技術的メカニズムを提供して、キーマテリアルを保護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

多くの規制要件やベストプラクティスには、基本的なセキュリティ制御として保管中のデータの暗号化が含まれています。この制御に準拠するには、ワークロードに、保管中のデータの暗号化に使用されるキーマテリアルを安全に保存および管理するメカニズムが必要です。

AWS は AWS Key Management Service (AWS KMS) を使用して AWS KMS キー用の高い耐久性と安全性を備えた冗長ストレージを提供します。多くの AWS サービスは AWS KMS と統合して、データ暗号化をサポートしています。AWS KMS は、FIPS 140-2 レベル 3 検証済みの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します。AWS KMS キーをプレーンテキストでエクスポートするメカニズムはありません。

マルチアカウント戦略を使用してワークロードをデプロイする場合、キーを使用するワークロードと同じアカウントに AWS KMS キーを保持することがベストプラクティスです。この分散型モデルでは、AWS KMS キーの管理責任はアプリケーションチームにあります。他のユースケースでは、組織は AWS KMS キーを一元管理されたアカウントに保存することもできます。この一元化された構造では、ワークロードアカウントが統合アカウントに保存されているキーにアクセスするために必要なクロスアカウントアクセスを可能にする追加のポリシーが必要ですが、単一のキーを複数の AWS アカウントで共有するユースケースにより適している可能性があります。

キーマテリアルを保管する場所にかかわらず、キーへのアクセスはキーポリシーおよび IAM ポリシーで厳重に管理する必要があります。キーポリシーは、AWS KMS キーへのアクセスを制御する主な方法です。さらに、AWS KMS キーの付与によって、ユーザーに代わってデータを暗号化および復号する AWS のサービスへのアクセスを提供できます。AWS KMS キーへのアクセスを制御するベストプラクティスを十分に確認してください。

暗号化キーの使用状況を監視して、異常なアクセスパターンを検出するのがベストプラクティスです。AWS マネージドキーと AWS KMS に保存されているカスタマーマネージドキーを使用して実行される操作は AWS CloudTrail でログインできるため、定期的に確認する必要があります。キー破壊イベントの監視には特に注意する必要があります。キーマテリアルの偶発的または悪意のある破壊を防ぐため、キー破壊イベントによってキーマテリアルがすぐに削除されることはありません。AWS KMS の削除には待機時間が設けられており、デフォルトで 30 日間です。管理者は削除アクションを確認し、必要に応じてリクエストをロールバックする時間を確保できます。

AWS の多くのサービスはわかりやすい方法で AWS KMS を使用します。唯一必要なのは、AWS マネージドキーとカスタマーマネージドキーのどちらを使用するかを決定することです。ワークロードでデータを暗号化または復号するために直接 AWS KMS を使用する場合、データを保護するためエンベロープ暗号化を使用するのがベストプラクティスです。AWS Encryption SDK は、アプリケーションにクライアント側の暗号化プリミティブを提供し、エンベロープ暗号化を実装して AWS KMS と統合することができます。

実装手順

1. キーに適したキー管理オプション (AWS 管理またはカスタマー管理) を決定します。
    - 使いやすさを考慮して、AWS はほとんどのサービスにおいて AWS 所有キーと AWS マネージドキーを提供しています。これにより、キーマテリアルやキーポリシーを管理しなくても保管時の暗号化が可能になります。
    - カスタマーマネージドキーを使用する場合は、俊敏性、セキュリティ、データ主権、可用性の最適なバランスを実現するデフォルトのキーストアを検討してください。他のユースケースでは、AWS CloudHSM または外部キーストアによりカスタムキーストアの使用が必要な場合もあります。
2. ワークロードに使用しているサービスのリストを確認して、AWS KMS がサービスとどのように統合されているかを理解します。例えば、EC2 インスタンスは暗号化された EBS ボリュームを使用できます。これにより、そのボリュームから作成された Amazon EBS スナップショットもカスタマーマネージドキーを使用して暗号化されていることを確認し、暗号化されていないスナップショットデータが誤って開示されるのを防ぐことができます。
    - AWS のサービスで AWS KMS を使用する方法
    - AWS のサービスが提供する暗号化オプションの詳細については、ユーザーガイドの「保管時の暗号化」トピックまたはサービスのデベロッパーガイドを参照してください。
3. AWS KMS の実装: AWS KMS を使用すると、キーの作成と管理が簡単になり、幅広い AWS のサービスやアプリケーションでの暗号化の使用を制御できます。
    - 開始方法: AWS Key Management Service (AWS KMS)

- AWS KMS キーへのアクセスを制御するベストプラクティスを確認してください。

4. AWS Encryption SDK の検討: アプリケーションがクライアント側でデータを暗号化する必要がある場合は、AWS KMS 統合済みの AWS Encryption SDK を使用してください。
   - AWS Encryption SDK
5. IAM Access Analyzer を有効化して、過度に広範な AWS KMS キーポリシーがないかどうかを自動的に確認し、通知を受け取るようにします。
6. Security Hub を有効化して、キーポリシーの設定ミス、削除予定のキー、自動ローテーションが有効になっていないキーがある場合に通知を受け取るようにします。
7. AWS KMS キーに適したログ記録レベルを決定します。AWS KMS への呼び出し (読み取り専用イベントを含む) はログに記録されるため、CloudTrail に関連する AWS KMS ログが膨大になる可能性があります。
   - 組織によっては、AWS KMS のログ記録アクティビティを別の証跡に分けた方がよい場合があります。詳細については、「AWS KMS デベロッパーガイド」の「CloudTrail で AWS KMS API コールをログに記録する」セクションを参照してください。

リソース

関連ドキュメント:

- AWS Key Management Service
- AWS cryptographic services and tools
- 暗号化によるデータの保護
- エンベロープ暗号化
- Digital sovereignty pledge
- Demystifying AWS KMS key operations, bring your own key, custom key store, and ciphertext portability
- AWS Key Management Service cryptographic details

関連動画:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS
- AWS data protection: Using locks, keys, signatures, and certificates

関連する例:

- AWS KMS での高度なアクセスコントロールメカニズムの実装

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は、保管時にデフォルトで暗号化する必要があります。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を活用し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、1 つまたは非常に少数のデータ暗号化キーを使用する場合、過度に許容されるアクセスから保護するのに役立ちます (「SEC07-BP01 データ分類スキームを理解する」を参照)。

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、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、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、デフォルトの暗号化を設定して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、Amazon S3 バケットなどで暗号化の使用状況を自動的に確認できます。

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用してデータを保護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、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 に対して保管中の暗号化を適用する: 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 マシンイメージ (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有効化して、保管中の Amazon RDS データベースクラスターとスナップショットに対して暗号化を設定します。

  データの分類ごとに適切なプリンシパルへのアクセスを制限するポリシーが適用される AWS KMS キーを作成して設定する: 例えば、本番データを暗号化するための AWS KMS キーを 1 つ作成し、開発データまたはテストデータを暗号化するための別のキーを作成します。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 AWS サービスで暗号化を設定する: 使用する他の AWS サービスについては、そのサービスのセキュリティドキュメントを参照して、サービスの暗号化オプションを確認してください。

リソース

関連ドキュメント:

- AWS Crypto Tools ドキュメント
- AWS Encryption SDK
- AWS KMS Cryptographic Details Whitepaper
- AWS Key Management Service
- AWS cryptographic services and tools
- Amazon EBS Encryption
- Default encryption for Amazon EBS volumes
- Amazon RDS リソースの暗号化
- Amazon S3 バケットのデフォルト暗号化を有効にする方法
- 暗号化によるデータの保護

関連動画:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC08-BP03 保管中のデータの保護を自動化する

自動化を利用して、保管中のデータの統制を検証し、適用します。 自動スキャンを使用してデータストレージソリューションの設定ミスを検出し、可能な場合はプログラムによる自動対応で修復を行います。 CI/CD プロセスに自動化を組み込んで、データストレージの設定ミスを検知し、本番環境に適用されないよう未然に防ぎます。

期待される成果: 自動システムは、コントロールの設定ミス、不正アクセス、予期しない使用方法がないか、データストレージの場所をスキャンして監視します。 データストレージの設定ミスが検出されると、自動修復が開始します。 自動化されたプロセスによってデータのバックアップが作成され、イミュータブル (変更不可能) なコピーがバックアップ元の環境の外部に保管されます。

一般的なアンチパターン:

- デフォルト設定で暗号化を有効にするオプションがサポートされているのに、そうしたオプションを検討しない。
- バックアップと復旧の自動化戦略を策定する際に、運用上のイベントだけでなくセキュリティイベントも考慮していない。
- ストレージサービスに対してパブリックアクセス設定を強制しない。
- 保管中のデータを保護するための統制の監視や監査をしていない。

このベストプラクティスを活用するメリット: 自動化は、データストレージの場所の設定ミスのリスクを防ぐのに役立ちます。設定ミスが本番環境に入り込まないように阻止できます。このベストプラクティスは、設定ミスが起きた場合の検出と修正にも役立ちます。

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: 中

実装のガイダンス

保管中のデータを保護するためのあらゆる取り組みにおいて、自動化は重要です。「SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する」では、AWS CloudFormation などの infrastructure as code (IaC) テンプレートを使用してリソース設定をキャプチャする方法を説明します。 これらのテンプレートはバージョン管理システムにコミットされ、CI/CD パイプラインを通じて AWS でリソースをデプロイするために使用されます。 データストレージソリューションの設定 (Amazon S3 バケットの暗号化設定など) を自動化する場合にも、これらの手法が同様に適用されます。

IaC テンプレートで定義された設定にミスがないか、AWS CloudFormation Guard のルールを CI/CD パイプライン内で使用してチェックできます。 AWS Config を使用して、CloudFormation やその他の IaC ツールでまだ利用できない設定をモニタリングし、設定ミスを確認できます。「SEC04-BP04 非準拠リソースの修復を開始する」で説明されているように、設定ミスに対して Config が生成するアラートは自動的に修正できます。

アクセス許可管理の戦略に自動化を組み込むことも、自動データ保護の要素として不可欠です。「SEC03-BP02 最小特権のアクセスを付与する」および「SEC03-BP04 アクセス許可を継続的に削減する」では、最小特権アクセスポリシーの設定について説明します。AWS Identity and

Access Management Access Analyzer によってポリシーは継続的に監視され、権限を削減できる場合はレポートが出力されます。 アクセス許可のモニタリングを自動化するだけでなく、Amazon GuardDuty を設定して、EBS ボリューム (EC2 インスタンスを介して)、S3 バケット、およびサポートされるAmazon Relational Database Service データベースで異常なデータアクセス動作を監視できます。

許可されていない場所に機密データが保存されていることを検知する場合にも、自動化が活躍します。「SEC07-BP03 識別および分類を自動化する」では、Amazon Macie を使用して S3 バケットで予期しない機密データをモニタリングし、自動応答を開始するアラートを生成する方法について説明します。

「REL09 バックアップデータ」のプラクティスに従って、自動データバックアップおよびリカバリ戦略を開発します。データのバックアップと復旧は、運用上のイベントと同様、セキュリティイベントから復旧するために重要です。

実装手順

1. データストレージの設定を IaC テンプレートに取り込みます。 CI/CD パイプラインで自動チェックを行い、設定ミスを検出します。
   a. IaC テンプレートには CloudFormation を、テンプレートの設定ミスをチェックするには CloudFormation Guard を使用できます。
   b. AWS Config を使用して、プロアクティブ評価モードでルールを実行します。この設定を使用して、リソースの作成前に CI/CD パイプラインのステップとしてリソースのコンプライアンスを確認します。
2. データストレージの設定ミスがないか、リソースを監視します。
   a. AWS Config を設定して、データストレージリソースの制御設定の変更をモニタリングし、設定ミスの検出時に修復アクションを呼び出すアラートを生成するようにします。
   b. 自動修復の詳細については、「SEC04-BP04 非準拠リソースの修復を開始する」を参照してください。
3. データアクセス許可を自動化により継続的に監視し、削減します。
   a. IAM Access Analyzer を継続的に実行して、アクセス許可を削減できる場合にアラートを生成できます。
4. 異常なデータアクセス動作を監視し、警告します。
   a. GuardDuty は、EBS ボリューム、S3 バケット、RDS データベースなどのデータストレージリソースについて、既知の脅威シグネチャとベースラインアクセス動作からの逸脱を監視します。

5. 機密データが予期しない場所に保存されていないか監視し、警告します。
    a. Amazon Macie を使用して、S3 バケットの機密データを継続的にスキャンします。
6. 暗号化した安全なデータバックアップの作成を自動化します。
    a. AWS Backup は、AWS の各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。 Elastic Disaster Recovery を使用すると、サーバーワークロード全体をコピーし、秒単位の目標復旧時点 (RPO) を提供する継続的なデータ保護を実現できます。 両方のサービスを連携するよう設定し、データバックアップの作成とフェイルオーバー先へのコピーを自動化できます。 そうしておくことで、運用上のイベントやセキュリティイベントの影響を受けた場合でも、データが常時利用可能になります。

リソース

関連するベストプラクティス:

- SEC01-BP06 標準的なセキュリティ統制のデプロイを自動化する
- SEC03-BP02 最小特権のアクセスを付与する
- SEC03-BP04 アクセス許可を継続的に削減する
- SEC04-BP04 非準拠リソースの修復を開始する
- SEC07-BP03 識別および分類を自動化する
- 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

関連ドキュメント:

- AWS 規範ガイダンス: Automatically encrypt existing and new Amazon EBS volumes
- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework (CSF)

関連する例:

- How to use AWS Config proactive rules and AWS CloudFormation Hooks to prevent creation of noncompliant cloud resources
- Automate and centrally manage data protection for Amazon S3 with AWS Backup
- AWS re:Invent 2023 - Implement proactive data protection using Amazon EBS snapshots
- AWS re:Invent 2022 - Build and automate for resilience with modern data protection

関連ツール:

- AWS CloudFormation Guard
- AWS CloudFormation Guard Rules Registry
- IAM Access Analyzer
- Amazon Macie
- AWS Backup
- Elastic Disaster Recovery

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: そのデータについて知る必要がある、許可されたユーザーのみがデータに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 (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を削減する計画を策定します。

Amazon S3 Glacier のボールトロックと Amazon S3 Object Lock は、Amazon S3 のオブジェクトに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によってロックされると、そのロックの有効期限が切れるまでルートユーザーであっても変更できません。

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、最小権限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それぞれ異なる AWS アカウントを使用し、それらのアカウントは AWS Organizations を使って管理します。
- 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レビューします。
- 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レビューします。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 を使用する: IAM Access Analyzer は Amazon S3 バケットを分析し、S3 ポリシーが外部エンティティにアクセスを許可したときに検出結果を生成します。
- 必要に応じて、Amazon S3 のバージョニングと Object Lock を使用します。
- Amazon S3 インベントリを使用する: 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およびレポートには Amazon S3 インベントリを使用します。
- Amazon EBS と AMI 共有アクセス許可をレビューする: 共有アクセス許可を使用すると、ワークロード外の AWS アカウントとイメージおよびボリュームを共有することができます。
- AWS Resource Access Manager の共有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 を使用すると、AWS Network Firewall ポリシー、Amazon Route 53 Resolver ルール、サブネットなど、Amazon VPC 内のリソースを共有することができます。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する

関連ドキュメント:

- AWS KMS Cryptographic Details Whitepaper
- Amazon S3 リソースへのアクセス許可の管理
- 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
- バージョニングの使用
- S3 Object Lock を使ってオブジェクトをロックする
- Sharing an Amazon EBS Snapshot
- 共有 AMI
- Amazon S3 で単一ページのアプリケーションをホストする

関連動画:

- Securing Your Block Storage on AWS

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することで、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 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

Transport Layer Security (TLS) 証明書は、ネットワーク通信を保護し、インターネットやプライベートネットワーク上のウェブサイト、リソース、ワークロードの ID を確立するために使用されます。

期待される成果: 公開鍵基盤 (PKI) で証明書をプロビジョニング、デプロイ、保存、更新できる、安全な証明書管理システム。安全な鍵と証明書の管理メカニズムは、証明書のプライベートキーの内容が漏洩するのを防ぎ、自動的に証明書の定期更新を行います。また、他のサービスと統合して、ワークロード内のマシンリソースに安全なネットワーク通信と ID を提供します。キーの内容は、決して人的 ID にアクセス可能なものであってはなりません。

一般的なアンチパターン:

- 証明書のデプロイまたは更新プロセス中に手動で手順を実行する。
- プライベート認証機関 (CA) を設計する際、CA 階層に十分な注意を払わない。
- 公共リソースに自己署名証明書を使用する。

このベストプラクティスを活用するメリット:

- 自動デプロイと自動更新により証明書管理を簡素化する
- TLS 証明書を使用して転送中のデータの暗号化を奨励する
- 認証機関による証明書アクションのセキュリティと可監査性を向上させる
- CA 階層のさまざまなレイヤーにおける管理業務を整理する

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: 高

実装のガイダンス

最新のワークロードでは、TLS などの PKI プロトコルを使用して暗号化されたネットワーク通信が広く利用されています。PKI 証明書の管理は複雑になる場合がありますが、証明書のプロビジョニング、デプロイ、更新を自動化することで、証明書管理に伴う手間を軽減できます。

AWS には、汎用 PKI 証明書を管理するための 2 つのサービス、AWS Certificate Manager と AWS Private Certificate Authority (AWS Private CA) があります。ACM は、証明書をプロビジョニング、管理、デプロイして、パブリックとプライベート両方の AWS ワークロードで使用できるようにするための主要なサービスです。ACM は AWS Private CA を使用して証明書を発行し、他の多くの AWS マネージドサービスと連携してワークロード用の安全な TLS 証明書を提供します。

AWS Private CA では、独自のルート認証機関または下位認証機関を確立し、API を通じて TLS 証明書を発行できます。こうした種類の証明書は、TLS 接続のクライアント側で信頼チェーンを制御し管理するシナリオで使用できます。TLS ユースケースに加えて、AWS Private CA は、Kubernetes ポッドへの証明書の発行、Matter デバイス製品認証、コード署名、カスタムテンプレートを使用したその他のユースケースにも使用できます。また、IAM Roles Anywhere を使用することで、プライベート CA によって署名された X.509 証明書が発行されたオンプレミスのワークロードに、一時的な IAM 認証情報を提供することもできます。

ACM と AWS Private CA に加えて、AWS IoT Core は、PKI 証明書のプロビジョニング、管理、IoT デバイスへのデプロイに特化したサポートを提供しています。AWS IoT Core は、公開鍵のインフラストラクチャに IoT デバイスを大規模にオンボーディングするための特殊な仕組みを備えています。

プライベート CA 階層を確立する際の考慮事項

プライベート CA を確立する必要がある場合、特別な注意を払って事前に CA 階層を適切に設計しておくことが重要です。プライベート CA 階層を作成する場合は、CA 階層の各レベルを個別の AWS アカウントにデプロイすることがベストプラクティスです。この意図的な手順により、CA 階層内の各レベルへの外部からのアクセスが減り、CloudTrail ログデータ内の異常をより簡単に発見できるようになります。また、いずれかのアカウントに不正アクセスがあった場合、アクセス範囲と影響が小さくなります。ルート CA はそれぞれ別のアカウントに保存し、1 件以上の中間 CA 証明書の発行にのみ使用すべきです。

次に、ルート CA のアカウントとは別のアカウントに 1 つ以上の中間 CA を作成し、エンドユーザー、デバイス、または他のワークロードに証明書を発行します。最後に、ルート CA から中間 CA に証明書を発行します。これにより、エンドユーザーまたはデバイスに証明書が発行されます。回復力の計画、クロスリージョンレプリケーション、組織全体での CA の共有など、CA デプロイの計画と CA 階層の設計の詳細については、「Planning your AWS Private CA deployment」を参照してください。

実装手順

1. ユースケースに必要となる適切な AWS サービスを判断します。
    - 多くのユースケースでは、AWS Certificate Manager を使用して既存の AWS パブリックキーインフラストラクチャを活用することができます。ACM は、ウェブサーバー、ロードバランサー、一般的に信頼されている証明書のその他の用途向けに TLS 証明書をデプロイする際に使用できます。
    - 独自のプライベート認証機関の階層を設定する必要がある場合や、エクスポート可能な証明書へのアクセスが必要な場合は、AWS Private CA の使用を検討します。それにより、ACM を使

用して、AWS Private CA を使ったさまざまな種類のエンドエンティティ証明書を発行できます。

- 組み込み型のモノのインターネット (IoT) デバイス向けに証明書を大規模にプロビジョニングする必要があるユースケースについては、AWS IoT Core の使用を検討します。

2. 可能な限り、証明書の自動更新を実装してください。
   - ACM が発行した証明書に、ACM マネージド型更新を、統合された AWS のマネージドサービスと併せて使用します。
3. ログ記録と監査証跡の設定:
   - CloudTrail logs を有効にして、認証機関を持つアカウントへのアクセスを追跡します。CloudTrail でログファイルの整合性検証を設定し、ログデータの信頼性を検証することを検討します。
   - プライベート CA が発行または取り消しした証明書を一覧表示する監査レポートを、定期的に生成し、レビューします。これらのレポートは S3 バケットにエクスポートできます。
   - プライベート CA をデプロイするときは、証明書失効リスト (CRL) を保存する S3 バケットも確立する必要があります。ワークロードの要件に基づいてこの S3 バケットを設定する方法については、「Planning a certificate revocation list (CRL)」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
- SEC08-BP01 安全なキー管理を実装する
- SEC09-BP03 ネットワーク通信を認証する

関連ドキュメント:

- How to host and manage an entire private certificate infrastructure in AWS
- How to secure an enterprise scale ACM Private CA hierarchy for automotive and manufacturing
- Private CA best practices
- How to use AWS RAM to share your ACM Private CA cross-account

関連動画:

- Activating AWS Certificate Manager Private CA (workshop)

関連する例:

- Private CA workshop
- IOT Device Management Workshop (including device provisioning)

関連ツール:

- Plugin to Kubernetes cert-manager to use AWS Private CA

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転送中も、データの機密性を保持できます。

期待される成果: 伝送中のデータはすべて、安全な TLS プロトコルと暗号スイートを使用して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を活用しない場合のリスクレベル: 高

実装のガイダンス

AWS のサービスには、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 内または Application Load Balancer の 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 VPC に VPN で接続することで、トラフィックの暗号化を円滑にすることができます。クライアントで AWS API の呼び出しに TLS 1.2 以降のバージョンが使われていることを確認します。これ以前のバージョンは AWS によって 2023 年 6 月に廃止されるためです。AWS は TLS 1.3 の使用を推奨しています。特別な要件がある場合は、AWS Marketplace でサードパーティーのソリューションを入手できます。

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。例えば、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: Amazon CloudFront を使用して HTTPS を設定し、自社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ます。
- ロードバランサーで安全なプロトコルを設定する: リスナーに接続するクライアントがサポートしている暗号スイートのなかで、もっとも堅牢な暗号スイートを提供しているセキュリティポリシーを選びます。Application Load Balancer 用の HTTPS リスナーを作成する
- Amazon Redshift で安全なプロトコルを設定する: Secure Socket Layer (SSL) または Transport Layer Security (TLS) 接続を必須とするように、クラスターを設定します。
- 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 バケットへのアップロード時の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、データへの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 を使用すると、AWS サービスで使用するパブリック TLS 証明書のプロビジョニング、管理、デプロイが行えます。

- プライベート PKI のニーズには AWS Private Certificate Authority の使用を検討する: AWS Private CA を使用すると、プライベート認証局 (CA) 階層を作成して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。こちらは暗号化された TLS チャネルの作成に使用できます。

リソース

関連ドキュメント:

- CloudFront で HTTPS を使用する
- 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
- Tutorial: Configure SSL/TLS on Amazon Linux 2
- SSL/TLS を使用した DB インスタンスへの接続の暗号化
- 接続のセキュリティオプションを設定する

SEC09-BP03 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。

サービス間、アプリケーション間、またはユーザーへの通信には常に、安全で認証済みのネットワークプロトコルを使用するようにワークロードを設計してください。認証と認可をサポートするネットワークプロトコルを使用すると、ネットワークフローをより強力に制御し、不正アクセスの影響を軽減できます。

期待される成果: ワークロードで、サービス間のデータプレーンとコントロールプレーンのトラフィックフローが明確に定義されています。技術上可能な場合は必ず、認証および暗号化されたネットワークプロトコルをトラフィックフローが使用する。

一般的なアンチパターン:

- ワークロード内のトラフィックフローが暗号化されていない、または認証されていない。
- 複数のユーザーやエンティティで認証情報を再利用している。
- アクセス制御のメカニズムとしてネットワーク統制にばかり依存している。
- 業界標準の認証メカニズムに頼る代わりに、カスタムの認証メカニズムを作成する。
- VPC 内のサービスコンポーネントや他のリソース間のトラフィックフローが必要以上に許可されている。

このベストプラクティスを活用するメリット:

- 不正アクセスによる影響が及ぶ範囲をワークロードの一部に制限します。
- アシュアランスのレベルを上げ、認証済みのエンティティだけがアクションを実行するように徹底します。
- 導入予定のデータ転送インターフェイスを明確に定義し、実際に導入して、サービスの分離を強化します。
- リクエストのアトリビューションと、明確に定義された通信インターフェイスにより、モニタリング、ログ記録、インシデント対応を強化します。
- ネットワーク統制に認証と認可の制御を組み合わせることで、ワークロードに多層防御を提供します。

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: 低

実装のガイダンス

ワークロードのネットワークトラフィックのパターンは、次の 2 つのカテゴリに分類できます。

- East-West トラフィックは、特定のワークロードを構成しているサービス間のトラフィックフローを表します。
- North-South トラフィックはワークロードとコンシューマー間のトラフィックフローを表します。

一般的には North-South トラフィックを暗号化し、認証済みプロトコルを用いて East-West トラフィックを保護する例はあまり見られません。最近のセキュリティ対策では、ネットワークの設計だけで、2 つのエンティティ間に信頼関係があるとは想定しないというのが通例となっています。2 つのサービスが共通のネットワーク境界内に存在する場合でも、それらのサービス間の通信を暗号化し、認証と認可を行うことがベストプラクティスです。

例えば、AWS サービス API は、リクエストの発信側のネットワークに関係なく、AWS Signature Version 4 (SigV4) 署名プロトコルを使用して発信者を認証します。この認証により、AWS API はアクションをリクエストした ID を検証し、その ID をポリシーと組み合わせて、アクションを許可するかどうかを判断する認可決定を行うことができます。

Amazon VPC Lattice や Amazon API Gateway などのサービスでは、同じ SigV4 署名プロトコルを使用して、独自のワークロードの East-West トラフィックに認証と認可を追加できます。SigV4 ベースの認証と認可が要求されるサービスに AWS 環境外のリソースから通信する必要がある場合は、非 AWS リソースに AWS Identity and Access Management (IAM) Roles Anywhere を使用するこ

とで、一時的な AWS 認証情報を取得できます。これらの認証情報は、SigV4 を使用してサービスへのリクエストに署名して、アクセスの認可を受けるために使用できます。

East-West トラフィックを認証するメカニズムとしては、TLS 相互認証 (mTLS) も一般的です。モノのインターネット (IoT)、ビジネス間アプリケーション、マイクロサービスの多くは、mTLS を採用しています。TLS 通信のクライアント側とサーバー側の両方が X.509 証明書を使用して、双方のアイデンティティを認証し合います。これらの証明書は AWS Private Certificate Authority (AWS Private CA) で発行できます。Amazon API Gateway や AWS App Mesh などのサービスを使用すると、ワークロード間またはワークロード内の通信で mTLS 認証を行うことができます。mTLS は TLS 通信の両側に認証情報を提供しますが、認証のメカニズムは提供しません。

最後に、OAuth 2.0 と OpenID Connect (OIDC) の 2 つのプロトコルは、ユーザーがサービスへのアクセスを制御する際によく利用されていますが、最近はサービス間のトラフィックにもよく利用されています。API Gateway の JSON ウェブトークン (JWT) オーソライザーを使用すると、OIDC または OAuth 2.0 の ID プロバイダーが発行した JWT を使用して、API ルートへのアクセスをワークロードで制限することができます。OAuth2 のスコープを基本的な承認決定のソースとして使用できますが、依然として承認チェックをアプリケーション層に実装する必要があります。OAuth2 スコープ単体で複雑な承認ニーズに対応することはできません。

実装手順

- ワークロードのネットワークフローを定義し文書化する: 多層防御戦略を実装するには、まずワークロードのトラフィックフローを定義します。
  - ワークロードを構成するさまざまなサービス間でデータがどのように転送されるかを明確に定義したデータフロー図を作成します。これらのフローを認証済みのネットワークチャネルに実際に流していく前に、まずこの図を用意します。
  - 開発段階とテスト段階でワークロードを計測して、ランタイム時のワークロードの動作がデータフロー図に正確に反映されていることを確認してください。
  - データフロー図は、脅威モデリングを行う (「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」を参照) ときにも役立ちます。
- ネットワーク統制を確立する: データフローに応じたネットワーク統制を確立するときは AWS の機能を使用することを検討します。ネットワーク境界は、それだけでは十分なセキュリティ統制にはなりませんが、ワークロードを保護する多層防御戦略の 1 層にはなります。
  - リソース間のデータフローの確立、定義、制限には、セキュリティグループを使用します。
  - AWS サービスと、AWS PrivateLink をサポートしているサードパーティーのサービスの両方と通信するときは、AWS PrivateLink の使用を検討します。AWS PrivateLink インターフェイスエ

ンドポイントを介して送信されるデータは、AWS ネットワークバックボーン内にとどまり、公開インターネットを経由しません。

- ワークロード内のサービス間に認証と認可を実装する: ワークロード内のトラフィックフローの認証と暗号化を実現するために最も適した AWS サービスのセットを選択します。
  - サービス間通信のセキュリティを保護するときは、Amazon VPC Lattice の使用を検討します。VPC Lattice では、SigV4 認証を認証ポリシーと組み合わせて使用することで、サービス間のアクセスを制御できます。
  - mTLS を使用するサービス間通信では、API Gateway または App Mesh の使用を検討します。AWS Private CA を使用すると、mTLS で使用する証明書を発行できる、プライベート CA 階層を作成できます。
  - OAuth 2.0 または OIDC を使用したサービスを統合するときは、API Gateway で JWT オーソライザーを使用することを検討します。
  - ワークロードと IoT デバイスとの通信には、AWS IoT Core の使用を検討します。これには、ネットワークトラフィックの暗号化と認証の方法が複数用意されています。
- 不正アクセスを監視する: 意図しない通信チャネル、不正なプリンシパルによる保護済みリソースへのアクセス、その他不適切なアクセスパターンを継続的にモニタリングします。
  - VPC Lattice を使用してサービスへのアクセスの管理するときは、VPC Lattice アクセスログを有効にしてモニタリングすることを検討します。これらのアクセスログには、リクエスト元のエンティティに関する情報、ソースとターゲットの VPC などのネットワーク情報、リクエストのメタデータが記録されています。
  - ネットワークフローのメタデータをキャプチャし、異常がないか定期的に点検するときは、VPC フローログを有効にすることを検討します。
  - セキュリティインシデントのプランニング、シミュレーション、対応に関する解説は、「AWS セキュリティインシデント対応ガイド」および「セキュリティの柱 - AWS Well-Architected Framework」内の「インシデント対応」のセク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:

- 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
- SEC02-BP02 一時的な認証情報を使用する
- 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

関連ドキュメント:

- Evaluating access control methods to secure Amazon API Gateway APIs
- REST API の相互 TLS 認証の設定
- How to secure API Gateway HTTP endpoints with JWT authorizer
- Authorizing direct calls to AWS services using AWS IoT Core credential provider
- AWS セキュリティインシデント対応ガイド

関連動画:

- AWS re:invent 2022: Introducing VPC Lattice
- AWS re:invent 2020: Serverless API authentication for HTTP APIs on AWS

関連する例:

- Amazon VPC Lattice Workshop
- Zero-Trust Episode 1 – The Phantom Service Perimeter workshop

インシデントへの対応

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?

成熟した予防的、発見的統制が実装されていても、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、影響を緩和するメカニズムを実装する必要があります。準備することで、インシデントの際にチームが効果的に動作し、問題を切り分け、封じ込め、フォレンジックを実行し、運用を既知の正常な状態に復元する能力に強く影響します。セキュリティインシデントが起こる前にツールとアクセス権を整備し、ゲームデー (実践訓練) を通じてインシデント対応を定期的に実施しておけば、ビジネスの中断を最小限に抑えながら復旧することができます。

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 セキュリティインシデント対応プレイブックを作成し、テストする
- 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 シミュレーション行う
- SEC10-BP08 インシデントから学ぶためのフレームワークを確立する

SEC10-BP01 重要な人員と外部リソースを特定する:

組織のインシデント対応体制を整えるため、組織内外の担当者、リソース、法的義務を特定します。

期待される成果: 主要な担当者、その連絡先情報、セキュリティイベントに対応する際のその役割から成る一覧を作成します。この情報を定期的に見直し、組織内外のツールの観点から人員配置の変更を反映させます。この情報の文書化にあたっては、セキュリティパートナー、クラウドプロバイダー、SaaS (Software-as-a-Service) アプリケーションなど、サードパーティーのサービスプロバイダーやベンダーをすべて考慮します。セキュリティイベントの発生時は、適切な責任とアクセス権を持つ担当者が状況を適切に理解して、対応と復旧にあたることができます。

一般的なアンチパターン:

- 主要担当者の連絡先と、セキュリティイベントへの対応時の役割や責任について最新情報をまとめたリストが用意されていない。
- イベントへの対応や復旧の際に、担当者、依存関係、インフラストラクチャ、ソリューションについて全員がわかっているものだと想定している。
- 主要なインフラストラクチャやアプリケーションの設計を記載したドキュメントまたはナレッジリポジトリがない。
- セキュリティイベント発生時の効果的な対応方法を新しい人員に指導する適切なオンボーディングプロセス (イベントシミュレーションの実施など) が用意されていない。
- 主要担当者が一時的に不在の場合や、セキュリティイベントの発生時に対応できない場合に備えたエスカレーションパスが用意されていない。

このベストプラクティスを活用するメリット: このベストプラクティスを活用することで、イベント発生時に適切な担当者とその役割とを特定するのにかかる、トリアージと対応の時間を短縮することができます。主要担当者とその役割の最新リストが用意してあれば、適切な担当者をイベントのトリアージと復旧に投入でき、イベント発生時の時間の無駄使いを極力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

組織内の主要な担当者を特定する: インシデント対応に必要な、組織内の担当者の連絡先一覧を保存します。この情報を定期的に見直し、組織編成の変更、昇進、チームの変更など、人員配置に変更があった場合は適宜更新してください。インシデント管理者、インシデント対応者、コミュニケーションリーダーなどの主要な役割については特に重要です。

- インシデント管理者: インシデント管理者は、イベント対応時のすべての権限を有します。
- インシデント対応者: インシデント対応者はインシデントの調査および修正を担当する人です。これらの人員はイベントの種類によって異なりますが、通常は、影響を受けたアプリケーションを担当する開発者や運用チームです。
- コミュニケーションリーダー: コミュニケーションリーダーは、組織内外とのコミュニケーション、特に公的機関、規制当局、顧客とのコミュニケーションを担当します。
- 対象分野のエキスパート (SME): 分散型の自律的なチームの場合は、ミッションクリティカルなワークロードに SME を指名しておくことが推奨されます。SME は、イベントに関与する重要なワークロードの運用とデータ分類に関する深い知識を共有してくれます。

主要連絡先の入手、対応プランの策定、オンコールスケジュールの自動化、エスカレーションプランの作成のため、AWS Systems Manager Incident Manager 機能を使用することを検討します。オンコールスケジュールでスタッフ全員を自動でローテーションさせ、ワークロードの責任を所有者間で分担できます。これにより、関連するメトリクスやログの生成、ワークロードにとって重要なアラームしきい値の定義など、優れた取り組みが促されます。

外部パートナーを特定する: 顧客向けに差別化されたソリューションを構築するため、多くの企業が、独立系ソフトウェアベンダー (ISV)、パートナー、下請け業者が構築したツールを使用しています。これらの関係先の担当者に、インシデントへの対応と復旧を支援してもらいます。サポートケースを通じて AWS の対象分野のエキスパートに迅速に連絡できるように、適切なレベルの AWS Supportにサインアップすることをお勧めします。ワークロード用のすべての重要なソリューションプロバイダーに対して、同様の取り決めを検討してください。一部のセキュリティイベントについては、上場企業は該当イベントとその影響を関連する公的機関や規制当局に通知する義務があります。関連部門や担当者の連絡先情報を管理し、更新します。

実装手順

1. インシデント管理ソリューションを設定します。

a. セキュリティツール用アカウントに Incident Manager をデプロイすることを検討してください。

2. インシデント管理ソリューションで連絡先を定義します。

a. インシデントの発生時に連絡が取れるように、連絡先ごとに少なくとも 2 種類の連絡チャネル (SMS、電話、E メールなど) を定義します。

3. 対応計画を定義します。

a. インシデント発生時の対応要員として最適な連絡先を特定します。個々の連絡先ではなく、対応担当者の役割に合わせたエスカレーション計画を定義します。インシデントの解決に直接関与していない場合でも、外部機関への情報提供を担当する可能性のある連絡先を含めておくことを検討してください。

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

関連ドキュメント:

- AWS セキュリティインシデント対応ガイド

関連する例:

- AWS customer playbook framework
- Prepare for and respond to security incidents in your AWS environment

関連ツール:

- AWS Systems Manager Incident Manager

関連動画:

- Amazon's approach to security during development

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 に準拠したワークロードを AWS で処理している場合は、「NIST SP 800-61 Computer Security Handling Guide」に従うと役に立ちます。同様に、欧州人の、個人を特定できる情報 (PII) のデータを含むワークロードを処理している場合は、「EU General Data Protection Regulation (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から自身をどう保護し、どう対処していくのかについてのシナリオを検討します。

AWS のワークロードに関するインシデント管理計画を策定するときは、AWS 責任共有モデルの検討から開始して、インシデント対応に向けた多層防御の戦略を構築していきます。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 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主要なコンセプトと基本の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

役割と責任を定義する:

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 Support
  - AWS Support は、AWS ソリューションの成功とオペレーションの正常性をサポートするツールと専門知識にアクセスできる一連のプランを用意しています。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。
  - AWS リソースに影響する問題に関してサポートを得るための連絡窓口として、AWS Management Console (サインインが必要) のサポートセンター を検討します。AWS Supportへのアクセスは AWS Identity and Access Management によって制御されます。AWS Supportの機能を利用する方法については、「Getting started with AWS Support」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、AWS 責任共有モデルの顧客側のセキュリティイベントが発生したときに顧客にサポートを提供する、24 時間対応の専門のグローバル AWS チームです。

  - 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 Support case から AWS CIRT と連携することができます。
- DDoS 対応のサポート
  - AWS が提供する 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、分散型サービス拒否攻撃 (DDoS) のマネージド型防御サービスです。Shield を使用すれば、常時検出と自動インライン緩和の機能により、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、DDoS 防御に AWS Supportを使う必要がなくなります。Shield は、AWS Shield Standard と AWS Shield Advanced の 2 種類に分かれています。両者の違いに関する詳細は、「AWS Shield の特徴」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。AMS は、ベストプラクティスを実行してインフラストラクチャを管理することで、運用のオーバーヘッドとリスクを減らします。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、ライフサイクル全般にわたるサービスを提供します。
  - 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 の顧客に共有されるため、顧客は AMS と対応策を練り、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が記されています。
- 役割と責任: インシデントに対応する利害関係者が一覧表示され、インシデント発生時のそれぞれの役割が詳しく記されています。
- コミュニケーションプラン: 連絡先とインシデント発生時の連絡方法が記されています。

- 通信手段のバックアップ: インシデント関連の通信のバックアップ方法としては、帯域外通信を確保する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各段階と取るべき措置: インシデント対応の各段階 (検出、分析、根絶、封じ込め、復旧など) を一覧にし、各段階で取るべき措置を大まかに記しています。
- インシデントの深刻度と優先順位の決定: インシデントの深刻度の分類方法、インシデントの優先付け方法、深刻度の定義がエスカレーション手順にどう影響するか、を詳しく説明しています。

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?)

関連ドキュメント:

- 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:

セキュリティインシデントが発生する前に、セキュリティイベントの調査を支援するフォレンジック機能の整備を検討します。

このベストプラクティスを活用しない場合のリスクレベル: 中

AWS には、従来のオンプレミスフォレンジックの概念が適用されます。AWS クラウドでフォレンジック機能の構築を開始するための重要な情報については、「Forensic investigation environment strategies in the AWS クラウド」を参照してください。

フォレンジックのための環境と AWS アカウント構造が整ったら、次の 4 つのフェーズにわたってフォレンジックに適した方法論を効果的に実行するために必要なテクノロジーを定義します。

- 収集: AWS CloudTrail、AWS Config、VPC フローログ、ホストレベルのログなどの関連 AWS ログを収集します。可能であれば、影響を受けた AWS リソースのスナップショット、バックアップ、メモリダンプを収集します。

- 調査: 関連する情報を抽出して評価することにより、収集されたデータを検証します。
- 分析: 収集したデータを分析してインシデントを解明し、そこから結論を導き出します。
- レポート: 分析フェーズから得られた情報を報告します。

実装手順

フォレンジック環境を準備する

AWS Organizations では、AWS リソースの拡大とスケールに合わせて、AWS 環境を一元的に管理および運用できます。AWS 組織を利用することで、AWS アカウントを統合して 1 つのユニットとして管理できるようになります。組織単位 (OU) を使用すると、アカウントをまとめてグループ化し、単一の単位として管理できます。

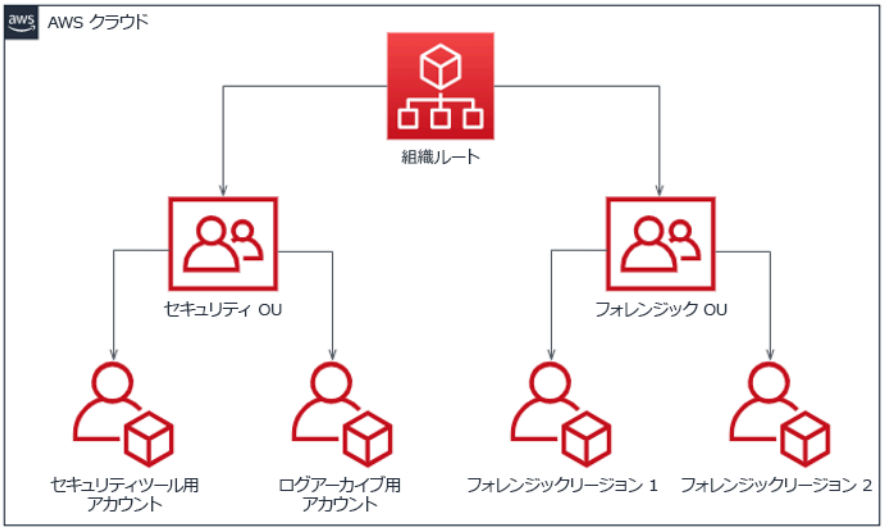
インシデント対応には、セキュリティ OU および フォレンジック OU を含むインシデント対応の機能をサポートする AWS アカウント構造があると便利です。セキュリティ OU 内には、次のアカウントが必要です。

- ログアーカイブ: 限られたアクセス許可を持つログアーカイブ用の AWS アカウントにログを集約します。
- セキュリティツール: セキュリティサービスをセキュリティツール用の AWS アカウントに一元化します。このアカウントは、セキュリティサービスの委任管理者として機能します。

フォレンジック OU 内では、お客様のビジネスモデルと運用モデルに最適なフォレンジックアカウントに応じて、フォレンジック用に 1 つのアカウントを実装するか、事業を展開するリージョンごとにアカウントを実装できます。リージョンごとにフォレンジックアカウントを作成すると、そのリージョン外での AWS リソースの作成をブロックし、リソースが意図しないリージョンにコピーされるリスクを低減できます。例えば、米国東部 (バージニア北部) リージョン (us-east-1) および米国西部 (オレゴン) (us-west-2) のみで運用する場合、フォレンジック OU には 2 つのアカウントがあります。1 つは us-east-1 用で、もう 1 つは us-west-2 用です。

複数のリージョンのフォレンジック AWS アカウントを作成できます。そのアカウントに AWS リソースをコピーする場合は、データ主権に関する要件に準拠しているか注意する必要があります。新しいアカウントのプロビジョニングには時間がかかるため、インシデントのかなり前にフォレンジックアカウントを作成して実装し、対応担当者が効果的に対応できるように準備しておくことが重要です。

次の図は、リージョンごとのフォレンジックアカウントを持つフォレンジック OU を含むアカウント構造の例を示しています。



インシデント対応のためのリージョンごとのアカウント構造

バックアップとスナップショットをキャプチャする

主要なシステムとデータベースのバックアップをセットアップすることは、セキュリティインシデントからの回復とフォレンジックのために重要です。バックアップを作成しておけば、システムを以前の安全な状態に復元できます。AWS では、さまざまなリソースのスナップショットを作成できます。スナップショットでは、こうしたリソースのポイントインタイムバックアップを作成できます。バックアップや復旧をサポートできる AWS のサービスは数多くあります。これらのサービス、バックアップと復旧のアプローチの詳細については、バックアップと復旧についての規範ガイダンスおよび「Use backups to recover from security incidents」を参照してください。

特にランサムウェアのような状況では、バックアップをしっかりと保護することが重要です。バックアップの保護に関するガイダンスについては、「Top 10 security best practices for securing backups in AWS」を参照してください。バックアップの保護に加えて、バックアップと復元のプロセスを定期的にテストして、導入しているテクノロジーとプロセスが想定どおりに機能することを確認する必要があります。

フォレンジックを自動化する

セキュリティイベント中、インシデント対応チームは、イベント前後の期間の証拠を、正確性を維持しながら迅速に収集して分析できなければなりません (特定のイベントやリソースに関連するログのキャプチャ、Amazon EC2 インスタンスのメモリダンプの収集など)。インシデント対応チームにとって、関連する証拠を手作業で収集することは困難であり、時間もかかります。多数のインスタンスやアカウントが対象となる場合は特にそうです。さらに、手作業による収集では人為的ミスが起こ

りやすくなります。このような理由から、フォレンジックの自動化を可能な限り開発し、実装する必要があります。

AWS には、フォレンジック用の自動化リソースが多数用意されており、これらのリソースは以下のリソースセクションに一覧表示されています。これらのリソースは、当社が開発し、お客様が実装したフォレンジックパターンの例です。手始めに参考にするリファレンスアーキテクチャとしては有効かもしれませんが、環境、要件、ツール、フォレンジックプロセスに基に変更するか、新しいフォレンジック自動化パターンを作成することを検討してください。

リソース

関連ドキュメント:

- AWS Security Incident Response Guide - Develop Forensics Capabilities
- AWS Security Incident Response Guide - Forensics Resources
- Forensic investigation environment strategies in the AWS クラウド
- How to automate forensic disk collection in AWS
- AWS 規範ガイダンス - Automate incident response and forensics

関連動画:

- Automating Incident Response and Forensics

関連する例:

- Automated Incident Response and Forensics Framework
- Automated Forensics Orchestrator for Amazon EC2

SEC10-BP04 セキュリティインシデント対応プレイブックを作成し、テストする

インシデント対応プロセスを準備する上で重要なのは、プレイブックを作成することです。インシデント対応プレイブックには、セキュリティイベントが発生したときに従うべき一連の規範的なガイダンスと手順が記載されています。明確な体制と手順があると、対応が簡単になり、人為的ミスの可能性が低くなります。

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

プレイブックは、次のようなインシデントシナリオ向けに作成する必要があります。

- 予想されるインシデント: プレイブックは、予測されるインシデントに合わせて作成する必要があります。これには、サービス拒否 (DoS)、ランサムウェア、認証情報の漏えいなどの脅威が含まれます。
- 既知のセキュリティ上の検出結果またはアラート: プレイブックは、既知のセキュリティ上の検出結果とアラート (GuardDuty の検出結果など) に基づいて作成する必要があります。GuardDuty の検出結果を受け取っても、どうすればよいかわからないといったことがあるかもしれません。そこで、GuardDuty の検出結果を誤って処理したり無視したりすることがないように、GuardDuty で検出される可能性のある問題ごとにプレイブックを作成しておきます。修正に関する詳細とガイダンスについては、GuardDuty のドキュメントで確認できます。なお、GuardDuty はデフォルトでは有効になっておらず、コストがかかりますので注意してください。GuardDuty の詳細については、「付録 A: クラウド機能定義 - 可視性とアラート」を参照してください。

プレイブックには、起こりうるセキュリティインシデントを適切に調査して対応するために、セキュリティアナリストが実行すべき技術的な手順を記載する必要があります。

実装手順

プレイブックに記載すべき項目には次のようなものがあります。

- プレイブックの概要: このプレイブックがどのようなリスクやインシデントシナリオに対応しているか。このプレイブックの目的は何か。
- 前提条件: このインシデントシナリオには、どのようなログ、検出メカニズム、自動ツールが必要か。どのような通知が想定されるか。
- コミュニケーションとエスカレーションに関する情報: 関与している人員およびその連絡先情報。各利害関係者の責任は何か。
- 対応ステップ: インシデント対応の各フェーズで、どのような戦術的措置を講じるべきか。アナリストはどのようなクエリを実行すべきか。望ましい結果を得るためにどのようなコードを実行すべきか。
  - 検知: インシデントはどのように検出されるか。
  - 分析: 影響範囲はどのように特定されるか。
  - 封じ込め: 影響範囲を限定するために、インシデントをどのように隔離するか。
  - 根絶: どのようにして脅威を環境から取り除くか。

  - 復旧: 影響を受けたシステムやリソースをどのようにして本番環境に戻すか。
- 期待される結果: クエリとコードが実行された後、プレイブックで想定される結果はどのようなものか。

リソース

関連する Well-Architected のベストプラクティス:

- SEC10-BP02 - インシデント管理計画を作成する

関連ドキュメント:

- Framework for Incident Response Playbooks
- Develop your own Incident Response Playbooks
- Incident Response Playbook Samples
- Building an AWS incident response runbook using Jupyter playbooks and CloudTrail Lake

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 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。適切な許可を持つユーザー、グループ、ロールを使用してタスクを実行し、AWS リソースにアクセスすることをお勧めします。ルートユーザーは、ルートユーザー認証情報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と多要素認証 (MFA) を適用する必要があります。インシデント対応プレイブックで AWS Management Consoleへのアクセスのみが要求されている場合、そのユーザーのアクセスキーが設定されてはならず、アクセスキー作成を明示的に禁止する必要があります。これは IAM ポリシーまたはサービスコントロールポリシー (SCP) で設定できます。詳細は、「AWS Security Best Practices for AWS Organizations SCPs」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 アカウントのルートユーザー 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、ルートユーザーのサインイン認証情報と 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 IAM ポリシーを設定するには、IAM Access Analyzer を使用して AWS CloudTrail ログに基づいてポリシーを生成することを検討してください。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、プレイブックを一とおり実行します。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。

インシデント対応アカウントを使用して、別の AWS 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。すべての `AssumeRole` リクエストが CloudTrail 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 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ために使用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インシデント対応 IAM ロールに関連する AssumeRole イベント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、すみ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 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

関連ドキュメント:

- Managing temporary elevated access to your AWS environment
- 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
- クロスアカウントアクセス用に MFA を設定する
- IAM Access Analyzer を使用した IAM ポリシーの設定
- マルチアカウント環境における AWS Organizations サービスコントロールポリシーのベストプラクティス
- How to Receive Notifications When Your AWS Account's Root Access Keys Are Used
- Create fine-grained session permissions using IAM managed policies

関連動画:

- Automating Incident Response and Forensics in AWS
- DIY guide to runbooks, incident reports, and incident response
- Prepare for and respond to security incidents in your AWS environment

関連する例:

- Lab: AWS Account Setup and Root User
- Lab: Incident Response with AWS Console and CLI

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事前にデプロイしておきます。

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

セキュリティ対応と運用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、既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。

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。

セキュリティ調査中、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、関連ログを確認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。クエリと取得のメカニズムとアラートを選択、有効化、保存、セットアップし、アラート発行を設定することが非常に重要となります。さらに、ログデータを検索するツールとして、Amazon Detective が有効です。

AWS では、200 を超えるクラウドサービスと数千の機能を提供しています。インシデント対応戦略をサポートし、簡素化できるサービスを確認することをお勧めします。

ログ記録に加えて、タグ付け戦略を策定して実装する必要があります。タグ付けを行うことで、AWS リソースの目的についての背景情報を付け加えることができます。タグ付けは自動化にも使用できます。

実装手順

分析とアラート発行のためのログを選択して設定する

インシデント対応のログ記録の設定については、次のドキュメントを参照してください。

- Logging strategies for security incident response
- 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

検出と対応をサポートするセキュリティサービスを有効にする

AWS は検出、予防、対応のネイティブ機能備えているほか、カスタムセキュリティソリューションの構築に使用できるサービスも提供しています。セキュリティインシデント対応に最も関連性の高いサービスのリストについては、「クラウド機能の定義」を参照してください。

タグ付け戦略を策定し、実装する

AWS リソースを取り巻くビジネスユースケースやかかわりのある内部関係者についての背景情報の入手は難しい場合があります。これを達成する方法の 1 つとして、タグを使用して、ユーザー定義のキーと値で構成されるメタデータを AWS リソースに割り当てる方法があります。タグを作成して、目的、所有者、環境、処理されるデータの種類など、任意の基準でリソースを分類できます。

一貫したタグ付け戦略があると、AWS リソースに関する背景情報をすばやく特定、識別できるため、応答時間を短縮し、組織の背景情報の把握に費やす時間を最小限に抑えることができます。タグは、対応の自動化を開始するためのメカニズムとしても機能します。タグ付けする対象の詳細については、「Tagging your AWS resources」を参照してください。まず、組織全体に導入するタグを定義する必要があります。その後、このタグ付け戦略を導入し、適用します。導入と適用の詳細については、「Implement AWS resource tagging strategy using AWS Tag Policies and Service Control Policies (SCPs)」を参照してください。

リソース

関連する Well-Architected の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 標準化した場所にログ、検出結果、メトリクスを取り込む

関連ドキュメント:

- Logging strategies for security incident response
- Incident response cloud capability definitions

関連する例:

- Amazon GuardDuty と Amazon Detective を使用した脅威の検出と対応
- Security Hub Workshop
- Vulnerability Management with Amazon Inspector

SEC10-BP07 シミュレーション行う

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、シミュレーション (ゲームデーとも呼ばれる) 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確立するメリット: シミュレーションには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

関連ドキュメント:

- AWS Incident Response Guide

関連動画:

- AWS GameDay - Security Edition

SEC10-BP08 インシデントから学ぶためのフレームワークを確立する

教訓フレームワークと根本原因分析プロセスを導入することは、インシデント対応能力の向上だけでなく、インシデントの再発防止にも役立ちます。各インシデントから学ぶことで、同じ失敗、露出、設定ミスの繰り返しを防ぐことができ、セキュリティ体制が強化されるだけでなく、予防できたはずの状況に無駄にする時間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: 中

実装のガイダンス

以下の点を大まかに確立して達成する教訓フレームワークを実装することが重要です。

- 事後検証会を実施するタイミング
- 事後検証会を通して行うこと
- 事後検証会の実施方法
- そのプロセスに関わる人物、またかかわり方
- 改善の余地がある領域の特定方法
- 改善事項を効果的に追跡、実装する方法

フレームワークは、個人に焦点を当てたり非難したりするのではなく、ツールやプロセスの改善に焦点を当てるべきです。

実装手順

前述の大局的な成果とは別に、プロセスから最大の価値 (実行可能な改善につながる情報) を引き出すためには、適切な質問を行うことが重要です。教訓についての議論を進めるうえで役立つ質問には次のようなものがあります。

- どのようなインシデントでしたか。
- インシデントが最初に特定されたのはいつでしたか。
- どのようにして特定されましたか。
- どのシステムからアクティビティについてのアラートが発行されましたか。
- どのようなシステム、サービス、データが関与しましたか。

- 具体的に何が起きましたか。
- 何がうまくいきましたか。
- 何がうまくいきませんでしたか。
- インシデントに対応できなかった、またはスケールに失敗したのはどのプロセスまたは手順ですか。
- 次の領域で改善できることは何でしょうか。
  - 人員
    - 連絡する必要があった担当者に実際に連絡がつきましたか。また、連絡先リストの情報は最新のものでしたか。
    - インシデントに効果的に対応して調査するために必要なトレーニングや能力を欠いていましたか。
    - 適切なリソースは用意されていましたか。
  - プロセス
    - 対応はプロセスと手順に従って進められましたか。
    - この (タイプの) インシデントについて、プロセスと手順が文書化され、利用可能になっていましたか。
    - 必要なプロセスや手順が欠けていましたか。
    - 対応担当者は、問題に対応するために必要な情報にタイムリーにアクセスできましたか。
  - テクノロジー
    - 既存のアラートシステムは、アクティビティを効果的に特定してアラートを出しましたか。
    - どうすれば検出までの時間を 50% 短縮できたでしょうか。
    - この (タイプの) インシデントに備えて、既存のアラートを改善する、または新しいアラートを作成する必要がありますか。
    - 既存のツールでインシデントを効果的に調査 (検索/分析) できましたか。
    - この (タイプの) インシデントをより早く特定するにはどうすればよいでしょうか。
    - この (タイプの) インシデントの再発を防ぐにはどうすればよいでしょうか。
    - 改善計画の所有者は誰ですか。また、その実施状況をどのように検証しますか。
    - 追加のモニタリング、予防的統制やプロセスを導入し、テストするまでのスケジュールはどのようになっていますか。

このリストはすべてを網羅しているわけではありませんが、インシデントから最も効果的に学び、セキュリティ体制の継続的な改善に向けて、組織とビジネスのニーズを見極め、その分析方法を特定す

るための出発点として活用いただくことを目的としています。最も重要なのは、事後検証会を標準的なインシデント対応プロセスと文書化の一部として取り入れ、想定されるものとして関係者全員にも定着させることです。

リソース

関連ドキュメント:

- AWS Security Incident Response Guide - Establish a framework for learning from incidents
- NCSC CAF guidance - Lessons learned

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問題が発生する可能性を軽減するのに役立ちます。

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: ソフトウェアは、セキュリティを念頭に置いて設計および構築する必要があります。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

組織のビルダーに対してトレーニングを実施します。脅威モデリングのコースから始めることは、セキュリティのトレーニングに役立つ良い基盤です。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たずねることなく、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 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

関連ドキュメント:

- AWS トレーニング と認定
- How to think about cloud security governance
- 脅威モデリングへのアプローチ方法
- Accelerating training – The AWS Skills Guild

関連動画:

- Proactive security: Considerations and approaches

関連する例:

- 脅威モデリングについてのワークショップ
- Industry awareness for developers

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特定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のセキュリティ問題のリスクが低減されます。

期待される成果: 自動テストの目的は、開発ライフサイクルを通じて潜在的な問題を早期に、かつ頻繁にプログラムで検出する方法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、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特定し、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目的とす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ぞれの目的とする成果の比較に基づいており、テストのライフサイクル全体を迅速化します。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、自動化に最適です。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取得することができます。静的または動的なコード分析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における潜在的なソフトウェアの問題の検出に役立ちます。

一般的なアンチパターン:

- 自動化テストのテストケースおよび結果のコミュニケーションが欠如している。

- リリース直前にのみ自動化テストを実施する。
- 自動化テストケースの要件を頻繁に変更する。
- セキュリティテストの結果の対処方法に関するガイダンスが欠如している。

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が低減します。
- 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り、一貫性が向上します。
- 本稼働ソフトウェアでセキュリティ問題が発生する可能性が低減されます。
- ソフトウェアの問題を早期に検出することで、検出から修正までの時間が短縮されます。
- 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が向上し、組織全体で改善が促進されます。

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: 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、セキュリティに重点を置い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、欠陥が発生しやすいコードを検出します。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 Quality Group は、Source Code Security Analyzer の比較を提供します。これには、Byte Code Scanner と Binary Code Scanner 用のオープンソースツールが含まれています。

潜在的な想定外の動作を識別するために、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) 手法によって静的テストを補完します。動的テストは、静的分析では検出できない潜在的な問題の検出に使用できます。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種類の潜在的な問題をチェックできます。Amazon CodeWhisperer は、ビルダーの IDE で、セキュリティスキャンなどのコード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、セキュリティ問題、見つけにくいバグを特定し、コード品質を向上させるためのレコメンデーションを提供します。

Security for Developers ワークショップでは、AWS CodeBuild、AWS CodeCommit、AWS CodePipeline などの AWS 開発者ツールを使用して、SAST および DAST テスト手法を含むリリースパイプラインの自動化を行います。

SDLC を進める中で、セキュリティチームと共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環として、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し、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。
- 修正が必要な問題を単に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。
- Security for Developers ワークショップでは、静的テストと動的テストをリリースパイプラインに統合する例を示します。
- 開発者 IDE と統合された Amazon CodeWhisperer や、コミット時にコードをスキャンするための Amazon CodeGuru Reviewer などの自動ツールを使用してテストまたはコード分析を実行すると、ビルダーは適切なタイミングでフィードバックを得ることができます。
- AWS Lambda を使用して構築する場合は、Amazon Inspector を使用して関数内のアプリケーションコードをスキャンできます。
- 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通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改善を支援します。
- 自動化ツールからの検出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

関連ドキュメント:

- Continuous Delivery and Continuous Deployment

- AWS DevOps コンピテンシーパートナー

- AWS Security Competency Partners for Application Security
- Choosing a Well-Architected CI/CD approach
- Amazon EventBridge および Amazon CloudWatch Events の CodeCommit イベントのモニタリング
- Secrets detection in Amazon CodeGuru Review
- Accelerate deployments on AWS with effective governance
- How AWS approaches automating safe, hands-off deployments

関連動画:

- Hands-off: Automating continuous delivery pipelines at Amazon
- Automating cross-account CI/CD pipelines

関連する例:

- Industry awareness for developers
- AWS CodePipeline Governance (GitHub)
- Security for Developers workshop

SEC11-BP03 定期的にペンテストを実施する

定期的にソフトウェアのペ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出できない、ソフトウェアの潜在的な問題の特定に役立ちます。また、検出統制の効率を理解する上でも役立ちます ペンテストでは、保護する必要があるデータを公開する、想定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行します。

期待される成果: ペンテストは、アプリケーションのセキュリティ特性を検出、修正、検証するために使用します。ソフトウェア開発ライフサイクル (SDLC) の一環として、定期的かつ計画的にペンテストを実施します。ペ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。ペンテストでの検出結果を分析し、自動化によって検出できる問題があるかどうかを確認します。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ンテストを実施する。
- 依存するサードパーティーツールやライブラリを除いてアプリケーションのペ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関する信頼性を向上させます。
- 望ましいアプリケーションパターンを特定する機会を創出して、ソフトウェアの品質をさらに向上させます。
- 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特定するフィードバックループを確立します。

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

ペンテストは、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出、修正、検証する、構造化されたセキュリティテストです。ペンテストは、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を作成し、実施します。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した場合は、必ずペンテストを実施する必要があります。

ペ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間近の状態で、なおかつ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

- ペンテストの範囲を設定するための構造化されたプロセスを用意し、このプロセスを脅威モデルに基づいて行うことは、コンテキストを維持するための良い方法です。

- ペ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、ペンテストプロセスの時間を短縮します。
- ペ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特定し、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

関連ドキュメント:

- AWS Penetration Testing provides detailed guidance for penetration testing on AWS
- Accelerate deployments on AWS with effective governance
- 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 Fault Injection Simulator

関連する例:

- Automate API testing with AWS CodePipeline (GitHub)
- Automated security helper (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、コードを記述したユーザーが、コードの品質をチェックする唯一のユーザーでないことを検証する上で役立ちます。

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と、記述されるソフトウェアの品質が向上し、チームの経験の浅いメンバーのスキルが向上して、自動化を使用できる場所を特定する機会が提供されます。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。
- ペンテストや後工程において検出される問題が低減する。
- チーム内での知識の移転が改善される。

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: 中

実装のガイダンス

全体的なコード管理フローの一環として、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ー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、プロセスを本番環境にデプロイする前に、コード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のオーケストレーションが容易になります。

実装手順

- コード管理フローの一環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には、Amazon CodeGuru Reviewer の使用を検討してください。

- 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、自動的に検出できるプロセスがないかを確認します。
- 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ステップを統合します。

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

関連ドキュメント:

- AWS CodeCommit リポジトリのプルリクエストを操作する
- Working with approval rule templates in AWS CodeCommit
- About pull requests in GitHub
- Amazon CodeGuru Reviewer での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

関連動画:

- Continuous improvement of code quality with Amazon CodeGuru

関連する例:

- Security for Developers workshop

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、記述されたコードに加えて、他のソフトウェアパッケージのセットで構成されます。これにより、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、そのパッケージの特性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をより良く理解できる。
- 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を低減する。

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: 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。更新が発生したときや新しい要件が発生したときに、リポジトリにパッケージを追加する効率的な方法を実装する必要があります。AWS CodeArtifact や同様の AWS パートナーソリューションなどの AWS のサービスでは、この機能を提供する方法を提供しています。

実装手順:

- ソフトウェアを開発し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。
- 最も頻繁に使用されるパッケージ、言語、および変更量が最も多いチームの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特定します。

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

関連ドキュメント:

- Accelerate deployments on AWS with effective governance
- Tighten your package security with CodeArtifact Package Origin Control toolkit
- Detecting security issues in logging with Amazon CodeGuru Reviewer
- Supply chain Levels for Software Artifacts (SLSA)

関連動画:

- Proactive security: Considerations and approaches
- The AWS Philosophy of Security (re:Invent 2017)
- When security, safety, and urgency all matter: Handling Log4Shell

関連する例:

- Multi Region Package Publishing Pipeline (GitHub)
- Publishing Node.js Modules on AWS CodeArtifact using AWS CodePipeline (GitHub)
- AWS CDK Java CodeArtifact Pipeline Sample (GitHub)
- Distribute private .NET NuGet packages with AWS CodeArtifact (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、人為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: データからユーザーを遠ざけることは、AWS クラウドでソフトウェアを安全に構築するための重要な原則です。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で人への依存を排除することで、テスト済みのソフトウェアが常に一貫してデプロイ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使用することで、コードを変更することなく、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。暗号化を使用し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:

- 本番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が向上します。
- 変更に失敗してビジネスの機能性に影響を与えるリスクが低減されます。
- 変更リスクの低減により、リリース頻度が向上します。
- デプロイ中に予期しないイベントが発生した場合に自動ロールバックが機能します。
- 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

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。AWS Systems Manager Parameter Store などの外部ソースから環境固有の設定データを取得できるように、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 や 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 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 CI/CD 機能を提供することで、セキュリティテストをパイプラインに簡単に統合できます。
- 「複数のアカウントで AWS 環境を構成する」ホワイトペーパーの「環境の分離」に関するガイダンスに従ってください。
- 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に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 や AWS Key Management Service (AWS KMS) などの暗号化ツールを使用し、デプロイするソフトウェアパッケージに署名して検証します。

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

関連ドキュメント:

- AWS CI/CD ワークショップ
- Accelerate deployments on AWS with effective governance
- 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
- 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

関連する例:

- 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で、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されるパイプラインは、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が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、長期的認証情報に依存せず、構築環境の整合性検証のためにステータスを送信するように設定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ない。
- 長期的認証情報、または強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:

- パイプラインによっ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が向上します。
- 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を備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

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こ

とで、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従うことは、CI/CD 環境を構築するための出発点として最適です。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に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から始めます。
- AWS IAM Access Analyzer を使用して、パイプラインの最小特権 IAM ポリシーをプログラムで生成することを検討してください。
- パイプラインをモニタリングやアラートと統合することで、予期しないアクティビティや異常なアクティビティの通知を受け取ることができます。AWS マネージドサービスでは、Amazon EventBridge を使用することで、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できます。

リソース

関連ドキュメント:

- AWS デプロイパイプラインリファレンスアーキテクチャ
- AWS CodePipeline のモニターリング
- 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 (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。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: セキュリティの所有権と意思決定をビルダーチームに根付かせるには、セキュリティの考え方についてビルダーをトレーニングするか、ビルダーチームに所属するまたは関連しているセキュリティ担当者によってトレーニングを強化します。いずれのアプローチも有効で、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としています。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設計思考を重視するのに役立ちます。セキュリティを重視したビルダーコミュニティ、またはビルダーチームと連携するセキュリティエンジニアグループのもう 1 つの利点は、ソフトウェアの記述方法をより深く理解できることです。この理解は、自動化機能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が短縮されます。
- セキュリティのレビュー段階でようやく検出されるセキュリティ問題が低減されます。
- 作成されるソフトウェアの全体的な品質が向上します。
- システム的な問題や価値の高い改善領域を特定し、理解する機会が創出されます。
- セキュリティレビューでの検出結果に基づくやり直しが低減されます。
- セキュリティ機能に関する認識が改善されます。

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

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、組織に最適と思われるプログラムの運用モデルを特定します。主な 2 つのパターンは、ビルダーへ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測定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、良い学習経験になります。このベストプラクティスは、主に組織の変化と文化に重点を置いています。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。
- プログラムに参加する人数、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測定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

関連ドキュメント:

- 脅威モデリングへのアプローチ方法
- How to think about cloud security governance

関連動画:

- Proactive security: Considerations and approaches

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

- 基礎
- ワークロードアーキテクチャ
- 変更管理
- 障害管理

基礎

Questions

- REL 1. Service Quotas と制約はどのように管理するのですか。
- 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。

REL 1. Service Quotas 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、Service Quotas (サービスの制限とも言います) があります。これらのクォータは、サービスを不正使用されないようにするために、必要以上のリソースを誤ってプロビジョニングすることを防ぎ、API オペレーションのリクエストレートを制限するためのものです。また、光ファイバーケーブルにビットをプッシュダウンできる速度や、物理ディスク上のストレージの量などのリソースの制限もあります。

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: お客様は、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、自動修復の手順などに、適切なガイドラインを設定して、サービスクォータや制約がサービスの低下や中断につながるレベルに達していないことを確認することによって、AWS アカウントでのサービスの低下や中断を防ぐことができます。

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ことで、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を活用し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できる AWS のサービスです。クォータ値を検索できるほか、Service Quotas コンソールから、または AWS SDK を使用して、クォータの増加をリクエストし追跡することもできます。AWS Trusted Advisor には、特定のサービスの特定の要素の使用状況および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: Amazon VPC クォータを参照のこと)。

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 には独自のサービスの制限ダッシュボードがあり、インスタンス、Amazon Elastic Block Store、Elastic IP アドレスの制限を管理するのに役立ちます。サービスクォータがアプリケーションのパフォーマンスに影響を及ぼし、ニーズに合わせて調整できないような事例が発生した場合は、AWS Supportに連絡し、緩和策の有無についてお問い合わせください。

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は特定のリージョンで使用される DL Amazon EC2 の数を制限していますが、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の最中に、この制限に達する場合があります。

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。これらはストレージ容量だったり (例えば gp2 のサイズ制限は 1 GB ~ 16 TB)、ディスクスループットだったりします。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、AWS Supportに連絡し、緩和策の有無についてお問い合わせください。修正されたクォータの調整に関する詳細は、「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 のモニタリングや管理に役立つ AWS サービスやツールは多数あります。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 Management Console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するには、使用されているサービス (IAM Access Analyzer など) を確認します。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特定します。各リージョンには約 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 リソースを特定します。
- AWS CloudFormation データを使用して、使用されている AWS リソースを特定します。AWS Management Console で作成された、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確認します。テンプレート自体にデプロイされるように設定されたリソースも確認できます。
- 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。

- 適用するサービスクォータを決定します。Trusted Advisor および Service Quotas から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で知らせる、自動モニタリングの方法を作成します (「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」を参照のこと)。
- サービスクォータが 1 つのリージョンで変更されたときに同一アカウント内の他のリージョンで変更されたかどうかを確認する、自動化されたプログラムによる方法を作成します (「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」を参照のこと)。
- 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、クォータに必要とされる変更を測定するための手順を作成します (「REL01-BP05 クォータ管理を自動化する」を参照のこと)。
- 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

関連ドキュメント:

- AWS Well-Architected Framework の信頼性の柱: 可用性
- AWS Service Quotas (旧称: サービスの制限)
- AWS Trusted Advisor ベストプラクティスチェック (「Service Limits」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
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- AWS のクォータモニタ
- AWS 障害分離境界
- 冗長性を実装した可用性
- AWS for Data
- 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
- AWS における Account-per-Tenant 型 SaaS 環境のライフサイクル管理
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

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下、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、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) 間のすべ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

関連ドキュメント:

- AWS Well-Architected Framework の信頼性の柱: 可用性
- AWS Service Quotas (旧称: サービスの制限)
- AWS Trusted Advisor ベストプラクティスチェック (「Service Limits」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
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- AWS のクォータモニタ
- AWS 障害分離境界
- 冗長性を実装した可用性
- AWS for Data
- 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

- AWS における Account-per-Tenant 型 SaaS 環境のライフサイクル管理
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

関連動画:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

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

Service Quotas、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ル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活用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 が表示されている場合`ADJUSTABLE = No`、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

関連ドキュメント:

- AWS Well-Architected Framework の信頼性の柱: 可用性
- AWS Service Quotas (旧称: サービスの制限)
- AWS Trusted Advisor ベストプラクティスチェック (「Service Limits」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
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- AWS のクォータモニタ
- AWS 障害分離境界
- 冗長性を実装した可用性
- AWS for Data
- 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
- AWS における Account-per-Tenant 型 SaaS 環境のライフサイクル管理
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- 「Service Quotas 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

- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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、使用量を予定どお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、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。

Benefits of establishing this best practice: 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活用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。CloudWatch Logs のメトリクスフィルターを使用して、ログのパターンを検索および抽出し、使用量がクォータのしきい値に近づいているかどうかを判断することもできます。

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。現在のリソース消費を収集するには、Amazon EC2 `DescribeInstances` API などのサービス API オペレーションを使用します。
- 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
  - 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

関連ドキュメント:

- AWS Well-Architected Framework の信頼性の柱: 可用性
- AWS Service Quotas (旧称: サービスの制限)
- AWS Trusted Advisor ベストプラクティスチェック (「Service Limits」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
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- AWS のクォータモニタ
- AWS 障害分離境界
- 冗長性を実装した可用性
- AWS for Data
- 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
- AWS における Account-per-Tenant 型 SaaS 環境のライフサイクル管理
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- 「Service Quotas 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
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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的に行うことができます。

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: 中

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
    - AWS のクォータモニタリング - AWS ソリューション
  - Amazon SNS および AWS Service Quotas API を使用して、クォータしきい値に基づいてトリガーされるレスポンスをセットアップします。
  - 自動化をテストします。
    - 制限のしきい値を設定します。
    - AWS Config、デプロイパイプライン、Amazon EventBridge、またはサードパーティーからの変更イベントと統合します。
    - 応答をテストするために、人為的に低いクォータしきい値を設定します。
    - 通知に対して適切なアクションを取り、必要に応じて AWS Supportに問い合わせる自動操作をセットアップします。
    - 変更イベントを手動で開始します。
    - 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: サービスの制限)
- AWS Trusted Advisor ベストプラクティスチェック (「Service Limits」セクションを参照)
- AWS 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

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

この記事では、リソースクォータと使用量の間にスペースを維持する方法と、それが組織にどのように役立つかについて説明します。リソースの使用が終了した後も、そのリソースの使用クォータは引き続き考慮される可能性があります。これにより、リソースに障害が発生したり、アクセスできなくなったりする可能性があります。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て、リソースの傷害を防ぎます。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します。

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。戦略の例としては、サービスの制限を使い果たさずに、アクセスできないリソースを置き換える追加リソースを作成してフェイルオーバー条件に対応することが挙げられます。

このベストプラクティスを活用しない場合のリスクレベル: 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。次のフェイルオーバーケースを検討してください。

- 中断された、またはアクセスできない VPC。
- アクセスできないサブネット。
- リソースのアクセシビリティに影響するアベイラビリティーゾーンの低下。
- 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。
- 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。
- リージョンまたはアベイラビリティーゾーンでの障害の影響を受けるリソースのサブセット。

ビジネスへの影響は異なる可能性があり、フェイルオーバーの決定は状況ごとに異なります。アプリケーションまたはサービスのフェイルオーバーを決定する前に、フェイルオーバーロケーションとリソースのクォータでリソース容量計画に対処します。

各サービスのクォータを確認するときは、アクティビティのピークが通常のピークよりも高いことを考慮してください。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アクセス不可能ですが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を維持します。
- サービスクォータを決定します。一般的なデプロイパターン、可用性要件、消費の増加を考慮します。
- 必要に応じてクォータの引き上げをリクエストします。クォータ引き上げリクエストの待機時間を予測します。
- 信頼性の要件 (「9 の数」としても知られる) を決定します。
- コンポーネント、アベイラビリティーゾーン、リージョンの喪失などの潜在的な障害シナリオを理解します。
- 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。

- 現在のクォータ制限に適切なバッファを含めます。バッファの例は 15% です。
- 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、消費の傾向を監視します。
- 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。これは実装すると、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

関連ドキュメント:

- AWS Well-Architected Framework の信頼性の柱: 可用性
- AWS Service Quotas (旧称: サービスの制限)
- AWS Trusted Advisor ベストプラクティスチェック (「Service Limits」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
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- AWS のクォータモニタ
- AWS 障害分離境界
- 冗長性を実装した可用性
- AWS for Data
- 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
- AWS における Account-per-Tenant 型 SaaS 環境のライフサイクル管理
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- 「Service Quotas 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
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

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。その中には、複数のクラウド環境 (パブリックアクセスとプライベートの両方) や、場合によっては既存のデータセンターのインフラストラクチャが含まれます。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スペースにおいて、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みなします。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を活用し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

クトが許可され、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、可用性と拡張性に優れたドメインネームシステム (DNS) のウェブサービスです。Route 53 は、ユーザーリクエストを 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接続します。詳細については、「Amazon Route 53 を DNS サービスとして設定する」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. ヘルスチェックのステータス監視と通知の受信
   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は、ロードバランサーのリージョンエンドポイントを指す Amazon Route 53 を使用して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、[ターゲットの正常性の評価] オプションを [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、「how AWS WAF works」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、「Getting started with AWS WAF」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、「how AWS WAF works with Amazon CloudFront features」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、「examples of DDoS resilient architectures」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、「Getting started with AWS Shield Advanced」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
- AWS Global Accelerator とは
- Amazon CloudFront とは何ですか?
- Amazon Route 53 とは?
- Elastic Load Balancing とは?
- 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
- Amazon Application Recovery Controller とは
- DNS フェイルオーバーのカスタムヘルスチェックの設定

関連動画:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Building resilient networks

関連する例:

- Amazon Application Recovery Controller (ARC) を使用したディザスタリカバリ
- Reliability Workshops
- AWS Global Accelerator Workshop

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

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接続に冗長性を実装して、接続の耐障害性を実現します。これは、ネットワーク障害発生時にも接続を維持できるように、2 つ以上のリンクとトラフィックパスをデプロイすることで可能になります。

一般的なアンチパターン:

- 単一のネットワーク接続に依存することで単一障害点が発生する。
- 1 つの VPN トンネルのみを使用するか、同じアベイラビリティーゾーンで終端する複数のトンネルを使用する。
- 1 社の ISP に VPN 接続を依存しているため、ISP が停止すると完全なネットワーク障害につながる可能性がある。
- ネットワーク中断時のトラフィック再ルーティングに不可欠な BGP などの動的ルーティングプロトコルを実装していない。
- VPN トンネルの帯域幅制限を無視し、バックアップ機能を過大評価している。

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により、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 Direct Connect を使用してオンプレミスネットワークを AWS に接続する場合、複数のオンプレミスの口ケーションと複数の AWS Direct Connect の口ケーションにある個別のデバイスで終端する個別の接続を使用して、ネットワークの耐障害性を最大限に高める (99.99% の SLA) ことができます。このトポロジは、デバイス障害、接続問題、および口ケーション全体での停止に対する耐障害性を提供します。または、複数の口ケーション (各オンプレミスの口ケーションを 1 つの Direct Connect 口ケーションに接続) への 2 つの個別の接続を使用することで、高い耐障害性 (99.9% の SLA) を実現できます。このアプローチにより、ファイバーの切断やデバイスの障害による接続の中断を防ぎ、口ケーション全体での障害を軽減できます。AWS Direct Connect Resiliency Toolkit は、AWS Direct Connect トポロジの設計に役立ちます。

プライマリ AWS Direct Connect 接続に対する費用対効果の高いバックアップとして、AWS Transit Gateway で終端する AWS Site-to-Site VPN を検討することもできます。この設定により、複数の VPN トンネルでの等コストマルチパス (ECMP) ルーティングが可能になり、各 VPN トンネルの上限が 1.25 Gbps であっても、最大 50 Gbps のスループットが可能になります。ただし、ネットワークの中断を最小限に抑え、安定した接続を提供するには、AWS Direct Connect が依然として最も効果的な選択肢であることに注意してください。

インターネット経由で VPN を使用してクラウド環境をオンプレミスのデータセンターに接続する場合は、単一のサイト間 VPN 接続の一部として 2 つの VPN トンネルを設定します。高可用性を実現

するために、各トンネルが異なるアベイラビリティーゾーンで終端するようにし、オンプレミスデバイスの障害を防ぐために冗長ハードウェアを使用する必要があります。さらに、1 社のインターネットサービスプロバイダー (ISP) の停止によって VPN 接続が完全に中断してしまうことを防ぐために、オンプレミスのロケーションで異なる ISP による複数のインターネット接続を利用することを検討してください。ルーティングとインフラストラクチャが多様な ISP、特に AWS エンドポイントに複数の物理パスがある ISP を選択すると、高い接続可用性が得られます。

複数の AWS Direct Connect 接続と複数の VPN トンネル (または両方の組み合わせ) による物理的な冗長性に加え、ボーダーゲートウェイプロトコル (BGP) の動的ルーティングを実装することも重要です。動的 BGP は、リアルタイムのネットワーク状態と設定されているポリシーに基づいて、1 つのパスから別のパスにトラフィックを自動的に再ルーティングします。この動的な動作は、リンクまたはネットワーク障害の発生時にネットワークの可用性とサービスの継続性を維持するうえで特に役立ちます。代替パスが瞬時に選択されるため、ネットワークの耐障害性と信頼性が高まります。

実装手順

- AWS とオンプレミス環境間に可用性の高い接続を確保します。
  - 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。
  - 複数の AWS Direct Connect ロケーションを使用して、高い可用性を確保します。
  - 複数の AWS リージョンを使用している場合は、2 つ以上のリージョンで冗長性を確保します。
- 可能な場合は AWS Transit Gateway を使用して、VPN 接続 を終端します。
- AWS Marketplace アプライアンスを評価して VPN を終端するか、SD-WAN を AWS に拡張します。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します。
  - 可用性のニーズを満たすために、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 Resiliency Toolkit を使用して開始します。

リソース

関連ドキュメント:

- AWS Direct Connect の耐障害性に関するレコメンデーション
- 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

- ルーティングポリシーと BGP コミュニティ
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- 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 Connectivity Options ホワイトペーパー
- スケーラブルでセキュアなマルチ VPC の AWS ネットワークインフラストラクチャの構築
- 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
- AWS Direct Connect Resiliency Toolkit を使用して開始する
- VPC エンドポイントおよび VPC エンドポイントサービス (AWS PrivateLink)
- Amazon VPC とは?
- Transit Gateway とは
- What is AWS Site-to-Site VPN?
- Direct Connect ゲートウェイの操作

関連動画:

- AWS re:Invent 2018: Amazon VPC 向けの高度な VPC 設計と新機能
- AWS re:Invent 2019: 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-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 の IP アドレス範囲は、将来の拡張やアベイラビリティーゾーン間でのサブネットへの IP アドレスの割り当てを考慮して、ワークロードの要件を満たすための十分な大きさが必要です。これには、ロードバランサー、EC2 インスタンス、コンテナ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 アドレススペース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 アドレススペースを割り当てます。
- VPC 内で複数のアベイラビリティーゾーンを網羅できるように、複数のサブネット用のスペースを確保します。

- 将来の拡張のために、未使用の CIDR ブロックスペースを VPC 内に残しておくことを検討します。
- 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 Amazon 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 アドレススペースがあることを確認します。各 Kubernetes ポッドにはデフォルトで VPC CIDR ブロックからルーティング可能なアドレスが割り当てられるため、Amazon Elastic Kubernetes Service (Amazon EKS) などの Kubernetes クラスターについても同様の検討が必要です。
- 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、使用できません。
- 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できませんが、重複していない CIDR ブロックは VPC に追加できます。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。
- 利用可能な VPC CIDR ブロックの最大サイズは /16、最小サイズは /28 です。
- 他の接続ネットワーク (VPC、オンプレミス、その他のクラウドプロバイダー) を検討し、IP アドレススペースが重複しないようにしてください。詳細については、「REL02-BP05接続されているすべてのプライベートアドレススペースにおいて、重複しないプライベート IP アドレス範囲を指定する」を参照してください。

期待できる成果: IP サブネットをスケールできるため、将来の成長に対応し、無駄を回避できます。

一般的なアンチパターン:

- 将来の成長が考慮されていないため、CIDR ブロックが小さすぎて再構成が必要になり、ダウンタイムが発生する可能性がある。
- Elastic Load Balancer 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。
- IP アドレスの消費をモニタリングできない状態で、自動スケーリングメカニズムを使用する。
- 将来の成長予測をはるかに超える過剰に大きな CIDR 範囲を定義したせいで、アドレス範囲が重複する他のネットワークとのピアリングが困難になる可能性がある。

このベストプラクティスを活用するメリット: 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引き続き可用性を提供できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

拡張、規制コンプライアンス、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実装が困難に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、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
  - 規制要件のためにネットワークの分離が必要かどうかを判断します。
  - 現在および将来のニーズに合わせて、適切なサイズの CIDR ブロックを持つ VPC を作成します。
    - 成長予測が不確かな場合は、将来の再構成のリスクを軽減するために、大きめの CIDR ブロックを選択しておいた方がよいでしょう。
  - デュアルスタック VPC の一部として、サブネットの IPv6 アドレス指定を使用することを検討してください。IPv6 は、多数の IPv4 アドレスが必要となる一時的なインスタンスやコンテナのフリートを含む、プライベートサブネットでの使用に適しています。

リソース

関連する Well-Architected のベストプラクティス:

- REL02-BP05 接続されているすべてのプライベートアドレススペースにおいて、重複しないプライベート IP アドレス範囲を指定する

関連ドキュメント:

- 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 Connectivity Options ホワイトペーパー
- 複数のデータセンターの HA ネットワーク接続

- 単一リージョンの複数 VPC 接続
- Amazon VPC とは?
- IPv6 on AWS
- リファレンスアーキテクチャの IPv6
- Amazon EKS が IPv6 サポートを開始
- VPC に関する推奨事項 - Classic Load Balancer
- アベイラビリティーゾーンサブネット - Application Load Balancer
- アベイラビリティーゾーン - Network Load Balancer

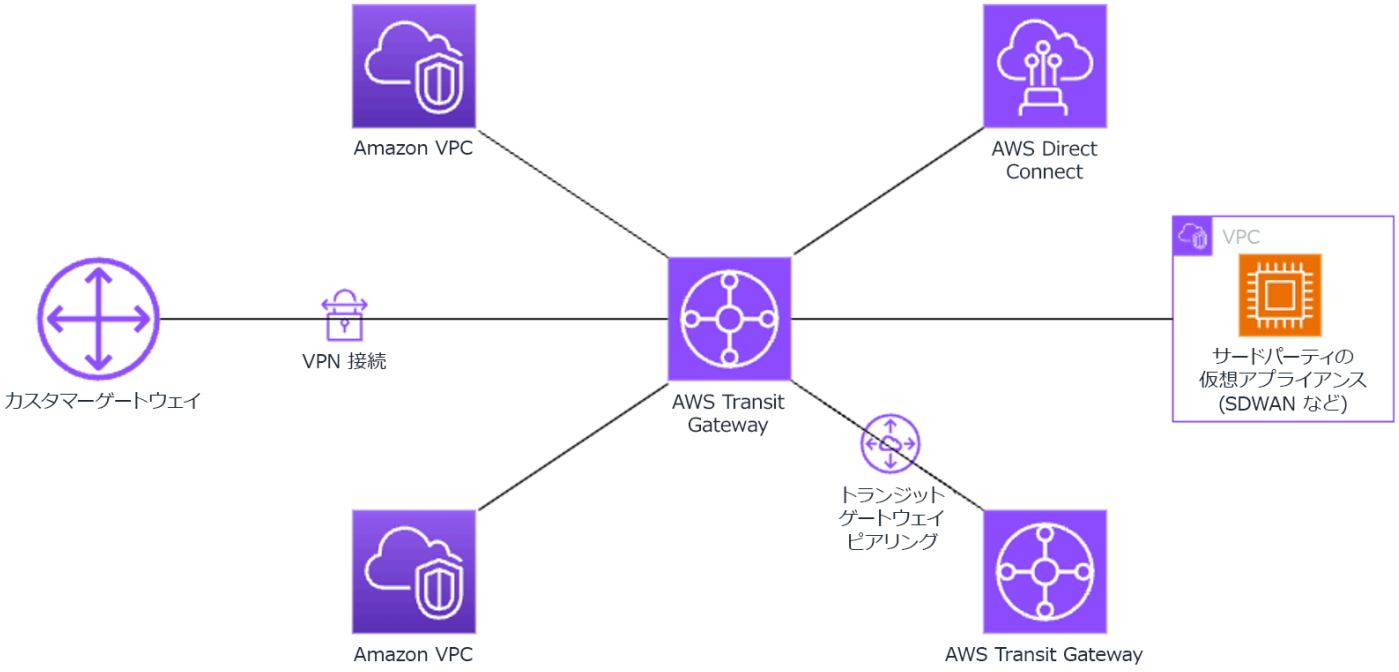
関連動画:

- AWS re:Invent 2018: Amazon VPC 向けの高度な VPC 設計と新機能 (NET303)
- AWS re:Invent 2019: 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 (NET406-R1)
- AWS re:Invent 2023: AWS の新機能のご紹介 成長と柔軟性を考慮したネットワーク設計 (NET310)

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

仮想プライベートクラウド (VPC) やオンプレミスネットワークなど、複数のプライベートネットワークを接続する場合は、メッシュトポロジではなくハブアンドスポークトポロジを選択します。各ネットワークが互いに直接接続され、複雑さと管理オーバーヘッドが増加するメッシュトポロジとは異なり、ハブアンドスポークアーキテクチャでは、接続が 1 つのハブを介して一元化されます。この一元化により、ネットワーク構造が簡素化され、運用性、スケーラビリティ、コントロールが強化されます。

AWS Transit Gateway は、AWS でハブアンドスポークネットワークを構築するために設計された、スケーラブルで可用性の高いマネージドサービスです。このサービスは、ネットワークのセントラルハブとして機能し、ネットワークのセグメンテーション、一元化されたルーティング、クラウド環境とオンプレミス環境両方へのシンプルな接続を提供します。次の図は、AWS Transit Gateway を使用してハブアンドスポークトポロジを構築する方法を示しています。



一般的なアンチパターン:

- ハブアンドスポークアーキテクチャでは、ルーティングポリシーが複雑になりすぎると、ネットワークの効率が低下し、トラブルシューティングと事前管理の両方が複雑になります。
- ハブ内のルーティングベースのセグメンテーションが不十分な場合、脆弱性が発生し、ネットワークが不正アクセスにさらされる可能性があります。
- 特にアベイラビリティーゾーンやリージョンを横断するトラフィックの場合、ハブを経由するトラフィックのデータ転送コストが高くなる可能性があるため、慎重な最適化が必要です。コストを抑えるには、効果的なトラフィック管理戦略が不可欠です。

このベストプラクティスを活用するメリット: 接続するネットワークの数が増えるにつれて、メッシュ接続の管理と拡張はますます困難になります。AWS Transit Gateway は、ハブアンドスポークトポロジの構築と運用のための、スケーラブルで信頼性の高いマネージドハブを提供します。AWS Transit Gateway を使用すると、接続を確立し、複数のネットワークにわたるトラフィックルーティングを一元化できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

- ネットワークを計画します。

- AWS Transit Gateway を作成します。
- VPC をアタッチします。
- 必要に応じて、VPN 接続または Direct Connect ゲートウェイを作成し、それらを Transit Gateway に関連付けます。
- Transit Gateway のルートテーブルを設定して、接続されている VPC と他の接続間のトラフィックのルーティング方法を定義します。
- パフォーマンスとコストを最適化するために Amazon CloudWatch を使用し、必要に応じて構成のモニタリングと調整を行います。

リソース

関連ドキュメント:

- Transit Gateway とは
- スケーラブルでセキュアなマルチ VPC の AWS ネットワークインフラストラクチャの構築
- AWS Transit Gateway のリージョン間ピアリングを使用したグローバルネットワークの構築
- Amazon Virtual Private Cloud の接続オプション
- 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

関連動画:

- AWS re:Invent 2023 - AWS ネットワーキングの基礎
- AWS re:Invent 2023 – 高度な VPC 設計と新機能

REL02-BP05 接続されているすべてのプライベートアドレススペースにおいて、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、Transit Gateway 経由での接続時、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。VPC とオンプレミス環境間の、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してください。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。これを自動化するには、IP アドレス管理 (IPAM) システムが役立ちます。

期待される成果:

- VPC、オンプレミス環境、または他のクラウドプロバイダー間で IP アドレス範囲が競合しません。
- 適切に IP アドレスを管理することで、ネットワークインフラストラクチャを容易にスケールし、規模の拡大やネットワーク要件の変更に対応できるようになります。

一般的なアンチパターン:

- オンプレミス、社内ネットワーク、または他のクラウドプロバイダーにあるものと同じ 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。
- スプレッドシートなど、手動の IP アドレス管理プロセスに依存する。
- CIDR ブロックサイズの過剰/過小なサイズ設定によって無駄な IP アドレスが発生する、またはワークロード用のアドレススペースに不足が発生する。

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を活用しない場合のリスクレベル: 中

実装のガイダンス

Amazon VPC IP Address Manager などの IPAM を使用して、CIDR の使用をモニタリングおよび管理します。AWS Marketplace は、複数の IPAM を提供しています。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。

実装手順

- 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。
  - Amazon VPC IP Address Manager を使用して、リソースを検出します。
- 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。
  - Amazon VPC IP Address Manager を使用して、リソースを検出します。

- 現在の使用量を記録します。
- 重複する IP 範囲を作成したかどうかを確認します。
- 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。新しいアドレス範囲に移行するか、重複する範囲を接続する必要がある場合は、プライベート NAT ゲートウェイや AWS PrivateLink などの手法を使用することを検討します。

リソース

関連するベストプラクティス:

- ネットワークの保護

関連ドキュメント:

- 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 Connectivity Options ホワイトペーパー
- 複数のデータセンターの HA ネットワーク接続
- Connecting Networks with Overlapping IP Ranges
- Amazon VPC とは?
- IPAM とは

関連動画:

- AWS re:Invent 2023 - 高度な VPC 設計と新機能
- AWS re:Invent 2019: 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
- AWS re:Invent 2023 - Ready for what's next? 成長と柔軟性を考慮したネットワーク設計
- AWS re:Invent 2021 - {New Launch} Manage your IP addresses at scale on AWS

ワークロードアーキテクチャ

Questions

- REL 3. どのようにワークロードサービスアーキテクチャを設計すればよいですか?
- REL 4. 障害を防ぐために、分散システムの操作をどのように設計しますか?
- REL 5. 障害を軽減または耐えるために、分散システムにおけるインタラクションをどのように設計しますか。

REL 3. どのようにワークロードサービスアーキテクチャを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、スケーラビリティ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検討します。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。

期待できる成果: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必要があります。

ワークロードのセグメント化方法を選択する場合は、複雑さとメリットのバランスを考慮してください。新製品のローンチ時に適しているものは、最初からスケールするように構築されたワークロードが必要とするものとは異なります。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。サービスをより細かく分割すると、明確に定義された小規模なチームがサービスを開発し、管理できるようになります。ただし、サービスが細かくなると、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。

このベストプラクティスを活用するメリット:

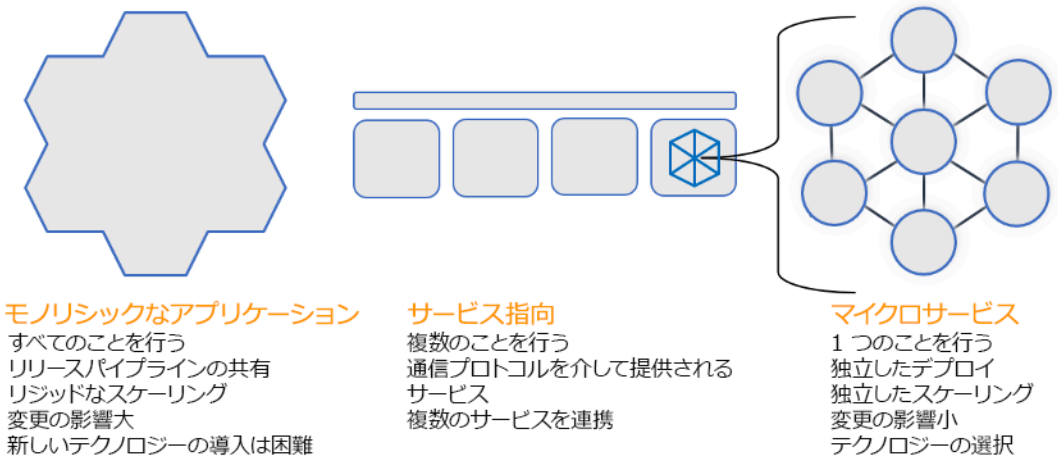
- より特化したセグメントは、高い俊敏性、組織の柔軟性、およびスケーラビリティにつながります。
- サービス中断の影響が小さくなります。
- アプリケーションコンポーネントには異なる可用性要件があり、より特化したセグメント化によってサポートすることができます。
- ワークロードをサポートするチームの責任が明確に定義されます。

この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 (まれにモノリシックアーキテクチャ) を選択します。モノリシック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、それぞれより細か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デプロイする際は、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がさらに複雑になることが挙げられます。AWS X-Ray をこの問題の解決に役立てることができます。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デプロイする必要があるため、運用が複雑になることも考慮する必要があります。



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、それぞれがより細かなセグメント化を実現するため、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。SOA は、マイクロサービスの複雑さを回避しながら、より細か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を参照してください。
- 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る必要があります。詳細については、AWS でのマイクロサービスの実装を参照してください。
- モノリスを細かなコンポーネントにリファクタリングするには、Strangler Fig のパターンに従うことを検討してください。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や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、ストラングラーパターンを使用して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には、これらの分散サービスと相互に通信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 はサービス指向アーキテクチャで使用することができ、信頼性の高いサービスの検出とアクセス性を提供します。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 に移行する場合、Amazon MQ は、クラウド内のレガシーアプリケーションを再設計するときに、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細か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:

- イテレーティブアプリモダナイゼーションワークショップ

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、疎結合するサービスとして設計されます。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。サービスは需要の変動に個別に対応し、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:

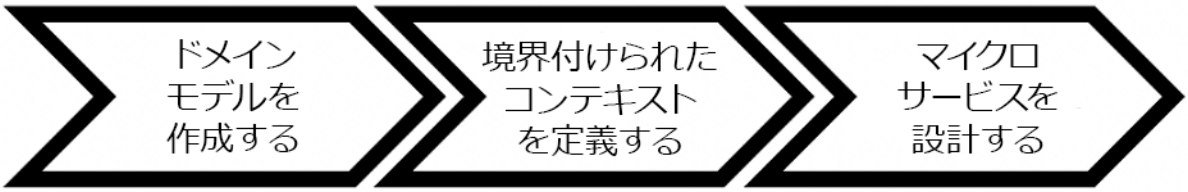
- チームは、特定のビジネスドメインではなく、UI や UX、ミドルウェア、データベースなどの特定の技術ドメインを中心に形成される。
- アプリケーションがドメインの担当範囲にまたがっている。限定されたコンテキストにまたがるサービスは、メンテナンスが難しく、大規模なテスト作業が必要になり、複数のドメインチームがソフトウェア更新に参加する必要がある。
- ドメインエンティティライブラリと同様に、ドメイン依存関係がサービス間で共有されるため、あるサービスドメインを変更すると、他のサービスドメインも変更する必要がある。
- サービス契約とビジネスロジックはエンティティを共通かつ一貫したドメイン言語で表現していないため、翻訳層が発生し、システムが複雑になり、デバッグ作業が増加する。

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個別の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。サービスは個別に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回復力要件を満たします。

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

ドメイン駆動型設計 (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、サービスにすることができます。



ドメイン駆動型設計

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。
- ドメインエンティティと関数がドメインコンテキストで形成されたら、境界コンテキストを使用してドメインをサービスに分割できます。境界コンテキストでは、類似の特徴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すると、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。
- モノリスをマイクロサービスに分解すると、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が概説されます。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、ドメイン駆動のアプローチに適しています。
- バブルコンテキストなどの戦術的な手法を使用すると、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、既存のアプリケーション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する破損防止層を使用して、小さな境界コンテキストが確立されます。

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、ドメイン駆動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。テスト駆動型開発 (TDD) と動作駆動型開発 (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 のサービスを選択します。
  - 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の管理ではなく、特定のドメインロジックに集中できます。
  - AWS のコンテナを使用すると、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 にヘキサゴナルアーキテクチャを構築することで、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、機能要件を満たして統合アダプターをアタッチするための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
- エリック・エヴァンスのドメイン駆動設計: ソフトウェアの核心にある複雑さに立ち向かう
- AWS でヘキサゴナルアーキテクチャを構築する

- マイクロサービスへのモノリスの分解
- 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
- 動作駆動型開発

関連する例:

- 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: 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ルインしなくても、吸収するリソース需要を個別にスケールできます。

一般的なアンチパターン:

- 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。
- バージョニング戦略を採用していないため、API コンシューマーに更新とリリースを強制します。または、サービス契約が進化するときに失敗します。
- 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。
- API 契約を使用せずにテストケースを開発し、モック API 実装を使用しないことで、サービスコンポーネントを個別にテストできます。

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た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。開発者は、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、および必須フィールドが存在することを確認し、開発プロセスの早期に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み、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

Amazon API Gateway、AWS AppSync、Amazon EventBridge などの AWS のサービスをアーキテクチャに組み込み、アプリケーションで API サービス契約を使用します。Amazon API Gateway は、ネイティブ AWS サービスやその他のウェブ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、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 スキーマを定義して設定する、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と、API の作成が簡素化され、AWS Serverless Application Model や AWS Cloud Development Kit (AWS CDK) などのコードツールとしての AWS インフラストラクチャと統合できます。
  - 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。
- AWS AppSync で GraphQL スキーマファイルを定義して、GraphQL API を定義して管理し、コントラクトインターフェイスを生成して、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できます。
- AWS AppSync と統合された AWS Amplify プロジェクトは、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、Amazon DynamoDB テーブル用の AWS AppSync GraphQL クライアントライブラリを生成します。
- 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、契約拡張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したりするファサードを確立し、契約の変更の吸収を簡素化できます。このアプローチにより、基盤となるサービスが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 への API Gateway 拡張機能の使用
- OpenAPI 仕様
- GraphQL: スキーマとタイプ
- Amazon EventBridge のコードバインディング

関連する例:

- Amazon API Gateway: OpenAPI を使用した REST API の設定
- Amazon API Gateway to Amazon DynamoDB CRUD application using OpenAPI
- 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 サービスの統合
- Amazon CloudFront でのヘッダーベースの 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をビルドする

関連動画:

- Using OpenAPI in AWS SAM to manage API Gateway

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に依存しています。これらのネットワークでデータ損失や遅延が発生しても、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及ぼさ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障害を防ぎ、平均故障間隔 (MTBF) を改善します。

ベストプラクティス

- REL04-BP01 依存している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 すべての応答にべき等性を持たせる

REL04-BP01 依存している分散システムの種類を特定する

分散システムには、同期、非同期、またはバッチの 3 つの種類があります。同期システムは、リクエストを可能な限り迅速に処理し、HTTP/S、REST、またはリモートプロシージャコール (RPC) プロトコルを使用して同期リクエスト呼び出しと応答呼び出しを行うことで相互に通信する必要があります。非同期システムは、個々のシステムを結合することなく、中間サービスを介して非同期的にデータを交換することによって相互に通信します。バッチシステムは大量の入力データを受け取り、人の介入なしに自動データプロセスを実行し、出力データを生成します。

望ましい結果: 同期、非同期、バッチの依存関係と効果的に相互作用するワークロードを設計します。

一般的なアンチパターン:

- ワークロードは依存関係からの応答を無期限に待つため、リクエストが受信されたかどうかが不明なまま、ワークロードクライアントがタイムアウトすることがあります。

- ワークロードは、相互に同期呼び出しを行う一連の依存システムを使用しています。これには、チェーン全体で正常に処理が行われる前に、各システムが利用可能な状態にあり、システムでリクエストを正常に処理する必要があるため、動作が脆弱になり、全体的な可用性が損なわれる可能性があります。
- ワークロードは依存関係と非同期で通信し、重複したメッセージを受信する場合が多いのにもかかわらず、1 回のみのメッセージ処理が保証されるという概念に基づいています。
- 適切なバッチスケジューリングツールを使用していないため、ワークロードは同じバッチジョブを同時に実行します。

このベストプラクティスを活用する利点: 特定のワークロードでは、同期、非同期、バッチのいずれかの通信スタイルを 1 つ以上実装するのが一般的です。このベストプラクティスは、それぞれの通信スタイルに関連するさまざまなトレードオフを特定し、ワークロードが依存関係の中断に耐えられるように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

以下のセクションでは、各種類の依存関係に関する一般的な実装ガイダンスと固有の実装ガイダンスの両方について説明します。

一般的な質問、または機能要望

- 依存関係が提供するパフォーマンスと信頼性のサービスレベル目標 (SLO) が、ワークロードのパフォーマンスと信頼性の要件を満たしていることを確認します。
- AWS オブザーバビリティサービスを使用して応答時間とエラー率をモニタリングし、依存関係がワークロードに必要なレベルでサービスを提供していることを確認します。
- 依存関係と通信する際に、ワークロードが直面する可能性のある課題を特定します。分散システムには、アーキテクチャの複雑さ、運用上の負担、コストが増加する可能性のあるさまざまな課題があります。一般的な課題には、レイテンシー、ネットワークの中断、データ損失、スケーリング、データ複製の遅延などがあります。
- 堅牢なエラー処理とログ記録を実装して、依存関係で問題が発生した場合の問題のトラブルシューティングに役立ててください。

同期依存関係

同期通信では、ワークロードは依存関係にリクエストを送信し、応答を待っている操作をブロックします。依存関係がリクエストを受け取ると、すぐに処理を試み、応答をワークロードに送り返します。同期通信の大きな課題は、一時的な結合が発生するため、ワークロードとその依存関係を同時に利用できるようにする必要があることです。ワークロードで依存関係との同期通信が必要な場合は、以下のガイダンスを検討します。

- ワークロードが 1 つの機能を実行するために複数の同期依存関係に依存するべきではありません。リクエストを正常に完了させるためにパス内のすべての依存関係が利用可能である必要があるため、依存関係の連鎖は全体的な脆弱性を高めます。
- 依存関係が正常でない場合や利用できない場合のエラー処理と再試行戦略を決定します。バイモーダル動作の使用は避けてください。バイモーダル動作とは、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。バイモーダル動作の詳細については、「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」を参照してください。
- ワークロードを待機させるより、フェイルファストの方がよいことを覚えておいてください。例えば、「AWS Lambda デベロッパーガイド」では、Lambda 関数を呼び出すときに再試行や失敗を処理する方法について説明します。
- ワークロードが依存関係を呼び出す際のタイムアウトを設定します。これにより、応答を待つ時間が長すぎたり、無期限に待ったりすることを回避できます。このトピックに関する役立つ説明は、「レイテンシーを考慮した Amazon DynamoDB アプリケーションのための AWS Java SDK HTTP リクエスト設定のチューニング」に記載されています。
- 1 つのリクエストを処理するためのワークロードから依存関係への呼び出し回数を最小限に抑えます。呼び出し回数の多さは、結合とレイテンシーの増加につながります。

非同期依存関係

ワークロードを依存関係から一時的に切り離すには、非同期で通信する必要があります。非同期アプローチを使用すると、ワークロードの依存関係や一連の依存関係からの応答の送信を待つことなく、他の処理を実行できます。

ワークロードで依存関係との非同期通信が必要な場合は、以下のガイダンスを検討します。

- ユースケースと要件に基づいて、メッセージングとイベントストリーミングのどちらを使用するかを決定します。メッセージングを使用すると、メッセージブローカーを介してメッセージを送受信することで、ワークロードが依存関係と通信できます。イベントストリーミングを使用すると、ワークロードとその依存関係は、ストリーミングサービスを使用して、できるだけ早く処理する必要のあるデータを継続的なストリームとして配信されるイベントを公開およびサブスクライブできます。

- メッセージングとイベントストリーミングではメッセージの処理方法が異なるため、以下に基づいてトレードオフを決定する必要があります。
  - メッセージ優先度: メッセージブローカーは、通常のメッセージよりも先に優先度の高いメッセージを処理できます。イベントストリーミングでは、すべてのメッセージの優先度が同じになります。
  - メッセージ消費: メッセージブローカーは、コンシューマーがメッセージを受信したかどうか確認します。イベントストリーミングのコンシューマーは、最後に読んだメッセージを常に把握しておく必要があります。
  - メッセージの順序: メッセージングでは、先入れ先出し (FIFO) アプローチを使用しない限り、メッセージの送信順序を正確に受信することは保証されません。イベントストリーミングでは、データが生成された順序が常に保持されます。
  - メッセージの削除: メッセージングでは、コンシューマーはメッセージを処理した後に削除する必要があります。イベントストリーミングサービスはメッセージをストリームに追加し、メッセージの保存期間が終了するまでストリームに残ります。この削除ポリシーにより、イベントストリーミングはメッセージの再生に適したものになります。
- 依存関係がいつ作業を完了したかをワークロードがどのように認識するかを定義します。ワークロードが Lambda 関数を非同期的に呼び出すと、Lambda はリクエストをキューに入れ、追加情報を含まない成功のレスポンスを返します。処理が完了すると、Lambda 関数は成功または失敗に基づいて設定可能な送信先に結果を送信できます。
- べき等性を活用して、重複メッセージを処理するワークロードを構築します。べき等性とは、同じメッセージに対してワークロードが複数回生成されても、ワークロードの結果は変化しないことを指します。ネットワーク障害が発生した場合、または確認応答が受信されていない場合、メッセージングサービスまたはストリーミングサービスがメッセージを再配信することに注意してください。
- ワークロードが依存関係から応答を受け取らない場合は、ワークロードはリクエストを再送信します。リトライ回数を制限して、ワークロードの CPU、メモリ、ネットワークリソースの消費を抑え、他のリクエストを処理できるようにすることを検討してください。「AWS Lambda ドキュメント」では、非同期呼び出しのエラーを処理する方法を示しています。
- 適切なオブザーバビリティ、デバッグ、トレースツールを活用して、ワークロードの非同期通信とその依存関係を管理し運用します。Amazon CloudWatch を使用して、メッセージングおよびイベントストリーミングサービスをモニタリングできます。また、AWS X-Ray を使用してワークロードを計測し、問題のトラブルシューティングに関するインサイトをすばやく得ることもできます。

バッチ依存関係

バッチシステムは、手動操作なしで、入力データを受け取り、処理するための一連のジョブを開始し、いくつかの出力データを生成します。データサイズにもよりますが、ジョブは数分から、場合によっては数日かかることもあります。ワークロードで依存関係とのバッチ通信を行う場合は、以下のガイダンスを検討します。

- ワークロードでバッチジョブを実行する時間枠を定義します。ワークロードでは、例えば 1 時間ごとまたは月末に、バッチシステムを呼び出す繰り返しパターンを設定できます。
- データ入力と処理済みデータ出力の場所を定義します。Amazon Simple Storage Service (Amazon S3)、Amazon Elastic File System (Amazon EFS)、Amazon FSx for Lustre などのストレージサービスを使用すると、大規模なワークロードのファイル読み書きに対応できます。
- ワークロードで複数のバッチジョブを呼び出す必要がある場合は、AWS Step Functions を活用して、AWS またはオンプレミスで実行されるバッチジョブのオーケストレーションを簡素化できます。このサンプルプロジェクトは、Step Functions、AWS Batch、および Lambda を使用したバッチジョブのオーケストレーションを示しています。
- バッチジョブをモニタリングして、ジョブの完了に本来よりも時間がかかっているなどの異常がないかを確認します。CloudWatch Container Insights などのツールを使用して、AWS Batch 環境やジョブをモニタリングできます。この場合、ワークロードによって次のジョブの開始が停止し、関連するスタッフに例外が通知されます。

リソース

関連ドキュメント:

- AWS クラウド Operations: モニタリングとオブザーバビリティ
- Amazon Builders' Library: 分散システムの課題
- 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
- AWS Lambda デベロッパーガイド: AWS Lambda でのエラー処理と自動再試行
- レイテンシーを考慮した Amazon DynamoDB アプリケーションのための AWS Java SDK HTTP リクエスト設定のチューニング
- AWS メッセージング
- ストリーミングデータとは
- AWS Lambda デベロッパーガイド: 非同期呼び出し
- Amazon Simple Queue Service に関するよくある質問: FIFO キュー
- Amazon Kinesis Data Streams デベロッパーガイド: 重複レコードの処理

- Amazon Simple Queue Service デベロッパーガイド: Amazon SQS で使用できる CloudWatch メトリクス
- Amazon Kinesis Data Streams デベロッパーガイド: Amazon CloudWatch による Amazon Kinesis Data Streams Service のモニタリング
- AWS X-Ray デベロッパーガイド: AWS X-Rayの概念
- GitHub の AWS サンプル: AWS Step Functions 複雑なオーケストレーターアプリ
- AWS Batch ユーザーガイド: AWS Batch CloudWatch Container Insights

関連動画:

- AWS Summit SF 2022 - AWS によるフルスタックの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)

関連ツール:

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Service (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx for Lustre
- AWS Step Functions
- AWS Batch

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、疎結合されています。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。

キューイングシステム、ストリーミングシステム、ワークフローなどの依存関係を疎結合化すると、システムへの変更や障害の影響を最小限に抑えることができます。疎結合化により、コンポーネントの動作が依存する他のシステムに影響を与えないように分離され、回復力と俊敏性が向上します。

密結合のシステムでは、あるコンポーネントを変更すると、そのコンポーネントに依存する他のコンポーネントも変更しなければならなくなり、結果として、すべてのコンポーネントのパフォーマンスが低下する可能性があります。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隔離することができます。

疎結合では、コードの変更やコンポーネントへの機能の追加を自由にできる一方で、そのコンポーネントに依存する他のコンポーネントへのリスクを最小限に抑えることができます。また、回復力を細分化でき、コンポーネントレベルでスケールアウトしたり、依存関係の根本的な実装さえも変更したりできます。

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、Amazon 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: 疎結合された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

Amazon SQS キューと AWS Step Functions 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。Amazon EventBridge を使用してイベント駆動型アーキテクチャを AWS クラウドに構築することもできます。Amazon EventBridge は、クライアント (イベントプロデューサー) が依存するサービス (イベントコンシューマー) から抽象化できます。Amazon Simple Notification Service (Amazon SNS) は、高スループットの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みなされ (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。

期待される成果: 疎結合の依存関係を実装すると、コンポーネントレベルへの障害の面積を最小限に抑えることができ、問題の診断と解決に役立ちます。また、開発サイクルが簡素化され、チームはモジュールレベルで変更を実装できるようになり、その部分に依存する他のコンポーネントのパフォー

マンスに影響は及びません。このアプローチでは、リソースのニーズに基づいてコンポーネントレベルでスケールアウトできるだけでなく、コスト効率良くコンポーネントを活用できるようになります。

一般的なアンチパターン:

- モノリシックワークロードのデプロイ。
- 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。
- 共有データを使用した密結合。疎結合のシステムでは、共有データベースや他の形で密結合されたデータストレージを介したデータの共有を避ける必要があります。そうしたデータ共有が密結合を持ち込み、スケーラビリティを妨げる可能性があります。
- バックプレッシャーを無視する。ワークロードには、コンポーネントが同じ速度でデータを処理できない場合に、データの受信を遅らせたり停止したりする機能が必要です。

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隔離するのに役立ち、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隔離されます。

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。疎結合のアプリケーションを構築するためのさまざまなソリューションがあります。フルマネージド型のキュー、自動化されたワークフロー、イベントへの対応、API などを実装するサービスがこれに当たりますが、いずれも、コンポーネントの動作を他のコンポーネントから隔離するのに役立ち、弾力性と俊敏性を高めます。

- イベント駆動型アーキテクチャの構築: Amazon EventBridge は、疎結合で分散されたイベント駆動型アーキテクチャの構築に役立ちます。
- 分散システムにキューを実装する: Amazon Simple Queue Service (Amazon SQS) を使用して、分散システムを統合および疎結合化できます。
- コンポーネントをマイクロサービスとしてコンテナ化: マイクロサービスを使用すると、チームは十分に定義された API を通して通信する小型の独立コンポーネントから構成されるアプリケーションを構築できます。Amazon Elastic Container Service (Amazon ECS) および Amazon Elastic Kubernetes Service (Amazon EKS) は、コンテナ化をすばやく実現できます。

- Step Functions を使用してワークフローを管理する: Step Functions は、複数の AWS サービスを柔軟なワークフローに調整するのに役立ちます。
- パブリッシュ/サブスクライブ (pub/sub) メッセージングアーキテクチャを活用する: Amazon Simple Notification Service (Amazon SNS) は、パブリッシャーからサブスクライバー (プロデューサーおよびコンシューマーとも呼ばれます) へのメッセージ配信を提供します。

実装手順

- イベント駆動型アーキテクチャのコンポーネントは、イベントによって開始されます。イベントは、ユーザーがカートに商品を追加するなど、システム内で発生するアクションです。アクションが正常に実行されると、システムの次のコンポーネントを起動するイベントが生成されます。
  - Amazon EventBridge を使用したイベント駆動型アプリケーションの構築
  - AWS re:Invent 2022: Amazon EventBridge を使用したイベント駆動型統合の設計
- 分散型メッセージングシステムには、キューベースのアーキテクチャを確立するために主に 3 つの部分を実装する必要があります。これには、分散型システムのコンポーネント、分離のために使用するキュー (Amazon SQS サーバー上で分散される)、キュー内のメッセージが該当します。一般的なシステムには、キューへのメッセージ配信を開始するプロデューサーと、キューからメッセージを受信するコンシューマーがあります。キューは、冗長性を確保するために、複数の Amazon SQS サーバーにメッセージを格納します。
  - Amazon SQS の基本的なアーキテクチャ
  - Amazon Simple Queue Service を使用して分散アプリケーション間でメッセージを送信する
- マイクロサービスをうまく利用すれば、疎結合のコンポーネントが独立したチームによって管理されるため、保守性とスケーラビリティが向上します。また、変更が必要になっても、動作を分離し、単一のコンポーネントに限定できます。
  - AWS でのマイクロサービスの実装
  - Let's Architect! コンテナを使用するマイクロサービスの設計
- AWS Step Functions では、分散型アプリケーションの構築、プロセスの自動化、マイクロサービスのオーケストレーションなどを行うことができます。複数のコンポーネントをオーケストレーションして 1 つのワークフローとしてまとめ、自動化することで、アプリケーション内の依存関係を分離できます。
  - AWS Step Functions と AWS Lambda を使用してサーバーレスワークフローを作成する
  - AWS Step Functions の開始方法

リソース

関連ドキュメント:

- Amazon EC2: べき等性を保証する
- Amazon Builders' Library: 分散システムの課題
- Amazon Builders' Library: 信頼性、動作の継続、1 杯のおいしいコーヒー
- What Is Amazon EventBridge?
- Amazon Simple Queue Service とは
- モノリスから卒業する
- AWS Step Functions および Amazon SQS を使用してキューベースのマイクロサービスをオーケストレーションする
- Amazon SQS の基本的なアーキテクチャ
- キューベースのアーキテクチャ

関連動画:

- AWS New York Summit 2019: イベント駆動型アーキテクチャと Amazon EventBridge 入門 (MAD205)
- AWS re:Invent 2018: 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: イベント駆動型アーキテクチャへの移行 (SVS308)
- AWS re:Invent 2019: Amazon SQS と Lambda を使用するスケーラブルなサーバーレスイベント駆動型アプリケーション
- AWS re:Invent 2022: Amazon EventBridge を使用したイベント駆動型統合の設計
- AWS re:Invent 2017: Elastic Load Balancing の詳細とベストプラクティス

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。

例えば、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。しかし、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毎回現在の状態のフルスナップショットを送信する必要があります。それぞれがビットで表される 100,000 個のサーバーヘルス状態は、12.5 KB のペイロードにすぎません。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を活用し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、疎結合されています。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。
  - Amazon Builders' Library: 信頼性、動作の継続、1 杯のおいしいコーヒー
  - AWS re:Invent 2018: 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ヘルスチェック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

関連ドキュメント:

- Amazon EC2: べき等性を保証する
- Amazon Builders' Library: 分散システムの課題
- Amazon Builders' Library: 信頼性、動作の継続、1 杯のおいしいコーヒー

関連動画:

- AWS New York Summit 2019: イベント駆動型アーキテクチャと Amazon EventBridge 入門 (MAD205)

- AWS re:Invent 2018: 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: 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: イベント駆動型アーキテクチャへの移行 (SVS308)

REL04-BP04 すべての応答にべき等性を持たせる

べき等性のサービスは、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。べき等性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。べき等性サービス 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のは困難です。つまり、アクションは 1 回だけ実行されるため、同一のリクエストを複数回行っても、リクエストを 1 回行ったのと効果は同じです。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: 中

実装のガイダンス

- すべての応答にべき等性を持たせます。べき等性のサービスは、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。べき等性サービス 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: べき等性を保証する

リソース

関連ドキュメント:

- Amazon EC2: べき等性を保証する
- Amazon Builders' Library: 分散システムの課題
- Amazon Builders' Library: 信頼性、動作の継続、1 杯のおいしいコーヒー

関連動画:

- AWS New York Summit 2019: イベント駆動型アーキテクチャと Amazon EventBridge 入門 (MAD205)
- AWS re:Invent 2018: 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: イベント駆動型アーキテクチャへの移行 (SVS308)

REL 5. 障害を軽減または耐えるために、分散システムにおけるインタラクション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及ぼさ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、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、グレー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 可能な限りシステム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、グレー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提供できます。

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みなしてください。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顧客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、そのコア機能とみなすべき内容によって異なります。例:

- 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、上位ランクの商品、顧客の注文状況など、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、エラーページを顧客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しなければなりません。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。一般的には、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. 高負荷のためにダウンストリームシステムがリクエスト処理不可: ダウンストリームシステムへのリクエストが一貫して失敗している場合、再試行を続けることは意味がありません。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. パラメータストアが使用不可: 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または非機能的依存関係が停止: 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停止している可能性がある: Amazon Relational Database Service は、ほぼ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を 1 つだけ持つことができます。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 Serverless 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

関連ドキュメント:

- Amazon API Gateway: API リクエストを調整してスループットを向上させる
- Circuit Breaker (「Release It!」書籍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 著「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
- Amazon Builders' Library: 乗り越えられないキューバックログの回避
- The Amazon Builders' Library: キャッシングの課題と戦略
- The Amazon Builders' Library: ジッターを伴うタイムアウト、再試行、およびバックオフ

関連動画:

- Retry, backoff, and jitter: 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高める

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ますが、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。
- 水平方向にスケール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則ってトークンバケットアルゴリズムを実装します。使用プランに従ってクライアントごとに設定することが可能です。さらに、Amazon Simple Queue Service (Amazon SQS) と Amazon Kinesis を使用することで、リクエストをバッファリングしてリクエストレートを均衡にし、対処可能なリクエストのスロットリングレートを高めることができます。最後に、AWS WAF を使用してレート制限を実装することで、異常に高い負荷を発生させる特定の API コンシューマーをスロットリングします。

実装手順

API Gateway で API のスロットリング制限を設定し、制限を超過したときに「`429 Too Many Requests`」エラーを返すようにします。AWS AppSync および API Gateway エンドポイントで AWS WAF を使用すれば、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。

非同期処理の場合、Amazon SQS を AWS Lambda のイベントソースとして設定しているときは、最大同時実行数を設定することで、イベント率の上昇によって、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が消費されることを回避できます。

API Gateway ではトークンバケットのマネージド実装が行われますが、API Gateway を使用できない場合は、お使いの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」内の「関連する例」を参照) を利用できます。

- API Gateway のスロットリング制限は、リージョンごとのアカウントレベル、ステージごとの API、使用プランのレベルごとの API キーで理解し、設定します。
- 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のトリガーとして設定されているときは、最大同時実行数は、サービスレベルの目標達成に十分に対応できる値、かつ他の Lambda 関数に影響を与える同時実行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ます。
- API Gateway を、Amazon SQS または Kinesis とのネイティブサービス統合と共に使用して、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

関連ドキュメント:

- Amazon API Gateway: スループットを高めるために API リクエストをスロットリングする
- AWS WAF: レートベースのルールステートメント
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source

- AWS Lambda: 最大同実行数

関連する例:

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Python token-bucket
- Node token-bucket
- .NET System Threading Rate Limiting

関連動画:

- Implementing GraphQL API security best practices with AWS AppSync

関連ツール:

- Amazon API Gateway
- AWS AppSync
- Amazon Simple Queue Service
- 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 に組み込まれていたりす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、オブザーバビリティ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ー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。
- 再試行を行う前に、サービスがべき等性を持っていることを確認します。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、再試行の設定オプションを把握します。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:

- REL04-BP04 すべての応答にべき等性を持たせる
- 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
- Amazon Builders' Library: ジッターを伴うタイムアウト、再試行、およびバックオフ
- Exponential Backoff and Jitter
- Making retries safe with idempotent APIs

関連する例:

- Spring Retry
- Resilience4j Retry

関連動画:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

関連ツール:

- AWS SDKs and Tools: Retry behavior
- 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たりしない。
- 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: フェイルファストなシステムはデバッグや修正が容易で、多くの場合、リリースが本稼働環境にパブリッシュされる前に、コーディングや構成上の問題を明らかにすることができます。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、障害をモニタリングし、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 Amazon SQS やその他キューテクノロジーを使用してキューを実装し、負荷を軽減している場合、そのシステムは、キューのバックログやメッセージ使用の失敗の、管理方法を検討しておく必要があります。

実装手順

- プログラムによるアサーションまたは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 を使用すると、アプリケーションのログパターンや SDK の計測に基づいてメトリクスとアラームを作成することができます。
- 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たり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
- Fail Fast
- 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: Zonal Shift
- Amazon Application Recovery Controller: トラフィックフェイルオーバーのルーティングコントロール

関連する例:

- Enterprise Integration Patterns: Dead Letter Channel

関連動画:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

関連ツール:

- Amazon Simple Queue Service
- 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、いずれかを決定します。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。AWSSDK と AWS CLI を使用すると、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行えます。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 を併用すれば、ローコードのサーキットブレーカーを構築して、AWS サービスおよび SDK との事前構築済みの統合を活用できます。AWS App MeshEnvoy はタイムアウトとサーキットブレーカーの機能を備えています。

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 または AWS CLI コマンドを使用するときは、`connectTimeoutInMillis` と `tlsNegotiationTimeoutInMillis` の AWS 設定デフォルト を設定し、デフォルトのタイムアウト値を設定します。
- コマンドラインオプション の `cli-connect-timeout` と `cli-read-timeout` を適用して、AWS のサービスの 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。
- 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、レイテンシーの外れ値に関する CloudWatch メトリクスと CloudWatch 異常検出を実装すると、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。
- Lambda 関数でタイムアウトを設定します。
- 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に再試行を行う必要があります。API Gateway は、ダウンストリームの統合に対しては 50 ミリ秒から 29 秒までの統合タイムアウトをサポートしており、統合リクエストがタイムアウトしたときは再試行を行いません。
- 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の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については、「App Mesh Envoy」の機能を確認して、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します。

- AWS Step Functions を使用して、リモートサービスを呼び出すための (特に、ワークロードを簡素化する目的で AWS ネイティブの SDK と、サポートされている Step Functions 統合とを呼び出すための)、ローコードのサーキットブレーカーを作成します。

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

関連ドキュメント:

- AWS SDK: 再試行とタイムアウト
- Amazon Builders' Library: ジッターを伴うタイムアウト、再試行、およびバックオフ
- Amazon API Gateway のクォータと重要な注意点
- 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
- AWSBotocore using the config object and Config Reference
- AWS SDK for .NET: Retries and Timeouts
- AWS Lambda: AWS Lambda 関数の設定

関連する例:

- Using the circuit breaker pattern with AWS Step Functions and Amazon DynamoDB
- Martin Fowler: CircuitBreaker

関連ツール:

- AWS SDK
- AWS Lambda
- Amazon Simple Queue Service
- AWS Step Functions

- AWS Command Line Interface

REL05-BP06 可能な限りシステムをステートレスにする

状態を必要としないシステム、または状態をオフロードするシステム (異なるクライアントリクエスト間にディスクやメモリ内のローカルに保存されたデータへの依存がない) にしてください。これにより、可用性に影響を与えることなく、サーバーをいつでも置き換えることができます。

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、水平方向にスケールできることです。

このベストプラクティスを活用するメリット: ステートレスに設計されたシステムは水平スケーリングへの適応性が高いため、トラフィックや需要の変化に応じてキャパシティを増やしたり減らしたりすることが可能です。また、本質的に耐障害性に優れており、アプリケーション開発に柔軟性と俊敏性をもたらします。

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: 中

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、水平スケーリングが可能であり、個別ノードの障害に耐性があります。アーキテクチャ内の状態を維持するアプリケーションのコンポーネントを分析して理解します。これにより、ステートレス設計への移行で考えられる影響を評価できます。ステートレスアーキテクチャはユーザーデータを切り離し、セッションデータをオフロードします。各コンポーネントを個別にスケールし、変化するワークロードの需要に対応することでリソースの使用率を最適化できる柔軟性が得られます。

実装手順

- アプリケーション内のステートフルなコンポーネントを特定して理解します。

- ユーザーデータをコアアプリケーションロジックから分離して管理することで、データを切り離します。
  - Amazon Cognito では、アイデンティティプール、ユーザープール、Amazon Cognito Sync などの機能を使用することでユーザーデータをアプリケーションコードから切り離すことができます。
  - AWS Secrets Manager を使用すると、シークレットを一元化された安全な場所に保管することでユーザーデータを切り離すことができます。つまり、アプリケーションコードにシークレットを保存する必要がないため、安全性が高まります。
  - イメージやドキュメントなどの構造化されていない大規模なデータを保存するときは Amazon S3 の使用を検討します。アプリケーションは必要に応じてこのデータを取得できるため、メモリに保存する必要はありません。
  - ユーザープロファイルなどの情報を保存するときは Amazon DynamoDB を使用します。アプリケーションでは、ほぼリアルタイムでこのデータをクエリできます。
- セッションデータをデータベース、キャッシュ、または外部ファイルにオフロードします。
  - セッションデータのオフロードに使用できる AWS のサービスには、Amazon ElastiCache、Amazon DynamoDB、Amazon Elastic File System (Amazon EFS)、Amazon MemoryDB などがあります。
- 選択したストレージソリューションでは、どの状態とユーザーデータを持続しておく必要があるのかを特定した後、ステートレスアーキテクチャを設計します。

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

関連ドキュメント:

- Amazon Builders' Library: 分散システムでのフォールバックの回避
- Amazon Builders' Library: 乗り越えられないキューバックログの回避
- Amazon Builders' Library: キャッシングの課題と戦略
- AWS のステートレスなウェブ層

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。

緊急レバーは、既知のテスト済みのメカニズムを使用して、コンポーネントや依存関係の動作を無効にしたり、スロットリングしたり、変更したりするためのものです。その効果として、想定外の需要増によるリソースの枯渇が原因となるワークロードの障害を軽減し、ワークロード内の重要ではないコンポーネントの障害の波及を抑制できます。

期待される成果: 緊急レバーを実装することで、ワークロードに欠かせないコンポーネントの可用性を維持するための、問題がないことが確認されているプロセスを確立できます。緊急レバーが作動している間、ワークロードは意図的に性能を落とし (グレースフルデグラデーション)、ビジネスに不可欠な機能を引き続き実行します。グレースフルデグラデーションの詳細は、「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、グレースフルデグラデーションを実装する」を参照してください。

一般的なアンチパターン:

- 重要ではない依存関係に障害が発生した場合に、主要ワークロードの可用性に影響が波及する。
- 重要ではないコンポーネントに障害が起きている間に、重要なコンポーネントの動作をテストまたは検証しない。
- 緊急レバーの作動または作動解除に関する決定的な基準が明確に定義されていない。

このベストプラクティスを活用するメリット: 緊急レバーを実装すれば、予期せぬ需要の急増や、重要度の低い依存関係における障害などに対処するためのプロセスを確立してリゾルバーに提供することで、ワークロードに不可欠なコンポーネントの可用性を高めることができます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

- ワークロードの重要なコンポーネントを特定します。
- 重要ではないコンポーネントに障害が起きても耐えられるように、ワークロードの重要なコンポーネントを設計し、構築します。
- 重要ではないコンポーネントで障害が発生している最中に、重要なコンポーネントの動作を検証するためのテストを実施します。
- 緊急レバーの手続き開始の基準となる適切な指標やトリガーを定義し、監視します。
- 緊急レバーを構成する手順 (手動または自動) を定義します。

実装手順

- ワークロード内のビジネスクリティカルなコンポーネントを特定します。
  - ワークロードの技術的なコンポーネントをそれぞれ適切なビジネス機能にマッピングし、重要または非重要にランク分けします。Amazon の重要な機能および非重要な機能の例については、「Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second」を参照してください。
  - これは技術上の決定でもビジネス上の決定でもあり、組織やワークロードによって異なります。
- 重要ではないコンポーネントに障害が起きても耐えられるように、ワークロードの重要なコンポーネントを設計し、構築します。
  - 依存関係の分析では、想定される障害モードをすべて検討し、緊急レバーのメカニズムを通じて、ダウンストリームのコンポーネントも重要な機能を利用できるか検証します。
- 緊急レバーが作動している間に、重要なコンポーネントの動作を検証するためのテストを実施してください。
  - バイモーダル動作は防止してください。詳細については、「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」を参照してください。
- 緊急レバーの手続き開始の基準となる指標を定義して監視し、警戒します。
  - ワークロードに応じて、監視対象として適切な指標を判断してください。指標の例としては、レイテンシーや、依存関係へのリクエストの失敗回数などが該当します。
- 緊急レバーを構成する手順 (手動または自動) を定義します。
  - これには、負荷制限、リクエストのスロットリング、グレースフルデグラデーションの実装などのメカニズムが含まれます。

リソース

関連する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、グレー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
- 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

関連ドキュメント:

- 安全なハンズオフデプロイメントの自動化

- プライムデーがいつ来ても大丈夫: Amazon.com の検索機能がカオスエンジニアリングで 1 秒に 84,000 件以上のリクエストを処理する方法

関連動画:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

変更管理

Questions

- REL 6. ワークロードのリソースをどのようにモニタリングするのですか。
- REL 7. 需要の変化に対応するためには、どのようにワークロードを設計すればよいですか。
- REL 8. どのように変更を実装するのですか。

REL 6. ワークロードのリソースをどのようにモニタリングするのですか。

ログやメトリクスは、ワークロードの状態に関するインサイトを得るための強力なツールです。ログやメトリクスをモニタリングし、しきい値を超えたときや、重要なイベントが発生したときに通知を送信するようにワークロードを設定することができます。モニタリングにより、ワークロードが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らを認識し、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 ログの分析
- 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のツールを使っ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は、メトリクスの季節的な変化と傾向の変化を考慮します。

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 canary」ともいう。「カナリアデプロイ」と混同しないこと) 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が実行するアクションに相当する多くの共通タスクを定期的に実行するものです。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、Synthetic canaries を作成してエンドポイントと API をモニタリングすることができます。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活用するメリット: 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. 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、ディスク I/O の平均、に関するメトリクスを収集します。CloudWatch にメトリクスをパブリッシュする AWS のサービスの一覧については、「CloudWatch メトリクスを発行する AWS のサービス」を参照してください。
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。メトリクスは、AWS CLI または API を使用して作成し、CloudWatch にパブリッシュすることもできます。
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発するかを決定します。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、偽陽性を最小限に抑えることができます。
4. アラートを定義し、アラートが起動し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。Amazon CloudWatch Alarms 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。合成トランザクションを使用すると、顧客が問題を検出する前に問題を検出できます。

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

関連ドキュメント:

- AWS Health Dashboard の使用開始 – アカウントヘルスの確認
- 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
- AWS Lambda での Amazon CloudWatch Logs の使用
- サーバーアクセスログによるリクエストのログ記録
- 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
- Amazon EC2 インスタンスに CloudWatch エージェントをインストールする
- カスタムメトリクスをパブリッシュする
- Amazon CloudWatch ダッシュボードの使用
- Amazon CloudWatch メトリクスを使用する
- canary の使用 (Amazon CloudWatch Synthetics)
- Amazon CloudWatch Logs とは

ユーザーガイド:

- 証跡の作成
- Amazon EC2 Linux インスタンスのメモリおよびディスクのメトリクスをモニタリングする
- コンテナインスタンスでの CloudWatch ログの使用

- VPC フローログ
- Amazon DevOps Guru とは
- What is AWS X-Ray?

関連ブログ:

- Amazon CloudWatch Synthetics と AWS X-Ray でのデバッグ

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
- Amazon 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
- Observability workshop

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、主要な集計と保存の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、アラートを呼び出すために定義した 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を活用し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は CloudWatch Logs に送信されたログデータを検索するための語句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、グラフを作成したり、アラームを設定したりできます。

  - 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 Amazon S3 に直接発行できます。ログの主な要件が Amazon S3 のストレージである場合、追加のインフラストラクチャをセットアップすることなく、簡単に、サービスにログを生成させて、直接 Amazon S3 に送信させることができます。
  - Amazon S3 に直接ログを送信する

リソース

関連ドキュメント:

- Amazon CloudWatch Logs Insights のサンプルクエリ
- Amazon CloudWatch Synthetics と AWS X-Ray でのデバッグ
- 1 つのオブザーバビリティワークショップ
- ログデータの検索およびフィルタリング
- Amazon S3 に直接ログを送信する
- Amazon 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

REL06-BP03 通知を送信する (リアルタイム処理とアラーム)

組織は、潜在的な問題を検出すると、その問題に迅速かつ効果的に対応するために、適切な担当者とシステムにリアルタイムの通知とアラートを送信します。

期待される成果: サービスとアプリケーションのメトリクスに基づいて関連するアラームを設定することで、運用にかかわるイベントに迅速に対応できます。アラームのしきい値を超えると、適切な担当者とシステムに通知され、根本的な問題に対処できます。

一般的なアンチパターン:

- アラームのしきい値を過度に高く設定し、重要な通知が送信されなくなる。
- アラームのしきい値を低くしすぎたことにより、過剰な通知のノイズが原因で重要なアラートへの対処が行われなくなる。

- 使用率が変わってもアラームとそのしきい値を更新しない。
- 自動アクションで対処するのが最適なアラームに対して、自動アクションを生成する代わりに担当者に通知を送信することで、余計な通知が送信されてしまう。

このベストプラクティスを活用するメリット: 適切な担当者とシステムにリアルタイムの通知とアラートを送信することで、問題を早期に検出し、運用にかかわるインシデントに迅速に対応できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

アプリケーションの可用性に影響を与え、自動対応のトリガーとなる可能性のある問題を検出しやすくするために、ワークロードにはリアルタイム処理とアラーム機能が備わっている必要があります。組織は、重要なイベントが発生したり、メトリクスがしきい値を超えたりしたときに通知を受け取ることができるよう、定義されたメトリクスを使用してアラートを作成することで、リアルタイムの処理とアラームの発行を実施できます。

Amazon CloudWatch では、静的しきい値、異常検出、およびその他の基準に基づく CloudWatch アラームを使用して、メトリクスアラームと複合アラームを作成できます。CloudWatch を使用して設定できるアラームの種類の詳細については、CloudWatch ドキュメントのアラームに関するセクションを参照してください。

CloudWatch ダッシュボードを使用して、チーム向けに AWS リソースのメトリクスとアラートをカスタマイズ表示できます。CloudWatch コンソールのカスタマイズ可能なホームページでは、複数のリージョンのリソースを 1 つのビューでモニタリングできます。

アラームは、Amazon SNS トピックへの通知の送信、Amazon EC2 アクションまたは Amazon EC2 Auto Scaling アクションの実行、OpsItem または AWS Systems Manager のインシデントの作成など、1 つまたは複数のアクションを実行できます。

Amazon CloudWatch は Amazon SNS を使用して、アラームの状態が変化したときに通知を送信し、パブリッシャー (プロデューサー) からサブスクライバー (コンシューマー) にメッセージを配信します。Amazon SNS 通知の設定の詳細については、「Configuring Amazon SNS」を参照してください。

CloudWatch アラームが作成、更新、削除されたり、状態が変更されたりするたびに、CloudWatch は EventBridge にイベントを送信します。こうしたイベントで EventBridge を使用して、アラームの

状態が変わるたびに通知したり、Systems Manager Automation を使用してアカウント内のイベントを自動的にトリガーしたりするなどのアクションを実行するルールを作成できます。

EventBridge を使うべき場合と Amazon SNS を使うべき場合

EventBridge と Amazon SNS はどちらもイベント駆動型アプリケーションの開発に使用できます。どちらを選ぶかは、具体的なニーズによって異なります。

Amazon EventBridge は、独自のアプリケーション、SaaS アプリケーション、AWS サービスからのイベントに反応するアプリケーションを構築する場合に推奨されます。EventBridge は、サードパーティーの SaaS パートナーと直接統合する唯一のイベントベースのサービスです。EventBridge は、デベロッパーがアカウントにリソースを作成することなく、200 を超える AWS サービスからイベントを自動的に取り込みます。

EventBridge では、定義済みの JSON ベースの構造がイベントに使用されており、ターゲットに転送するイベントを選択する際にイベント本文全体に適用されるルールを作成できます。EventBridge は現在、AWS Lambda、Amazon SQS、Amazon SNS、Amazon Amazon Kinesis Data Streams、Amazon Data Firehose など、20 を超える AWS サービスをターゲットとしてサポートしています。

Amazon SNS は、高いファンアウトを必要とするアプリケーション (数千または数百万のエンドポイント) に推奨されます。よく見られるパターンは、お客様が Amazon SNS をルールのターゲットとして使用し、必要なイベントをフィルタリングして複数のエンドポイントに分散させるというものです。

メッセージは構造化されておらず、任意の形式にすることができます。Amazon SNS では Lambda、Amazon SQS、HTTP/S エンドポイント、SMS、モバイルプッシュ、メールの 6 種類のターゲットへのメッセージ転送をサポートしています。Amazon SNS の通常のレイテンシーは 30 ミリ秒未満です。AWS のさまざまなサービス (Amazon EC2、Amazon S3、Amazon RDS など 30 以上のサービス) で、Amazon SNS メッセージを送信するようにサービスを設定できます。

実装手順

1. Amazon CloudWatch アラームを使用してアラームを作成します。
    a. メトリクスアラームは、単一の CloudWatch メトリクス、または CloudWatch メトリクスに依存する式をモニタリングします。アラームは、メトリクスまたは式の値としきい値との比較に基づいて、複数の時間間隔にわたって 1 つまたは複数のアクションを開始します。アクションでは、Amazon SNS トピックに通知を送信したり、Amazon EC2 アクションまたは Amazon EC2 Auto Scaling アクションを実行したりできます。また、AWS Systems Manager で OpsItem またはインシデントを作成できます。

b. 複合アラームは、作成した他のアラームのアラーム条件を考慮するルール式で構成されます。複合アラームは、すべてのルール条件が満たされた場合にのみアラーム状態になります。複合アラームのルール式で指定されるアラームには、メトリクスアラームや追加の複合アラームを含めることができます。複合アラームは、状態が変更されたときに Amazon SNS 通知を送信できます。また、ALARM 状態になったときに Systems Manager の OpsItems またはインシデントを作成できますが、EC2 アクションまたは Auto Scaling アクションを実行することはできません。

2. Amazon SNS 通知を設定します。CloudWatch アラームを作成する際には、アラームの状態が変化したときに通知を送信する Amazon SNS トピックを含めることができます。
3. 指定された CloudWatch アラームに一致するルールを EventBridge に作成します。各ルールは、Lambda 関数を含む複数のターゲットをサポートします。例えば、使用可能なディスク容量が少なくなったときに起動するアラームを定義できます。このアラームにより、領域をクリーンアップする Lambda 関数が EventBridge ルールを介してトリガーされます。EventBridge ターゲットの詳細については、「EventBridge targets」を参照してください。

リソース

関連する Well-Architected の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)
- REL12-BP01 プレイブックを使用して障害を調査する

関連ドキュメント:

- Amazon CloudWatch
- CloudWatch Logs insights
- Amazon CloudWatch アラームの使用
- Amazon CloudWatch ダッシュボードの使用
- Amazon CloudWatch メトリクスを使用する
- Amazon SNS 通知の設定
- CloudWatch 異常検出
- CloudWatch Logs data protection
- Amazon EventBridge

- Amazon Simple Notification Service

関連動画:

- reinvent 2022 observability videos
- AWS re:Invent 2022 - Observability best practices at Amazon

関連する例:

- 1 つのオブザーバビリティワークショップ
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

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)。

アラームの自動リアルタイム処理が実装されているため、アラームがトリガーされたときにシステムが迅速に是正措置を講じ、障害やサービスの低下を防ぐことができます。アラームへの自動対応には、障害が起きたコンポーネントの交換、コンピューティングキャパシティの調整、正常なホスト、アベイラビリティーゾーン、その他のリージョンへのトラフィックのリダイレクト、オペレーターへの通知などがあります。

期待される成果: リアルタイムのアラームが特定され、アラームの自動処理が設定され、サービスレベル目標とサービスレベルアグリーメント (SLA) を達成するための適切なアクションが呼び出されます。自動処理は、単一コンポーネントの自己修復アクティビティからサイト全体のフェイルオーバーまで多岐にわたります。

一般的なアンチパターン:

- 主要なリアルタイムアラームの明確なインベントリまたはカタログがない。
- 重大なアラームへの自動対応 (例えば、コンピューティングが枯渇しそうになると、オートスケーリングが行われる) が欠如している。
- アラームへの対応が矛盾している。
- オペレーターがアラート通知を受け取ったときに従うべき標準作業手順書 (SOP) がない。
- 構成変更がモニタリングされていない。構成変更が検出されないと、ワークロードのダウンタイムが生じる可能性があります。

- 意図しない構成変更を取り消す戦略がない。

このベストプラクティスを活用するメリット: アラーム処理を自動化することで、システムの回復力を向上させることができます。システムが自動的に是正措置を講じるため、人が介入することでミスが生じやすい手作業を減らすことができます。ワークロードの可用性の目標を達成し、サービスの中断を低減します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

アラートを効果的に管理し、対応を自動化するには、重要度と影響に基づいてアラートを分類し、対応手順を文書化し、対応計画を立ててからタスクをランク付けします。

特定のアクション (たいていはランブックに詳細が記載されている) が必要なタスクを特定し、ランブックとプレイブックをすべて調べて、どのタスクを自動化できるか判断します。アクションを定義できる場合、たいていは自動化できます。アクションを自動化できない場合は、手作業による手順を SOP に記録し、オペレーターにその手順の訓練をします。手作業のプロセスは継続的に見直し、アラートへの対応を自動化する計画を立て、実践できる余地がないか検討してください。

実装手順

1. アラームのインベントリを作成する: すべてのアラームのリストを取得するには、Amazon CloudWatch コマンド `describe-alarms` を使用して AWS CLI を使用できます。設定したアラームの数によっては、ページ分割を使用して各呼び出しのアラームのサブセットを取得するか、または AWS SDK を使用して API コールを使用してアラームを取得できます。
2. すべてのアラームアクションを文書化する: 手動か自動かにかかわらず、すべてのアラームとそのアクションでランブックを更新します。AWS Systems Manager には、定義済みのランブックが用意されています。詳細については、「ランブックの使用」を参照してください。ランブックコンテンツを表示する方法の詳細については、「View runbook content」を参照してください。
3. アラームアクションを設定して管理する: アクションが必要なアラームについては、CloudWatch SDK を使用して自動アクションを指定します。例えば、アラームに応じてアクションを作成して有効にする、またはアラームに応じてアクションを無効にする形で、CloudWatch アラームに基づいて Amazon EC2 インスタンスの状態を自動的に変更できます。

   Amazon EventBridge を使用すると、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。ルールを作成して、注目しているイベントと、イベントがルールに一致した場合に実行するアクションを指定できます。自動的に開始できるアク

ションには、AWS Lambda 関数の呼び出し、Amazon EC2 Run Command の呼び出し、Amazon Kinesis Data Streams へのイベントのリレー、「EventBridge を使用して Amazon EC2 を自動化する」の表示が含まれます。

4. 標準作業手順書 (SOP): アプリケーションコンポーネントに基づいて、AWS Resilience Hub は複数の SOP テンプレートを推奨します。これらの SOP を使用して、アラートが発生した場合にオペレーターが従うべきプロセスをすべて文書化できます。また、Resilience Hub のレコメンデーションに基づいて SOP を作成することもできます。その場合は、回復ポリシーを関連付けた Resilience Hub のアプリケーションと、そのアプリケーションに対する回復力評価の履歴が必要です。SOP のレコメンデーションは、回復力の評価を受けて作成されます。

   Resilience Hub は Systems Manager と連携して、SOP の基礎として使用できる多数の SSM ドキュメントを提供することで、SOP の手順を自動化します。例えば、Resilience Hub は既存の SSM 自動化ドキュメントに基づいてディスク容量を追加するための SOP を推奨する場合があります。

5. Amazon DevOps Guru を使用して自動化アクションを実行する: Amazon DevOps Guru を使用して、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できます。DevOps Guru を使用すると、Amazon CloudWatch メトリクス、AWS Config、AWS CloudFormation、AWS X-Ray など、複数のソースからの運用データのストリームをほぼリアルタイムでモニタリングできます。また、DevOps Guru を使用して OpsCenter で OpsItems を自動的に作成し、イベントを EventBridge に送信して追加の自動化を行うこともできます。

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)
- 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)
- 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

関連ドキュメント:

- AWS Systems Manager Automation
- AWS リソースのイベントでトリガーされる EventBridge ルールの作成
- 1 つのオブザーバビリティワークショップ

- Amazon 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
- Amazon DevOps Guru とは
- オートメーションドキュメント (プレイブック) の使用

関連動画:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

関連する例:

- Reliability Workshops
- Amazon CloudWatch and Systems Manager Workshop

REL06-BP05 ログの分析

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。詳細については、Amazon Athena ユーザーガイドでサポートされる SerDes およびデータ形式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
サードパーティーによって提供されています。このようなツールには、New
Relic、Splunk、Loggly、Logstash、CloudHealth、Nagios などがあります。ただし、システムやアプリケーションログの外で行うデータ生成は各クラウドプロバイダーに固有であり、また多くの場合サービスごとに固有です。

モニタリングプロセスで見落とされがちな点は、データ管理です。モニタリングのためのデータ保存要件を決定し、それに応じたライフサイクルポリシーを適用する必要があります。Amazon S3 は、S3 バケットレベルのライフサイクル管理をサポートしています。このライフサイクル管理には、バケット内のパスごとに異なる管理方法を適用できます。ライフサイクルの最終段階では、データを Amazon S3 Glacier に移行して長期保存し、保存期間の終了後には期限切れにすることができます。S3 Intelligent-Tiering ストレージクラスは、パフォーマンスへの影響や運用のオーバーヘッドなしに、データを最も費用対効果の高いアクセス階層に自動的に移動することにより、コストを最適化できるように設計されています。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

- CloudWatch Logs Insights を使用すると、Amazon CloudWatch Logs のログデータをインタラクティブに検索し分析することが可能になります。
  - Analyzing Log Data with CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights のサンプルクエリ
- Amazon CloudWatch Logs を使用して、Amazon Athena でデータのクエリを実行できる Amazon S3 にログを送信します。
  - Amazon Athena で Amazon S3 サーバーアクセスログを分析する方法を教えてください。
    - 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。これにより、各クエリで Athena が分析するデータの量が減ります。
      - 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

関連ドキュメント:

- Amazon CloudWatch Logs Insights のサンプルクエリ
- Analyzing Log Data with CloudWatch Logs Insights
- Amazon CloudWatch Synthetics と AWS X-Ray でのデバッグ
- S3 バケットのライフサイクルポリシーを作成する方法
- Amazon Athena で Amazon S3 サーバーアクセスログを分析する方法を教えてください。
- 1 つのオブザーバビリティワークショップ

- Amazon 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

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: ログを保存してその内容を調査します。
- Amazon CloudWatch Logs Insights: 大量のログを数秒で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、運用パフォーマンスのレビューとチーム間での学習の共有を目的として、毎週ミーティングを実施しています。AWS には非常に多くのチームが存在するため、レビューするワークロードをランダムに選択するために The wheel を作成しました。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: 中

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

関連ドキュメント:

- Amazon CloudWatch Logs Insights のサンプルクエリ
- Amazon CloudWatch Synthetics と AWS X-Ray でのデバッグ
- 1 つのオブザーバビリティワークショップ
- Amazon 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
- 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ー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、サードパーティー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。

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して、アプリケーションを通過するリクエストの全体像を把握できます。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

- Amazon S3、AWS Lambda、Amazon API Gateway など、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 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 Management Console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ー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。
- Amazon CloudWatch Synthetic の Canary および Amazon CloudWatch RUM の X-Ray トレースを使用して、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。
- プライマリトレースソリューションにサードパーティー製ツールを使用している場合は、Datadog、New Relic、Dynatrace などのサードパーティートレースツールの X-Ray 統合を有効にします。

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
- Amazon CloudWatch Synthetics と AWS X-Ray でのデバッグ
- Amazon 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
- Integrating AWS X-Ray with other AWS services
- AWS Distro for OpenTelemetry と AWS X-Ray
- Amazon CloudWatch: 合成モニタリング (canary)
- Amazon CloudWatch: CloudWatch RUM
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm
- 可用性およびその他: AWS の分散システムの回復力の理解と向上

関連する例:

- 1 つのオブザーバビリティワークショップ

関連動画:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

関連ツール:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7. 需要の変化に対応するためには、どのようにワークロードを設計すればよいですか。

スケーラブルなワークロードでは、リソースを自動的に追加または削除することで、任意の時点での需要により合致できる伸縮性を得られます。

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ルしたりする場合は、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。サードパーティーのツールや 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 AWS サービスには、Amazon S3、Amazon CloudFront 、AWS Auto Scaling、AWS Lambda、Amazon DynamoDB、AWS Fargate、および Amazon Route 53 が含まれます。

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。他にも、Amazon EC2 インスタンスおよびスポットフリート、Amazon ECS タスク、Amazon DynamoDB テーブルとインデックス、Amazon Aurora レプリカなどのリソースのスケーリングプランを作成することもできます。

EC2 インスタンスをスケールする場合は、複数のアベイラビリティーゾーン (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ー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ー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ルされます。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、高いリクエストレートに対応するため自動的にスケールされます。例えば、アプリケーションでバケット内のプレフィックスごとに 1 秒あたり 3,500 回以上の PUT/COPY/POST/

DELETE リクエストまたは 5,500 回以上の GET/HEAD リクエストを達成できます。バケット内のプレフィックスの数に制限はありません。読み取りを並列化することによって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。たとえば、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と、読み取りパフォーマンスを 1 秒あたり 55,000 回の読み取りリクエストにスケールできます。

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ル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。
  - What is AWS Auto Scaling?
    - 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、該当するものに対して 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?
  - Application Load Balancer は、パスごとに負荷を分散できます。
    - 「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 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。
        - 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 または信頼された DNS プロバイダーを使用します。
    - What is Amazon Route 53?
  - 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。
      - レコードを使用する
- 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。AWSGlobal 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 秒以内に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 Balancer、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 とは何ですか?
    - 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、サードパーティーの 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

関連ドキュメント:

- APN パートナー: 自動化されたコンピューティングソリューションの作成を支援できるパートナー
- AWS Auto Scaling: スケーリングプランの仕組み
- AWS Marketplace: 自動スケーリングで使用できる製品
- DynamoDB Auto Scaling によるスループットキャパシティの自動管理
- 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?
- Amazon EC2 Auto Scaling とは
- What is AWS Auto Scaling?
- Amazon CloudFront とは何ですか?
- Amazon Route 53 とは?
- Elastic Load Balancing とは?
- Network Load Balancer とは?
- 「Application Load Balancer とは?」
- レコードを使用する

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

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ルするか、オートメーションを開始してリソースを自動的にスケールします。

スケーリングはワークロードに合わせて手動で調整できます。例えば、Auto Scaling グループの EC2 インスタンスの数の変更や、DynamoDB テーブルのスループットの変更は、AWS Management Consoleまたは AWS CLI で行うことができます。ただし、可能な限り自動化を使用する必要があります (「リソースを取得またはスケールするときに自動化を使用する」を参照)。

期待される成果: 障害やカスタマーエクスペリエンスの低下が検知された時点で、可用性を回復するためのスケーリングアクティビティ (自動または手動) が開始されます。

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: 中

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オブザーバビリティとモニタリングを実装して、カスタマーエクスペリエンスを監視し、障害を検知します。必要なリソースをスケールする手順 (手動または自動) を定義します。詳細については、「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」を参照してください。

実装手順

- 必要なリソースをスケールする手順 (手動または自動) を定義します。
  - スケーリングの手順は、ワークロード内のさまざまなコンポーネントの設計方法に応じて異なります。
  - また、使用されている基盤のテクノロジーによっても異なります。
    - AWS Auto Scaling を使用するコンポーネントでは、スケーリングプランを使用して、リソースをスケールするための一連の指示を設定できます。AWS CloudFormation を使用または AWS リソースにタグを追加する場合、アプリケーションごとに異なる一連のリソース用にスケーリングプランを設定できます。Auto Scaling は、各リソースに合わせてカスタマイズされたスケーリング戦略の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、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。詳細については、「How scaling plans work」を参照してください。
    - Amazon EC2 Auto Scaling は、アプリケーションの負荷を処理するために適切な数の Amazon EC2 インスタンスを利用できるようにします。Auto Scaling グループと呼ばれる EC2 インスタンスの集合を作成します。Auto Scaling グループごとにインスタンスの最小数と最大数

を指定でき、グループがこれらの制限を下回る/上回ることがないように Amazon EC2 Auto Scaling が調整します。詳細については、「What is Amazon EC2 Auto Scaling?」を参照してください。

- Amazon DynamoDB Auto Scaling は Application Auto Scaling サービスを使用し、実際のトラフィックパターンに応じてプロビジョンドスループットキャパシティをユーザーに代わって動的に調節します。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で、プロビジョニングされた読み込み/書き込みキャパシティーが拡張され、トラフィックの急激な増加をスロットリングなしに処理できるようになります。詳細については、「DynamoDB Auto Scaling によるスループットキャパシティの自動管理」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:

- 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
- 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

関連ドキュメント:

- AWS Auto Scaling: スケーリングプランの仕組み
- DynamoDB Auto Scaling によるスループットキャパシティの自動管理
- Amazon EC2 Auto Scaling とは

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、これらの自動スケーリング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、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。

Amazon EC2 Auto Scaling の予測自動スケーリングは、機械学習によって各リソースのワークロード履歴を分析し、今後のワークロード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数 (EC2 インスタンス、同時実行の Lambda 関数など) 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: 中

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。
    - Telling Stories About Little's Law
  - 使用状況の履歴パターンがある場合は、Amazon EC2 Auto Scaling のスケジュールされたスケーリングを設定します。
    - 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。
    - Amazon EC2 Auto Scaling の予測スケーリング

リソース

関連ドキュメント:

- AWS Marketplace: 自動スケーリングで使用できる製品
- DynamoDB Auto Scaling によるスループットキャパシティの自動管理
- 新機能 – Machine Learning を中核とする 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
- Telling Stories About Little's Law
- Amazon EC2 Auto Scaling とは

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、本稼働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。テスト環境の支払いは実行時にのみ発生するため、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稼働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:

- 本番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。
- 予想される負荷を下回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: 負荷がかかるとアーキテクチャのどのコンポーネントに障害が発生するかを把握し、問題への対処に間に合うように、その負荷に近づいていることを示す、監視するべきメトリクスを特定して、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: 中

実装のガイダンス

- 負荷テストを実行して、容量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、本番環境で受け取るものと同様の代表的なトラフィックを使用する必要がありま

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。さまざまなリクエストが混在している可能性があるため、トラフィックの組み合わせを特定するときは、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します。
  - 最初は小さな容量に対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容量に負荷をかけることで、すぐに効果が現れます。
  - 大きな容量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、できるだけ本番環境に近い環境でテストする必要があります。

リソース

関連ドキュメント:

- 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
- 「アプリケーションの負荷テスト」

関連動画:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing

REL 8. どのように変更を実装するのですか。

変更制御は、新しい機能をデプロイしたり、ワークロード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検証したりするために必要です。これらの変更がコントロールされていない場合、変更による影響を予測したり、変更によって発生する問題に対応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 自動化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に、プロセスを配置します。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始めて、それをコードで実装し、必要に応じて自動実行を呼び出します。

高度に自動化された最新のワークロードに対しても、ランブックはゲームデーの実行や、厳格なレポートおよび監査の要件を満たすのに役立ちます。

プレイブックは特定のインシデントに対応するために使用し、ランブックは特定の成果を達成するために使用します。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対応するために使用します。

一般的なアンチパターン:

- 本番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して、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活用するメリット: 効果的な変更計画を作成すると、影響を受けるすべてのシステムを認識できるため、変更を正常に実行する能力が向上します。テスト環境で変更を検証すると、信頼性が強化されます。

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

- ランブックに手順を文書化することで、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 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ー) を使用してインフラストラクチャを定義することで、バージョン管理ソフトウェアを使用して、バージョン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ープロバイダー) 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ー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 フレームワーク: 概念: ランブック
- What is AWS CloudFormation?
- AWS CodeCommit とは

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

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、パイプラインの本稼働前にステージングされた、本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。

期待される成果: 自動化を使用して機能をテストし、関連付けられたテストデータによってテスト期間および費用を削減して、テスト結果の精度を高めます。デプロイプロセスの一部として機能テストを組み込むことで、リリースパイプラインを自動化して、アプリケーションとインフラストラクチャを迅速かつ確実に更新できます。

一般的なアンチパターン:

- テストをデプロイパイプラインの外部で手動で実行する。
- 手動の緊急ワークフローにより、自動化のテストステップを省略する。
- スケジュールを短縮するために、確立されたテスト計画やプロセスを無視する。

このベストプラクティスを活用するメリット: 機能テストにより、システムが指定された要件に従って動作することを検証できます。テストでは、ユーザーインターフェイス、API、データベース、ソースコードなどのコンポーネントの想定された動作順序を一貫して検証します。システムのこれらのコンポーネントを検証する際に、機能テストは各機能が想定どおりに動作することを検証します。これにより、ユーザーの期待とソフトウェアの整合性の両方を満たすことができます。通常のデプロイの一部として機能テストを組み込み、すべての変更を自動化してデプロイすることで、人為的ミスが発生する可能性を軽減できます。

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

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。成功基準が満たされない場合、パイプラインは停止またはロールバックされます。AWS CodePipeline は、自動テスト用の継続的デリバリーパイプラインを提供するため、テスターはテストとデプロイのプロセス全体を自動化できます。AWS CodeBuild や AWS CodeDeploy などの AWS サービスと統合して、ソフトウェア開発ライフサイクルの構築、テスト、デプロイの各フェーズを自動化します。

実装手順

- パイプラインを設定する: AWS CodePipeline コンソールまたは AWS Command Line Interface (CLI) を使用して、ソース、ビルド、テスト、デプロイの各ステージを設定します。
  - ソースを定義する: AWS CodePipeline を使用すると、GitHub 、AWS CodeCommit、Bitbucket などのバージョン管理システムからソースコードを自動的に取得できます。これにより、最新のコードが常にテストに使用されることを確認できます。

- ビルドとテストを自動化する: AWS CodeBuild はコードを自動的にビルドしてテストし、テストレポートを生成します。JUnit、NUnit、TestNG などの一般的なテストフレームワークをサポートします。
- コードをデプロイする: コードの構築とテストが完了すると、AWS CodeDeploy は、Amazon EC2 インスタンス、AWS Lambda 関数、オンプレミスサーバーなどのテスト環境にコードをデプロイします。
- パイプラインのモニタリング: AWS CodePipeline はパイプラインの進行状況と各ステージのステータスを追跡します。品質チェックを使用して、テストの実行ステータスに基づいてパイプラインをブロックすることもできます。パイプラインステージの障害や、パイプラインの完了に関する通知を受け取ることもできます。

リソース

関連ドキュメント:

- AWS CodeBuild で AWS CodePipeline を使用してコードをテストし、ビルドを実行する
- AWS CodeBuild でのログ記録とモニタリング
- 機能テストのインジケータ

REL08-BP03 デプロイの一部として回復力テストを統合する

意図的にシステムに障害を導入することで回復力テストを統合し、障害発生時のシステムの能力を測定します。回復力テストは、システムでの予期しない障害の特定に重点を置いているため、通常デプロイサイクルに統合されるユニットテストや機能テストとは異なります。本番稼働前に回復力テストの統合を始めても問題ありませんが、ゲームデーの一環として、これらのテストを本番環境に実装するという目標を設定します。

期待される成果 回復力テストは、本番環境の劣化に耐えられるシステムの能力に対する信頼を構築するのに役立ちます。テストによって障害につながる可能性のある弱点を特定することで、システムを改善し、障害や劣化を自動的かつ効率的に軽減できます。

一般的なアンチパターン:

- デプロイプロセスにおけるオブザーバビリティとモニタリングの欠如
- システム障害の解決を人間に依存する
- 低品質の分析メカニズム

- システムの既知の問題に重点を置き、未知の問題点を特定するためのテストを行わない
- 障害を特定できるが、解決できない
- 検出結果とランブックが文書化されていない

このベストプラクティスを活用する利点: デプロイに統合された回復力テストは、システムの未知の問題のうち、テストを実行しないと気付かないものを特定するのに役立ちます。これらの問題は本番環境のダウンタイムにつながる可能性があります。システムでのこれらの未知の問題の特定は、検出結果の文書化、CI/CD プロセスへのテストの統合、およびランブックの作成に役立ち、効率的で反復可能なメカニズムを通じて、障害の緩和を簡素化します。

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: 中

実装のガイダンス

システムのデプロイに統合できる最も一般的な回復力テストの形式は、ディザスタリカバリとカオスエンジニアリングです。

- 大規模なデプロイには、ディザスタリカバリ計画と標準運用手順 (SOP) の更新を含めます。
- 信頼性テストを自動デプロイパイプラインに統合します。AWS Resilience Hub などのサービスを CI/CD パイプラインに統合して、すべてのデプロイの一部として自動的に評価される、継続的な耐障害性評価を確立します。
- AWS Resilience Hub でアプリケーションを定義します。耐障害性評価では、アプリケーションの AWS Systems Manager ドキュメントとして復旧手順を作成するのに役立つコードスニペットが生成され、推奨される Amazon CloudWatch モニタとアラームのリストが提供されます。
- ディザスタリカバリ計画と SOP が更新されたら、ディザスタリカバリテストを実施して効果を確認します。ディザスタリカバリテストは、イベント後にシステムを復旧して通常の運用に戻ることができるかどうかを判断するのに役立ちます。さまざまなディザスタリカバリ戦略をシミュレートし、計画が稼働時間の要件を満たすのに十分であるかどうかを確認できます。一般的なディザスタリカバリ戦略には、バックアップと復元、パイロットライト、コールドスタンバイ、ウォームスタンバイ、ホットスタンバイ、アクティブ - アクティブなどがあり、コストと複雑さは戦略によって異なります。ディザスタリカバリテストの前に、目標復旧時間 (RTO) と目標復旧時点 (RPO) を定義して、シミュレートする戦略の選択を簡素化することをお勧めします。AWS には、計画とテストの開始に役立つ AWS Elastic Disaster Recovery などのディザスタリカバリツールが用意されています。
- カオスエンジニアリングのテストでは、ネットワーク障害やサービス障害などのシステムの中断を発生させます。制御された障害を使用してシミュレーションを行うことで、発生した障害の影響を抑えながら、システムの脆弱性を発見できます。他の戦略と同様に、AWS Fault Injection Service

のようなサービスを使用して、非本番環境で制御された障害シミュレーションを実行し、本番環境にデプロイする前に確信を得ることができます。

リソース

関連ドキュメント:

- レジリエンステストで障害に関する実験を行いリカバリに備える
- アプリケーションを AWS Resilience Hub と AWS CodePipeline で継続的に評価する
- AWS で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の戦略
- カオスエンジニアリングを使用したワークロードのレジリエンスの検証
- 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ワークショップ

関連動画:

- AWS re:Invent 2020: Testing Resilience using Chaos Engineering
- Improve Application Resilience with AWS Fault Injection Service
- Prepare & Protect Your Applications From Disruption With AWS Resilience Hub

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、本稼働ワークロードで更新、セキュリティパッチ適用、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。

イミュータブルなインフラストラクチャのデプロイ戦略に従って、ワークロードデプロイの信頼性、一貫性、再現性を高めましょう。

期待される成果: イミュータブルなインフラストラクチャでは、ワークロード内でインフラストラクチャリソースを実行するためのインプレース変更はできません。代わりに、変更の必要が生じた場合は、必要な変更をすべて適用した新しいインフラストラクチャリソース一式を、既存のリソースと並行してデプロイします。このデプロイは自動的に検証され、検証に合格すると、トラフィックが新しいリソース一式に徐々にシフトします。

このデプロイ戦略は、ソフトウェアの更新、セキュリティパッチの適用、インフラストラクチャの変更、構成の更新、アプリケーションの更新などに適用されます。

一般的なアンチパターン:

- 実行中のインフラストラクチャリソースにインプレース変更を実装する。

このベストプラクティスを活用するメリット:

- 環境全体での一貫性の向上: 環境全体でインフラストラクチャリソースに違いがないため、一貫性が向上し、テストが簡素化されます。
- 設定ドリフトの削減: インフラストラクチャリソースを既知のバージョン管理された設定に置き換えることで、インフラストラクチャが既知のテスト済みで信頼できる状態に設定され、設定ドリフトを回避できます。
- 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、何も変更されないため、デプロイプロセスの一貫性と信頼性が向上します。
- デプロイの簡素化: 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。新たにデプロイするだけでアップグレードされます。
- 迅速なロールバックとリカバリプロセスによる安全なデプロイ: 以前の動作バージョンが変更されないため、デプロイはより安全です。エラーが検出された場合はロールバックできます。
- セキュリティ体制の強化: インフラストラクチャへの変更を許可しないことで、リモートアクセスメカニズム (SSH など) を無効にすることができます。これにより、攻撃ベクトルが減少し、組織のセキュリティ体制が強化されます。

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: 中

実装のガイダンス

オートメーション

イミュータブルなインフラストラクチャのデプロイ戦略を定義するときは、再現性を高め、人為的ミスの可能性を最小限に抑えるために、可能な限り自動化を使用することをお勧めします。詳細については、「REL08-BP05 自動化による変更のデプロイ」および「安全かつハンドオフのデプロイの自動化」を参照してください。

Infrastructure as Code (IaC) では、インフラストラクチャのプロビジョニング、オーケストレーション、およびデプロイのステップがプログラムによって説明的、宣言的な方法で定義され、ソース管理システムに保存されます。IaC を活用することで、インフラストラクチャのデプロイを簡単に自動化し、インフラストラクチャの不変性を実現できます。

デプロイパターン

ワークロードの変更が必要な場合、イミュータブルなインフラストラクチャのデプロイ戦略では、必要な変更をすべて適用済みの、新しいインフラストラクチャリソース一式をデプロイすることが義務付けられています。この新しいリソースセットでは、ユーザーへの影響を最小限に抑えるロールアウトパターンに従うことが重要です。このデプロイには、主に 2 つの戦略があります。

カナリアデプロイは、通常、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、発生し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は、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、完全にデプロイされるまで、希望の速度でデプロイを続行できます。AWS CodeDeploy では、デプロイ設定で カナリアデプロイを有効にすることができます。

ブルー/グリーンデプロイはカナリアデプロイに似ていますが、アプリケーションのフリート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と 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。

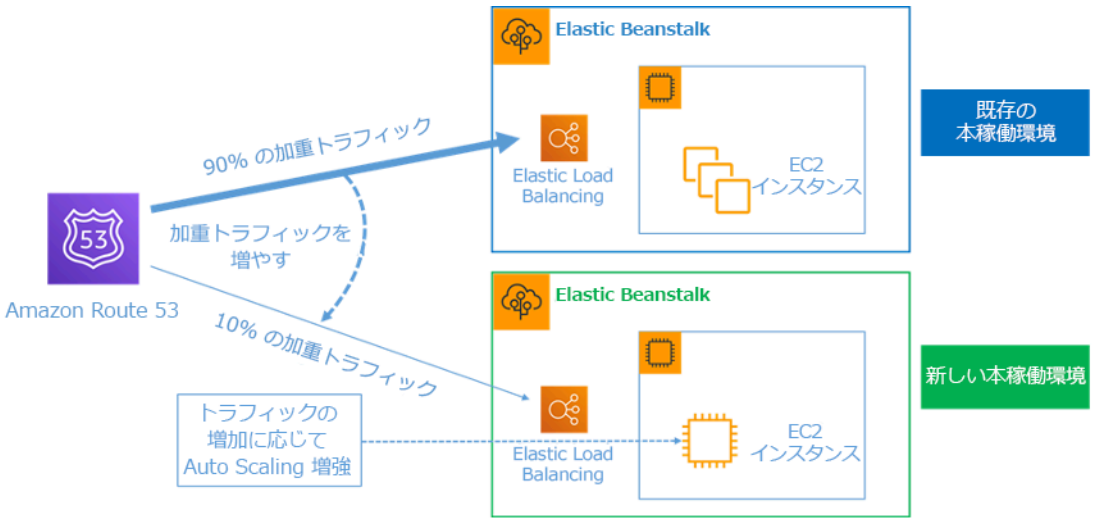


図 8: AWS Elastic Beanstalk と Amazon Route 53 によるブルー/グリーンデプロイ

ドリフト検出

ドリフトは、インフラストラクチャリソースの状態や構成が想定とは異なる変更として定義されます。適切に管理されていない構成変更は、イミュータブルなインフラストラクチャの概念と矛盾します。イミュータブルなインフラストラクチャを正常に実装するには、このような構成変更を検出し、修復する必要があります。

実装手順

- 実行中のインフラストラクチャリソースのインプレース変更は禁止します。
  - AWS Identity and Access Management (IAM) を使用すると、AWS のサービスとリソースにアクセスできるユーザーまたは内容を指定し、きめ細かいアクセス許可を一元管理し、アクセスを分析して、AWS 全体のアクセス許可を調整できます。
- インフラストラクチャリソースのデプロイを自動化して再現性を高め、人為的ミスを最小限に抑えます。
  - AWS ホワイトペーパーの「DevOps の概要」で説明されているように、自動化は AWS サービスの基礎であり、すべてのサービスと機能で内部的にサポートされています。
  - Amazon マシンイメージ (AMI)を事前に作成すると、起動時間を短縮できます。EC2 Image Builder は、カスタマイズされた、安全かつ最新の Linux または Windows カスタム AMI の作成、メンテナンス、検証、共有、デプロイを自動化するのに役立つフルマネージド AWS サービスです。
  - 次のサービスは自動化に対応しています。
    - AWS Elastic Beanstalk は、Java、.NET、PHP、Node.js、Python、Ruby、Go、Docker で開発されたウェブアプリケーションを、Apache、NGINX、Passenger、IIS などの一般的なサーバーに迅速にデプロイしてスケールするためのサービスです。
    - AWS Proton は、プラットフォームチームがインフラストラクチャのプロビジョニング、コードのデプロイ、モニタリング、更新に必要なさまざまなツールを接続および調整するのに役立ちます。AWS Proton は、サーバーレスおよびコンテナベースのアプリケーションのコードプロビジョニングおよびデプロイとして、インフラストラクチャを自動化できるようにします。
  - IaC を活用することで、インフラストラクチャのデプロイを簡単に自動化し、インフラストラクチャの不変性を実現できます。AWS は、プログラムにより記述的、宣言的な方法でインフラストラクチャの作成、デプロイ、メンテナンスを可能にするサービスを提供しています。
    - AWS CloudFormation は、開発者が順序正しく予測可能な方法でAWS リソースを作成するのに役立ちます。リソースは、JSON または YAML 形式でテキストファイルに書き込まれます。テンプレートには、作成および管理されるリソースのタイプに応じて、特定の構文と構造が必要です。AWS Cloud9 などのコード工ディタを使用して JSON または YAML でリソースを作成し、バージョン管理システムにチェックインすると、CloudFormation が、指定されたサービスを安全で繰り返し可能な方法で構築します。
    - AWS Serverless Application Model (AWS SAM) は、AWS でサーバーレスアプリケーションを構築するために使用できるオープンソースフレームワークです。AWS SAM は他の AWS サービスと統合された、AWS CloudFormation の拡張機能です。

    - AWS Cloud Development Kit (AWS CDK) は、使い慣れたプログラミング言語を使用して、クラウドアプリケーションリソースをモデル化およびプロビジョニングするために使用できるオープンソースのソフトウェア開発フレームワークです。AWS CDK により、TypeScript、Python、Java、.NET を使用してアプリケーションインフラストラクチャをモデル化できます。AWS CDK は AWS CloudFormation をバックグラウンドで使用し、安全で繰り返し可能な方法でリソースをプロビジョニングします。
    - AWS Cloud Control API では、開発者がクラウドインフラストラクチャを簡単に、一貫した方法で管理できるように、作成、読み取り、更新、削除、一覧表示 (CRUDL) API の共通セットが導入されています。Cloud Control API の共通 API を使用すると、開発者は AWS およびサードパーティーサービスのライフサイクルを均一に管理できます。
- ユーザーへの影響を最小限に抑えるデプロイパターンを実装します。
  - カナリアデプロイ:
    - API Gateway の Canary リリースデプロイの設定
    - AWS App Mesh を使用して、Amazon ECS のカナリアデプロイでパイプラインを作成する
  - ブルー/グリーンデプロイ: AWS ホワイトペーパーの「ブルー/グリーンデプロイ」では、ブルー/グリーンデプロイ戦略を実装する手法の例について説明しています。
- 構成または状態のドリフトを検出します。詳細については、「スタックとリソースに対するアンマネージド型構成変更の検出」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:

- REL08-BP05 自動化を使用して変更をデプロイする

関連ドキュメント:

- Automating safe, hands-off deployments
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- Infrastructure as Code
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

関連動画:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

REL08-BP05 自動化を使用して変更をデプロイする

デプロイとパッチ適用を自動化することで、悪影響を排除します。

本稼働システムに変更を加えることは、多くの組織にとって最大級のリスクの 1 つです。当社は、ソフトウェアで対処するビジネス上の問題と同じくらい、デプロイを最優先の課題であると考えています。これは今日、変更のテストとデプロイ、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。

期待される成果: 広範な本稼働前テスト、自動ロールバック、ずらされた本稼働デプロイを使用して、自動デプロイの安全性をリリースプロセスに構築します。この自動化により、デプロイの失敗による本番環境への潜在的な影響が最小限に抑えられ、開発者は本番環境へのデプロイを積極的に監視する必要がなくなります。

一般的なアンチパターン:

- 手動で変更を行う。
- 手動の緊急ワークフローにより、自動化のステップを省略する。
- スケジュールを短縮するために、確立された計画やプロセスを無視する。
- ベイク時間を考慮せずに、急激なフォローオンデプロイを実行する。

このベストプラクティスを活用するメリット: 自動化を使用してすべての変更をデプロイすることで、人為的ミスが発生する可能性を排除し、本番環境を変更する前にテストする機能を提供します。本番環境の稼働前にこのプロセスを実行することで、計画が完了していることを確認できます。さらに、リリースプロセスに自動ロールバックを組み込むことで、本番環境の問題を特定し、ワークロードを以前の動作状態に戻すことができます。

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

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。このために不可欠な部分は、継続的インテグレーションと継続的デリバリー/デプロイ (CI/CD) 文化の導入です。これにより、コミットまたはコードの変更は、ビルドおよびテスト段階から本番環境へのデプロイまで、さまざまな自動化されたステージゲートを通過します。

運用上の最も難しい手順には人間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。

実装手順

デプロイを自動化して手動操作をなくすには、以下の手順に従います。

- コードを安全に保存するためのコードリポジトリを設定する: AWS CodeCommit を使用して、安全な Git ベースのリポジトリを作成します。
- ソースコードをコンパイルし、テストを実行して、デプロイアーティファクトを作成するように継続的統合サービスを設定する: この目的のためにビルドプロジェクトを設定するには、「コンソールを使用した AWS CodeBuild の開始方法」を参照してください。
- アプリケーションのデプロイを自動化し、エラーが発生しやすい手動デプロイに依存せずにアプリケーションの更新の複雑さを処理するデプロイサービスを設定する: AWS CodeDeploy は、Amazon EC2、AWS Fargate、AWS Lambda、オンプレミスサーバーなどのさまざまなコンピューティングサービスへのソフトウェアデプロイを自動化します。これらのステップを設定するには、「CodeDeploy の開始方法」を参照してください。
- リリースパイプラインを自動化する継続的な配信サービスを設定して、アプリケーションとインフラストラクチャの更新をより迅速かつ確実にする: リリースパイプラインの自動化に役立つ AWS CodePipeline の使用を検討してください。詳細については、「CodePipeline チュートリアル」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:

- OPS05-BP04 構築およびデプロイ管理システムを使用する
- OPS05-BP10 統合とデプロイを完全自動化する
- OPS06-BP02 デプロイをテストする
- OPS06-BP04 テストとロールバックを自動化する

関連ドキュメント:

- AWS CodePipeline を利用したネストされた AWS CloudFormation スタックの継続的デリバリー
- AWS CodeCommit、AWS CodeBuild、AWS CodeDeploy、AWS CodePipelineでCI/CDを完成
- 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

- 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
- Amazon Builders' Library: 継続的デリバリーによる高速化
- AWS CodePipeline とは
- What is CodeDeploy?
- AWS Systems Manager Patch Manager
- Amazon SESとは
- Amazon Simple Notification Service とは

関連動画:

- AWS Summit 2019: AWS における CI/CD

障害管理

Questions

- REL 9. データのバックアップはどのように行うのですか。
- REL 10. ワークロードを保護するための障害分離はどのように使用するのですか。
- 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. どのように信頼性をテストしますか?
- REL 13. ディザスタリカバリ (DR) はどのように計画するのですか?

REL 9. データのバックアップはどのように行うのですか。

データ、アプリケーション、設定をバックアップすると、目標復旧時間 (RTO) や目標復旧時点 (RPO) の要件を満たすことができます。

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てバックアップする、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てバックアップする、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。データを喪失した場合は、実装された戦略によって、定義された RPO および RTO 内でデータを復旧または再生成できます。

クラウド成熟フェーズ: 基礎

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を活用しない場合のリスクレベル: 高

実装のガイダンス

すべての 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 のサービスでは、バックアップを別の AWS リージョンにコピーできます。AWS Backup は、AWS のサービス間でデータ保護を一元化および自動化する機能を提供するツールです。AWS Elastic Disaster Recovery では、完全なサーバーワークロードをコピーし、オンプレミス、クロス AZ、またはクロスリージョンの継続的なデータ保護を維持し、目標復旧時点 (RPO) を秒単位で測定できます。

Amazon S3 をセルフマネージドデータソースおよび 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。サードパーティーのバックアップソフトウェアも使用できます。

AWS Storage Gateway または AWS DataSync を使用して、オンプレミスのデータを AWS クラウドにバックアップできます。Amazon S3 バケットを使用して、このデータを AWS に保存できます。Amazon S3 は、Amazon S3 Glacier や S3 Glacier Deep Archive などの複数のストレージ階層を提供し、データストレージのコストを削減します。

他のソースからデータを再生成することによって、データリカバリのニーズを満たすこともできます。例えば、Amazon ElastiCache レプリカノードまたは Amazon RDS リードレプリカを使用して、プライマリを喪失した場合にデータを再現できます。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) を満たせる場合は、バックアップが不要な可能性があります。別の例として、Amazon EMR を使用している場合、Amazon S3 から Amazon EMR にデータを再現できる限り、HDFS データストアをバックアップする必要がない可能性があります。

バックアップ戦略を選択するときは、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、バックアップの種類 (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さに応じて異なります。この時間は、ワークロードの RTO 以内である必要があります。

実装手順

1. ワークロードのすべてのデータソースを特定する。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多数のリソースに保管できます。「リソース」セクションを参照して、データが保管されているさまざまな AWS の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. 重要性に基づいてデータソースを分類する。データセットごとにワークロードに対する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度が高く、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 の要件も異なります。
3. AWS またはサードパーティーのサービスを使用して、データのバックアップを作成します。AWS Backup は、AWS 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 1 秒未満の自動データ複製を処理します。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。以下に記載されている「リソース」を参照して、さまざまな AWS のサービスからデータバックアップを作成する方法に関する情報を確認してください。

4. バックアップしないデータにデータ再生成メカニズムを確立する。さまざまな理由から、他のソースから再現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現するよりも長く、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現する方法について、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現できるデータの例にあてはまるかもしれません。これらの分析結果がどこかに保管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によって、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としては他にも、キャッシュ (Amazon ElastiCache など) や RDS リードレプリカなどが挙げられます。
5. データをバックアップするサイクルを確立する。データソースのバックアップ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損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

関連ドキュメント:

- AWS Backup とは
- AWS DataSync とは
- ボリュームゲートウェイとは
- APN パートナー: バックアップを支援できるパートナー
- AWS Marketplace: バックアップに活用できる製品
- Amazon EBS スナップショット

- Amazon EFS のバックアップ
- Amazon FSx for Windows File Server のバックアップ
- ElastiCache for Redis のバックアップと復元
- Neptune での DB クラスタースナップショットの作成
- DB スナップショットの作成
- スケジュールに従ってトリガーする EventBridge ルールの作成
- Amazon S3 によるクロスリージョンレプリケーション
- EFS-to-EFS AWS Backup
- Amazon S3 へのログデータのエクスポート
- オブジェクトのライフサイクル管理
- DynamoDB のオンデマンドバックアップと復元
- DynamoDB のポイントインタイムリカバリ
- Amazon OpenSearch Service でのインデックススナップショットの操作
- AWS Elastic Disaster Recovery とは

関連動画:

- 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 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: AWS Backup の詳細と Rackspaceについて (STG341)

関連する例:

- Well-Architected ラボ - Bi-Directional Cross-Region Replication (CRR) for Amazon S3 を実装する
- Well-Architected ラボ: データのバックアップと復元のテスト
- Well-Architected ラボ: 分析ワークロードのフェイルバックによるバックアップと復元
- Well-Architected ラボ: ディザスタリカバリ - バックアップと復元

REL09-BP02 バックアップを保護し、暗号化する

認証と認可を使用して、バックアップへのアクセスを制御および検出します。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 Identity and Access Management (IAM) などによる認証と認可を使用して、バックアップへのアクセスを制御および検出します。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、保管中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。また、自分でキーを用意し、そのキーを管理することもできます。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 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 ボリュームを暗号化します。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、保管中のデータをすべて暗号化します。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 に保存されているデータを暗号化します。ファイルシステムを作成するときに暗号化を設定します。
- 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。
- デフォルトまたはカスタムの Elastic Disaster Recovery 用 Amazon EBS 暗号化を使用するかどうかを選択します。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管中の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

関連ドキュメント:

- AWS Marketplace: バックアップに活用できる製品
- Amazon EBS 暗号化
- Amazon S3: 暗号化によるデータの保護
- CRR 追加設定: AWS KMS に保存された暗号化キーを使用したサーバー側の暗号化 (SSE) で作成されたオブジェクトをレプリケートする
- 保管時の DynamoDB 暗号化
- Amazon RDS リソースを暗号化する
- Amazon EFS のデータとメタデータの暗号化
- AWS でのバックアップの暗号化
- 暗号化テーブルの管理
- セキュリティの柱 - AWS Well-Architected Framework
- AWS Elastic Disaster Recovery とは

関連する例:

- Well-Architected ラボ: Amazon S3 の双方向クロスリージョンレプリケーション (CRR) を実装する

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) によって通知される定期的なスケジュール、またはデータセット内の変更に基づいて、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。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活用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、AWS データソースの自動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。これらのサービスはネイティブバックアップ機能も提供します。ポイントインタイムリカバリによる自動バックアップを提供する AWS サービスには、Amazon DynamoDB、Amazon RDS、Amazon Keyspaces (Apache Cassandra 向け) などがあります。これらは、バックアップ履歴内の特定の時点に復元できます。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 および Amazon DynamoDB は、ポイントインタイムリカバリと継続的なバックアップを提供します。Amazon S3 バージョニングは、有効化すると自動的に行われます。Amazon EBS スナップショットの作成、保持、削除を自動化するには、Amazon Data Lifecycle Manager を使うこと

ができます。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。

バージョニングに加えて、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. 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、「REL09-BP01 バックアップが必要なすべてのデータを特定してバックアップする、またはソースからデータを再現する」を参照してください。
2. ワークロードの RPO を決定します。詳細については、「REL13-BP01 ダウンタイムやデータ損失に関する復旧目標を定義する」を参照してください。
3. 自動バックアップまたはマネージドサービスを使用します。AWS Backup はフルマネージド型のバックアップサービスであり、AWS のサービス、クラウド内、およびオンプレミス間で簡単に一元化およびデータ保護を自動化できます。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を作成します。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 を使用して自動バックアップを作成する方法の実践的なガイダンスについては、「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. オンプレミスデータソースおよびメッセージキューなど、自動バックアップソリューションやマネージドサービスによってサポートされていないデータソースに関しては、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 に基づく頻度で呼び出せます。

実装計画に必要な工数レベル: 低

リソース

関連ドキュメント:

- APN パートナー: バックアップを支援できるパートナー
- AWS Marketplace: バックアップに活用できる製品
- スケジュールに従ってトリガーする EventBridge ルールの作成
- AWS Backup とは
- AWS Step Functions とは
- AWS Elastic Disaster Recovery とは

関連動画:

- AWS re:Invent 2019: AWS Backup の詳細と Rackspace について (STG341)

関連する例:

- Well-Architected ラボ: データのバックアップと復元のテスト

REL09-BP04 データの定期的な復旧を行ってバックアップの完全性とプロセスを確認する

復旧テストを実行して、バックアッププロセスの実装が目標復旧時間 (RTO) と目標復旧時点 (RPO) を満たしていることを検証します。

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ことで、データの紛失や破損を心配せずに、必要なときにデータを復元できること、ワークロードの RTO 内で復元と復旧が可能であること、ならびにデータ損失がワークロードの RPO の範囲内であることを確認できます。

このベストプラクティスを活用しない場合のリスクレベル: 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、テスト環境を構築し、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、設定されたポリシーに基づいて時間の経過とともにポイントインタイム状態が記録されます。Elastic Disaster Recovery は、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して、スナップショットの整合性を検証するのに役立ちます。

実装手順

1. バックアップ中のデータソースと、これらのバックアップの保存場所を特定します。実装のガイダンスについては、「REL09-BP01 バックアップが必要なすべてのデータを特定してバックアップする、またはソースからデータを再現する」を参照してください。
2. 各データソースのデータ検証の基準を確立します。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. データの重要性に基づいた RTO と RPO を確立します。実装のガイダンスについては、「REL13-BP01 ダウンタイムやデータ損失に関する復旧目標を定義する」を参照してください。
4. データ復旧機能を評価します。バックアップおよび復元戦略をレビューして、RTO および RPO を満たせるかどうかを理解し、必要に応じて戦略を調整します。AWS Resilience Hub を使用して、ワークロードのアセスメントを実行できます。アセスメントは、回復力ポリシーに対してアプリケーション設定を評価し、RTO および RPO 目標を満たすことができるかどうかを報告します。
5. 本番環境で確立済みのデータ復元プロセスを使用して、テスト復元を行います。プロセスは、オリジナルデータソースのバックアップ方法、バックアップそのもののフォーマットとストレージ場所、またはデータが他のソースから再生成されるかどうかによって異なります。例えば、AWS Backup などのマネージドサービスを使用している場合、バックアップを新しいリソース に復元するだけで済みます。AWS Elastic Disaster Recovery を使用した場合、リカバリドリルを起動します。
6. データ検証のために以前に確立した基準に基づいて、復元されたリソースからのデータ復旧を検証します。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. 復元と復旧に必要な時間を測定し、確立された RTO と比較します。このプロセスは、ワークロードの RTO の範囲内ですか? 例えば、復元プロセスが開始されたときのタイムスタンプと復旧検証が完了したときのタイムスタンプを比較して、このプロセスの所要時間を計算します。すべての AWS API 呼び出しにはタイムスタンプが付けられており、AWS CloudTrail で情報を確認できます。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。自動プロセスを使用する場合、Amazon DynamoDB などのサービスを使用してこの情報を保存できます。さらに、多くの AWS のサービスは、特定のアクションが発生したときのタイムスタンプ付きの情報を提供するイ

ベント履歴を備えています。AWS Backup 内では、バックアップおよび復元アクションはジョブと呼ばれ、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. 復元と復旧に必要な時間がワークロードについて確立された RTO を超えている場合は、関係者に通知します。このラボで示すように自動化を実装する場合、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関係者に送信できます。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 を使用してタスクを OpsItems として作成したりできます。
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呼び出すことができます。AWS Backup でデータリカバリの検証を自動化する方法を学びます。この Well-Architected ラボでは、いくつかのステップを自動化するための方法を解説します。

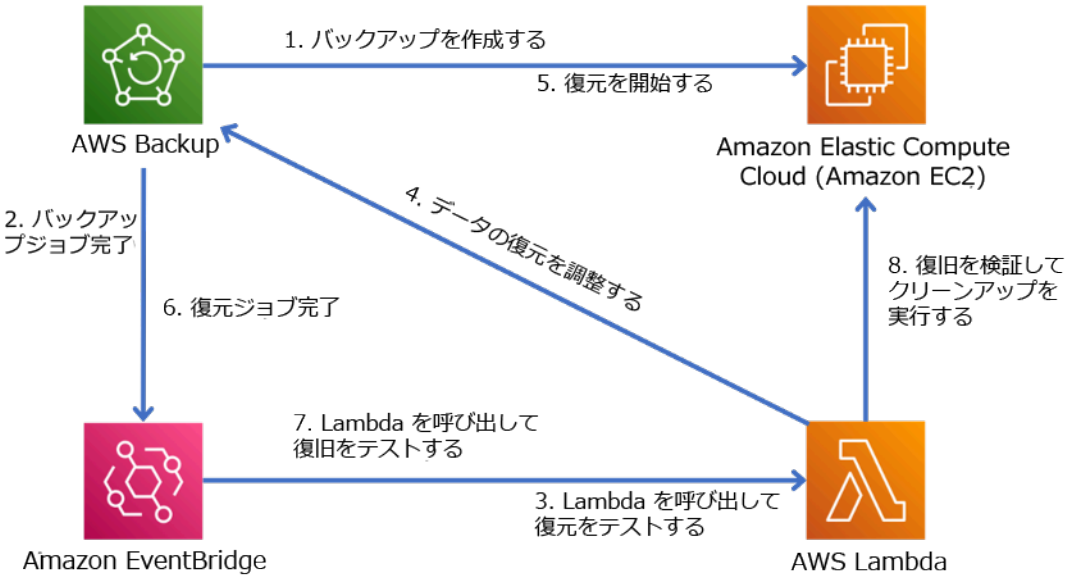


図 9: 自動化されたバックアップおよび復元プロセス

実装計画に必要な工数レベル: 検証基準の複雑さに応じて中から高

リソース

関連ドキュメント:

- AWS Backup でデータリカバリ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: バックアップに活用できる製品
- スケジュールに従ってトリガーする EventBridge ルールの作成
- DynamoDB のオンデマンドバックアップと復元
- AWS Backup とは
- AWS Step Functions とは
- AWS Elastic Disaster Recovery とは
- AWS Elastic Disaster Recovery

関連する例:

- Well-Architected ラボ: データのバックアップと復元のテスト

REL 10. ワークロードを保護するための障害分離はどのように使用するのですか。

障害を分離した境界によって、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限定できます。境界の外部のコンポーネントは、障害の影響を受けません。障害を分離した複数の境界を使用して、ワークロードへの影響を制限することができます。

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ー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。

アベイラビリティーゾーン (AZ)

AWS リージョンは、互いに独立するように設計された複数のアベイラビリティーゾーンで構成されます。各アベイラビリティーゾーンは、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ーゾーンは、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、アベイラビリティー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 つだけで済みます。地理的には分離されていても、アベイラビリティーゾーンは、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ーゾーンにまたがり、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ーゾーンを使用することができます。

アベイラビリティー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ー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、アベイラビリティー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ー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。

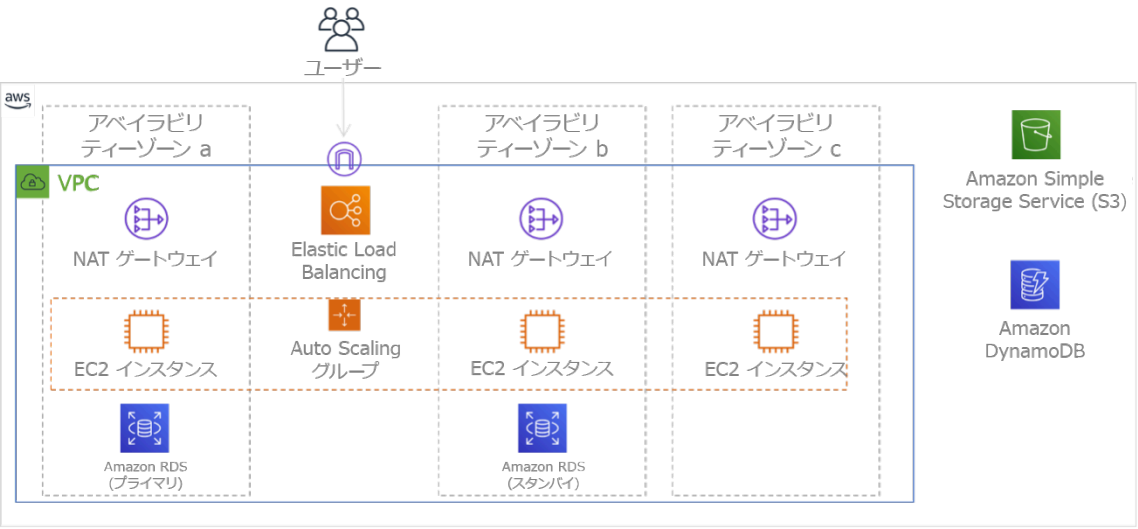


図 9: 3 つのアベイラビリティーゾーンにまたがってデプロイされる多階層アーキテクチャ。Amazon S3 と Amazon DynamoDB は常に自動的にマルチ AZ です。ELB も 3 つのゾーンすべてにデプロイされます。

AWS の制御プレーンには通常リージョン全体 (複数のアベイラビリティーゾーン) のリソース管理能力がありますが、一部のコントロールプレーン (Amazon EC2 や Amazon EBS など) は単一アベイラビリティーゾーンに対してデータをフィルタリングする能力を持っています。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 アベイラビリティーゾーンからのみ 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 リージョン内でアベイラビリティー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ています。Amazon CloudFront は、このネットワークを使用して、低レイテンシーでエンドユーザーにコンテンツを配信します。AWSGlobal 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にオンボーディングできます。Amazon API Gateway では、CloudFront ディストリビューションを使用してエッジ最適化 API エンドポイントが可能になり、最も近いエッジロケーションを介したクライアントアクセスが容易になります。

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達成するためのディザスタリカバリ戦略として一般的です。これらの戦略の詳細については、「ディザスタリカバリ (DR) の計画」を参照してください。ここでは、時間の経過とともに平均稼働時間目標を実現しようとする可用性に焦点を当てています。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)。

> **推奨事項**
>
> 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)、および Amazon DynamoDB グローバルテーブルを使用したデータの継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、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と、地理的近接性やレイテンシーベースのポリシーなど、利用可能な他の複数のポリシーをサポートしています。Global Accelerator は、AWS エッジサーバーの広範なネットワークを自動的に活用して、トラフィックを AWS ネットワークバックボーンにできるだけ早くオンボードするため、リクエストのレイテンシーが低くなります。

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貴社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ーゾーンと 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ー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、および 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。Amazon EC2 Auto Scaling、Amazon 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へのインスタンスの分散
    - リージョンとアベイラビリティーゾーンの選択
  - 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するための AWS Lambda 関数の設定
  - ElastiCache クラスターで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の選択
- 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ー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: 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用アーキテクチャパターン (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
  - AWS Outposts とは

- 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:

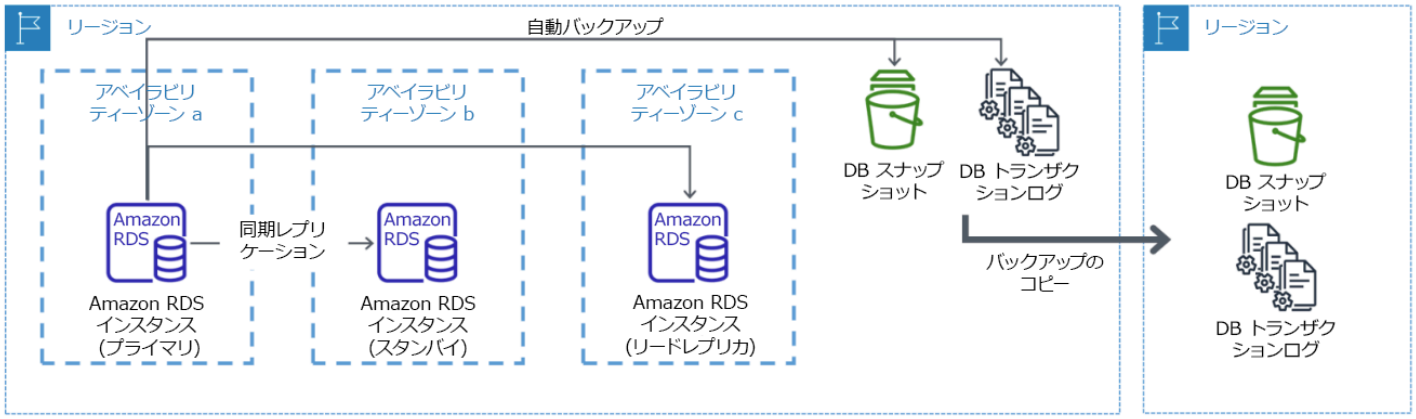
- AWS グローバルインフラストラクチャ
- AWS Local Zones のよくある質問
- Amazon ECS タスク配置戦略
- リージョンとアベイラビリティーゾーンの選択
- 例: 複数のアベイラビリティーゾーンへのインスタンスの分散
- グローバルテーブル: DynamoDB を使用した複数リージョンレプリケーション
- Amazon Aurora Global Database の使用
- AWS のサービスによるマルチリージョンアプリケーションの作成 (ブログシリーズ)
- AWS Outposts とは

関連動画:

- 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用アーキテクチャパターン (ARC209-R2)
- AWS re:Invent 2019: AWS グローバルネットワークインフラストラクチャの革新性とオペレーション (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、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ーゾーン (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。



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、防御レイヤー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 99.99% の可用性の違いは、1 か月あたり 3.5 時間に相当します。期待されるワークロードで 99.99% の可用性を達成するには、複数の AZ 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ー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ー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ーゾーンで構成され、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ー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ーゾーンの中断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ー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ー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 を使用して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。次に Elastic Load Balancing は、異常なゾーンのインスタンスを検出し、トラフィックを正常なゾーンにルーティングするソリューションを提供します。
      i. Application Load Balancer の開始方法
      ii. Network Load Balancer の使用開始
   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ー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 配置
      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 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

c. ワークロードの前に Application Load Balancer 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 の標準アクセラレーターのエンドポイント 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. AWS リージョン間での Amazon EventBridge イベントの送受信
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 Global Database 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 サービス機能については、「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」のブログシリーズを参照してください。

実装計画に必要な工数レベル: 中から高

リソース

関連ドキュメント:

- 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
- 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
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- クラウド内の災害対策は異なる
- グローバルテーブル: DynamoDB を使用した複数リージョンレプリケーション

関連動画:

- 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用アーキテクチャパターン (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:

- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- 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ーゾーンを使用して、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 のディザスタリカバリ
- 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 のデータレジデンシーを構築する方法
- 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

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活用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ロ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することが可能です。EMR File System (EMRFS) を使用することで EMR のデータを Amazon S3 に保存することができ、そのデータを、今度は複数のアベイラビリティーゾーンまたは AWS リージョンを横断してレプリケートすることができます。

同様に、Amazon Redshift でも、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ーゾーンにクラスターがデフォルトでプロビジョニングされます。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ー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は、Elastic Disaster Recovery を使用することでワークロードを別のアベイラビリティーゾーンまたはリージョンに保護することができます。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実現します。

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を利用して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 インスタンスの自動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されている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」
- インスタンスの耐障害性
- Amazon ECS サービスを自動的にスケールする
- 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を使用します。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、多くの一般的な障害はセル自体に封じ込められるため、障害の分離を強化できます。このように障害を分離することで、コードのデプロイに失敗したり、リクエストが破損したり特定の障害モードを呼び出したり (ポイズンピルリクエスト) するなど、他の方法では封じ込めが難しい障害が起きても回復が可能になります。

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: 高

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。境界の外部のコンポーネントは、障害の影響を受けません。障害を分離した複数の境界を使用して、ワークロードへの影響を制限することができます。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、サービスの粒度に従うか、サービスのワークロードをセル間のインタラクションを最小限にして分割できる自然な方法に従う必要があります。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。

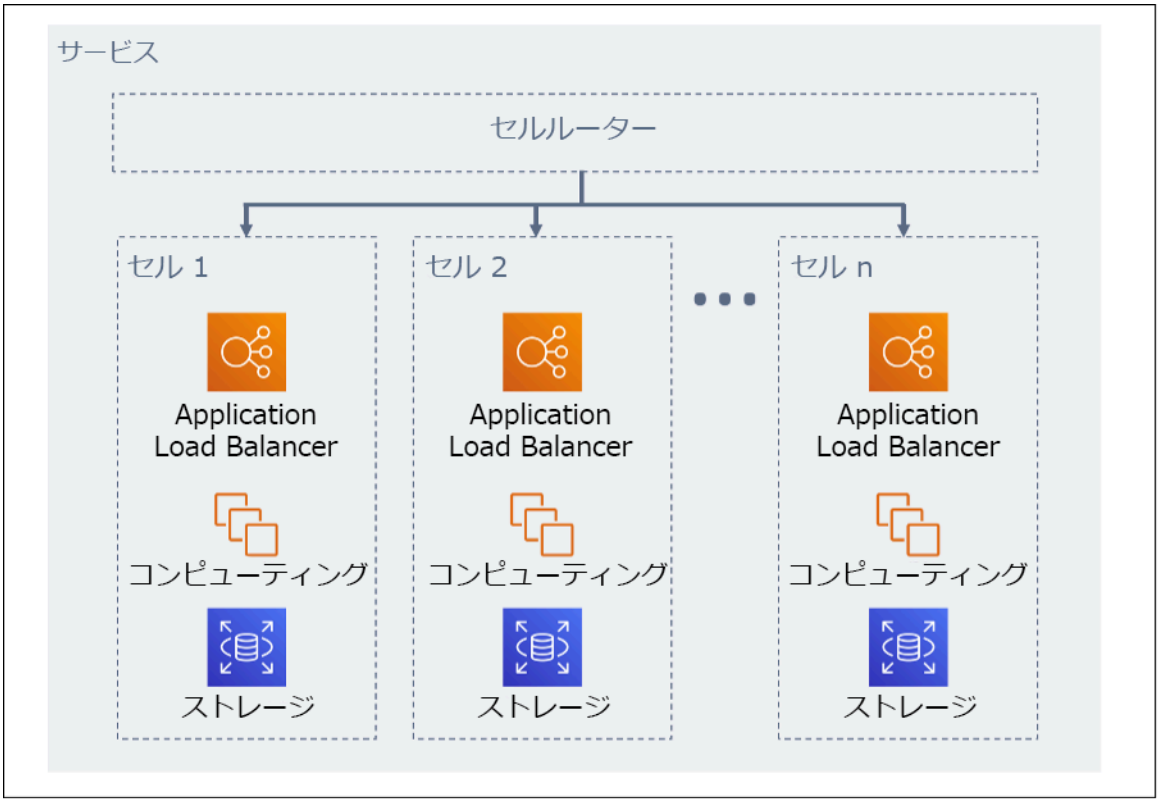


図 11: セルベースアーキテクチャ

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ときは特に注意が必要です。
   - 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。たとえば、`customer ID` や `resource ID` などがあります。
   - 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: アップストリームのサービスは、リソースのライフサイクルを通じて 1 つのセルとしかやり取りできません。
   - 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で共有されたコンポーネントであるため、セルと同じ区分化の戦略をとることはできません。
   - 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「信頼性、動作の継続、1 杯のおいしいコーヒー」で述べているとおり、信頼性の高いシステムを生み反脆弱性を最小限に抑えるものは、シンプルな設計と継続的な取り組みです。
4. セルのサイズ: セルにはサイズの上限を設定する必要があり、それを超えるサイズは許容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、「REL07-BP04 ワークロードの負荷テストを実施する」を参照してください。

- 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ル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: さまざまな障害に対抗するには、耐障害性を多層的に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、「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」を参照してください。
6. コードのデプロイ: コードの変更は、すべてのセルに同時にデプロイするのではなく段階的にデプロイする方法が推奨されます。
   - 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。

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

関連ドキュメント:

- 信頼性、動作の継続、1 杯のおいしいコーヒー
- AWS and Compartmentalization
- シャッフルシャーディングを使ったワークロードの分離
- 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: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

関連する例:

- Well-Architected ラボ - 障害分離とシャッフルシャーディング

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、回復力を考慮して設計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障害やパフォーマンスの低下の発生をすみ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主要業績評価指標 (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、これを測定する主要業績評価指標 (KPI) が検出および修正戦略の一部である必要があります。

期待される成果: ワークロードの重要なコンポーネントは個別にモニタリングされ、障害が発生したタイミングと場所で障害を検出してアラートを出します。

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する。
- 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ない。
- 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ない。
- ワークロードの顧客向けインターフェイスのみがアクティブにモニタリングされる。
- 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ない。
- 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がない。
- 作成されたモニタが多すぎる。

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

モニタリングの対象となるすべてのワークロードを特定します。モニタリングする必要があるワークロードのすべてのコンポーネントを特定したら、次はモニタリングの間隔を決定します。モニタリングの間隔は、障害の検出にかかる時間に基づいた復旧を開始する速さに直接影響します。平均検出時間 (MTTD) は、障害が発生してから修理作業が開始されるまでの時間です。サービスのリストは広範囲かつ完全でなければなりません。

モニタリングは、アプリケーション、プラットフォーム、インフラストラクチャ、ネットワークを含むアプリケーションスタックのすべてのレイヤーを対象とする必要があります。

モニタリング戦略では、Gray failure の影響を考慮する必要があります。Gray failure の詳細については、ホワイトペーパー「Advanced Multi-AZ Resilience Patterns」の「Gray failures」を参照してください。

実装手順

- 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。
- コンポーネントとマネージドサービスの詳細な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。
  - 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、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。
  - Lambda、API Gateway、Amazon EKS、Amazon ECS、すべてのロードバランサーのタイプに不可欠なサーバーレスコンポーネントの、モニタリング要件を決定します。
  - Amazon S3、Amazon FSx、Amazon EFS、Amazon EBS のストレージコンポーネントのモニタリング要件を決定します。
- 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。ワークロードには主要なビジネス機能が実装されており、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。
- ユーザー Canary 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 (「canary テスト」ともいう。「カナリアデプロイ」と混同しないこと) 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。
- 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。
- ワークロードのいずれかの要素が正常に動作していない場合にこれを検出し、リソースを自動的にスケールする時期を示す、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 メールを介して送信したり、Auto Scaling と連携させてワークロードリソースをスケールアップ/スケールダウンするのに使用したりすることができます。
- 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を示したりするために使用できます。

- サービスに分散トレースモニタリングを作成します。分散型モニタリングにより、アプリケーションと基盤となるサービスがどのように動作しているかを理解して、パフォーマンスの問題や エラーの根本原因を特定し、トラブルシューティングすることができます。
- モニタリングシステム (CloudWatch または X-Ray を使用) のダッシュボードとデータ収集を別のリージョンとアカウントに作成します。
- Amazon Health Aware のモニタリングを統合することで、パフォーマンス低下の可能性がある AWS リソースを視覚的にモニタリングできます。ビジネスに不可欠なワークロードの場合、このソリューションによって、AWS サービスに対するプロアクティブでリアルタイムのアラートへのアクセスが可能になります。

リソース

関連するベストプラクティス:

- 可用性
- 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

関連ドキュメント:

- 合成モニタリング (canary)
- インスタンスの詳細モニタリングを有効または無効にする
- 拡張モニタリング
- Auto Scaling グループとインスタンスを Amazon CloudWatch でモニタリングする
- カスタムメトリクスを発行する
- Amazon CloudWatch でのアラームの使用
- Amazon CloudWatch ダッシュボードの使用
- クロスアカウントクロスリージョンダッシュボード
- AWS X-Ray のよくある質問
- 可用性について
- AWS Health Aware — 組織および個人の AWS アカウントの AWS Health アラートのカスタマイズ

関連動画:

- グレー障害の緩和

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高める
- 1 つのオブザーバビリティワークショップ: X-Ray の探究

関連ツール:

- CloudWatch
- CloudWatch X-Ray

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します。ロケーション障害 (アベイラビリティーゾーンや 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。

サービスを設計するときは、リソース、アベイラビリティーゾーン、またはリージョンに負荷を分散します。そのため、個々のリソースの障害は、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。障害発生時にサービスがどのように検出され、ルーティングされるかを検討してください。

障害復旧を念頭に置いてサービスを設計します。AWS では、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。

フェイルオーバーを可能にするパターンとデザインは、AWS プラットフォームサービスごとに異なります。AWS ネイティブのマネージドサービスの多くは、複数のアベイラビリティーゾーン (Lambda や API Gateway など) にネイティブに対応しています。他の AWS サービス (EC2 や EKS など) では、AZ 間でのリソースまたはデータストレージのフェイルオーバーをサポートするための特定のベストプラクティス設計が必要です。

モニタリングは、フェイルオーバーリソースが正常であることを確認し、リソースのフェイルオーバーの進行状況を追跡して、ビジネスプロセスの復旧をモニタリングするために設定する必要があります。

期待される成果: システムは、新しいリソースを自動または手動で使用してパフォーマンスの低下から復旧できます。

一般的なアンチパターン:

- 障害を想定した計画が、計画と設計の段階に含まれていない。
- RTO と RPO が確立されていない。
- モニタリングが不十分で、障害が発生しているリソースを検出できない。
- 障害ドメインの適切な隔離。
- マルチリージョンのフェイルオーバーが考慮されていない。
- フェイルオーバーを決定する際の障害検出の感度が高すぎる、または過剰である。
- フェイルオーバー設計のテストや検証を行っていない。
- 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。
- すぐにフェイルバックするのを防ぐための減衰期間を十分に設けていない。

このベストプラクティスを活用するメリット: 適切に機能低下し、迅速に回復することで、障害発生時でも信頼性を維持する耐障害性の高いシステムを構築できます。

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

Elastic Load Balancing や Amazon EC2 Auto Scaling などの AWS サービスは、複数のリソースおよびアベイラビリティー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、アベイラビリティーゾーンの障害を、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、設計はさらに複雑です。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。ただし、リードレプリカをプライマリに昇格させ、トラフィックを新しいエンドポイントに向けるには、やはりフェイルオーバーが必要です。Amazon Route 53、Amazon Application Recovery Controller (ARC)、Amazon CloudFront、AWS Global Accelerator は、AWS リージョン間でトラフィックをルーティングするために役立ちます。

Amazon S3、Lambda、API Gateway、Amazon SQS、Amazon SNS、Amazon SES、Amazon Pinpoint、Amazon ECR、AWS Certificate Manager、EventBridge、Amazon DynamoDB などの AWS サービスは、AWS によって複数のアベイラビリティーゾーンに自動的にデプロイされます。

障害が発生した場合、これらの AWS サービスはトラフィックを正常な口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ーゾーンに冗長的に保存され、使用可能な状態を維持します。

Amazon RDS、Amazon Aurora、Amazon Redshift、Amazon EKS、Amazon ECS の場合、マルチ AZ は設定オプションです。フェイルオーバーが開始すると、AWS はトラフィックを正常なインスタンスに転送させることができます。このフェイルオーバーアクションは、AWS が行うことも、必要に応じてお客様が実行することもできます。

Amazon EC2 インスタンス、Amazon Redshift、Amazon ECS タスク、Amazon EKS ポッドの場合は、デプロイ先のアベイラビリティーゾーンを選択します。設計によっては、Elastic Load Balancing に、異常なゾーンにあるインスタンスを検出してトラフィックを正常なゾーンにルーティングするソリューションが用意されています。Elastic Load Balancing は、オンプレミスのデータセンター内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トラフィックフェイルオーバーの場合、再ルーティングでは、インターネットドメインを定義し、ヘルスチェックなどのルーティングポリシーを割り当ててトラフィックを正常なリージョンにルーティングする手段として、Amazon Route 53、Amazon Application Recovery Controller、AWS Global Accelerator、Route 53 Private DNS for VPC、または CloudFront を活用できます。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、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任意の AWS リージョンのエンドポイントにルーティングすることで、パフォーマンスと信頼性を高めます。

実装手順

- すべての適切なアプリケーションとサービスのフェイルオーバー設計を作成します。各アーキテクチャコンポーネントを分離し、各コンポーネントの RTO と RPO を満たすフェイルオーバー設計を作成します。
- フェイルオーバープランに必要なすべてのサービスを使用して、下位環境 (開発環境やテスト環境など) を構成します。Infrastructure as Code (IaC) を使用してソリューションをデプロイし、再現性を確保します。
- フェイルオーバー設計を実装してテストするために、2 つ目のリージョンなどの復旧サイトを設定します。必要に応じて、テスト用のリソースを一時的に設定して、追加コストを抑えることができます。
- どのフェイルオーバープランを AWS で自動化するか、どのフェイルオーバープランを DevOps プロセスで自動化できるか、どのフェイルオーバープランを手動で行うかを判断します。各サービスの RTO と RPO を文書化して測定します。

- フェイルオーバープレイブックを作成し、各リソース、アプリケーション、サービスをフェイルオーバーするためのすべての手順を含めます。
- フェイルバックプレイブックを作成し、各リソース、アプリケーション、サービスをフェイルバックするためのすべての手順を (タイミングとともに) 含めます。
- プレイブックを開始してリハーサルするための計画を立てます。シミュレーションとカオステストを使用して、プレイブックの手順と自動化をテストします。
- ロケーション障害 (アベイラビリティーゾーンや 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。フェイルオーバーテストの前に、クォータ、自動スケーリングのレベル、実行中のリソースを確認してください。

リソース

関連する Well-Architected のベストプラクティス:

- REL13 - ディザスタリカバリ (DR) を計画する
- REL10 - 障害部分を切り離してワークロードを保護する

関連ドキュメント:

- RTO と RPO のターゲットを設定する
- Route 53 加重ルーティングを使用したフェイルオーバー
- Amazon Application Recovery Controller を使用したディザスタリカバリ
- EC2 with autoscaling
- ECS Deployments - Multi-AZ
- ECS Deployments - Multi-AZ
- Amazon Application Recovery Controller を使用したトラフィックの切り替え
- Lambda を使用した Application Load Balancer リクエストの処理
- ACM Replication and Failover
- パラメータストアのレプリケーションとフェイルオーバー
- ECR クロスリージョンレプリケーションとフェイルオーバー
- Secrets Manager クロスリージョンレプリケーションの設定
- 新機能 – Amazon Elastic File System (EFS) のレプリケーション

- EFS クロスリージョンレプリケーションとフェイルオーバー
- ネットワークフェイルオーバー
- MRAP を使用した S3 エンドポイントフェイルオーバー
- S3 のクロスリージョンレプリケーションの作成
- AWS でのクロスリージョンフェイルオーバーとグレースフルフェイルバックに関するガイダンス
- マルチリージョンの Global Accelerator によるフェイルオーバー
- DRS によるフェイルオーバー
- Amazon Route 53 を用いたディザスタリカバリ (DR) のメカニズム

関連する例:

- AWS でのディザスタリカバリ
- AWS の Elastic Disaster Recovery

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。パフォーマンスの低下は、内部のサービスメカニズムによって自動的に修復される場合もあれば、修復アクションによってリソースを再起動または削除する必要がある場合もあります。

セルフマネージドアプリケーションやクロスリージョン修復では、復旧設計と自動修復プロセスを既存のベストプラクティスから引き出すことができます。

リソースを再起動または削除する機能は、障害を修復するための重要なツールです。ベストプラクティスは、可能な限りサービスをステートレスにすることです。これにより、リソースの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 (コンピューティングインスタンス、サーバーレス関数など) 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。

再起動または再試行は、ネットワークリクエストにも適用されます。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。再起動の機能は、復旧指向コンピューティング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。

期待される成果: 障害の検出を修正するために、自動アクションが実行されます。

一般的なアンチパターン:

- 自動スケーリングなしでリソースをプロビジョニングする。
- 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する。
- 自動復旧を使用せずに、複数のロケーションにデプロイできないアプリケーションをデプロイします。
- 自動スケーリングと自動復旧が修復に失敗するアプリケーションを手動で修復する。
- フェイルオーバーデータベースの自動化はない。
- トラフィックを新しいエンドポイントに再ルーティングする自動化された方法が足りない。
- ストレージのレプリケーションはない。

このベストプラクティスを活用するメリット: 自動修復により、平均回復時間を短縮し、可用性を向上させることができます。

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

Amazon EKS やその他の Kubernetes サービスの設計には、レプリカセットまたはステートフルセットの最小値と最大値の両方、およびクラスターとノードグループの最小サイズが含まれている必要があります。これらのメカニズムは、Kubernetes コントロールプレーンを使用して障害を自動的に修正しながら、継続的に利用可能な処理リソースを最小限に抑えます。

コンピューティングクラスターを使用してロードバランサーを介してアクセスされる設計パターンでは、Auto Scaling グループを活用する必要があります。Elastic Load Balancing (ELB) は、受信アプリケーショントラフィックを 1 つ以上のアベイラビリティーゾーン (AZ) にある複数のターゲットと仮想アプライアンスに自動的に分散します。

ロードバランシングを使用しないクラスター化されたコンピューティングベースの設計では、少なくとも 1 台のノードが失われることを想定したサイズにする必要があります。これにより、新しいノードを復旧している間も、サービスが容量が減少する可能性がある状態で稼働し続けることができます。サービスの例としては、Mongo、DynamoDB Accelerator、Amazon Redshift、Amazon

EMR、Cassandra、Kafka、MSK-EC2、Couchbase、ELK、Amazon OpenSearch Service などがあります。これらのサービスの多くは、追加の自動修復機能を使用して設計できます。一部のクラスターテクノロジーでは、ノードが失われたときにアラートを生成し、新しいノードを再作成するための自動または手動のワークフローをトリガーする必要があります。このワークフローは、AWS Systems Manager を使用して自動化でき、問題を迅速に修正できます。

Amazon EventBridge を使用すれば、CloudWatch アラームなどのイベントや、その他の AWS サービスの状態の変化などを、モニタリングおよびフィルタリングすることができます。イベント情報に基づいて、AWS Lambda、Systems Manager Automation、または他のターゲットを呼び出して、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みなして代替インスタンスを起動します。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、静的安定性が高可用性のために優先されます。

実装手順

- 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。Auto Scaling 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、キャパシティを追加および削除できます。
- 前述のコンピューティングインスタンスの場合は、ロードバランシングを使用して適切なロードバランサーのタイプを選択します。
- Amazon RDS の修復を検討します。スタンバイインスタンスでは、スタンバイインスタンスへの自動フェイルオーバーを設定します。Amazon RDS リードレプリカの場合、リードレプリカをプライマリにするには自動化されたワークフローが必要です。
- 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、EBS ボリュームと Amazon Elastic File System または File Systems for Lustre および Windows へのマウントポイントも保持されます。AWS OpsWorks を使用することで、レイヤーレベルで EC2 インスタンスの自動修復を設定できます。
- 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。
- Amazon EventBridge を使用すれば、CloudWatch アラームなどのイベントや、その他の AWS サービスの状態の変化などを、モニタリングおよびフィルタリングすることができます。イベント

情報に基づいて、AWS Lambda (または他のターゲット) を呼び出し、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。

リソース

関連するベストプラクティス:

- 可用性
- 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

関連ドキュメント:

- AWS Auto Scaling の仕組み
- Amazon EC2 の自動復旧
- 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: 障害の発生したインスタンスを自動修復機能によって交換する
- What is AWS Step Functions?
- What is AWS Lambda?
- What Is Amazon EventBridge?
- Amazon CloudWatch でのアラームの使用
- Amazon RDS フェイルオーバー
- SSM - Systems Manager Automation
- 回復力のあるアーキテクチャのベストプラクティス

関連動画:

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

関連する例:

- Auto Scaling ワークショップ
- Amazon RDS フェイルオーバーワークショップ

関連ツール:

- CloudWatch
- CloudWatch X-Ray

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作成、読み取り、記述、更新、削除、一覧表示 (CRUDL) に使用される管理 API を提供し、データプレーンは日々のサービストラフィックを処理します。回復力に影響を与える可能性のあるイベントへの回復または軽減対応を実装するときは、サービスの回復、再スケーリング、復元、修復、またはフェイルオーバーに最小限のコントロールプレーンのオペレーションを使用することに焦点を当ててください。これらのパフォーマンスの低下イベント中は、データプレーンのアクションがどのアクティビティよりも優先されるはずです。

例えば、新しいコンピューティングインスタンスの起動、ブロックストレージの起動、キューサービスの記述などは、すべてコントロールプレーンのアクションです。コンピューティングインスタンスを起動後、コントロールプレーンは、容量のある物理ホストの検索、ネットワークインターフェイスの割り当て、ローカルブロックストレージボリュームの準備、認証情報の生成、セキュリティルールの追加など、複数のタスクを実行する必要があります。コントロールプレーンは複雑なオーケストレーションになりがちです。

期待される成果: リソースに障害が発生すると、システムは、トラフィックを障害のあるリソースから正常なリソースに移行することで、自動または手動で回復できます。

一般的なアンチパターン:

- トラフィックを再ルーティングするための DNS レコードの変更への依存。
- リソースのプロビジョニングが不十分なため、障害が発生したコンポーネントの交換をコントロールプレーンのスケーリング操作に依存。
- あらゆるカテゴリの障害を修復するために、広範囲にわたるマルチサービス、マルチ API コントロールプレーンアクションに依存。

このベストプラクティスを活用するメリット: 自動修復の成功率が高くなると、平均復旧時間が短縮され、ワークロードの可用性が向上します。

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: 中。特定の種類のサービス低下の場合、コントロールプレーンが影響を受けます。コントロールプレーンを広範囲に使用して修復すると、復旧時間 (RTO) と平均復旧時間 (MTTR) が長くなる可能性があります。

実装のガイダンス

データプレーンのアクションを制限するには、サービスの復元に必要なアクションについて各サービスを評価します。

DNS トラフィックをシフトするときは Amazon Application Recovery Controller を活用します。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機能を継続的にモニタリングし、複数の AWS リージョン、アベイラビリティーゾーン、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 ルーティングポリシーにコントロールプレーンが使用されているため、復旧の際にコントロールプレーンに依存しないようにします。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。これらはグローバルに配布され、「100% 利用可能」のサービスレベルアグリーメント (SLA) 向けに設計されています。

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、米国東部 (バージニア北部) 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。

コンピューティングインフラストラクチャは静的に安定するように設計して、インシデント中にコントロールプレーンを使用しないようにします。例えば、Amazon EC2 インスタンスを使用している場合は、新しいインスタンスを手動でプロビジョニングしたり、Auto Scaling グループにインスタンスの追加を指示したりすることは避けてください。回復性を最大限に高めるには、フェイルオーバーに使用するクラスターに十分な容量をプロビジョニングしてください。この容量のしきい値を制限する必要がある場合は、エンドツーエンドのシステム全体にスロットルを設定して、限られたリソースに到達するトラフィックの合計を安全に制限してください。

Amazon DynamoDB、Amazon API Gateway、ロードバランサー、AWS Lambda サーバーレスなどのサービスでは、これらのサービスを使用するとデータプレーンを活用できます。ただし、新しい関数、ロードバランサー、API ゲートウェイ、または DynamoDB テーブルの作成はコントロールプレーンのアクションであり、イベントの準備やフェイルオーバーアクションのリハーサルとして、パ

フォーマンス低下の前に完了しておく必要があります。Amazon RDS では、データプレーンのアクションによりデータへのアクセスが可能になります。

データプレーン、コントロールプレーン、および、AWS が高可用性の目標を満たすサービスをいかに構築しているのか、についての詳細は、「アベイラビリティーゾーンを使用した静的安定性」を参照してください。

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。

実装手順

パフォーマンス低下のイベント後に復元する必要のある各ワークロードについて、フェイルオーバーランブック、高可用性設計、自動修復設計、または HA リソース復元プランの評価を行います。コントロールプレーンのアクションとみなされる可能性のある各アクションを特定します。

コントロールアクションをデータプレーンアクションに変更することを検討します。

- 自動スケーリング (コントロールプレーン) を事前スケールされた Amazon EC2 リソース (データプレーン) に変更します。
- Amazon EC2 インスタンススケーリング (コントロールプレーン) を AWS Lambda スケーリング (データプレーン) に変更します。
- Kubernetes を使用するあらゆる設計とコントロールプレーンのアクションの性質を評価します。ポッドの追加は Kubernetes のデータプレーンのアクションです。アクションはノードの追加ではなく、ポッドの追加に限定する必要があります 過剰にプロビジョニングされたノードを使用することは、コントロールプレーンのアクションを制限するための推奨される方法です。

データプレーンのアクションが同じ修復に影響を与えられる代替アプローチを検討してください。

- Route 53 レコードの変更 (コントロールプレーン) または Amazon Application Recovery Controller (データプレーン)
- 自動化がさらに進んだアップデート用の Route 53 ヘルスチェック

サービスがミッションクリティカルな場合は、影響を受けていないリージョンでより多くのコントロールプレーンとデータプレーンのアクションを実行できるように、セカンダリリージョンのサービスを検討してください。

- プライマリリージョンの Amazon EC2 Auto Scaling または Amazon EKS と、セカンダリリージョンの Amazon EC2 Auto Scaling または Amazon EKS とを比較し、セカンダリリージョンにトラフィックをルーティングします (コントロールプレーンのアクション)。

- セカンダリプライマリでリードレプリカを作成するか、プライマリリージョンで同じアクションを試みる (コントロールプレーンのアクション)

リソース

関連するベストプラクティス:

- 可用性
- 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: AWS Marketplace: 耐障害性に活用できる製品
- Amazon Builders' Library: 小さなサービスに制御を任せて分散型システムのオーバーヘッドを回避する
- 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)
- AWS Elemental MediaStore Data Plane
- Amazon Application Recovery Controller を使用して回復力の高いアプリケーションを構築する、パート 1: 単一リージョンスタック
- Amazon Application Recovery Controller を使用して回復力の高いアプリケーションを構築する、パート 2: マルチリージョンスタック
- Amazon Route 53 を用いたディザスタリカバリ (DR) のメカニズム
- Amazon Application Recovery Controller とは
- Kubernetes のコントロールプレーンとデータプレーン

関連動画:

- Back to Basics - Using Static Stability
- Building resilient multi-site workloads using AWS global services

関連する例:

- Amazon Application Recovery Controller の紹介

- Amazon Builders' Library: 小さなサービスに制御を任せて分散型システムのオーバーヘッドを回避する
- Amazon Application Recovery Controller を使用して回復力の高いアプリケーションを構築する、パート 1: 単一リージョンスタック
- Amazon Application Recovery Controller を使用して回復力の高いアプリケーションを構築する、パート 2: マルチリージョンスタック
- アベイラビリティーゾーンを使用した静的安定性

関連ツール:

- Amazon CloudWatch
- AWS X-Ray

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

ワークロードは静的に安定し、1 つの通常モードでのみ動作する必要があります。バイモーダル動作とは、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。

例えば、別のアベイラビリティーゾーン (AZ) で新しいインスタンスを起動して、AZ の障害からの復旧を試みることができます。これにより、障害モード中にバイモーダル応答が発生する可能性があります。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。この例では、これらのインスタンスは障害発生前に 2 番目の AZ でプロビジョニングしておく必要があります。この静的に安定した設計により、確実にワークロードが単一のモードでのみ動作するようになります。

期待される成果: ワークロードは、通常モードと障害モードでバイモーダル動作を示しません。

一般的なアンチパターン:

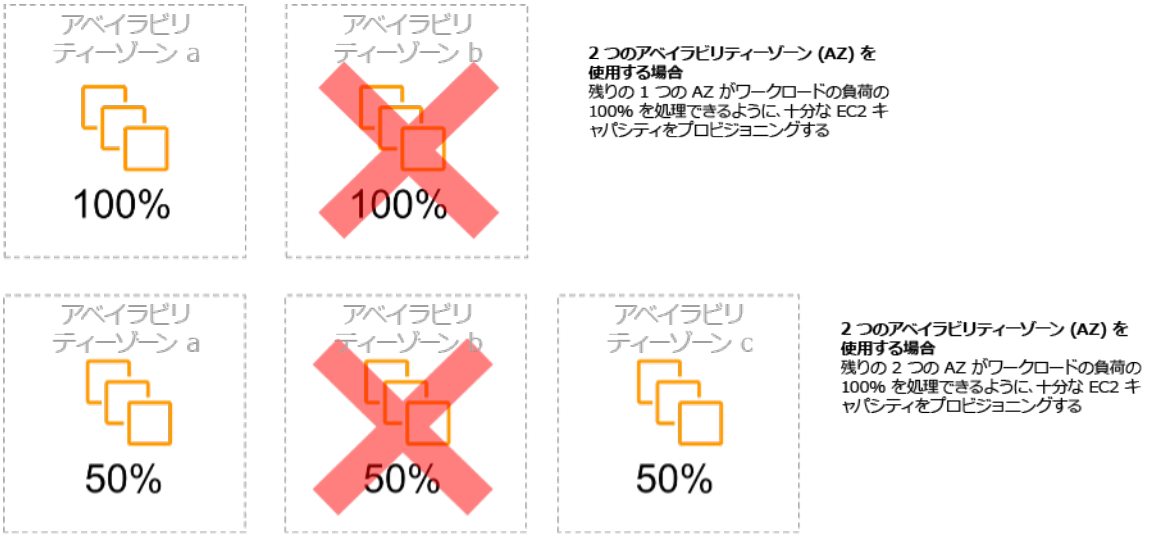
- 障害の範囲に関係なく、リソースが常にプロビジョニング可能であると仮定する。
- 障害発生時にリソースを動的に取得しようとする。
- 障害が発生するまで、ゾーンまたはリージョン間で適切なリソースをプロビジョニングしない。
- コンピューティングリソースにのみ静的に安定した設計を検討する。

このベストプラクティスを活用するメリット: 静的に安定した設計で実行されるワークロードは、通常のイベント時でも障害発生時でも予測可能な結果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: 中

実装のガイダンス

バイモーダル動作とは、例えばアベイラビリティーゾーン (AZ) 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。バイモーダル動作の例は、安定した Amazon EC2 設計により、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が各 AZ にプロビジョニングされる場合です。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。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。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。



複数のアベイラビリティーゾーン (AZ) にわたる EC2 インスタンスの静的安定性

これは、このモデルのコストと、すべてのレジリエンスケースでワークロードを維持することのビジネス価値に照らして検討する必要があります。コンピューティングキャパシティを少なくプロビジョニングし、障害発生時に新しいインスタンスの起動に依存するほうがコストは低くなりますが、大規模な障害 (AZ やリージョンの障害など) の場合、このアプローチは、運用プレーンと、影響を受けていないゾーンまたはリージョンでリソースが十分に利用可能であることの両方に依存するため、あまり効果的ではありません。

また、ソリューションでは、信頼性とワークロードに必要なコストを比較検討する必要があります。静的に安定したアーキテクチャは、複数の AZ に分散されているコンピューティングインスタンス、データベースリードレプリカ設計、Kubernetes (Amazon EKS) クラスター設計、マルチリージョンフェイルオーバーアーキテクチャなど、さまざまなアーキテクチャに適用されます。

各ゾーンでより多くのリソースを使用することにより、より静的に安定した設計を実装することもできます。ゾーンを追加することで、静的安定性に必要な追加のコンピューティング量を減らすことができます。

バイモーダル動作の例に、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します。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、アラームを開始します。

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、ワークロードの需要を大幅に変更し、障害が発生する可能性が高くなります。

重要なワークロードを評価して、どのワークロードにこのタイプのレジリエンス設計が必要かを判断します。重要と思われるものについては、各アプリケーションコンポーネントを確認する必要があります。静的安定性評価を必要とするサービスの種類の例は次のとおりです。

- コンピューティング: Amazon EC2、EKS-EC2、ECS-EC2、EMR-EC2
- データベース: Amazon Aurora、Amazon RDS、Amazon Redshift。
- ストレージ: Amazon S3 (単一ゾーン)、Amazon EFS (マウント)、Amazon FSx (マウント)
- ロードバランサー: 特定の設計で

実装手順

- 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します。この場合、1 つのアベイラビリティーゾーンまたはリージョンが削除された場合にワークロード容量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またはリージョンにプロビジョニングします。正常なリソースへのルーティングには、次のようなさまざまなサービスを使用できます。
  - クロスリージョン DNS ルーティング
  - MRAP Amazon S3 マルチリージョンのルーティング
  - AWS Global Accelerator
  - Amazon Application Recovery Controller

- 単一のプライマリインスタンスまたはリードレプリカが失われた場合に備えて、データベースリードレプリカを設定します。トラフィックがリードレプリカによって処理されている場合、各アベイラビリティーゾーンと各リージョンでの量は、そのゾーンまたはリージョンに障害が発生した場合の全体的な必要量と同じにします。
- アベイラビリティーゾーンに障害が発生した場合に保存されるデータに対して静的に安定するように設計された Amazon S3 ストレージに重要なデータを設定します。Amazon S3 One Zone-IA ストレージクラスを使用する場合、そのゾーンが失われると、保存されたデータへのアクセスが最小限に抑えられるため、静的に安定しているとみなすべきではありません。
- ロードバランサーは、誤って、または設計により、特定のアベイラビリティーゾーン (AZ) を処理するように設定されている場合があります。この場合、静的に安定した設計では、ワークロードをより複雑な設計の複数の AZ に分散することが考えられます。セキュリティ、レイテンシー、またはコスト上の理由から、元の設計を使用してゾーン間のトラフィックを削減できる場合があります。

リソース

関連する Well-Architected のベストプラクティス:

- 可用性
- 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-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

関連ドキュメント:

- ディザスタリカバリディザスタリカバリプランの依存関係を最小化する
- 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
- 障害分離境界
- アベイラビリティーゾーンを使用した静的安定性
- マルチゾーン RDS
- ディザスタリカバリディザスタリカバリプランの依存関係を最小化する
- クロスリージョン DNS ルーティング
- MRAP Amazon S3 マルチリージョンのルーティング
- AWS Global Accelerator
- Amazon Application Recovery Controller

- 1 ゾーン Amazon S3
- クロスゾーンロードバランシング

関連動画:

- Static stability in AWS: 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

しきい値の違反が検出されると、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。

回復力を備えたシステムは、低下イベントがすぐに適切なチームに通知されるよう設計されています。これらの通知は、1 つまたは複数の通信チャネルを通じて送信する必要があります。

期待される成果: エラー率、レイテンシー、その他の主要業績評価指標 (KPI) メトリクスなどのしきい値を超えると、直ちにオペレーションチームにアラートが送信されます。これにより、これらの問題をできるだけ早く解決し、ユーザーへの影響を回避するか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:

- 送信するアラームが多すぎる。
- 実行できないアラームを送信する。
- アラームのしきい値の設定が高すぎる (感度が高すぎる) か、低すぎる (感度が低すぎる)。
- 外部依存関係に対するアラームを送信しない。
- モニタリングとアラームを設計する際にグレー障害を考慮していない。
- ヒーリングのオートメーションを実行しているが、ヒーリングが必要となったことを適切なチームに通知しない。

このベストプラクティスを活用するメリット: 復旧の通知により、運用チームやビジネスチームはサービスの低下を認識し、直ちに対応して平均検出時間 (MTTD) と平均修復時間 (MTTR) の両方を最

小限に抑えることができます。復旧イベントの通知により、発生頻度の低い問題を無視することもなくなります。

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 適切なモニタリングとイベント通知のメカニズムを実装しないと、自動ヒーリングで対処されるものも含め、問題のパターンを検出できなくなる可能性があります。チームは、ユーザーがカスタマーサービスに連絡した場合、または偶然に見つけた場合にしか、システムの低下を認識できなくなります。

実装のガイダンス

モニタリング戦略の定義において、アラームのトリガーは一般的なイベントです。このイベントには、アラームの識別子、アラームの状態 (IN ALARM や OK) およびアラームをトリガーした原因の詳細が含まれる可能性があります。多くの場合、1 件のアラームイベントが検出されると、1 件の E メール通知が送信されます。これは、1 件のアラームにつき 1 つのアクションの例です。アラーム通知は、問題があることを適切な担当者に通知するため、オブザーバビリティにおいて非常に重要です。ただし、オブザーバビリティソリューションでイベントに対するアクションが洗練されると、人間の介入を必要とせずに問題を自動的に修正できます。

KPI モニタリングアラームを設定すると、しきい値を超えたときに適切なチームにアラートが送信されます。これらのアラートを使用して、低下の修復を試みる自動プロセスをトリガーすることもできます。

より複雑なしきい値のモニタリングには、複合アラームを検討する必要があります。複合アラームでは、多くの KPI モニタリングアラームを使用して、運用上のビジネスロジックに基づいてアラートを作成します。CloudWatch アラームは、Amazon SNS 統合または Amazon EventBridge を使用して、E メールを送信するか、サードパーティーのインシデント追跡システムにインシデントをログ記録するように設定できます。

実装手順

モニタリング対象のワークロードに応じて、次のようなさまざまなタイプのアラームを作成します。

- アプリケーションアラームは、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するために使用します。
- インフラストラクチャアラームは、リソースをスケールするタイミングを示します。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 メールを介して送信したり、Auto Scaling と連携させてワークロードリソースをスケールイン/スケールアウトするのに使用したりすることができます。
- シンプルな静的アラームを作成して、メトリクスが指定された評価期間の静的しきい値を超える状況をモニタリングできます。

- 複合アラームは、複数のソースからの複雑なアラームに対応できます。
- アラームを作成したら、適切な通知イベントを作成します。Amazon SNS API を直接呼び出して、通知を送信し、修正やコミュニケーションのための自動化をリンクすることができます。
- Amazon Health Aware のモニタリングを統合することで、パフォーマンス低下の可能性がある AWS リソースを視覚的にモニタリングできます。ビジネスに不可欠なワークロードの場合、このソリューションによって、AWS サービスに対するプロアクティブでリアルタイムのアラートへのアクセスが可能になります。

リソース

関連する Well-Architected のベストプラクティス:

- 可用性

関連ドキュメント:

- 静的しきい値に基づいて CloudWatch アラームを作成する
- What Is Amazon EventBridge?
- Amazon Simple Notification Service とは
- カスタムメトリクスを発行する
- Amazon CloudWatch でのアラームの使用
- AWS Health Aware — 組織および個人の AWS アカウントの AWS Health アラートのカスタマイズ
- CloudWatch 複合アラームの設定
- re:Invent 2022 で公開された AWS オブザーバビリティに関する最新情報

関連ツール:

- CloudWatch
- CloudWatch X-Ray

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および管理できます。

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 メトリクスが高すぎに設定されている。
- 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を活用しない場合のリスクレベル: 中

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。
- 依存関係を減らすと、可用性にプラスの影響を与えることができます。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。
- 分散システムの可用性を向上させる要素は、障害の頻度が少ない (MTBF が長い)、障害検出時間が短い (MTTD が短い)、修理時間が短い (MTTR が短い) という 3 つです。
- 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損失に関する復旧目標を定義する
- ワークロードの状態の理解

関連ドキュメント:

- 冗長性を実装した可用性
- 信頼性の柱 - 可用性
- 可用性の測定
- AWS 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
- AWS サービスレベルアグリーメント (SLA)
- Guidance for Cell-based Architecture on AWS
- AWS インフラストラクチャ
- Advanced Multi-AZ Resiliance Patterns whitepaper

関連サービス:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

REL 12. どのように信頼性をテストしますか?

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、期待する回復性を実現することを検証する唯一の方法は、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呼び出されるまたは実行されるか、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。
    - AWS Systems Manager を使った運営計画の自動化
    - AWS Systems Manager Run Command
    - AWS Systems Manager Automation
    - What is AWS Lambda?
    - What Is Amazon EventBridge?
    - Amazon CloudWatch でのアラームの使用

リソース

関連ドキュメント:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- AWS Systems Manager を使った運営計画の自動化
- Amazon CloudWatch でのアラームの使用
- canary の使用 (Amazon CloudWatch Synthetics)
- What Is Amazon EventBridge?
- AWS Lambda とは

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、寄与因子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。

期待される成果: チームが合意済みの一貫したアプローチで、インシデント後の分析を取り扱います。そのメカニズムの 1 つがエラーの修正 (COE) プロセスです。COE プロセスは、チームがインシデントの根本原因を特定、理解、対処するのに役立つと同時に、同じインシデントの再発を防止するメカニズムとガードレールの構築にも有益です。

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。
- 原因究明よりも責任を追及するばかりで恐怖心を煽り、オープンなコミュニケーションが妨げられる。
- インサイトを共有できない。インシデント分析の結果を少人数のグループで抱え込み、他の人が教訓を活かせなくなります。
- 組織の知識をキャプチャするメカニズムがない。学んだ教訓を最新のベストプラクティスという形で保存しないと貴重なインサイトが失われ、同様または類似の根本原因でインシデントが再発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: 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

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。

COE プロセスの基礎は、問題を文書化して対処することです。重要な根本原因を文書化し、その原因をレビューして確実に解決するための標準化した手法を定義しておくことを推奨します。インシデント後の分析プロセスには明確に担当者を割り当てます。インシデントの調査とフォローアップの監督を担当するチームまたは個人を指名してください。

責任を追及するのではなく、学習と改善を重視する文化を醸成してください。個人の責任を問うのではなく、インシデントの再発防止が目標であることを強調します。

インシデント後の分析を実施するための明確な手順を策定します。これらの手順では、実行すべき手順、収集する情報、分析中に対処すべき重要な問題をまとめておく必要があります。インシデントを徹底的に調査し、直接的な原因だけでなく、根本原因と寄与因子を特定します。5 つの why 分析などの手法を用いて、根本的な問題を掘り下げてください。

インシデント分析から学んだ教訓のリポジトリを維持します。こうした組織の知識は、将来のインシデントや予防策の参考になります。インシデント後の分析から得られた結果とインサイトを共有し、誰でも参加できるインシデント後のレビューミーティングを開催して、学んだ教訓について話し合うことを検討してください。

実装手順

- インシデント後の分析を行う際は、非難の場にならないようにします。これにより、インシデントにかかわった人々は、提案された是正措置を冷静に検討し、誠実な自己評価とチーム間のコラボレーションに努めることができます。
- 重大な問題を文書化する標準化された方法を定義します。このようなドキュメントの構造の例は次のとおりです。
  - 何が起きたのか。
  - 顧客とビジネスにどのような影響がありましたか?
  - 根本原因は何でしたか?
  - これをサポートするデータはありますか?
    - 例えば、メトリクスとグラフ
  - 重要な柱、特にセキュリティとは何を意味するでしょうか?
    - 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、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意思決定がエンジニアリングの優先度を左右する可能性があります。開発環境では信頼性を妥協することでコストを削減するという最適化を#う場合や、ミッションクリティカルなソリューションでは、信頼性を最適化するためにコストをかける場合などがあります。セキュリティは常に最優先事項です。顧客を保護する必要があるからです。
  - どのような教訓を学びましたか?

  - どのような是正措置を講じましたか?
    - アクション項目
    - 関連項目
- インシデント後の分析を実施するための明確に定義された標準運用手順を作成します。
- 標準化されたインシデント報告プロセスを用意します。初回のインシデントレポート、ログ、コミュニケーション、インシデントの発生中に取られたアクションなど、すべてのインシデントを包括的に文書化します。
- インシデントが生じても必ずしもシステム停止を伴うわけではありません。ニアミスである場合や、システムがビジネス機能を果たしながら予想外の方法で動作している場合があります。
- フィードバックと教訓に基づいて、インシデント後の分析プロセスを継続的に改善します。
- 重要な検出結果をナレッジ管理システムでキャプチャし、開発者ガイドやデプロイ前のチェックリストに追加すべきパターンを検討します。

リソース

関連ドキュメント:

- correction of error (COE) を開発すべき理由

関連動画:

- Amazon's approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon

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用のステージングサーバーにデプロイされます。ステージングサーバーから、CodePipeline は統合やロードなどの色々な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 (canary 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

- 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
  - canary 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
- canary 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ル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たりし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とき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、負荷テストを実行します。
      - 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

悪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て、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障害に耐え、想定外の中断から影響を最小限に抑えて回復できることを確信できます。

一般的なアンチパターン:

- 回復力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る負荷による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無視する。

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で、回復力設計の復旧手順が実際の障害発生時にも機能するという確信を得られます。

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: 中

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ー、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、放置しておくと可用性やオペレーションに悪影響を及ぼす可能性がある、ワークロードの欠陥が浮き彫りになります。

> **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、本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

システムがこれらの混乱に耐えられる場合は、カオス実験を自動化されたリグレッションテストとして維持する必要があります。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 パイプライン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。例えば、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の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。

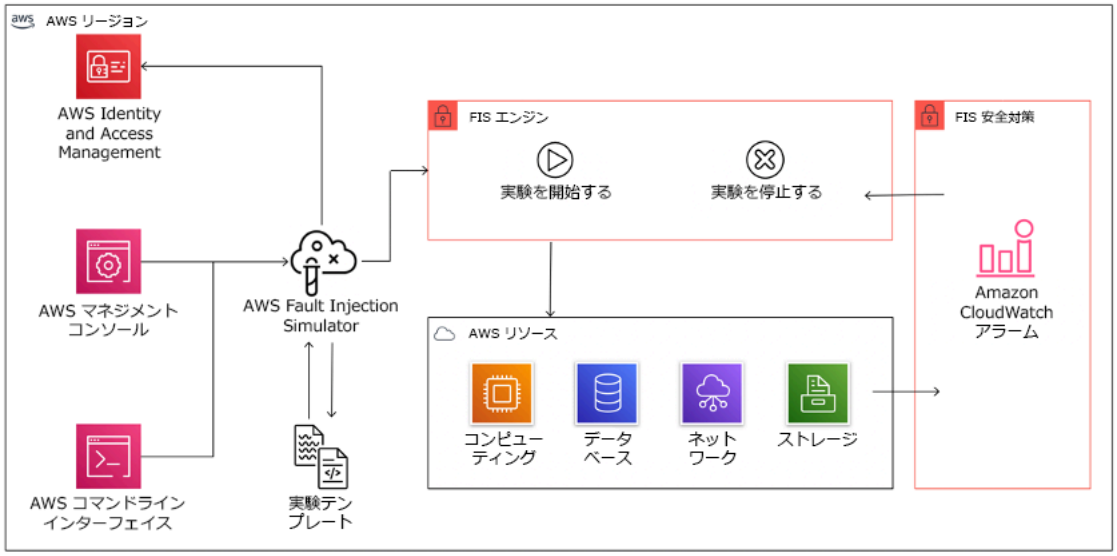
アプリケーションレベルの障害 (クラッシュなど) では、メモリや CPU の枯渇などのストレス要因から始めます。

断続的なネットワーク中断による外部依存関係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、数秒から数時間続く指定期間の間、コンポーネントがサードパーティープロバイダーへのアクセスをブロックすることで、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。。

その他の劣化状態では、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への影響を含むこれらの障害実験には、名前の解決、DNS サービスへの到達、依存サービスへの接続ができないことなどが含まれる可能性があります。

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フルマネージドサービスであり、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の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)、Amazon RDS など、さまざまな種類のリソースに障害を同時発生させるのに役立ちます。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 またはメモリへのストレス、スロットリング、レイテンシー、パケット損失などが含まれます。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、想定外の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。



AWS Fault Injection Service を AWS リソースと統合することで、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できるようになります。

フォールトインジェクション実験を実行するための、いくつかのサードパーティーオプションがあります。これには、Chaos Toolkit、Chaos Mesh、Litmus Chaos などのオープンソースツールや、Gremlin などの商用オプションが含まれます。AWS に挿入できる障害の範囲を拡張するために、AWS FIS を Chaos Mesh および Litmus Chaos と統合することで、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を調整できます。例えば、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

1. どの障害を実験に使用するかを決定します。

   回復力に対するワークロードの設計を評価します。Well-Architected Framework のベストプラクティスを使用して作成するこのような設計では、重要な依存関係、過去のイベント、既知の問題、コンプライアンス要件に基づくリスクを考慮します。回復力を維持するために想定した設計の各要素と、それを低減するために設計する障害をリストアップします。このようなリストの作成の詳細については、「Operational Readiness Review」ホワイトペーパーを参照してください。このホワイトペーパーでは、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について説明します。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分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 の「障害モードと継続的回復力」で詳しく説明しています。
2. 各障害に優先度を割り当てます。

「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。

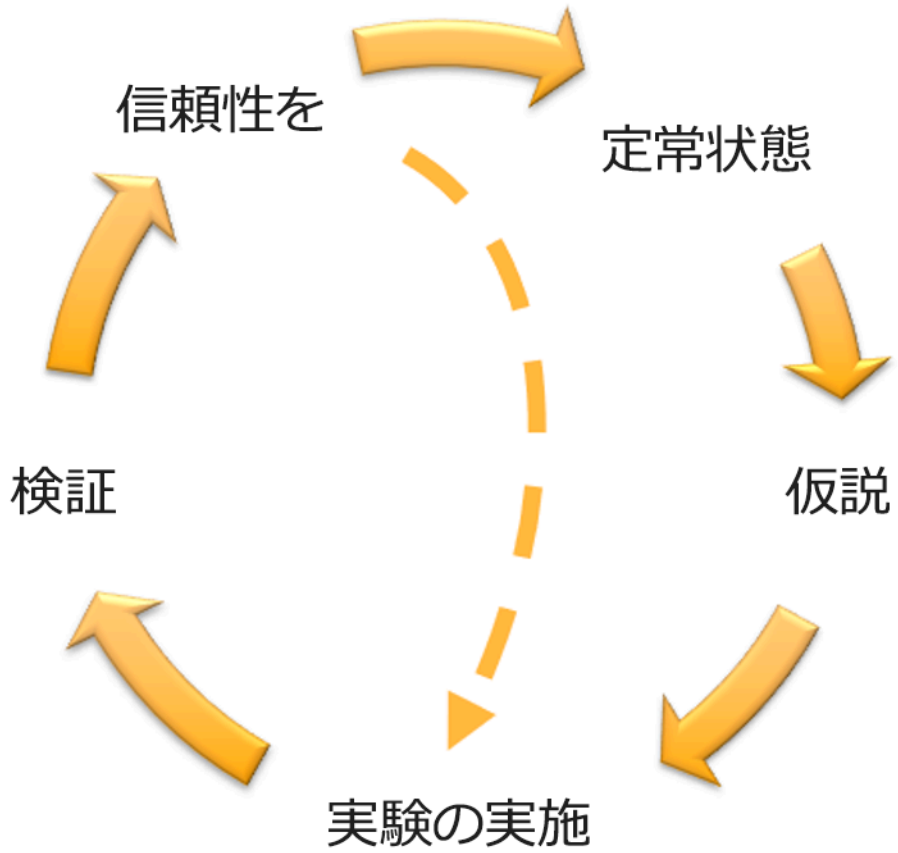
ある障害の発生頻度を考慮する場合、利用可能であれば、そのワークロードの過去のデータを分析します。利用できない場合は、類似の環境で実行されている他のワークロードのデータを使用します。

ある障害の影響を考慮する場合、一般的に、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。ワークロードの設計と目的も考慮します。例えば、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。

3. 実施するそれぞれの実験に関して、次の図の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 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。

a. 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します。

   ワークロードは、信頼性が高く、期待どおりに動作していれば、定常状態を示します。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。一定の割合の障害は許容範囲内である可能性があるため、定常状態は、必ずしも障害発生時にワークロードに影響が及ば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される基準値であり、定義した仮説が次のステップで想定どおりにならない場合に、異常を強調します。

   例えば、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できます。

b. 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ます。

良い仮説は、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、特定の種類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ます。なぜならば、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、仮説の中で特定する必要があります。

次のテンプレートを仮説に使用できます (ただし、他の表現も許容されます)。

> **Note**
>
> #####が発生した場合、#######のワークロードには、##################を維持するための###############。

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、ポッドはスケジューリングされて、実験開始後 8 分以内に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)。

c. 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わかっている場合は、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必要があります。既知の未知は、認識しているが完全には理解していないモノであり、未知なる未知は、認識も完全に理解もしていないモノです。壊れているとわ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用意

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わかったら、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 のワークロードの場合、AWS FIS はアクションと呼ばれる多くの事前定義された障害シミュレーションを提供します。AWS Systems Manager ドキュメントを使用して、AWS FIS で実行するカスタムアクションを定義することもできます。

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、お勧めできません。

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します。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 を使用したさまざまな実験については、「Resilient and Well-Architected Apps with Chaos Engineering lab」も参照してください。AWS Resilience Hub はワークロードを分析し、AWS FIS で実装および実行することを選択できる実験を作成します。

> **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。

可能であれば、実験はコントロールと実験的なシステムデプロイの両方をスピンアップするカナリアデプロイを使用して、実際の負荷下の本番環境で実行する必要があります。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方法です。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、本稼働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、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本番稼働前の環境で実験を行ってください。

実験が本稼働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は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、障害を挿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。合成モニタ (ユーザー canary とも呼ばれます) は、通常、ユーザープロキ

シとして含める必要があるメトリクスの 1 つです。AWS FIS 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、テンプレートごとに最大 5 つの停止条件を許可します。

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する必要がありますが、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、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どおりに作動するか、例外をキャッチするためのオブザーバビリティ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行する場合は、すべての責任者に情報が十分に伝達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、何が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提供します。

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る必要があります。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ます) を設定することができます。ポストアクションは、ターゲットをアクションが実行される前の状態に戻します。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、障害を検出して処理する方法を説明するプレイブックの一部である必要があります。

d. 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、以下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。その出力を短期間測定することによって、システムの定常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。カオスエンジニアリングでは、実験中のシステムの動作パターンに焦点を当て、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、システムが実際に動作す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、定常状態のメトリクスが含まれています。

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 とも呼ばれます) を含める必要があります。

e. 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は、AWS Well-Architected の信頼性の柱のベスト プラクティスを適用して、障害を軽減するためにワークロードの設計をどのように改善できるかを調査します。その他のガイダンスとリソースは、「AWS ビルダーライブラリ」にあります。このライブラリには、ヘルスチェックの改善方法や、アプリケーションコードでのバックオフによる再試行方法などに関する記事がホストされています。

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。

4. 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する必要があります。ワークロードが実験の仮説を満たしたら、CI/CD パイプラインのリグレッション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。これを行う方法については、AWS CodePipeline を使用して AWS FIS 実験を実行する方法に関するブログを参照してください。CI/CD パイプラインでの AWS FIS 反復実験に関するこのラボでは、実践的に作業できます。

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 (REL12-BP06 定期的にゲームデーを実施する を参照)。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、例外的なイベントの発生時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。

5. 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。結果の例として、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 を使用した実験ログ記録も、このデータキャプチャの一部です。

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

関連ドキュメント:

- What is AWS Fault Injection Service?
- What is AWS Resilience Hub?
- 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
- カオス実験のカナリアデプロイ

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、Amazon S3 の回復力のテスト
- 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。これらは定期的に実施し、チームが対応方法に基づいてマッスルメモリを蓄積できるようにする必要があります。

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。障害イベントも含む、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うことを確認し、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: 中

実装のガイダンス

- ゲームデーを計画して、定期的にランブックおよびプレイブックを使ってみます。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

関連動画:

- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)

  関連する例:

- AWS Well-Architected Labs - Testing Resiliency

REL 13. ディザスタリカバリ (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。これらは、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、データの場所と機能を考慮して、目標を達成するための戦略を実装します。ワークロードのディザスタリカバリを提供することのビジネス価値を伝える際、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損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損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 (RPO) が定義されます。

目標復旧時間 (RTO) は、サービスの中断から復旧までの最大許容遅延時間です。これにより、サービスが利用できなくなったときに許容できる時間枠が決まります。

目標復旧時点 (RPO) は、最後に実行されたデータ復旧時点からの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

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、DR 戦略の選択と実装のために技術チームによって使用されます。

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階層に割り当てられ、関連する 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。このような階層化ができない場合は、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の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: 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など)。

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する必要があります)。

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おそ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 実装の指針として必要になります。

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

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。一般的に、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種類によって異なります。例えば、最大 1 時間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 (および評判への影響)、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、さまざまな形態をとります。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関係者を決定し、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関係者と協力して、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、the section called “実装ワークシート” を使用できます。

2. 以下の質問に回答して、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。必ず、このステップに入るときに計算した元の値よりも厳しいカテゴリ (低い RTO および RPO) を選択します。この結果、値の変化が不適切に大きくなる場合は、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、または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実行できます。
5. 組織の事業継続計画 (BCP) の一部であるこのワークロードのディザスタリカバリ計画 (DRP) を、ワークロードチームや関係者がアクセスできる場所に文書化します。
   a. RTO および RPO と、これらの値を決定す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 に加えて、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始められます。
7. 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” に従って DR 戦略 (または DR 戦略の概念実証) を実装したら、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 (復旧ポイント機能) 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、ビジネス関係者と協力して目標を調整するか、DR 戦略を変更し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. 影響は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及ぶ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損失した場合、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するソースデータの重要度に基づいて RPO を選択します。
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択します。
   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択します。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、必要に応じて、ダウンストリームの重要な依存関係と協力して、復旧能力を改善します。

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を考慮してください。

4. 停止の種類 (リージョンと AZ など) に応じた異なる RTO と 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の、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ことや、給与システムがトランザクションを送信できないことによる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ていますか?
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? 義務を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。このワークシートは、特定のニーズに応じて調整できます (質問を追加するなど)。

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
[3b] 調整されたダウンストリームの依存関係						合は、依存関係を調整し、調整した値をここに入力します
[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

実装計画に必要な工数レベル: 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”

関連ドキュメント:

- AWS アーキテクチャブログ: ディザスタリカバリシリーズ
- AWS 上のワークロードのディザスタリカバリ: クラウドにおけるリカバリ (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: ディザスタリカバリを支援できるパートナー
- AWS Marketplace: ディザスタリカバリに使用できる製品

関連動画:

- 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用アーキテクチャパターン (ARC209-R2)
- AWS でのワークロードのディザスタリカバリ

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を活用しない場合のリスクレベル: 高 計画され、実装され、テストされた DR 戦略がなければ、災害発生時に復旧目標を達成できない可能性があります。

実装のガイダンス

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損失に関する復旧目標を定義する」を参照)。

単一の AWS リージョン内の複数のアベイラビリティーゾーン (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、コストと複雑さが低く、かつ RTO と RPO が長い順にリストされます。リカバリリージョンとは、ワークロードに使用されるプライマリ以外の AWS リージョンを指します。

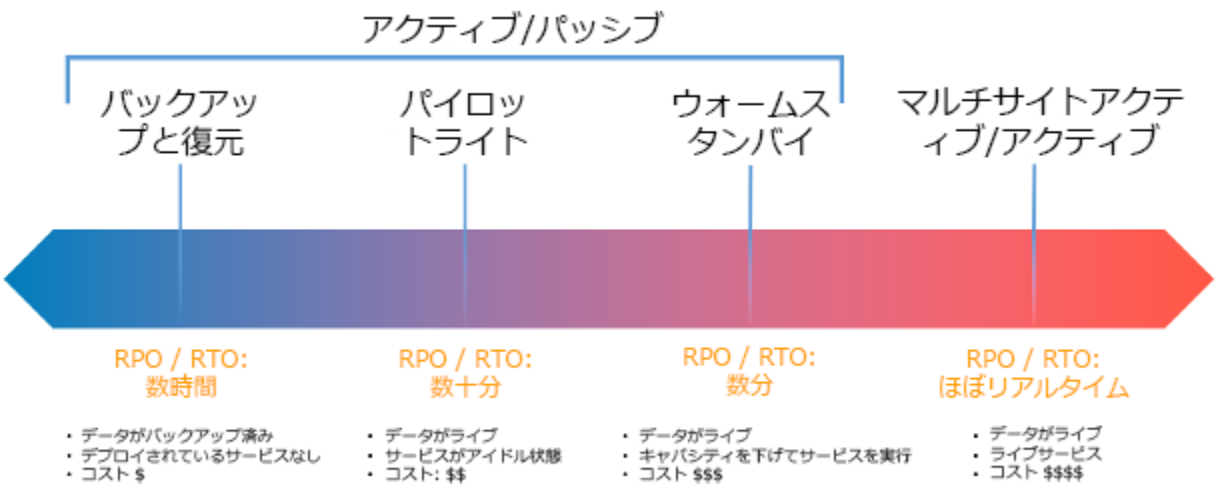


図 17: ディザスタリカバリ (DR) 戦略

- バックアップと復元 (数時間での RPO、24 時間以下での RTO): データとアプリケーションを復旧リージョンにバックアップします。自動化されたバックアップまたは連続バックアップを使用すると、ポイントインタイムリカバリ (PITR) 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。

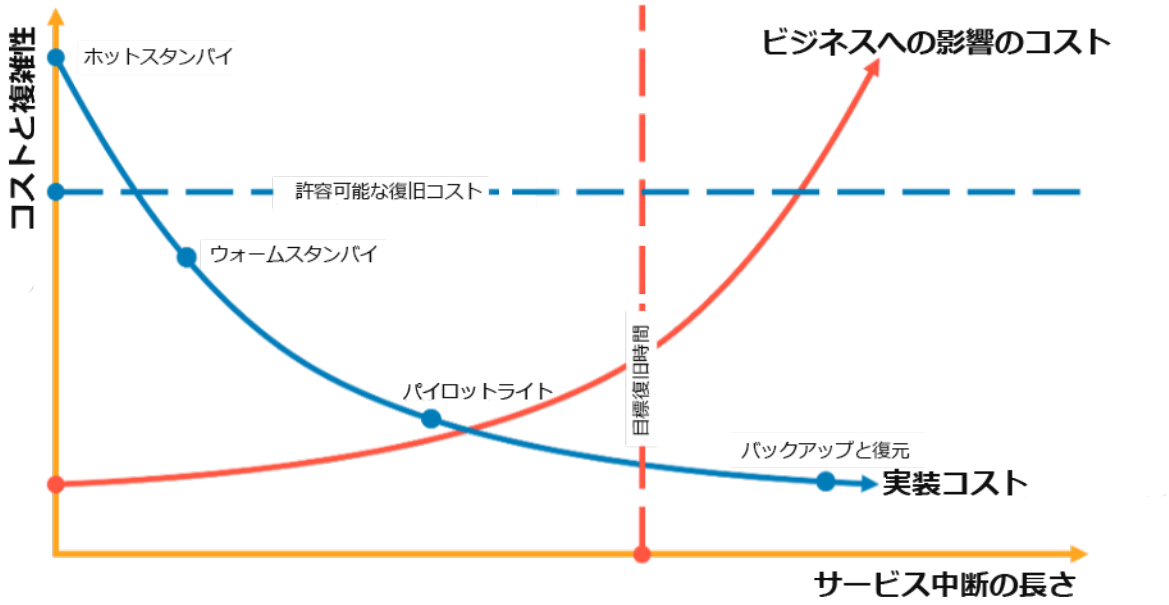


図 18: RTO とコストに基づく DR 戦略の選択を示すグラフ

詳細については、「ビジネス継続計画 (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ーゾーンを 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、実装の複雑さが最も少ない戦略ですが、ワークロードの復元に必要な時間と労力が多く、より高い RTO と RPO につながります。常にデータのバックアップを取り、これらを別のサイト (別の AWS リージョンなど) にコピーすることをお勧めします。

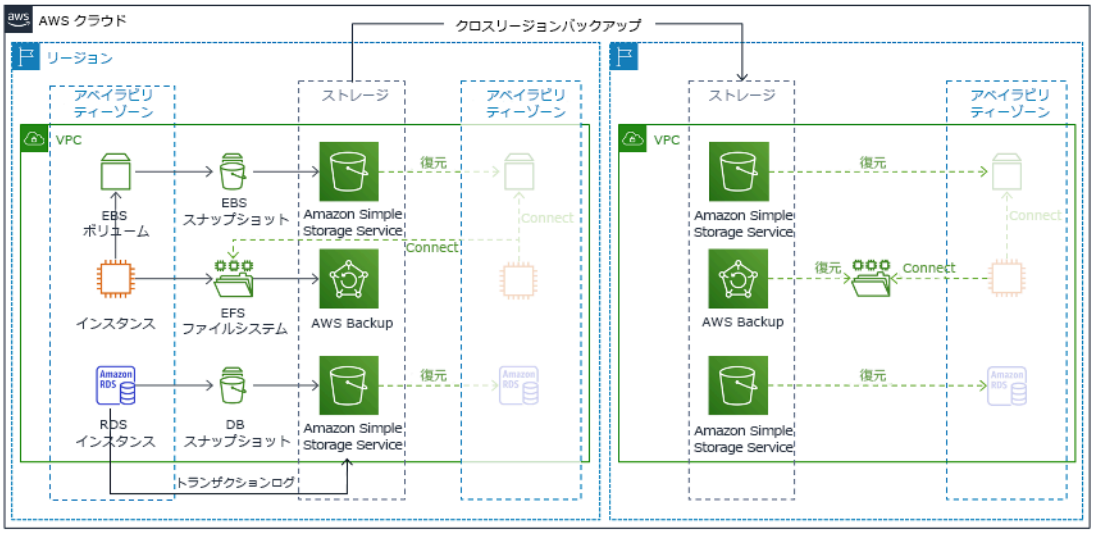


図 19: バックアップと復元アーキテクチャ

この戦略の詳細については、「AWS でのディザスタリカバリ (DR) アーキテクチャ、パート II: 迅速な復旧によるバックアップと復元」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。

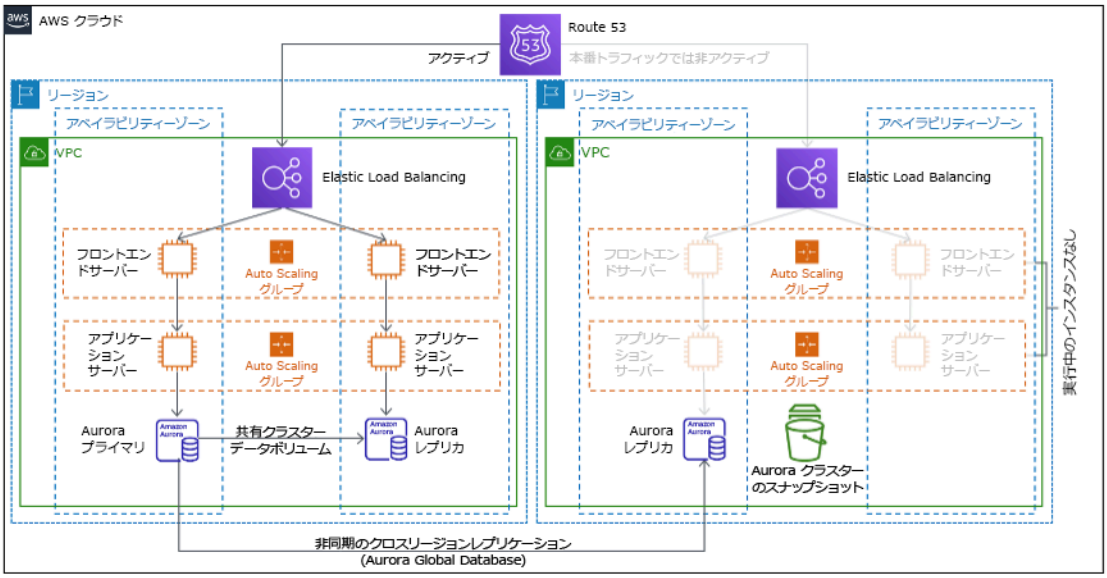


図 20: パイロットライトアーキテクチャ

この戦略の詳細については、「AWS でのディザスタリカバリ (DR) アーキテクチャ、パート III: パイロットライトとウォームスタンバイ」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アプローチでは、本番稼働環境の完全に機能する縮小コピーを別のリージョンに用意します。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。

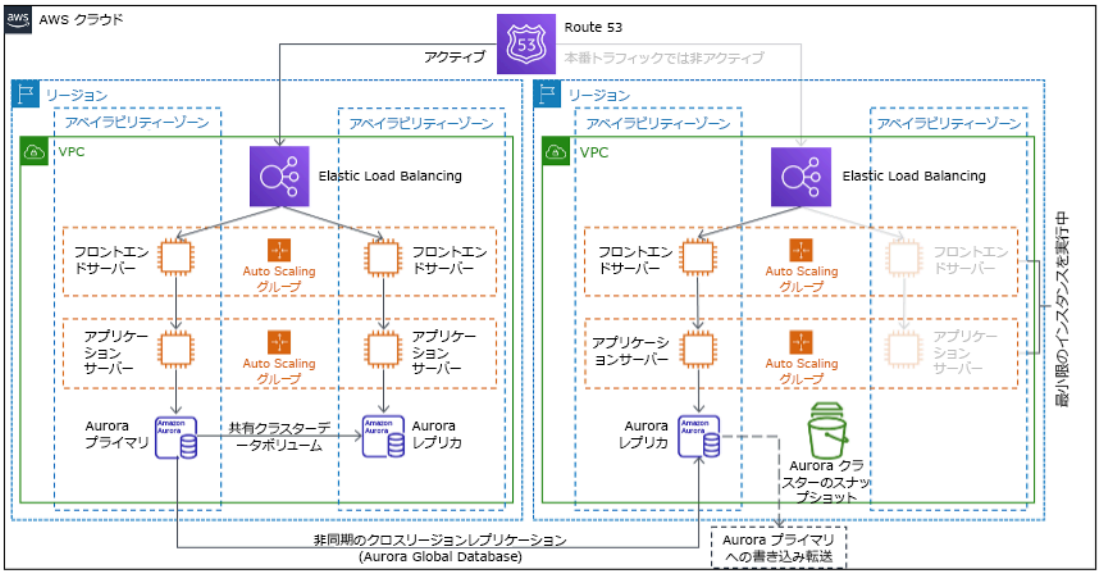


図 21: ウォームスタンバイアーキテクチャ

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。必要に応じてキャパシティが利用可能であることを保証するには、EC2 インスタンスのキャパシティ予約の使用を検討してください。AWS Lambda を使用する場合、プロビジョニングされた同時実行はランタイム環境を提供して、関数の呼び出しにすぐに応答する準備を整えることができます。

この戦略の詳細については、「AWS でのディザスタリカバリ (DR) アーキテクチャ、パート III: パイロットライトとウォームスタンバイ」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部として、複数のリージョンでワークロードを同時実行できます。マルチサイトアクティブ/アクティブは、デプロイされたすべてのリージョンからのトラフィックを処理します。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。

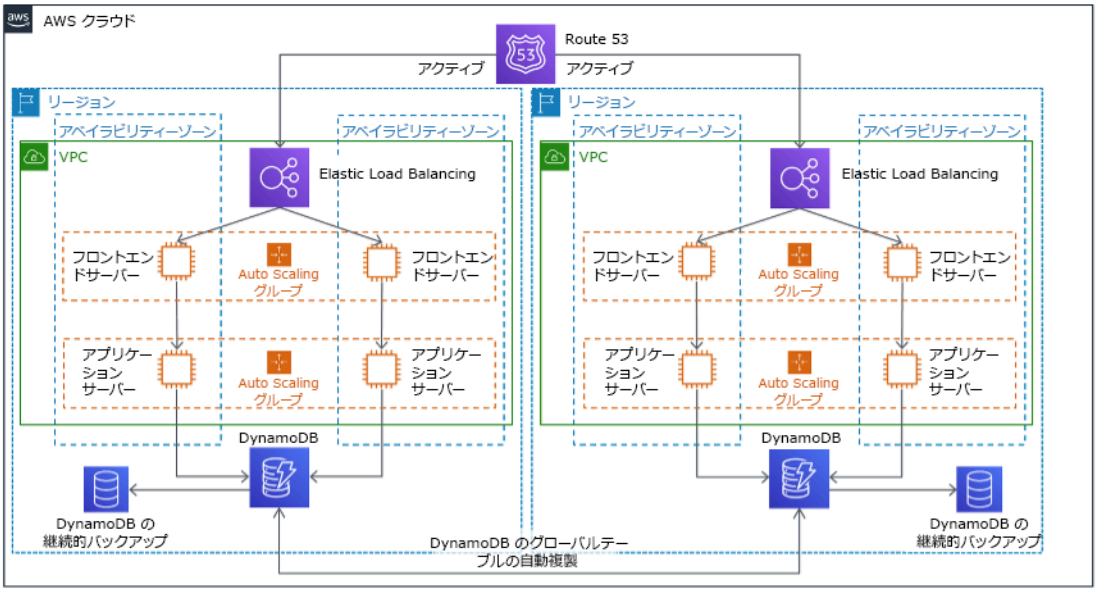


図 22: マルチサイトアクティブ/アクティブアーキテクチャ

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のパイロットライト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 はより優れた代替アプローチを提供できる場合があります。Elastic Disaster Recovery は、ウォームスタンバイと同様の RPO と RTO ターゲットを提供できますが、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、プライマリリージョンからリカバリリージョンにデータをレプリケートします。継続的なデータ保護を使用して、数秒で測定される RPO と数分で測定できる RTO を実現します。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 では、サービスが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。

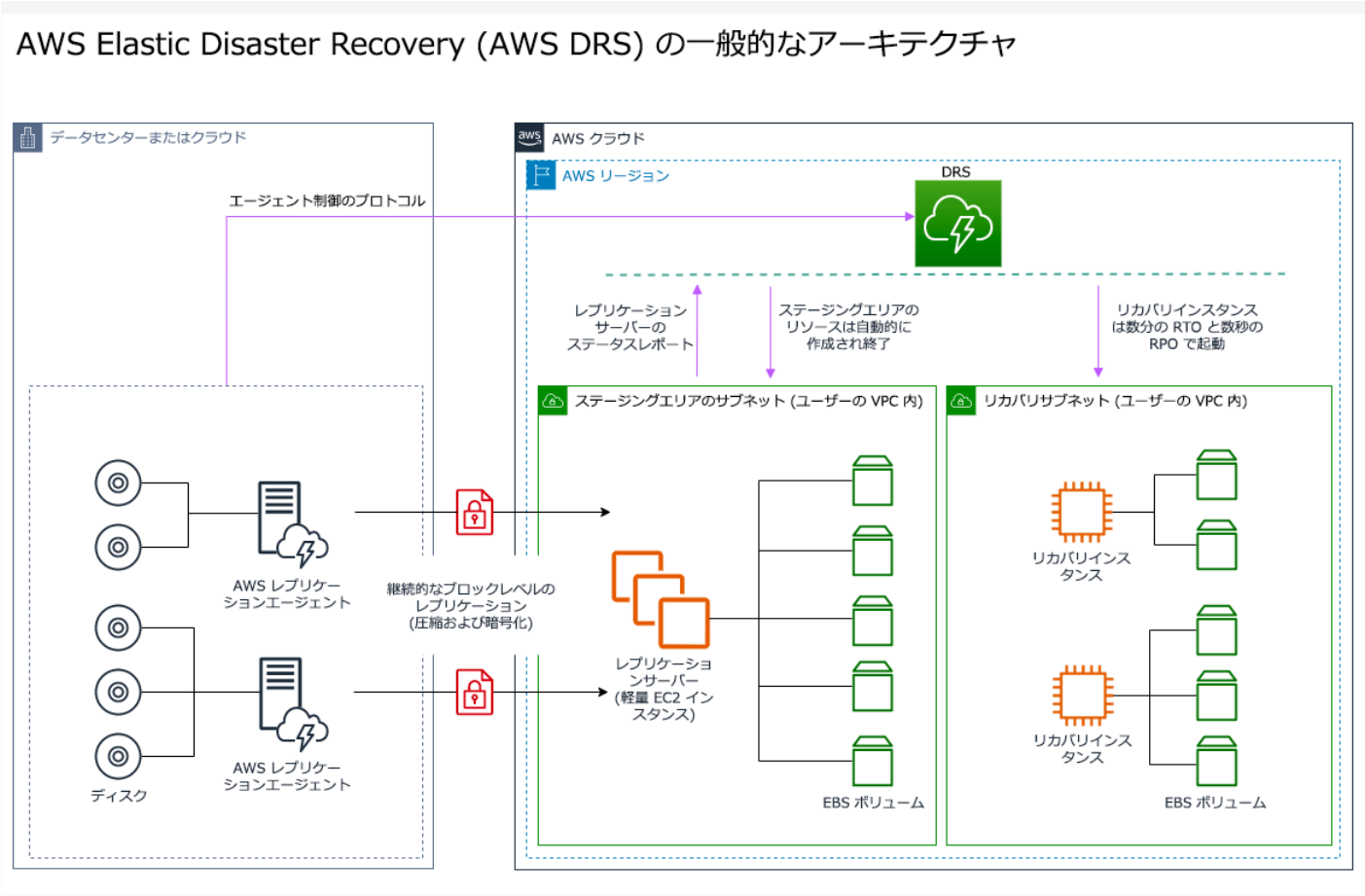


図 23: AWS Elastic Disaster Recovery アーキテクチャ

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では、Amazon EC2 インスタンスと Elastic Load Balancer にリソースが複数の AZ にデプロイされ、リクエストがアクティブに処理されるため、マルチサイトアクティブ/アクティブアプローチを使用します。このアーキテクチャはホットスタンバイでもあります。プライマリ Amazon RDS インスタンスが停止 (または AZ 自体が停止) すると、スタンバイインスタンスはプライマリに昇格されます。

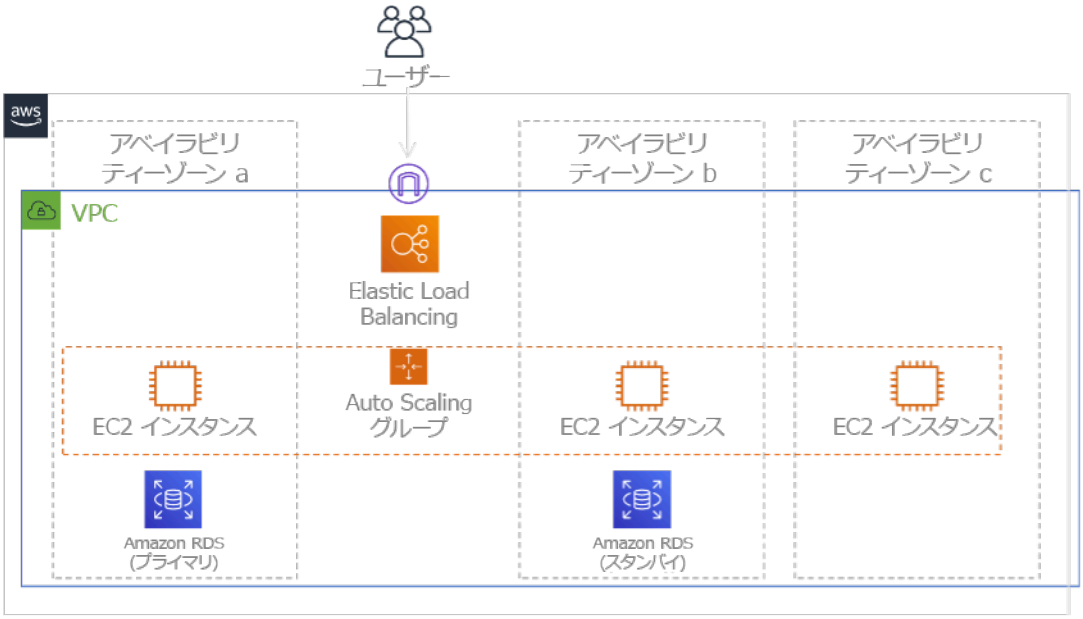


図 24: マルチ AZ アーキテクチャ

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 ボリュームまたは Amazon Redshift クラスターなど、単一のゾーンに制約されるデータの場合に特に重要です。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。

単一リージョンに対するあまり一般的でない代替アプローチとして、マルチ AZ DR があります。これはブログ記事「Amazon Application Recovery Controller を使用して回復力の高いアプリケーションを構築する、パート 1: 単一リージョンスタック」で説明されています。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 リソースには、AWS CloudFormation などのコードとしてインフラストラクチャ、または Hashicorp Terraform などのサードパーティーツールを使用します。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 を使用できます。マルチサイトアクティブ/アクティブとホットスタンバイ戦略の場合、復旧リージョンにデプロイされるインフラストラクチャはプライマリリージョンと同じリソースを持ちます。パイロットライトとウォームスタンバイ戦略の場合、デプロイされたインフラストラクチャを本番稼働で使用するには追加のアクションが必要です。CloudFormation パラメータおよび条件付きロジックを使用すると、デプロイされるスタックがアクティブかスタンバイかを単一のテンプレートで制御できます。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、データソースを AWS リージョン内でバックアップし、それらのバックアップをリカバリリージョンにコピーします。AWS Backup は、これらのリソースのバックアップを設定、スケジュール、モニタリングす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/アクティブの場合、Amazon Relational Database Service (Amazon RDS) DB インスタンスまたは Amazon DynamoDB テーブルなど、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に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。

   リージョン間での AWS サービスの運用方法の詳細については、「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」に関するブログシリーズを参照してください。

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。詳細については、「REL09-BP01 バックアップが必要なすべてのデータを特定してバックアップする、またはソースからデータを再現する」を参照してください。インフラストラクチャを再構築するには、Amazon Virtual Private Cloud (Amazon VPC)、サブネット、セキュリティグループに加えて、EC2 インスタンスなどのリソースの作成も必要です。復元プロセスの大部分を自動化できます。詳細については、このブログ記事を参照してください。

5. 必要なとき (災害発生時) にフェイルオーバーするトラフィックを再ルーティングする方法を決定し、実装します。

   このフェイルオーバー操作は、自動または手動で開始できます。ヘルスチェックまたはアラームに基づくフェイルオーバーの自動開始を使用するときには、不要なフェイルオーバー (誤ったアラーム) によって、使用できないデータやデータ損失などのコストが発生するため、注意が必要です。そのため、多くの場合、手動によるフェイルオーバーの開始が使用されます。この場合でも、フェイルオーバーのステップを自動化できるため、手動開始はボタンを押すようなものです。

   AWS サービスを使用するときに検討すべき、いくつかのトラフィック管理オプションがあります。1 つのオプションは、Amazon Route 53 を使用することです。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。手動で開始されるフェイルオーバーを実装するには、Amazon Application Recovery Controller を使用できます。これは、リカバリリージョンにトラフィックを再ルーティングするための高可用性データプレーン API を提供します。フェイルオーバーを実装するときには、「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」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。

   これらのオプションの詳細については、「ディザスタリカバリホワイトペーパー」の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 グローバルテーブルを使用している場合、プライマリリージョンのテーブルが使用できなくなった場合でも、オンラインに復帰すると、DynamoDB が保留中の書き込みの伝播を再開します。Amazon Aurora Global Database を使用していて、管理された計画的なフェイルオーバー を使用している場合、Aurora Global Database の既存のレプリケーショントポロジは維持されます。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。

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、「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てバックアップする、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損失に関する復旧目標を定義する”

関連ドキュメント:

- AWS アーキテクチャブログ: ディザスタリカバリシリーズ
- AWS でのワークロードの災害対策: クラウド内での復旧 (AWS ホワイトペーパー)
- クラウド内での災害対策オプション
- Build a serverless multi-region, active-active backend solution in an hour
- Multi-region serverless backend — reloaded
- 別の AWS リージョンでのリードレプリカの作成
- Route 53: Configuring DNS Failover
- S3: クロスリージョンレプリケーション
- AWS Backup とは
- Amazon Application Recovery Controller とは
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: 開始方法 - AWS
- APN Partner: partners that can help with disaster recovery
- AWS Marketplace: ディザスタリカバリに使用できる製品

関連動画:

- Disaster Recovery of Workloads on AWS
- 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用アーキテクチャパターン (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

関連する例:

- Well-Architected Lab - Disaster Recovery - DR 戦略を解説する一連のワークショップ

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を実施して、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を活用し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。例えば、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 戦略のドリルインスタンスを実装および起動します。AWS Elastic Disaster Recovery は、ドリルを効率的に実行する機能を提供して、フェイルオーバーイベントの準備に役立つます。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

関連ドキュメント:

- APN パートナー: ディザスタリカバリ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: ディザスタリカバリに活用できる商品
- AWS Elastic Disaster Recovery
- AWS 上のワークロードのディザスタリカバリ: クラウドにおけるリカバリ (AWS ホワイトペーパー)
- AWS Elastic Disaster Recovery 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
- AWS Fault Injection Simulator とは

関連動画:

- 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用アーキテクチャパターン
- AWS re:Invent 2019: AWS のバックアップおよび復元、ディザスタリカバリソリューション

関連する例:

- Well-Architected ラボ - 回復性テスト

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。例えば、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。ドリフトを検出し、AWS Systems Manager Automation を呼び出して修正し、アラームを生成できます。また、AWS CloudFormation は、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活用するメリット: 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を活用しない場合のリスクレベル: 中

実装のガイダンス

- デ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ディザスタリカバリ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
- 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

関連ドキュメント:

- APN パートナー: ディザスタリカバリ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: ディザスタリカバリに活用できる商品
- AWS Systems Manager Automation
- AWS 上のワークロードのディザスタリカバリ: クラウドにおけるリカバリ (AWS ホワイトペーパー)
- 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: 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用アーキテクチャパターン (ARC209-R2)

REL13-BP05 復旧を自動化する

AWS またはサードパーティー製ツールを使用して、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ーゾーンに負荷を分散できます。また、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。Amazon 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、アベイラビリティーゾーン、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。

既存の物理データセンター、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の場合、AWS Elastic Disaster Recovery を使用して AWS で自動ディザスタリカバリ戦略をセットアップします。Elastic Disaster Recovery は、AWS でクロスリージョンおよびクロスアベイラビリティーゾーンの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: 中

実装のガイダンス

- 復旧経路を自動化します。復旧時間を短縮するには、ディザスタリカバリプランに従って、障害発生時に IT システムを迅速にオンラインに戻してください。
  - 自動フェイルオーバーとフェイルバックには Elastic Disaster Recovery を使用します。Elastic 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、ファイルを含む) をターゲット AWS アカウントと優先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複製を行います。災害が発生した場合、Elastic Disaster Recovery を使用して復旧することを選択すると、Elastic Disaster Recovery は、レプリケートされたサーバーを AWS のリカバリリージョンで完全にプロビジョニングされたワークロードに変換するのを自動化します。
    - Elastic Disaster Recovery の自動フェイルオーバーとフェイルバック
    - AWS Elastic Disaster Recovery のリソース

リソース

関連ドキュメント:

- APN パートナー: ディザスタリカバリ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: ディザスタリカバリに活用できる商品
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- AWS 上のワークロードのディザスタリカバリ: クラウドにおけるリカバリ (AWS ホワイトペーパー)

関連動画:

- 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用アーキテクチャパターン (ARC209-R2)

パフォーマンス効率

パフォーマンス効率の柱には、クラウドリソースを効率的に使用してパフォーマンス要件を満たし、需要の変化や技術の進歩に合わせてこ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

- アーキテクチャの選択
- コンピューティングとハードウェア
- データ管理
- ネットワークとコンテンツ配信
- プロセスと文化

アーキテクチャの選択

Questions

- PERF 1. ワークロードに適切なクラウドリソースとアーキテクチャを選択するにはどうすればよいでしょうか?

PERF 1. ワークロードに適切なクラウドリソースとアーキテクチャを選択するにはどうすればよいでしょうか?

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し、異なる機能を有効化して、パフォーマンスを向上させます。

ベストプラクティス

- PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する
- PERF01-BP02 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使用して、アーキテクチャパターンとベストプラクティスについて学ぶ
- PERF01-BP03 アーキテクチャに関する意思決定においてコストを考慮する
- PERF01-BP04 トレードオフが顧客とアーキテクチャの効率にどのように影響するかを評価する
- PERF01-BP05 ポリシーとリファレンスアーキテクチャを使用する
- PERF01-BP06 ベンチマークを使用してアーキテクチャに関する意思決定を行う

- PERF01-BP07 データ駆動型のアプローチでアーキテクチャを選択する

PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する

利用可能なサービスや設定について継続的に学び、発見します。これにより、アーキテクチャに関する意思決定をより適切に行い、ワークロード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。

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。
- クラウドへの移行後、アプリケーションをモダナイズしない。
- 永続化する必要があるすべてのものに対して、1 つのストレージタイプのみを使用する。
- 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する。
- 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: 新しいサービスと設定を検討することで、パフォーマンスを大幅に向上させ、コストを削減し、ワークロードの維持に必要な労力を最適化できる場合があります。また、クラウド対応製品の価値実現までの時間を短縮できる可能性もあります。

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: 高

実装のガイダンス

AWS では、パフォーマンスを向上させ、クラウドワークロードのコストを削減するための新しいサービスや機能を継続的にリリースしています。クラウドでのパフォーマンスの有効性を維持するためには、こうした新しいサービスや機能に関する最新情報を常に把握しておくことが重要です。ワークロードアーキテクチャのモダナイズは、生産性の向上、イノベーションの促進、成長機会の拡大にも役立ちます。

実装手順

- 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を棚卸しします。どのカテゴリの製品について詳しく調べるかを決めます。
- AWS の提供サービスを調べ、パフォーマンスの向上、コストの削減、運用の煩雑さの軽減に役立つ関連サービスと設定オプションを特定、把握します。
  - Amazon Web Services Cloud

  - AWS Academy
  - AWS の最新情報
  - AWS ブログ
  - AWS スキルビルダー
  - AWS イベントスケジュール
  - AWS トレーニング と認定
  - AWS Youtube Channel
  - AWS Workshops
  - AWS コミュニティ
- Amazon Q を使用してサービスの関連情報やアドバイスを入手しましょう。
- サンドボックス (非実稼働) 環境を使用して、追加コストをかけずに新しいサービスについて学び、試してみます。
- 新しいクラウドサービスや機能について継続的に学びましょう。

リソース

関連ドキュメント:

- Overview of Amazon Web Services
- Amazon EC2 の特徴
- AWS パートナーラーニングプランでステップバイステップで学ぶ
- AWS トレーニングと認定
- My learning path to become an AWS solutions architect
- 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 情報センター
- AWS でモダンアプリケーションを構築する

関連動画:

- AWS re:Invent 2023 - What's new with Amazon EC2

- AWS re:Invent 2022 - Reduce your operational and infrastructure costs with Amazon ECS
- AWS re:Invent 2023 - Build with the efficiency, agility & innovation of the cloud with AWS
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- This is my Architecture

関連する例:

- AWSサンプル
- AWS SDK Examples

PERF01-BP02 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使用して、アーキテクチャパターンとベストプラクティスについて学ぶ

アーキテクチャに関する意思決定の指針として、ドキュメント、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、適切なパートナーなどのクラウド企業のリソースを活用します。こうしたリソースを利用することで、アーキテクチャを評価、改善し、最適なパフォーマンスを実現できます。

一般的なアンチパターン:

- AWS を一般的なクラウドプロバイダーとして使用する。
- 意図されていない方法で AWS のサービスを利用する。
- ビジネス上の背景を考慮せずに、すべてのガイダンスに従う。

このベストプラクティスを活用するメリット: クラウドプロバイダーや適切なパートナーからのガイダンスを利用することで、ワークロードに適したアーキテクチャを選択し、自信を持っ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: 中

実装のガイダンス

AWS には、効率的なクラウドワークロードの構築と管理に役立つさまざまなガイダンス、ドキュメント、リソースが用意されています。AWS ドキュメントには、コードサンプル、チュートリアルのほか、サービスについての詳細な説明が記載されています。ドキュメントの他にも、AWS では、お客様がクラウドサービスのさまざまな側面を探求し、AWS で効率的なクラウドアーキテクチャを実

装するのに役立つトレーニングおよび認定プログラム、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を提供しています。

こうしたリソースを活用して、貴重な知識やベストプラクティスに関するインサイトを導き出し、AWS クラウドでの時間の節約、成果の向上につなげましょう。

実装手順

- AWS ドキュメントとガイダンスを確認し、ベストプラクティスに従ってください。こうしたリソースは、サービスの効果的な選定および設定、パフォーマンスの向上に役立ちます。
  - AWS ドキュメントへようこそ (ユーザーガイド、ホワイトペーパーなど)
  - AWS ブログ
  - AWS トレーニング と認定
  - AWS Youtube Channel
- AWS パートナーイベント (AWS Global Summits、AWS re:Invent、ユーザーグループ、ワークショップなど) に参加して、AWS のエキスパートから AWS のサービスを利用する際のベストプラクティスを学びましょう。
  - AWS パートナーラーニングプランでステップバイステップで学ぶ
  - AWS イベントスケジュール
  - AWS Workshops
  - AWS コミュニティ
- 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と 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おけ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。
- サービスを効果的に使用するために技術的なサポートが必要な場合は、AWS Support をご利用ください。当社のサポートプランは、お客様にパフォーマンスを最適化しリスクとコストを管理しながら AWS を最大限に活用していただけるように、適切なツールと専門知識を備えて設計されています。

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 情報センター
- AWS エンタープライズサポート

関連動画:

- This is my Architecture
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Implementing distributed design patterns on AWS
- AWS re:Invent 2023 - Application architecture as code

関連する例:

- AWSサンプル
- AWS SDK Examples
- AWS Analytics Reference Architecture

PERF01-BP03 アーキテクチャに関する意思決定においてコストを考慮する

アーキテクチャに関する意思決定でコストを考慮すると、クラウドワークロードのリソース使用率とパフォーマンス効率が向上します。クラウドワークロードがコストに及ぼす影響を認識していれば、効率的なリソースを活用し、無駄な作業を減らせる可能性が高くなります。

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。
- 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。
- ストレージライフサイクルポリシーを定義しない。
- AWS クラウドの新しいサービスや機能を確認しない。
- あなたは、ブロックストレージのみを使用します。

このベストプラクティスを活用するメリット: 意思決定にコストを考慮することで、より効率的なリソースを使用し、他の投資を検討できるようになります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

コストを考慮してワークロードを最適化することで、リソース使用率を向上させ、クラウドワークロードの無駄を省くことができます。アーキテクチャ上の意思決定にコストを考慮に入れることには、通常、ワークロードコンポーネントのライトサイジングと伸縮性の有効化が含まれます。これにより、クラウドワークロードのパフォーマンス効率が向上します。

実装手順

- クラウドワークロードの予算制限などのコスト目標を設定します。
- ワークロードのコストを左右する主要コンポーネント (インスタンスやストレージなど) を特定します。AWS Pricing Calculator と AWS Cost Explorer を使ってワークロードの主なコスト要因を特定できます。
- クラウドの料金モデル (オンデマンド、リザーブドインスタンス、Savings Plans、スポットインスタンスなど) について理解します。
- Well-Architected コスト最適化のベストプラクティスを使用して、これらの主なコスト要因を最適化します。
- コストを継続的にモニタリングおよび分析して、ワークロードにおけるコスト最適化の機会を特定します。
  - AWS Budgets を使用して、許容できないコストに関するアラートを受け取ります。
  - AWS Compute Optimizer または AWS Trusted Advisor を使用して、コスト最適化の推奨事項を取得します。
  - AWS コスト異常検出を使用して、異常なコストの検出と根本原因の分析とを自動化します。

リソース

関連ドキュメント:

- AWS Billing and Cost Managementとは
- AWS によるコスト最適化
- Choosing an AWS cost management strategy
- AWS コスト管理初心者ガイド
- Cost Intelligence Dashboard の詳細な概要
- AWS アーキテクチャセンター
- AWS ソリューションライブラリ

- AWS 情報センター

関連動画:

- This is my Architecture
- AWS re:Invent 2023 - What's new with AWS cost optimization
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

関連する例:

- AWS Compute Optimizer Demo code
- Cost Optimization Workshop
- Cloud Financial Management Technical Implementation Playbooks
- Startup optimization: Tuning application performance for maximum efficiency
- Serverless Optimization Workshop (Performance and Cost)
- Scaling cost effective architectures

PERF01-BP04 トレードオフが顧客とアーキテクチャの効率にどのように影響するかを評価する

パフォーマンス関連の改善を評価する際には、どの選択肢が顧客、そしてワークロードの効率性に影響するかを特定します。例えば、key-value データストアを使用することでシステムパフォーマンスが向上する場合、その結果整合性の特性がお客様に与える影響を評価することが重要です。

一般的なアンチパターン:

- 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改善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で許容でき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

パフォーマンスと顧客への影響の観点から、アーキテクチャの重要な領域を特定します。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。

実装手順

- ワークロード要件と SLA を理解します。
- 評価要因を明確に定義します。要因は、ワークロードのコスト、信頼性、セキュリティ、パフォーマンスに関連する場合があります。
- 要件に対応できるアーキテクチャとサービスを選択します。
- 実験と概念実証 (POC) を実施して、トレードオフ要因と顧客やアーキテクチャ効率への影響を評価します。通常、可用性とパフォーマンスが高く、安全なワークロードは、顧客体験を向上させるものの、消費するクラウドリソースは多くなります。ワークロードの複雑さ、パフォーマンス、コストのトレードオフを理解します。通常、2 つの要素に優先順位を付けると、3 つ目の要素が犠牲になります。

リソース

関連ドキュメント:

- The Amazon Builders' Library
- Amazon QuickSight KPI
- Amazon CloudWatch RUM
- X-Ray Documentation
- 回復性のパターンとトレードオフを理解して、効率的なクラウド設計を実現

関連動画:

- Optimize applications through Amazon CloudWatch RUM
- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

関連する例:

- Amazon CloudWatch Synthetics を使用してページのロード時間を測定する
- Amazon CloudWatch RUM Web Client

PERF01-BP05 ポリシーとリファレンスアーキテクチャを使用する

サービスと設定を選択するときは、ワークロードの設計と実装をより効率的に行うために、社内ポリシーと既存のリファレンスアーキテクチャを使用します。

一般的なアンチパターン:

- 会社の管理諸経費に影響を与える可能性のある幅広い種類のテクノロジーを許可している。

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を活用しない場合のリスクレベル: 中

実装のガイダンス

リソースとアーキテクチャを選択する際の内部ポリシーを設けることで、アーキテクチャを選択する際に従うべき標準とガイドラインが得られます。これらのガイドラインは、適切なクラウドサービスを選択する際の意思決定プロセスを合理化し、パフォーマンス効率の向上に役立ちます。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し、サービスをクラウドデプロイに統合します。次に、パフォーマンステストを使用して、引き続きパフォーマンス要件を満たせることを確認します。

実装手順

- クラウドワークロードの要件を明確に把握します。
- 社内ポリシーと外部ポリシーを確認し、最も関連性の高いポリシーを特定します。
- AWS または業界のベストプラクティスで提供されている適切なリファレンスアーキテクチャを使用します。
- ポリシー、標準、リファレンスアーキテクチャ、および一般的な状況に対応する規範的なガイドラインで構成される一連の流れを作成します。そうすることで、チームがより迅速に行動できるようになります。該当する場合は、業種に合わせてアセットを調整します。
- サンドボックス環境のワークロードに対して、これらのポリシーとリファレンスアーキテクチャを検証します。

- 業界標準や AWS の最新情報を常に把握して、ポリシーとリファレンスアーキテクチャがクラウドワークロードの最適化に役立つことを確認します。

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 情報センター
- AWS アーキテクチャブログ

関連動画:

- This is my Architecture
- AWS re:Invent 2022 - Accelerate value for your business with SAP & AWS reference architecture

関連する例:

- AWSサンプル
- AWS SDK Examples

PERF01-BP06 ベンチマークを使用してアーキテクチャに関する意思決定を行う

既存のワークロードのパフォーマンスをベンチマークに照らして評価すると、クラウドでのパフォーマンスを把握し、そのデータに基づいてアーキテクチャに関する意思決定を行うことができます。

一般的なアンチパターン:

- ワークロードの特性を反映していない一般的なベンチマークを使用している。
- 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

総合テストでベンチマークを使用して、ワークロードのコンポーネントがどのように機能するかを評価します。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、または TPC-DS などの業界標準テストを使用して、ワークロードのベンチマークを実施することができます。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、アーキテクチャで実行する予定の特定タイプの操作を対象とする場合に便利です。

ベンチマーキングを実施するときは、有効な結果が得られるようにテスト環境の暖気運転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、時系列での変動を捉える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。

実装手順

- 計画と定義:
  - ベンチマークの目標、ベースライン、テストのシナリオ、メトリクス (CPU 使用率、レイテンシー、スループットなど)、および KPI を定義します。
  - ユーザーエクスペリエンスに関するユーザー要件や、応答時間やアクセシビリティなどの要素に焦点を当てます。
  - ワークロードに適したベンチマークツールを特定します。AWS のサービス (Amazon CloudWatch など) またはワークロードと互換性のあるサードパーティーツールを使用できます。
- 設定とインストルメント化:
  - 環境を設定し、リソースを設定します。
  - モニタリングとログ記録を実装してテスト結果をキャプチャします。
- ベンチマークとモニタリング:
  - ベンチマークテストを実行し、テスト中のメトリクスをモニタリングします。

- 分析と文書化:
  - ベンチマークプロセスと調査結果を文書化します。
  - 結果を分析して、ボトルネック、傾向、改善が必要な領域を特定します。
  - テスト結果を使用して構造に関する意思決定を行い、ワークロードを調整します。これには、サービスの変更や新機能の導入が含まれる場合があります。
- 最適化と反復:
  - ベンチマークに基づいてリソースの設定と割り当てを調整します。
  - 調整後にワークロードを再テストして、改善点を検証します。
  - 学んだことを文書化し、このプロセスを繰り返して改善すべき他の領域を特定します。

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 情報センター
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- ゲノミクスワークフローパート 5: 自動ベンチマーク
- Amazon SageMaker JumpStart でのエンドポイントデプロイのベンチマークと最適化

関連動画:

- AWS re:Invent 2023 - Benchmarking AWS Lambda cold starts
- Benchmarking stateful services in the cloud
- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

関連する例:

- AWSサンプル
- AWS SDK Examples
- 分散負荷テスト
- Amazon CloudWatch Synthetics を使用してページのロード時間を測定する
- Amazon CloudWatch RUM Web Client

PERF01-BP07 データ駆動型のアプローチでアーキテクチャを選択する

アーキテクチャを選択するための明確でデータ駆動型のアプローチを定義して、特定のビジネスニーズを満たす適切なクラウドサービスと設定が使用されていることを確認します。

一般的なアンチパターン:

- 現在のアーキテクチャが静的であり、今後更新されないと考えている。
- 推測や仮定を基にアーキテクチャを選択している。
- 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: アーキテクチャを選択するための明確なアプローチを持つことで、ワークロードの設計にデータを活用し、情報に基づいた意思決定を長期的に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: 中

実装のガイダンス

クラウドに関する社内の経験や知識、あるいは公開されているユースケース、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、アーキテクチャのリソースとサービスを選択します。ワークロードで利用できるサービスについて、実験とベンチマークを促す明確なプロセスを用意してください。

重要なワークロードのバックログには、ビジネスやユーザーに関連する機能を提供するユーザー関連の背景情報だけでなく、ワークロードのアーキテクチュラルランウェイを形成するテクニカルな背景情報も含める必要があります。このランウェイは、テクノロジーの進歩と新しいサービスからの情報を基に形成され、データと適切な理由に基づいてこうしたテクノロジーやサービスが採用されます。これにより、アーキテクチャの将来性が保たれ、停滞化を防ぐことができます。

実装手順

- 主要な利害関係者と協力して、パフォーマンス、可用性、コストに関する考慮事項を反映したワークロード要件を定義します。ユーザー数やワークロードの使用パターンなどの要素を考慮してください。
- アーキテクチュラルランウェイやテクノロジーバックログを作成して、機能的バックログとともに優先順位を付けます。
- さまざまなクラウドサービスを審査および評価します (詳細については、PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する を参照してください)。
- マイクロサービスやサーバーレスなど、パフォーマンス要件を満たすさまざまなアーキテクチャパターンを検討します (詳細については、PERF01-BP02 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使用して、アーキテクチャパターンとベストプラクティスについて学ぶ を参照してください)。

- 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 アーキテクチャセンター、AWS Partner Network などのリソースを参考にし、ワークロードに最適なアーキテクチャ選びに役立てます。

- ワークロードのパフォーマンスを評価するのに役立つスループットや応答時間などのパフォーマンスメトリクスを定義します。
- 定義したメトリクスを試用し、選択したアーキテクチャのパフォーマンスを検証します。
- アーキテクチャの最適なパフォーマンスを維持するために、継続的にモニタリングし、必要に応じて調整を行います。
- 選択したアーキテクチャと決定事項を、今後のアップデートや学習の参考として文書化します。
- 学習したことや新しいテクノロジー、さらに現在のアプローチで必要とされる変更や問題を示す指標に基づいて、アーキテクチャの選択アプローチを継続的に見直し、更新します。

リソース

関連ドキュメント:

- AWS ソリューションライブラリ
- AWS 情報センター
- AWS でエンドツーエンドのデータ駆動型アプリケーションを構築するためのアーキテクチャパターン

関連動画:

- This is my Architecture
- AWS re:Invent 2021 - Data-driven enterprise: Going from vision to value
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results

関連する例:

- AWSサンプル
- AWS SDK Examples

コンピューティングとハードウェア

Questions

- PERF 2. コンピューティングリソースを選択し、ワークロードで使用するにはどうすればよいでしょうか?

PERF 2. コンピューティングリソースを選択し、ワークロードで使用するにはどうすればよいでしょうか?

特定のワークロードに対する最適なコンピューティングの選択は、アプリケーションの設計、利用パターン、および構成設定に応じて異なります。アーキテクチャでは、各種コンポーネントに異なるコンピューティングを使用し、異なる機能を有効化してパフォーマンスを向上させることができます。アーキテクチャに誤ったコンピューティングを選択することは、パフォーマンス効率の低下につながる可能性があります。

ベストプラクティス

- PERF02-BP01 ワークロードに最適なコンピューティングオプションを選択する
- PERF02-BP02 利用可能なコンピューティング設定と機能について理解する
- 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 コンピューティングリソースの設定とライトサイジングを行う
- PERF02-BP05 コンピューティングリソースを動的にスケールする

- PERF02-BP06 最適化された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する

PERF02-BP01 ワークロードに最適なコンピューティングオプションを選択する

ワークロードに最適なコンピューティングオプションを選択することで、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、ワークロードを維持するために必要な運用工数を軽減できます。

一般的なアンチパターン:

- オンプレミスで使用されていたものと同じコンピューティングオプションを使用している。
- クラウドコンピューティングのオプション、機能、ソリューション、および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オプ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オプションを過剰に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: コンピューティング要件を特定し、利用可能なオプションに照らし合わせて評価することで、ワークロードのリソース効率を高めることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

クラウドワークロードを最適化してパフォーマンスを効率化するには、ユースケースとパフォーマンス要件に最適なコンピューティングオプションを選択することが重要です。AWS では、クラウド内のさまざまなワークロードに対応するさまざまなコンピューティングオプションを用意しています。例えば、Amazon EC2 を使用して仮想サーバーを起動・管理する、AWS Lambda を使用してサーバーのプロビジョニングや管理を行うことなくコードを実行する、Amazon ECS や Amazon EKS を使用してコンテナを実行・管理する、AWS Batch を使用して大量のデータを並列処理するといったことが可能です。スケールとコンピューティングニーズを基に、状況に最適なコンピューティングソリューションを選んで構成する必要があります。また、コンピューティングソリューションにもそれぞれ利点と欠点があるため、1 つのワークロードで複数のタイプを使用することも検討できます。

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わせて適切なコンピューティングオプションを選択する方法を説明します。

実装手順

- ワークロードのコンピューティング要件を把握します。主な要件には、処理ニーズ、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。
- ワークロードのさまざまな AWS コンピューティングサービスについて説明します。詳細については、「PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する」を参照してください。AWS の主要なコンピューティングオプション、その特徴、一般的な使用例は次のとおりです。

AWS のサービス	主な特徴	一般的なユースケース
Amazon Elastic Compute Cloud (Amazon EC2)	ハードウェア、ライセンス要件、さまざまなインスタンスファミリーの幅広い選択肢、プロセッサタイプおよびコンピューティングアクセラレーターの専用オプション	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
Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)	簡単なデプロイ、一貫性のある環境、スケーラビリティ	マイクロサービス、ハイブリッド環境
AWS Lambda	イベントに対応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。	マイクロサービス、イベント駆動型アプリケーション
AWS Batch	ジョブ要件に基づいてオンデマンドインスタンスまたはスポットインスタンスを使用するオプションを備え、Amazon Elastic Container Service (Amazon	HPC、機械学習モデルのトレーニング

AWS のサービス	主な特徴	一般的なユースケース
	ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、AWS Fargate コンピューティングリソースを効率的かつ動的にプロビジョニングおよびスケールします。	
Amazon Lightsail	小規模なワークロードを実行するための事前構成済みの Linux および Windows アプリケーション	シンプルなウェブアプリケーション、カスタムウェブサイト

- 各コンピューティングオプションに関連するコスト (時間単位の料金やデータ転送など) と管理諸経費 (パッチ適用やスケーリングなど) を評価します。
- 非運用環境で実験とベンチマーキングを行い、どのコンピューティングオプションがワークロード要件に最も適しているかを特定します。
- 実験を通じて新しいコンピューティングソリューションを特定したら、移行を計画し、パフォーマンスメトリクスを検証します。
- AWS が提供する Amazon CloudWatch などのモニタリングツールや AWS Compute Optimizer などの最適化サービスを使用して、実際の使用パターンに基づいてコンピューティングを継続的に最適化します。

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon EKS コンテナ: Amazon 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 Function Configuration
- 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:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AMS
- AWS re:Invent 2023 - What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Powering next-gen Amazon Elastic Compute Cloud: Deep dive on the Nitro System
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- AWS re:Invent 2019 - Amazon Elastic Compute Cloud foundations
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- Amazon EC2 foundations
- 「Deploy ML models for inference at high performance and low cost」

関連する例:

- ウェブアプリケーションをコンテナに移行する
- サーバーレスの Hello World を実行する
- Amazon EKS ワークショップ
- Amazon EC2 ワークショップ
- Amazon Elastic Compute Cloud Auto Scaling による効率的で回復力のあるワークロード
- コンテナサービスと共に AWS Graviton に移行する

PERF02-BP02 利用可能なコンピューティング設定と機能について理解する

コンピューティングサービスで利用できる設定オプションと機能を理解しておくと、適切な量のリソースをプロビジョニングしてパフォーマンス効率を向上させることができます。

一般的なアンチパターン:

- コンピューティングオプションや利用可能なインスタンスファミリーをワークロードの特性に照らして評価しない。
- ピーク需要の要件を満たすために、コンピューティングリソースを過剰に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 のコンピューティングの機能と設定に精通していれば、ワークロードの特性とニーズに合わせて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

各コンピューティングソリューションは、さまざまなワークロードの特性と要件に対応するために、それぞれ異なる設定や機能を備えています。こうしたオプションがワークロードをどのように補完するかを理解し、アプリケーションにどの設定オプションが最適か判断します。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。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、現在のコンピューティングオプションがワークロードに適しているかどうかを、CPU とメモリの観点から確認できます。

実装手順

- ワークロード要件 (CPU ニーズ、メモリ、レイテンシーなど) を把握します。
- AWS ドキュメントとベストプラクティスで、コンピューティングパフォーマンスの向上に役立つ推奨構成オプションについて確認します。考慮すべき主な設定オプションは次のとおりです。

設定オプション	例
インスタンスタイプ	• コンピューティング最適化インスタンスは、高い vCPU 対メモリ比を必要とするワークロードに最適です。 • 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。 • 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。
料金モデル	• 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

設定オプション	例
	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。 • Savings Plans を使用すると、1 年または 3 年の単位で一定量のコンピューティング容量を確約することで、オンデマンドインスタンスと比較して大幅にコストを削減できます。 • スポットインスタンスでは、ステートレスで耐障害性のあるワークロードで、未使用のインスタンス容量を割引価格で利用できます。
Auto Scaling	自動スケーリング設定を使用して、コンピューティングリソースをトラフィックパターンに一致させます。
サイジング	• Compute Optimizer を使用すると、機械学習によって、コンピューティング特性に最適なコンピューティング設定についての推奨事項が提示されます。 • AWS Lambda Power Tuning を使用して、Lambda 関数に最適な設定を選択します。
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	• 拘束コンピューティングインスタンスは、CPU ベースの代替インスタンスよりも効率的にグラフィック処理やデータパターンマッチングなどの機能を実行できます。 • 機械学習のワークロードには、AWS Trainium、AWS Inferentia、Amazon EC2 DL1 などのワークロード専用ハードウェアを活用してください。

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
- Amazon EC2 インスタンスタイプ
- Amazon EC2 インスタンスのプロセッサ状態制御
- Amazon EKS コンテナ: Amazon 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 Function Configuration

関連動画:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations
- AWS re:Invent 2022 – Optimizing Amazon EKS for performance and cost on AWS

関連する例:

- Compute Optimizer demo code
- Amazon EC2 spot instances workshop
- Efficient and Resilient Workloads with Amazon EC2 AWS Auto Scaling
- Graviton developer workshop
- AWS for Microsoft workloads immersion day
- AWS for Linux workloads immersion day
- AWS Compute Optimizer Demo code
- Amazon EKS ワークショップ

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関連のメトリクスを記録および追跡することで、コンピューティングリソースのパフォーマンスをよりよく理解し、パフォーマンスと使用率を向上させます。

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。
- 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: パフォーマンス関連のメトリクスを収集することで、アプリケーションのパフォーマンスとビジネス要件の整合性をとり、ワークロードのニーズを満たすことができます。また、ワークロードにおけるリソースのパフォーマンスと使用率を継続的に改善するのにも役立ちます。

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

クラウドワークロードでは、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。AWS クラウドでは、メトリクスの収集は、セキュリティ、コスト効率、パフォーマンス、持続可能性を向上させるために不可欠なステップです。AWS では、Amazon CloudWatch のようなモニタリングサービスを使用して、有益なインサイトにつながるさまざまなパフォーマンス関連のメトリクスを提供しています。CPU 使用率、メモリ使用率、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などのメトリクスにより、使用率レベルやパフォーマンスのボトルネックを把握できます。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。 理想は、コンピューティングリソースに関連するすべてのメトリクスを単一のプラットフォームで収集し、コストと運用上の目標をサポートするための保持ポリシーを実装することです。

実装手順

- ワークロードに関連するパフォーマンス関連の指標を特定します。リソース使用率やクラウドワークロードの動作状況 (応答時間やスループットなど) に関するメトリクスを収集する必要があります。
  - Amazon EC2 のデフォルトのメトリクス
  - Amazon ECS のデフォルトのメトリクス
  - Amazon EKS のデフォルトのメトリクス

  - Lambda のデフォルトのメトリクス
  - Amazon EC2 のメモリとディスクのメトリクス
- ワークロードに適したログ記録とモニタリングのソリューションを選んでセットアップします。
  - AWS ネイティブオブザーバビリティ
  - AWS Distro for OpenTelemetry
  - Amazon Managed Service for Prometheus
- ワークロード要件に基づいて、メトリクスに必要なフィルターと集計を定義します。
  - Amazon CloudWatch Logs とメトリクスフィルターでカスタムアプリケーションメトリクスを定量化する
  - Amazon CloudWatch での戦略的タグ付けによるカスタムメトリクスの収集
- セキュリティと運用の目標に合わせて、メトリクスのデータ保持ポリシーを設定します。
  - CloudWatch メトリクスのデフォルトのデータ保持
  - CloudWatch ログのデフォルトのデータ保持
- パフォーマンス関連の問題に積極的に対応できるよう、必要に応じて、メトリクスのアラームと通知を作成します。
  - Amazon CloudWatch 異常検知によりカスタムメトリクスのアラームを作成
  - Amazon CloudWatch RUM で特定のウェブページのメトリクスとアラートを作成する
- 自動化を利用して、メトリクス・ログ集計エージェントをデプロイします。
  - AWS Systems Manager オートメーション
  - OpenTelemetry Collector

リソース

関連ドキュメント:

- モニタリングとオブザーバビリティ
- ベストプラクティス: AWS によるオブザーバビリティの実装
- 「Amazon 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、ログ、トレースを収集する
- AWS Lambda の Amazon CloudWatch Logs へのアクセス
- Amazon ECS のモニタリングツール

- カスタムメトリクスをパブリッシュする
- AWS の回答: 統合ログ管理
- CloudWatch メトリクスを発行する AWS のサービス
- Prometheus と Grafana を使用して AWS Fargate で Amazon EKS をモニタリングする

関連動画:

- AWS re:Invent 2023 – [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 – Implementing application observability
- AWS re:Invent 2023 – Building an effective observability strategy
- AWS re:Invent 2023 – Seamless observability with AWS Distro for OpenTelemetry
- Application Performance Management on AWS

関連する例:

- AWS での Linux ワークロード Immersion Day - Amazon CloudWatch
- Amazon ECS クラスターとコンテナのモニタリング
- Amazon CloudWatch ダッシュボードによるモニタリング
- Amazon EKS ワークショップ

PERF02-BP04 コンピューティングリソースの設定とライトサイジングを行う

ワークロードのパフォーマンス要件に合わせてコンピューティングリソースの設定とライトサイジングを行うことで、リソースの過不足を防ぎます。

一般的なアンチパターン:

- ワークロードのパフォーマンス要件を無視した結果、コンピューティングリソースのプロビジョニングが過剰になったり不足したりする。
- 使用できる最大または最小のインスタンスのみをすべてのワークロードに対して選択する。
- 管理を容易にするため、1 つのインスタンスファミリーのみを使用する。
- AWS Cost Explorer または Compute Optimizer からのライトサイジングに関する推奨事項を無視する。
- 新しいインスタンスタイプが適合するかどうかについてワークロードを再評価しない。

- 組織で使用できるインスタンス設定としてごく少数のみを認証する。

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リソースをライトサイジングすることで、リソースの過剰プロビジョニングやプロビジョニング不足を回避できるため、クラウドでの運用が最適化されます。通常、コンピューティングリソースのサイズを適切に設定すると、パフォーマンスが上がり、カスタマーエクスペリエンスが向上すると同時に、コストも削減されます。

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: 中

実装のガイダンス

ライトサイジングにより、組織はビジネスニーズに対応しながら、効率的かつ費用対効果の高い方法でクラウドインフラストラクチャを運用できます。クラウドリソースの過剰プロビジョニングは追加コストにつながる可能性がありますが、プロビジョニング不足はパフォーマンスの低下やカスタマーエクスペリエンスの低下につながる可能性があります。AWS は、履歴データを使用してコンピューティングリソースを適切なサイズにするためのレコメンデーションを提供する AWS Compute Optimizer や AWS Trusted Advisor などのツールを提供します。

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 Fleet 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 グループを作成する
  - Karpenter の統合機能を用いた Kubernetes コンピューティングコストの最適化
- ワークロードのさまざまなパフォーマンス特性とそれらの特性とメモリ、ネットワーク、CPU 使用率との関連を分析します。このデータを使用して、ワークロードのプロファイルとパフォーマンス目標に最適なリソースを選択します。
- Amazon CloudWatch などのモニタリングツールを使用して、AWS リソースの使用状況をモニタリングします。
- コンピューティングリソースの適切な構成を選択します。
  - エフェメラルワークロードの場合は、`CPUUtilization` などのインスタンスの Amazon CloudWatch メトリクスを評価して、インスタンスが十分に活用されていないか、過剰に活用されているかを特定します。
  - 安定したワークロードの場合は、AWS のライトサイジングツール (AWS Compute Optimizer、AWS Trusted Advisor など) を定期的にチェックし、インスタンスの最適化とライトサイジングの機会を特定します。

- 構成の変更は、本番環境に実装する前に非運用環境でテストします。
- 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、ワークロードのニーズと照らし合わせます。

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
- 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 Function Configuration
- Amazon EC2 インスタンスのプロセッサ状態制御

関連動画:

- Amazon EC2 foundations
- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

関連する例:

- AWS Compute Optimizer Demo code
- Amazon EKS ワークショップ
- 適切なサイジングの推奨事項

PERF02-BP05 コンピューティングリソースを動的にスケールする

クラウドの伸縮性を利用して、ニーズに合わせてコンピューティングリソースを動的にスケールアップまたはスケールダウンすることで、ワークロードのキャパシティが過剰または過少になるのを防ぐことができます。

一般的なアンチパターン:

- アラームに対応するために手動でキャパシティを増やす。
- オンプレミスと同じサイズ設定ガイドライン (通常は静的インフラストラクチャ) を使用する。
- 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、キャパシティを増加させたままにする。

このベストプラクティスを活用するメリット: コンピューティングリソースの伸縮性を設定してテストすることで、コストの節約、パフォーマンスベンチマークの維持、トラフィックの変化に応じた信頼性の向上に役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS は、需要の変化に対応するためのさまざまなスケーリングメカニズムを通じて、リソースを動的にスケールアップまたはスケールダウンする柔軟性を備えています。コンピューティング関連のメトリクスと組み合わせると、動的スケーリングにより、ワークロードが自動的に変化に対応し、最適なコンピューティングリソースを使用して目標を達成できるようになります。

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モニタリングし、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。
- サービスのスケーリング: 設計により自動的にスケールされるサービス (サーバーレスなど) を選択します。

ワークロードのデプロイメントで、確実にスケールアップおよびスケールダウンイベントを対処できるようにしてください。

実装手順

- コンピューティングインスタンス、コンテナ、関数のいずれにも、伸縮性の仕組みが備わっています。サービスの機能として実装されている場合も、自動スケーリングと組み合わせて実現する場合もあります。自動スケーリングメカニズムの例を次に示します。

自動スケーリングメカニズム	使用する場所
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適切な数の Amazon EC2 インスタンスがあることを確認できます。
Application Auto Scaling	個別の AWS サービス (AWS Lambda 関数や Amazon Elastic Container Service (Amazon ECS) サービスなど) のリソースを、Amazon EC2 を超えて自動的にスケールします。
Karpenter のご紹介 – オープンソースの高性能 Kubernetes Cluster Autoscaler	Kubernetes クラスターを自動的にスケールする。

- スケーリングは大抵、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取り上げられます。AWS Glue のようなコンピューティング以外のサービスの設定も、考慮してください。
- スケーリングのメトリクスが、デプロイされているワークロードの特性と一致していることを確認します。動画トランスコーディングアプリケーションをデプロイしようとする場合、100% の CPU 使用率が想定されるため、プライマリメトリクスにするべきではありません。代わりに、トランスコーディングジョブのキュー深度を使用してください。必要に応じて、スケーリングポリシーにカスタマイズされたメトリクスを使用できます。適切なメトリクスを選ぶには、Amazon EC2 の以下のガイダンスを考慮してください。
  - メトリクスは有効な利用率メトリクスでなければならず、インスタンスのどの程度ビジーかを記述する必要があります。
  - メトリクス値は Auto Scaling グループ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 グループには、手動スケーリングの代わりに動的スケーリングを使用してください。また、動的スケーリングではターゲット追跡スケーリングポリシーを使用することをお勧めします。

- ワークロードのデプロイがスケーリングイベント (アップとダウン) の両方に対応処理できることを確認します。例えば、アクティビティ履歴を使用して、Auto Scaling グループのスケーリングアクティビティを検証できます。
- 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 の予測スケーリング」を参照してください。

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
- 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 Function Configuration
- Amazon EC2 インスタンスのプロセッサ状態制御
- Amazon ECS クラスターの Auto Scaling を深く探る
- Karpenter のご紹介 – オープンソースの高性能 Kubernetes Cluster Autoscaler

関連動画:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

関連する例:

- Amazon EC2 Auto Scaling Group Examples
- Amazon EKS ワークショップ

- IPv6 での実行により Amazon EKS ワークロードをスケールする

PERF02-BP06 最適化された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する

ハードウェアアクセラレーターを使用すると、CPU ベースの代替手段よりも効率的に特定の機能を実行できます。

一般的なアンチパターン:

- ワークロードで、より高いパフォーマンスとより低いコストを実現できる専用のインスタンスに対する汎用インスタンスのベンチマーキングを行っていない。
- 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。
- GPU の使用状況を監視していない。

このベストプラクティスを活用するメリット: GPU (グラフィックス処理ユニット) や FPGA (フィールドプログラマブルゲートアレイ) などのハードウェアベースのアクセラレーターを使用することで、特定の処理機能をより効率的に実行できます。

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: 中

実装のガイダンス

高速コンピューティングインスタンスを使用すると、GPU や 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にアクセスできます。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、必要のない場合は自動で廃止することで、全体的なパフォーマンス効率を向上させることができます。

実装手順

- 要件に対応できる高速コンピューティングインスタンスを特定します。
- 機械学習のワークロードには、AWS Trainium、AWS Inferentia、Amazon EC2 DL1 などのワークロード専用ハードウェアを活用してください。AWSInf2 インスタンスなどの 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。

- 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、「Amazon CloudWatch で NVIDIA GPU メトリクスを収集する」のように、CloudWatch エージェントを使用して GPU の `utilization_gpu` や `utilization_memory` などのメトリクスを収集できます。
- 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 設定の最適化
  - Deep Learning AMI での GPU のモニタリングと最適化
  - Amazon SageMaker でのディープラーニング学習時における、GPU パフォーマンスチューニングのための I/O 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

関連ドキュメント:

- Amazon Elastic Container Service での GPU の使用
- GPU インスタンス
- AWS Trainium を含むインスタンス
- AWS Inferentia を持つインスタンス
- Let's Architect! Architecting with custom chips and accelerators

- 高速コンピューティング
- Amazon EC2 VT1 インスタンス
- 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 SageMaker でコンピュータビジョン推論に最適な AI アクセラレータとモデルコンパイルを選択

関連動画:

- AWS re:Invent 2021 - How to select Amazon Elastic Compute Cloud GPU instances for deep learning
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances

- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

関連する例:

- Amazon SageMaker and NVIDIA GPU Cloud (NGC)
- Use SageMaker with Trainium and Inferentia for optimized deep learning training and inferencing workloads
- Optimizing NLP models with Amazon Elastic Compute Cloud Inf1 instances in Amazon SageMaker

データ管理

Questions

- PERF 3. ワークロード内のデータはどのように保存、管理、アクセスすればよいでしょうか?

PERF 3. ワークロード内のデータはどのように保存、管理、アクセスすればよいでしょうか?

特定のシステムに最適なデータ管理ソリューションは、データの種類 (ブロック、ファイル、またはオブジェクト)、アクセスパターン (ランダムまたはシーケンシャル)、必要なスループット、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)、および可用性と耐久性に関する制約に応じて異なります。優れた設計のワークロードは、さまざまな機能によってパフォーマンスを向上させることができる専用のデータストアを使用します。

ベストプラクティス

- PERF03-BP01 データアクセスとストレージ要件に最適な専用データストアを使用する
- PERF03-BP02 データストアで利用可能な設定オプションを評価する
- PERF03-BP03 データストアのパフォーマンスメトリクスを収集・記録する
- PERF03-BP04 データストアのクエリパフォーマンスを向上させるための戦略を実装する
- PERF03-BP05 キャッシュを利用するデータアクセスパターンを実装する

PERF03-BP01 データアクセスとストレージ要件に最適な専用データストアを使用する

データの特性 (共有可能、サイズ、キャッシュサイズ、アクセスパターン、レイテンシー、スループット、データの持続性など) を理解して、ワークロードに適した専用データストア (ストレージまたはデータベース) を選択します。

一般的なアンチパターン:

- 特定のタイプのデータストア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。
- すべてのワークロードのデータの保存とアクセスの要件が類似していると考えている。
- 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: データの特性と要件を理解することで、ワークロードのニーズに適した、最も効率的でパフォーマンスの高いストレージテクノロジーを特定できます。

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: 高

実装のガイダンス

データストレージを選択して実装する際は、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要件をサポートしていることを確認します。AWS では、ブロックストレージ、オブジェクトストレージ、ストリーミングストレージ、ファイルシステム、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、台帳などのデータベースをはじめとした、さまざまなデータストレージとデータベーステクノロジーを提供しています。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。データの特性と要件を理解することで、モノリシックなストレージテクノロジーや制約の多い汎用的なアプローチから脱却し、データの適切な管理に集中できます。

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を棚卸しします。
- 次のようなデータの特性と要件を理解して文書化します。
  - データタイプ (非構造化、半構造化、リレーショナル)
  - データ量と増加
  - データ保存期間: 永続、一時的、一過性
  - ACID 特性 (原子性、一貫性、独立性、耐久性) の要件

- データアクセスパターン (読み取りが多い、または書き込みが多い)
- レイテンシー
- スループット
- IOPS (1 秒あたりの入出力操作数)
- データ保持期間

- PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する で説明している、データ特性を満たし、AWS のワークロードに利用できるさまざまなデータストア (ストレージおよびデータベースサービス) について学びます。AWS の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	AWS サービス	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する複数のオプションがあります。Amazon S3 との間でオブジェクトを転送し、オブジェクトにアクセスするには、Transfer Acceleration やアクセスポイント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 S3 Glacier	データアーカイブ用に構築されています。
ストリーミングストレージ	Amazon Kinesis Amazon Managed Streaming for Apache Kafka (Amazon MSK)	ストリーミングデータを効率的に取り込み保存します。
共有ファイルシステム	Amazon Elastic File System (Amazon EFS)	複数のタイプのコンピューティングソリューションから

タイプ	AWS サービス	主な特徴
		アクセスできるマウント可能なファイルシステムです。
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 のレイテンシー、スループット、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適したファイルシステムを選択する際には、考慮する必要があります。
ブロックストレージ	Amazon Elastic Block Store (Amazon EBS)	Amazon Elastic Compute Cloud (Amazon EC2) のために設計された、スケーラブルな高性能ブロックストレージサービスです。Amazon EBS には、トランザクション、IOPS を多用するワークロード用の SSD ベースのストレージと、スループットを多用するワークロード用の HDD ベースのストレージが含まれています。

タイプ	AWS サービス	主な特徴
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。
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。
ドキュメントデータベース	Amazon DocumentDB	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。

タイプ	AWS サービス	主な特徴
インメモリデータベース	Amazon ElastiCache、Amazon MemoryDB for Redis	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。インメモリデータベースは、アプリケーションキャッシュ、セッション管理、ゲームリーダーボード、低レイテンシーの ML 特徴量ストア、マイクロサービスメッセージングシステム、および高スループットのストリーミングメカニズムに使用できます。
グラフデータベース	Amazon Neptune	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に使用されます。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。

タイプ	AWS サービス	主な特徴
時系列データベース	Amazon Timestream	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すために使用されます。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。
ワイドカラム	Amazon Keyspaces (Apache Cassandra 向け)	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。
台帳	Amazon Quantum Ledger Database (Amazon QLDB)	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。

- データプラットフォームを構築する場合は、AWS で最新のデータアーキテクチャを活用し、データレイク、データウェアハウス、専用データストアを統合します。
- ワークロードのデータストアを選択する際に考慮すべき主なポイントは次のとおりです。

質問	考慮事項
データはどのように構造化されていますか。	• 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、または Amazon DocumentDB などの NoSQL データベースを検討してください。 • キーと値のデータについては、DynamoDB、Amazon ElastiCache (Redis OSS)、または Amazon MemoryDB を検討してください。
どのレベルの参照整合性が必要ですか。	• 外部キーの制約については、Amazon RDS や 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。 • 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。
ACID (atomicity、consistency、isolation、durability) への準拠は必要ですか。	• リレーショナルデータベースに関連付けられた ACID プロパティが必要な場合は、Amazon RDS や Aurora などのリレーショナルデータベースを検討してください。 • NoSQL データベースに強力な整合性が必要な場合は、DynamoDB で強力な整合性のある読み込みを使用できます。

質問	考慮事項
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。	• DynamoDB や Amazon Quantum Ledger Database (Amazon QLDB) などのサーバーレスデータベースは、動的にスケールされます。 • 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使用して水平方向に分割する必要があります。
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。	• 読み取り負荷の高いワークロードは、データベースが DynamoDB の場合、ElastiCache や DAX などのキャッシュレイヤーの恩恵を受けることができます。 • Amazon RDS などのリレーショナル データベースを使用して、読み取りを読み取りレプリカにオフロードすることもできます。
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。	• 高スループットのそのまま読み取るトランザクション処理については、DynamoDB などの NoSQL データベースを検討します。 • 一貫性のある高スループットで複雑な読み取りパターン (join など) には、Amazon RDS を使用します。 • 分析クエリの場合は、Amazon Redshift などの列指向データベース、または Amazon S3 へのデータのエクスポートと、Athena または Amazon QuickSight を使用した分析の実行を検討してください。

質問	考慮事項
データにはどのレベルの耐久性が必要ですか。	• 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、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。 • 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発揮します。 • Amazon S3 は、99.999999999% (イレブンナイン) の耐久性を備えています。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ために、Amazon S3 へのデータのエクスポートをサポートしています。
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。	• 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。 • AWS Database Migration Service と AWS Schema Conversion Tool を活用して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実行する
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。	• 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。

質問	考慮事項
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。	• 外部ツールに依存している場合は、サポートするデータベースとの互換性を維持する必要がある場合があります。Amazon RDS は、Microsoft SQL Server、Oracle、MySQL、PostgreSQL など、サポートしてい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。

- 非運用環境で実験とベンチマーキングを行い、どのデータストアがワークロード要件に対応できるかを特定します。

リソース

関連ドキュメント:

- Amazon EBS ボリュームの種類
- Amazon EC2 ストレージ
- Amazon EFS: Amazon EFS のパフォーマンス
- Amazon FSx for Lustre のパフォーマンス
- Amazon FSx for Windows File Server のパフォーマンス
- Amazon S3 Glacier: S3 Glacier ドキュメント
- Amazon S3: リクエストレートとパフォーマンスに関する考慮事項
- AWS でのクラウドストレージ
- Amazon EBS I/O の特性
- AWS でのクラウドデータベース
- AWS データベースのキャッシュ
- DynamoDB Accelerator
- Amazon Aurora のベストプラクティス
- Amazon Redshift のパフォーマンス
- Amazon Athena パフォーマンスに関するヒントのトップ 10
- Amazon Redshift Spectrum のベストプラクティス
- Amazon DynamoDB のベストプラクティス

- Amazon EC2 と Amazon RDS のどちらかを選ぶか
- Amazon ElastiCache 実装のベストプラクティス

関連動画:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimizing storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2022: Building modern data architectures on AWS
- AWS re:Invent 2022: Building data mesh architectures on AWS
- AWS re:Invent 2023: Deep dive into Amazon Aurora and its innovations
- AWSre:Invent 2023: Advanced data modeling with Amazon DynamoDB
- AWS re:Invent 2022: Modernize apps with purpose-built databases
- Amazon DynamoDB deep dive: Advanced design patterns

関連する例:

- AWS 目的別データベースワークショップ
- 開発者向けデータベース
- AWS モダンデータアーキテクチャ Immersion Day
- AWS でデータメッシュを構築
- Amazon S3 の例
- Amazon Redshift データ共有を使用したデータパターンの最適化
- データベースの移行
- MS SQL Server - AWS Database Migration Service (AWS DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune サンプル

PERF03-BP02 データストアで利用可能な設定オプションを評価する

データストアで使用できるさまざまな機能と設定オプションを理解して評価し、ワークロードに合わせてストレージ容量とパフォーマンスを最適化します。

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している。
- 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。
- 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。
- データストアのスケーリング特性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: データストア設定を確認し、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: 中

実装のガイダンス

ワークロードには、データストレージとアクセス要件に基づいて 1 つまたは複数のデータストアを使用できます。パフォーマンス効率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、適切なデータストア設定を判別する必要があります。データストアのオプションを検討する際には、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、整合性要件、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。こうしたさまざまな設定オプションを試し、パフォーマンス効率のメトリクスを改善します。

実装手順

- データストアの現在の設定 (インスタンスタイプ、ストレージサイズ、データベースエンジンのバージョンなど) を把握します。
- AWS ドキュメントとベストプラクティスで、データストアのパフォーマンス向上に推奨される設定オプションについて確認します。考慮すべき主なデータストアのオプションは次のとおりです。

設定オプション	例
読み取りのオフロード (リードレプリカやキャッシュなど)	• 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。

設定オプション	例
	• Amazon ElastiCache クラスターを作成し、アプリケーションで最初にキャッシュから読み取り、要求されたアイテムが存在しない場合はデータベースにフォールバックするように設定できます。 • 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。 • DynamoDB などのサーバーレスデータベースは自動的にスケールします。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。

設定オプション	例
書き込みのスケーリング (パーティションキーのシャーディングやキューの導入など)	• 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。 • 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) を調整することで、書き込みスループットをスケールできます。 • それでも、特定のパーティションキーのスループット制限に達すると、ホットパーティションで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。

設定オプション	例
データセットのライフサイクルを管理するためのポリシー	• Amazon S3 ライフサイクルを使用すると、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、変更中、または予測不可能な場合は、Amazon S3 Intelligent-Tiering を使用できます。これにより、アクセスパターンがモニタリングされ、アクセスされていないオブジェクトが低コストのアクセス階層に自動的に移動します。Amazon S3 ストレージレンズメトリクスを活用して、ライフサイクル管理の最適化の機会とギャップを特定できます。 • Amazon EFS のライフサイクル管理では、ファイルシステムのファイルストレージが自動的に管理されます。
接続管理とプーリング	• Amazon RDS Proxy を使用して、Amazon RDS および Aurora でデータベースへの接続を管理できます。 • 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ルのポリシーを検討してください。

- 非運用環境で実験とベンチマーキングを行い、どの設定オプションがワークロード要件に対応できるかを特定します。
- 実験が終わったら、移行を計画し、パフォーマンスメトリクスを検証します。
- AWS のモニタリングツール (Amazon CloudWatch など) と最適化ツール (Amazon S3 ストレージレンズなど) を使用して、実際の使用パターンに基づいてデータストアを継続的に最適化します。

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 ボリュームの種類
- Amazon EC2 ストレージ
- Amazon EFS: Amazon EFS のパフォーマンス
- Amazon FSx for Lustre のパフォーマンス
- Amazon FSx for Windows File Server のパフォーマンス
- Amazon S3 Glacier: S3 Glacier ドキュメント
- Amazon S3: リクエストレートとパフォーマンスに関する考慮事項
- Amazon EBS I/O の特性
- AWS でのクラウドデータベース
- AWS データベースのキャッシュ
- DynamoDB Accelerator
- Amazon Aurora のベストプラクティス
- Amazon Redshift のパフォーマンス
- Amazon Athena パフォーマンスに関するヒントのトップ 10
- Amazon Redshift Spectrum のベストプラクティス
- Amazon DynamoDB のベストプラクティス

関連動画:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimize storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2023: What's new with AWS file storage
- AWS re:Invent 2023: Dive deep into Amazon DynamoDB

関連する例:

- AWS 目的別データベースワークショップ
- 開発者向けデータベース
- AWS モダンデータアーキテクチャ Immersion Day

- Amazon EBS Autoscale
- Amazon S3 の例
- Amazon DynamoDB の例
- AWS データベース移行サンプル
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for PostgreSQL DB

PERF03-BP03 データストアのパフォーマンスメトリクスを収集・記録する

データストアに関連するパフォーマンスメトリクスを追跡して記録することで、データ管理ソリューションのパフォーマンスを把握できます。こうしたメトリクスは、データストアの最適化を行い、ワークロードの要件が満たされていることを確認し、ワークロードのパフォーマンスを明確に把握するのに役立ちます。

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、データストアのパフォーマンス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

データストア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、異常を検出できるだけでなく、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満たしていることを確認できます。

メトリクスは、データストアをサポートする基盤システムとデータストア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、キャッシュヒット率、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、データストアのメトリクスには 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管理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。

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。

実装手順

- データストアで追跡すべき主要なパフォーマンスメトリクスを特定します。
  - Amazon S3 のメトリクスとディメンション
  - Amazon RDS インスタンスでのメトリクスのモニタリング
  - Amazon RDS での Performance Insights を使用した DB 負荷のモニタリング
  - Enhanced Monitoring の概要
  - DynamoDB のメトリクスとディメンション
  - DynamoDB Accelerator のモニタリング
  - Amazon CloudWatch による Amazon BMemoryDB のモニタリング
  - モニタリングすべきメトリクス
  - Amazon Redshift クラスターパフォーマンスのモニタリング
  - Timestream のメトリクスとディメンション
  - Amazon Aurora の Amazon CloudWatch メトリクス
  - Amazon Keyspaces (Apache Cassandra 向け) でのログ作成とモニタリング
  - Amazon Neptune リソースのモニタリング
- 承認されたロギングおよびモニタリングソリューションを使用して、これらのメトリクスを収集します。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。

- データストアのモニタリングに、パフォーマンスの異常を検出する機械学習ソリューションが役立つかどうかを確認します。
  - 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レコメンデーションを提供します。
- セキュリティと運用の目標に合わせて、モニタリングおよびログ記録ソリューションのデータ保持を設定します。
  - CloudWatch メトリクスのデフォルトのデータ保持
  - CloudWatch ログのデフォルトのデータ保持

リソース

関連ドキュメント:

- AWS データベースのキャッシュ
- Amazon Athena パフォーマンスに関するヒントのトップ 10
- Amazon Aurora のベストプラクティス
- DynamoDB Accelerator
- Amazon DynamoDB のベストプラクティス
- Amazon Redshift Spectrum のベストプラクティス
- Amazon Redshift のパフォーマンス
- AWS でのクラウドデータベース
- Amazon RDS Performance Insights

関連動画:

- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Database Performance Monitoring and Tuning with Amazon DevOps Guru for Amazon RDS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- Best Practices for Monitoring Redis Workloads on Amazon ElastiCache

関連する例:

- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS モニタリングワークショップ
- AWS 目的別データベースワークショップ

PERF03-BP04 データストアのクエリパフォーマンスを向上させるための戦略を実装する

データを最適化し、データクエリを改善する戦略を実装して、ワークロードのスケーラビリティとパフォーマンスを向上させます。

一般的なアンチパターン:

- データストア内のデータをパーティション化しない。
- データストアへのデータの格納に 1 つのファイル形式のみを使用する。
- データストアでインデックスを使用しない。

このベストプラクティスを活用するメリット: データとクエリのパフォーマンスを最適化することで、効率性の向上、コストの削減、ユーザーエクスペリエンスの改善につながります。

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

データ最適化とクエリチューニングはデータストアのパフォーマンス効率の重要な側面であり、クラウドワークロード全体のパフォーマンスと応答性に影響を与えます。クエリが最適化されていないと、リソースの使用量が増え、ボトルネックが発生し、データストアの全体的な効率が低下する可能性があります。

データ最適化には、効率的なデータストレージとアクセスを確保する手法がいくつかあります。これは、データストアでのクエリパフォーマンスの向上にも役立ちます。主な戦略には、データのパーティション化、データ圧縮、データ非正規化などがあり、ストレージとアクセスの両方でデータを最適化するのに役立ちます。

実装手順

- データストアで実行される重要なデータクエリを把握して分析します。
- データストア内で処理速度の遅いクエリを特定し、クエリプランを使用して現在の状態を把握します。

  - Amazon Redshift でのクエリプランの分析
  - Athena での EXPLAIN および EXPLAIN ANALYZE の使用
- クエリのパフォーマンスを向上させるための戦略を実装します。主な戦略には次のものがあります。
  - 列ファイル形式 (Parquet や ORC など) を使用します。
  - データストア内のデータを圧縮して、ストレージ容量と I/O 操作を削減する。
  - データのパーティション化によりデータを細かく分割し、データスキャン時間を短縮する。
    - Athena でのデータのパーティション化
    - パーティションとデータの分散
  - クエリでよく使用される列にデータインデックスを作成する。
  - 頻繁に実行するクエリにはマテリアライズドビューを使用する。
    - マテリアライズドビューを理解する
    - Amazon Redshift でのマテリアライズドビューの作成
  - クエリに適した結合操作を選択する。2 つのテーブルを結合する場合、結合の左側に大きい方のテーブルを指定し、結合の右側に小さい方のテーブルを指定します。
  - 分散キャッシュソリューションでレイテンシーを改善し、データベースの I/O 操作の数を減らす。
  - バキューム処理、インデックスの再作成、統計の実行などの定期的なメンテナンス。
- 非運用環境で実験し、戦略をテストする。

リソース

関連ドキュメント:

- Amazon Aurora のベストプラクティス
- Amazon Redshift のパフォーマンス
- Amazon Athena パフォーマンスに関するヒントのトップ 10
- AWS データベースのキャッシュ
- Amazon ElastiCache 実装のベストプラクティス
- Athena でのデータのパーティション化

関連動画:

- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Optimize Amazon Athena Queries with New Query Analysis Tools

関連する例:

- Amazon S3 Select - Querying data without servers or databases
- AWS 目的別データベースワークショップ

PERF03-BP05 キャッシュを利用するデータアクセスパターンを実装する

頻繁にアクセスされるデータを高速に取得できるようにデータをキャッシュする利点が得られるアクセスパターンを実装します。

一般的なアンチパターン:

- 頻繁に変更されるデータをキャッシュする。
- あたかも永続的に保存され、常に利用できるかのように、キャッシュされたデータに依存する。
- キャッシュされたデータの一貫性が考慮されない。
- キャッシュ実装の効率をモニタリングしない。

このベストプラクティスを活用するメリット: データをキャッシュに保存すると、読み取りレイテンシー、読み取りスループット、ユーザーエクスペリエンス、全体的な効率が向上し、コストも削減されます。

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: 中

実装のガイダンス

キャッシュとは、同じデータに対する今後のリクエストの処理を高速化したり効率性を向上したりするために、データを保存することを目的としたソフトウェアまたはハードウェアコンポーネントです。キャッシュに保存されたデータは、失われた場合でも、前の計算を繰り返すか、別のデータストアから取得することで再構築できます。

データキャッシュは、アプリケーション全体のパフォーマンスを向上させ、基盤となるプライマリデータソースの負担を軽減するうえで、最も効果的な戦略の 1 つです。データは、クライアント側

のキャッシュと呼ばれるリモート呼び出しを行うアプリケーションや、リモートキャッシュと呼ばれる高速セカンダリサービスを使用して、アプリケーションの複数のレベルでキャッシュできます。

クライアント側のキャッシュ

クライアント側のキャッシュを使用すると、各クライアント (バックエンドデータストアにクエリを実行するアプリケーションまたはサービス) は、独自のクエリの結果を指定された期間、ローカルに保存できます。これにより、最初にローカルのクライアントキャッシュを確認することで、ネットワーク経由でデータストアに送信されるリクエストの数を低減できます。結果がキャッシュに存在しない場合、アプリケーションはデータストアにクエリを実行し、その結果をローカルに保存できます。このパターンにより、各クライアントは可能な限り最も近い場所 (クライアント自体) にデータを保存できるため、レイテンシーを最小限に抑えることができます。また、バックエンドデータストアが使用できない場合でも、クライアントは引き続きクエリの一部を処理できるため、システム全体の可用性が向上します。

この方法の欠点の 1 つは、複数のクライアントが関係する場合、同じキャッシュデータをローカルに保存する可能性があることです。その結果、このようなクライアント間でストレージが重複して使用されることになり、データの不整合が発生します。あるクライアントがクエリの結果をキャッシュし、1 分後に別のクライアントが同じクエリを実行して別の結果を取得する場合もあります。

リモートキャッシュ

クライアント間でデータが重複する問題を解決するには、高速外部サービスまたはリモートキャッシュを使用して、クエリされたデータを保存します。ローカルデータストアをチェックする代わりに、各クライアントはバックエンドデータストアへのクエリを実行する前にリモートキャッシュをチェックします。この戦略により、クライアント間の応答の一貫性が強化され、保存されたデータの効率が向上し、ストレージスペースがクライアントとは別個にスケールされるため、キャッシュされたデータの量が増大します。

リモートキャッシュの欠点は、リモートキャッシュをチェックするために追加のネットワークホップが必要になるため、システム全体のレイテンシーが増大する可能性がある点です。クライアント側のキャッシュをリモートキャッシュと併用してマルチレベルキャッシュを行うことで、レイテンシーを短縮できます。

実装手順

- キャッシュの利点を活用できるデータベース、API、ネットワークサービスを特定します。読み取りワークロードが高いサービス、読み取りと書き込み率が高いサービス、またはスケールするのにコストがかかるサービスなどが、キャッシュの候補となります。
  - データベースのキャッシュ

  - API キャッシュを有効にして応答性を強化する
- アクセスパターンに最適なキャッシュ戦略の種類を特定します。
  - キャッシュ戦略
  - AWS キャッシュソリューション
- データストアのキャッシュのベストプラクティスに従います。
- すべてのデータに対して有効期間 (TTL) などのキャッシュ無効化戦略を設定し、データの鮮度とバックエンドデータストアへの負荷の軽減の間でバランスをとります。
- 自動接続再試行、エクスポネンシャルバックオフ、クライアント側のタイムアウト、接続プーリングなどの機能が利用できる場合は、クライアントで有効にします。これにより、パフォーマンスと信頼性が向上します。
  - ベストプラクティス: Redis クライアントと Amazon ElastiCache (Redis OSS)
- 80% 以上を目標にキャッシュヒットレートをモニタリングします。値が低い場合は、キャッシュサイズが不十分であるか、キャッシュの利点を活用できないアクセスパターンである可能性があります。
  - モニタリングすべきメトリクス
  - Amazon ElastiCache での Redis ワークロードのモニタリングのベストプラクティス
  - Amazon CloudWatch を使用して Amazon ElastiCache (Redis OSS) でベストプラクティスをモニタリングする
- データ複製を実装して読み取りを複数のインスタンスにオフロードし、データ読み取りのパフォーマンスと可用性を向上させます。

リソース

関連ドキュメント:

- Amazon ElastiCache Well-Architected レンズの使用
- Amazon CloudWatch を使用して Amazon ElastiCache (Redis OSS) でベストプラクティスをモニタリングする
- モニタリングすべきメトリクス
- Amazon ElastiCache ホワイトペーパーによるスケールに応じたパフォーマンス
- キャッシングの課題と戦略

関連動画:

- Amazon ElastiCache ラーニングパス
- Amazon ElastiCache ベストプラクティスによる成功のための設計
- AWS re:Invent 2020 - Amazon ElastiCache ベストプラクティスによる成功のための設計
- AWS re:Invent 2023 - [LAUNCH] Introducing Amazon ElastiCache Serverless
- AWS re:Invent 2022 - 5 great ways to reimagine your data layer with Redis
- AWS re:Invent 2021 - Deep dive on Amazon ElastiCache (Redis OSS)

関連する例:

- Amazon ElastiCache (Redis OSS) で MySQL データベースのパフォーマンスを強化する

ネットワークとコンテンツ配信

Questions

- PERF 4. ワークロード内のネットワークリソースはどのように選択して構成すればよいでしょうか?

PERF 4. ワークロード内のネットワークリソースはどのように選択して構成すればよいでしょうか?

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

- PERF04-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
- PERF04-BP02 使用可能なネットワーク機能を評価する
- PERF04-BP03 ワークロードに適した専用接続または VPN を選択する
- PERF04-BP04 ロードバランシングを使用してトラフィックを複数のリソースに分散する
- PERF04-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
- PERF04-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
- PERF04-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

PERF04-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に与える影響を分析して理解し、効率的なパフォーマンスとユーザーエクスペリエンスの向上を実現します。

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。
- クラウドネイティブなネットワークセキュリティツールを使用せずに、すべてのトラフィックを中央のファイアウォール経由でルーティングする。
- 実際の使用要件を理解せずに AWS Direct Connect 接続をプロビジョニングする。
- 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: ネットワーキング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メンテナンスを軽減できます。

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

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る可能性があります。

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。

実装手順:

- 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、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。

- VPC、AWS Direct Connect、Elastic Load Balancing (ELB)、Amazon Route 53 などの主要な AWS ネットワーキングサービスについて理解します。
- 次の主要なネットワーク特性を把握します。

特性	ツールとメトリクス
基礎的なネットワーク特性	• VPC フローログ • AWS Transit Gateway フローログ • AWS Transit Gateway のメトリクス • AWS PrivateLink のメトリクス
アプリケーションネットワークの特性	• Elastic Fabric Adapter • AWS App Mesh のメトリクス • Amazon API Gateway のメトリクス
エッジネットワークの特性	• Amazon CloudFront のメトリクス • Amazon Route 53 のメトリクス • AWS Global Accelerator のメトリクス
ハイブリッドネットワークの特性	• AWS Direct Connect のメトリクス • AWS Site-to-Site VPN のメトリクス • AWS Client VPN のメトリクス • AWS クラウド WAN のメトリクス
セキュリティネットワークの特性	• AWS Shield、AWS WAF、AWS Network Firewall のメトリクス
トレーシングの特性	• AWS X-Ray • VPC Reachability Analyzer • Network Access Analyzer • Amazon Inspector • Amazon CloudWatch RUM

- 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。

- インスタンスが同じ VPC 内にある場合、一部の要因が Amazon EC2 ネットワークのパフォーマンスに影響を与える可能性があるため、ネットワークスループットをベンチマークします。同じ VPC 内で Amazon 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。
- 負荷テストを実行し、ネットワーキングソリューションとオプションを試します。

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 が提供するネットワーク製品
- 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

関連動画:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - What can networking do for your application?
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2023 - A developer’s guide to cloud networking
- AWS re:Invent 2019 - Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2019 - Optimizing Network Performance for Amazon EC2 Instances
- AWS Summit Online - Improve Global Network Performance for Applications
- AWS re:Invent 2020 - Networking best practices and tips with the Well-Architected Framework
- AWS re:Invent 2020 - AWS networking best practices in large-scale migrations

関連する例:

- 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 ネットワーキングワークショップ
- ハンズオン Network Firewall ワークショップ
- AWS でのネットワークの監視と診断
- AWS でのネットワーク設定ミスの特定と対処

PERF04-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、分析によって測定します。例えば、レイテンシー、ネットワーク距離、またはジッターを低減するために利用できるネットワークレベルの機能を活用します。

一般的なアンチパターン:

- 本社の物理的な所在地を理由に、1 つのリージョン内に留まる。
- セキュリティグループの代わりにファイアウォールを使用してトラフィックをフィルタリングする。
- セキュリティグループ、エンドポイントポリシー、その他のクラウドネイティブ機能に頼らず、トラフィック検査のために TLS を破る。
- セキュリティグループではなく、サブネットベースのセグメンテーションのみを使用する。

このベストプラクティスを活用するメリット: 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、ワークロードパフォ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顧客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

AWS は、ネットワークパフォーマンスの向上に役立つ AWS Global Accelerator および Amazon CloudFront などのサービスを提供します。また、ほとんどの AWS サービスには、ネットワークトラフィックを最適化するための機能 (Amazon S3 Transfer Acceleration 機能など) があります。

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。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し、測定されたパフォーマンスとユーザーエクスペリエンスの両方に与える影響をいかに理解できるかにかかっています。

実装手順

- ワークロードコンポーネントのリストを作成します。
  - AWS クラウド WAN を使用して組織のネットワークを構築、管理、監視することを検討します。
  - Amazon CloudWatch Logs メトリクスを使用して、グローバルおよびコアネットワークをモニタリングします。Amazon CloudWatch RUM を活用します。この製品は、ユーザーのデジタルエクスペリエンスを特定、理解、改善するインサイトを提供します。
  - AWS Network Manager を使用して、AWS リージョンとアベイラビリティーゾーン間、および各アベイラビリティーゾーン内のネットワークレイテンシーの合計を確認します。アプリケーションのパフォーマンスが基盤となる AWS ネットワークのパフォーマンスとどのように関連しているかを把握します。
  - 既存の設定管理データベース (CMDB) ツールまたは AWS Config などのサービスを使用して、ワークロードのインベントリとその設定方法を作成します。
- 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、ワークロードの帯域幅、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。
- これが新しいワークロードの場合は、負荷テストを実行してパフォーマンスのボトルネックを特定します。
- 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。次の主要なネットワークオプションと機能を確認してください:

改善の機会	ソリューション
ネットワークパスまたはルート	Network Access Analyzer を使用して、パスまたはルートを識別します。
ネットワークプロトコル	「PERF04-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」を参照してください。

改善の機会	ソリューション
ネットワークトポロジ	複数アカウントを接続する際の VPC ピアリングおよび AWS Transit Gateway の運用上およびパフォーマンス上のトレードオフを評価します。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、AWS Resource Access Manager を使用します。 「PERF04-BP03 ワークロードに適した専用接続または VPN を選択する」を参照してください。

改善の機会	ソリューション
ネットワークサービス	AWS Global Accelerator は、AWS 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% 向上するネットワークサービスです。 Amazon CloudFront は、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。 Lambda@edge の機能を使用して、CloudFront が提供するコンテンツをカスタマイズしてユーザーに近づけ、レイテンシーを削減し、パフォーマンスを向上させることができます。 Amazon Route 53 には、レイテンシーベースのルーティング、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オプションが用意されており、世界中の視聴者のワークロードのパフォーマンスを向上させるのに役立ちます。ワークロードがグローバルに分散している場合に、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できます。

改善の機会	ソリューション
ストレージリソース機能	Amazon S3 Transfer Acceleration は、外部ユーザーが Amazon S3 へのデータのアップロードに CloudFront のネットワーキング最適化を活かすことができる機能です。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。 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 に書き込んだりできます。

改善の機会	ソリューション
コンピューティングリソース機能	Amazon EC2 インスタンス、コンテナ、および Lambda 関数が使用する 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確認して、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には、CloudWatch Metrics および ethtool を使用します。`ethtool` コマンドは 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、CloudWatch にカスタムメトリックスとして公開できます。 Amazon Elastic Network Adapter (ENA) は、クラスタープレイスメントグループ内のインスタンスに対しより高いスループットを提供することで、さらなる最適化を実現します。 Elastic Fabric Adapter (EFA) は Amazon EC2 インスタンス向けのネットワークインターフェイスで、高レベルのノード間通信を必要とするワークロードを AWS で大規模に実行することを可能にします。 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 専用の追加容量を提供します。

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 が提供するネットワーク製品
- 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:

- AWS re:Invent 2023 – 新機能のご紹介 成長と柔軟性を考慮したネットワーク設計
- AWS re:Invent 2023 – 高度な VPC 設計と新機能
- AWS re:Invent 2023 – クラウドネットワーキング開発者ガイド
- AWS re:Invent 2022 – AWS ネットワーキングインフラストラクチャの詳細
- AWS re:Invent 2019 – AWS とハイブリッド AWS ネットワークインフラストラクチャへの接続
- AWS re:Invent 2018 – 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
- AWS Global Accelerator

関連する例:

- 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 ネットワーキングワークショップ
- ネットワークの監視と診断
- AWS でのネットワーク設定ミスの特定と対処

PERF04-BP03 ワークロードに適した専用接続または VPN を選択する

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ハイブリッド接続が必要な場合は、パフォーマンス要件を満たす十分な帯域幅をプロビジョニングします。ハイブリッドワークロードの帯域幅とレイテンシーの要件を見積もります。これらの値によってサイズ要件が決まります。

一般的なアンチパターン:

- ネットワークの暗号化要件に対して VPN ソリューションのみを評価する。
- バックアップ接続または冗長接続の選択肢を評価しない。
- ワークロードのすべての要件 (暗号化、プロトコル、帯域幅、トラフィックのニーズ) を特定しない。

このベストプラクティスを活用するメリット: 適切な接続ソリューションを選択して構成することで、ワークロードの信頼性を高め、パフォーマンスを最大化できます。ワークロード要件を特定して事前計画を行い、ハイブリッドソリューションを評価することで、時間対価値を高めながら、コストの高いネットワークの物理的変更と運用上の諸経費を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

帯域幅要件に基づいてハイブリッドネットワークアーキテクチャを開発します。AWS Direct Connect を使用すると、オンプレミスネットワークを AWS とプライベートに接続できます。安定したパフォーマンスを実現しながら、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ます。一時的な接続のみが必要な場合、コストが重要な場合、または AWS Direct Connect 使用時に耐障害性のある物理ネットワーク接続が確立されるのを待つ間、不測の事態に備えて使用されます。

帯域幅の要件が高い場合は、AWS Direct Connect または VPN サービスを複数使用することを検討します。トラフィックはサービス間で負荷分散できますが、レイテンシーと帯域幅の違いから、AWS Direct Connect と VPN 間でのロードバランシングはお勧めしません。

実装手順

- 既存のアプリケーションの帯域幅とレイテンシーの要件を見積もります。
  - AWS に移行する既存のワークロードについては、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。

  - モニタリングデータがない新規または既存のワークロードついては、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。
- 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 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。
  - AWS Direct Connect は、専用接続またはホスト接続を使用して、50 Mbps から 100 Gbps 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よ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を備えた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ルできます。
  - 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 のドキュメントに従って、適切な接続オプションを選択します。
  - AWS Direct Connect を使用する場合は、接続に適した帯域幅を選択してください。
  - 複数の口ケーションで AWS Site-to-Site VPN を使用して AWS リージョンに接続する場合、高速 Site-to-Site VPN 接続を使用してネットワークパフォーマンスを向上させます。
  - ネットワーク設計が AWS Direct Connect 経由の IPSec VPN 接続で構成されている場合、プライベート IP VPN を使用してセキュリティを向上させ、セグメンテーションを実現することを検討してください。AWSサイト間プライベート IP VPN は、トランジット仮想インターフェイス (VIF) 上にデプロイされます。
  - AWS Direct Connect SiteLink では、AWS リージョンをバイパスして AWS Direct Connect 口ケーション間で最も速い接続でデータを送信し、世界中のデータセンター間で低レイテンシーな冗長接続を実現します。
- 本番環境にデプロイする前に、接続設定を検証します。セキュリティとパフォーマンスのテストを実施して、帯域幅、信頼性、レイテンシー、コンプライアンスの要件を満たしていることを確認します。
- 接続のパフォーマンスと使用状況を定期的に監視し、必要に応じて最適化します。

決定論的なパフォーマンスについてのフローチャート

リソース

関連ドキュメント:

- AWS が提供するネットワーク製品
- AWS Transit Gateway
- VPC エンドポイント
- スケーラブルでセキュアなマルチ VPC の AWS ネットワークインフラストラクチャの構築
- クライアント VPN

関連動画:

- AWS re:Invent 2023 – AWS でのハイブリッドネットワーク接続の構築
- AWS re:Invent 2023 – AWS への安全なリモート接続
- AWS re:Invent 2022 – Amazon CloudFront でのパフォーマンス最適化
- AWS re:Invent 2019 – AWS とハイブリッド AWS ネットワークインフラストラクチャへの接続
- AWS re:Invent 2020 – AWS Transit Gateway Connect

関連する例:

- 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 ネットワーキングワークショップ

PERF04-BP04 ロードバランシングを使用してトラフィックを複数のリソースに分散する

トラフィックを複数のリソースやサービスに分散させ、ワークロードがクラウドの伸縮性を活用できるようにします。また、ロードバランシングを使用して暗号化ターミネーションをオフロードし、パフォーマンスと信頼性を向上させ、トラフィックを効率的に管理およびルーティングすることもできます。

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。
- ワークロードがロードバランサーなしで直接インターネットに公開されている。
- 既存のロードバランサーを介して、すべてのインターネットトラフィックをルーティングしている。
- 汎用 TCP ロードバランシングを使用して、各コンピューティングノードが SSL 暗号化を処理するようにしている。

このベストプラクティスを活用するメリット: ロードバランサーは、単一のアベイラビリティーゾーンまたは複数のアベイラビリティーゾーンにおけるアプリケーショントラフィックのさまざまな負荷を処理し、ワークロードの高可用性、自動スケーリング、効率的な使用を可能にします。

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

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て使用率を改善します。

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、または維持性など) 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。

AWS は、アプリケーションでロードバランシングを使用するためのモデルを複数提供しています。Application Load Balancer は、HTTP および HTTPS トラフィックのロードバランシングに最適で、マイクロサービスとコンテナを含めたモダンアプリケーションアーキテクチャの実現を目的とする高度なリクエストルーティングを提供します。

Network Load Balancer は、きわめて高いパフォーマンスが要求される TCP トラフィックのロードバランシングに最適です。超低レイテンシーを維持しながら 1 秒あたり数百万件ものリクエストを処理することができ、突発的、または変動しやすいトラフィックパターンを処理するために最適化されています。

Elastic Load Balancing は、統合された証明書管理と SSL/TLS 復号化を提供するため、ロードバランサーの SSL 設定を一元的に管理し、CPU に負荷のかかる SSL/TLS 処理をお客様のワークロードからオフロードすることができます。

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 HTTP/2 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、HTTP/2 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。

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があります。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。AWS Local Zones または AWS Outposts で Application Load Balancer を利用して、ワークロードをユーザーに近づけることも検討できます。

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許可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。

ロードバランサーと統合された自動スケーリングを使用します。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。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。

- 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 でのネットワークロードバランシング

実装手順

- トラフィック量、可用性、アプリケーションのスケーラビリティなど、ロードバランシングの要件を定義します。
- アプリケーションに適したロードバランサータイプを選択します。
  - HTTP/HTTPS ワークロードには Application Load Balancer を使用します。
  - 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。
  - 両方の製品の機能を活用する場合は、両方の組み合わせ (ALB を NLB のターゲット) を使用します。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 に公開する場合は利用します。
  - すべてのロードバランサーの比較については、「ELB 製品比較」を参照します。
- 可能であれば SSL/TLS オフロードを使用します。
  - AWS Certificate Manager と統合された Application Load Balancer と Network Load Balancer の両方を使用して HTTPS/TLS リスナーを設定します。
  - 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許可する必要があります。
  - 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。
- 適切なルーティングアルゴリズムを選択します (ALB のみ)。
  - 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 2 つのルーティングアルゴリズムオプションがあります。
  - 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。
  - 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。
- クロスゾーンまたはゾーン分離を検討します。
  - 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おり、ALB ではターゲットグループごとにオフにできます。
  - 可用性と柔軟性の向上のために、クロスゾーンをオンにします。クロスズーンは ALB ではデフォルトでオフになっており、NLB ではターゲットグループごとにオフにできます。

- HTTP ワークロードの HTTP キープアライブをオンにします (ALB のみ)。この機能を使用すると、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 での実現方法については、「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。
- ロードバランサーのモニタリングをオンにします。
  - Application Load Balancer および Network Load Balancer のアクセスログを有効にしてください。
  - ALB の場合に考慮すべき主なフィールドは `request_processing_time`、`request_processing_time`、および `response_processing_time` です。
  - NLB の場合に考慮すべき主なフィールドは `connection_time` および `tls_handshake_time` です。
  - 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 を使用して、ALB ログと NLB ログの両方をクエリできます。
  - `TargetResponseTime` for ALB などパフォーマンス関連のメトリクス用アラームを作成します。

リソース

関連ドキュメント:

- ELB 製品の比較
- AWS グローバルインフラストラクチャ
- アベイラビリティーゾーンのアフィニティを使用してパフォーマンス向上とコスト削減を実現
- Amazon Athena でのログ分析ステップバイステップガイド
- 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
- Elastic Load Balancing を使用して 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させる

関連動画:

- AWS re:Invent 2023: アプリケーションに対するネットワーキングの利点
- AWS re:Inforce 2022: Elastic Load Balancing を使用してセキュリティ体制を大規模に向上する方法
- AWS re:Invent 2018: Elastic Load Balancing: 詳細とベストプラクティス
- AWS re:Invent 2021: AWS ワークロードに最適なロードバランサーの選定方法
- AWS re:Invent 2019: さまざまなワークロードでの Elastic Load Balancing の活用方法

関連する例:

- Gateway Load Balancer
- Amazon Athena を使用したログ分析の CDK と AWS CloudFormation サンプル

PERF04-BP05 パフォーマンスを高めるネットワークプロトコルを選択する

ワークロードのパフォーマンスに対する影響に基づいて、システムとネットワーク間における通信のためのプロトコルを決定します。

スループットの達成には、レイテンシーと帯域幅間の関係が関与します。ファイル転送で Transmission Control Protocol (TCP) を使用している場合は、レイテンシーが高くなると全体的なスループットを低下させる可能性が高くなります。これを解決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もありますが、1 つの解決策として User Datagram Protocol (UDP) を使用する方法があります。

一般的なアンチパターン:

- パフォーマンス要件に関係なく、すべてのワークロードに TCP を使用する。

このベストプラクティスを活用する利点: 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、コネクションレス型 UDP では高速通信が可能ですが、再送信や高い信頼性は提供されません。TCP はフル機能のプロトコルですが、パケットの処理にはより大きなオーバーヘッドが必要です。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

アプリケーションに合わせて異なるプロトコルを選択する能力があり、この分野の専門知識を持っている場合は、異なるプロトコルを使用してアプリケーションとエンドユーザーエクスペリエンスを最適化してください。このアプローチは非常に難しいため、先に他の方法でアプリケーションを最適化したことがある場合にのみ試してください。

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、HTTP や HTTPS などの TCP ベースのプロトコルを使用して TCP ソケットを開き、アプリケーションコンポーネント間の通信を行っています。TCP はアプリケーションコンポーネント間のシンプルで信頼性の高い転送メカニズムであるため、メールやファイルのデータ転送などのアプリケーションでも一般的に TCP が利用されます。TCP で TLS を使用すると通信のオーバーヘッドが増加し、レイテンシーが増加して、スループットが低下する可能性がありますが、セキュリティ上のメリットがあります。このオーバーヘッドは主にハンドシェイ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、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 Transport Layer Security (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

- AWS Global Accelerator および AWS Transfer Family サービスを使用して、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を改善します。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ルおよび管理できます。
- 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。最初のバイトまでの時間 (TTFB) やラウンドトリップタイム (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。ワークロードがユーザーに動的コンテンツを提供する場合、Amazon CloudFront の使用を検討します。これにより、動的コンテンツの各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。
- 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。この場合、Elastic Load Balancing で SSL/TLS オフロードを行い、バックエンドインスタンスの代わりに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ことを検討します。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、キャパシティ増大につながります。
- Network Load Balancer (NLB) を使用して、認証と認可、ログ記録、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デプロイし、ワークロードのパフォーマンスと信頼性を向上させます。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。
- ハイパフォーマンスコンピューティング (HPC) ワークロードでは、Elastic Network Adapter (ENA) Express 機能を検討してください。SRD プロトコルを使用して、EC2 インスタンス間のネットワークトラフィックに対してより高いシングルフロー帯域幅 (25 Gbps) と低いテールレイテンシー (99.9 パーセンタイル) を提供し、ネットワークパフォーマンスを改善します。
- Application Load Balancer (ALB) を使用して、ワークロードコンポーネント間または gRPC クライアントとサービス間の gRPC (Remote Procedure Calls) トラフィックをルーティングし、ロード

バランスします。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

関連ドキュメント:

- Kubernetes への UDP トラフィックのルーティング方法
- 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 が提供するネットワーク製品
- 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

関連動画:

- AWS re:Invent 2022 – 次世代 Amazon Elastic Compute Cloud 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
- AWS re:Invent 2022 – アプリケーションネットワーキングの基礎

関連する例:

- 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 ネットワーキングワークショップ

PERF04-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。

一般的なアンチパターン:

- すべてのワークロードリソースを 1 つの地理的場所に統合する。
- ワークロードのエンドユーザーではなく、自分の所在地に最も近いリージョンを選んでいる。

このベストプラクティスを活用する利点: ユーザーエクスペリエンスは、ユーザーとアプリケーションの間のレイテンシーの影響を大きく受けます。適切な AWS リージョンと AWS のプライベートなグローバルネットワークを使用することで、レイテンシーを減らし、リモートユーザーにより良いエクスペリエンスを提供できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

Amazon EC2 インスタンスなどのリソースは、AWS リージョン、AWS ローカルゾーン、AWS Outposts または AWS Wavelength ゾーン内のアベイラビリティーゾーンに配置されます。この口ケーション選択が、特定のユーザーの口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 および AWS Global Accelerator などのエッジサービスを利用して、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を改善することもできます。

Amazon EC2 にはネットワーク用のプレイスメントグループが用意されています。プレイスメントグループは、レイテンシーを減らすためにインスタンスを論理的にグループ化したものです。サポートされているインスタンスタイプを使ったプレイスメントグループと Elastic Network Adapter (ENA) を使用することにより、ワークロードを低レイテンシー、低ジッターの 25 Gbps ネットワークに参加させることができます。プレイスメントグループは、低ネットワークレイテンシー、高ネットワークスループット、またはその両方からメリットを得るワークロードに推奨されます。

レイテンシーの影響を受けやすいサービスは、AWS グローバルネットワークを使用して Amazon CloudFront などのエッジロケーションで提供されます。これらのエッジロケーションは一般に、コンテンツ配信ネットワーク (CDN) およびドメインネームシステム (DNS) などのサービスを提供します。これらのサービスをエッジで使用することにより、ワークロードがコンテンツまたは DNS 解決のリクエストに低いレイテンシーで応答できるようになります。これらのサービスは、コンテンツのジオターゲティング (エンドユーザーの位置に基づいて異なるコンテンツを提供) などの地理的なサービス、またはエンドユーザーを最寄りのリージョンに誘導するレイテンシーベースルーティング (最小レイテンシー) も提供します。

エッジサービスを使用してレイテンシーを低減し、コンテンツキャッシングを有効化します。これらのアプローチから最大限のメリットを得るために、DNS と HTTP/HTTPS の両方でキャッシュ制御を正しく設定してください。

実装手順

- VPC のネットワークインターフェイスとの間で行き来する IP トラフィックに関する情報を把握します。
  - VPC フローログを使用した IP トラフィックのログ記録
  - AWS Global Accelerator でクライアント IP アドレスが保持される方法
- 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。
  - Amazon CloudWatch や AWS CloudTrail などのモニタリングツールを使用して、ネットワークアクティビティに関するデータを収集します。
  - 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。
  - データがある場所: 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。
  - ユーザーがいる場所: ユーザー向けアプリケーションの場合は、ワークロードのユーザーに近いリージョン (1 つまたは複数) を選択します。
  - その他の制約:「ワークロードのリージョンを選択する際の考慮事項」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。
- ビデオレンダリングなどのワークロードは AWS Local Zones を使用して実行します。Local Zones では、エンドユーザーの近くにコンピューティングリソースやストレージリソースがあるというメリットが得られます。
- AWS にデプロイしている他のワークロードとシームレスに連携しながら、オンプレミスに残す必要があるワークロードには AWS Outposts を使用します。
- 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。
- 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データに使用すると、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	どのようなときに使うか
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。

- 以下のように、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	どのようなときに使うか
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 Functions	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	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。

- 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、静的エニーキャスト IP アドレスおよび TCP ターミネーションをエッジロケーションで提供するネットワーキングサービスの恩恵を受けることができます。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% 向上させ、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できます。AWS Global Accelerator では、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 アドレスを利用できます。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。

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 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
- 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
- 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 および AWS Outposts などエッジ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 ローカルゾーン
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN
- Amazon Route 53

関連動画:

- AWS Local Zones 解説動画
- AWS Outposts: 概要と機能紹介
- AWS re:Invent 2023 - エッジワークロードとオンプレミスワークロードの移行戦略
- AWS re:Invent 2021 - AWS Outposts: AWS をオンプレミスに
- AWS re:Invent 2020 - AWS Wavelength: 5G エッジでアプリを超低レイテンシーに実行
- AWS re:Invent 2022 - AWS Local Zones: 分散エッジ用にアプリケーションを構築
- AWS re:Invent 2021 - Amazon CloudFront で低レイテンシーウェブサイトを構築
- AWS re:Invent 2022 - AWS Global Accelerator でパフォーマンスと可用性を向上
- AWS re:Invent 2022 - AWS で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 - Amazon Route 53 でグローバルなトラフィック管理を実現

関連する例:

- AWS Global Accelerator カスタムルーティングワークショップ
- エッジ関数でリライトとリダイレクトを処理

PERF04-BP07 メトリクスに基づいてネットワーク設定を最適化する

収集して分析したデータを使用して、ネットワーク設定の最適化に関する十分な知識に基づいた意思決定を行います。

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ことで、ネットワークパフォーマンスを把握してネットワーク設定を最適化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: 低

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 AWS ネットワーク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

- レイテンシーやパケットロスなど、収集する主要なパフォーマンスメトリクスを特定します。AWS には、これらのメトリクスの収集に役立ついくつかのツールが用意されています。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。

AWS ツール	使用する場所
Amazon VPC IP Address Manager	IPAM を使用して、AWS とオンプレミスのワークロードの IP アドレスを、プランニング、追跡、モニタリングします。これは IP アドレスの使用と割り当てを最適化するためのベストプラクティスです。
VPC フローログ	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。
AWS Transit Gateway フローログ	AWS Transit Gateway フローログを使用して、Transit Gateway で送受信される IP トラフィックに関する情報をキャプチャします。
DNS クエリのログ記録	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ログに記録します。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、もしくは DNS

AWS ツール	使用する場所
	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。
Reachability Analyzer	Reachability Analyzer は、ネットワークの到達可能性を解析したりデバッグしたりするときに使用します。Reachability Analyzer は、VPC 内のソースリソースと送信先リソースとの間の接続テストを実行するための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。
Network Access Analyzer	Network Access Analyzer を使用すると、リソースへのネットワークアクセスの状況を把握するのに役立ちます。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。

AWS ツール	使用する場所
Amazon CloudWatch	Amazon CloudWatch を使用してネットワークのオプションに適したメトリクスをオンにします。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 ゲートウェイ、AWS Transit Gateway、VPN トンネル、AWS Network Firewall、Elastic Load Balancing、AWS Direct Connect などのメトリクスをオンにすることができます。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。
AWS Network Manager	AWS Network Manager を使用すれば、AWS グローバルネットワークの過去および現在のパフォーマンスをモニタリングして、オペレーションやプランニングの目的に生かすことができます。Network Manager では、AWS リージョン間、アベイラビリティーゾーン間、および各アベイラビリティーゾーン内のネットワークレイテンシーを集計できるため、アプリケーションのパフォーマンスが、基盤となる AWS ネットワークのパフォーマンスとどのように関係しているのかをより詳細に把握することができます。
Amazon CloudWatch RUM	Amazon CloudWatch RUM を使用して、ユーザーエクスペリエンスの特定、把握、改善に役立つインサイトが得られるメトリクスを収集します。

- VPC AWS Transit Gateway とフローログを使用して、上位の送信元やアプリケーショントラフィックのパターンを特定します。

- VPC、サブネット、ルーティングなど、現在のネットワークアーキテクチャを評価して最適化します。例として、異なる VPC ピアリングや AWS Transit Gateway がアーキテクチャ内のネットワークの改善にどのように役立つかを評価できます。
- ネットワーク内のルーティングパスを評価して、宛先間の最短パスが常に使用されていることを確認します。その際、Network Access Analyzer が役立ちます。

リソース

関連ドキュメント:

- パブリック DNS クエリログ記録
- IPAM とは
- Reachability Analyzer とは
- Network Access Analyzer とは
- VPC の CloudWatch メトリクス
- Apache Parquet 形式の VPC フローログでパフォーマンスを最適化し、ネットワーク分析のコストを削減する
- Amazon CloudWatch メトリクスでグローバルネットワークおよびコアネットワークをモニタリングする
- ネットワークトラフィックとリソースの継続的モニタリング

関連動画:

- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2020 – Networking best practices and tips with the AWS Well-Architected Framework
- AWS re:Invent 2020 – Monitoring and troubleshooting network traffic

関連する例:

- AWS ネットワーキングワークショップ

- AWS Network Monitoring
- AWS でのネットワークの監視と診断
- Finding and addressing network misconfigurations on AWS

プロセスと文化

Questions

- PERF 5. 組織の慣行と文化は、ワークロードのパフォーマンス効率にどのように貢献していますか?

PERF 5. 組織の慣行と文化は、ワークロードのパフォーマンス効率にどのように貢献していますか?

ワークロードを設計する際には、効率的で高性能なクラウドワークロードをより良く実行するために採用できる原則と慣行があります。クラウドワークロードのパフォーマンス効率を高める文化を採用するには、以下の重要な原則と慣行を検討します。

ベストプラクティス

- PERF05-BP01 ワークロードの状態とパフォーマンスを測定するための主要業績評価指標 (KPI) を設定する
- PERF05-BP02 モニタリングソリューションを活用して、パフォーマンスが最も重要な分野について把握する
- PERF05-BP03 ワークロードのパフォーマンス向上プロセスを定める
- PERF05-BP04 ワークロードの負荷テストを実施する
- PERF05-BP05 自動化でパフォーマンス関連の問題をプロアクティブに修正する
- PERF05-BP06 ワークロードとサービスを最新の状態に保つ
- PERF05-BP07 メトリクスを定期的に見直す

PERF05-BP01 ワークロードの状態とパフォーマンスを測定するための主要業績評価指標 (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健全性とパフォーマンスを測定するのに役立ちます。

一般的なアンチパターン:

- ワークロードについて把握す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、こうした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。
- 定量的で測定可能な KPI を定義していない。
- 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活用するメリット: ワークロードの正常性とパフォーマンスを表す具体的な KPI を特定することで、チームの優先順位をすり合わせ、目指すべきビジネス成果とは何かを定義できます。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を活用しない場合のリスクレベル: 高

実装のガイダンス

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、こうした要因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。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、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。

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 は文書化され、ビジネス目標と戦略をサポートするしきい値を含み、かつモニタリング対象のメトリクスに対応付けられている必要があります。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。

実装手順

- ステークホルダーを特定する: 開発チームやオペレーションチームなど主要なビジネスステークホルダーを特定し、文書化します。

- 目標を決める: 特定したステークホルダーと一緒にワークロードの目標を決め、文書化します。スループット、応答時間、コストなど、ワークロードのパフォーマンス上重要となる側面に加え、ユーザー満足度などのビジネス目標も考慮に入れてください。
- 業界のベストプラクティスを確認する: 業界のベストプラクティスを確認して、ワークロードの目標に沿った関連 KPI を特定します。
- メトリクスの特定: パフォーマンス目標とビジネス目標の測定に役立つ、ワークロードの目標に沿ったメトリクスを特定します。これらのメトリクスに基づいて KPI を設定します。メトリクスの例として、平均応答時間や同時ユーザー数などの測定値があります。
- KPI を定義し記録する: 業界のベストプラクティスとワークロードの目標を使用して、ワークロード KPI のターゲットを設定します。この情報を使用して、重要度またはアラームレベルの KPI しきい値を設定します。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。
- モニタリングを実装する: Amazon CloudWatch や AWS Config などのモニタリングツールを使用して、メトリクスを収集し KPI を測定します。
- KPI を視覚的に伝える: Amazon QuickSight などのダッシュボードツールを使用して KPI を視覚化し、ステークホルダーに伝えます。
- 分析して最適化する: KPI を定期的に確認、分析し、改善が必要なワークロードの領域を特定します。利害関係者と協力してこれらの改善を実装します。
- 見直してブラッシュアップする: メトリクスと KPI は定期的に、特にビジネス目標やワークロードのパフォーマンスに変更があったときなどは見直しを行ってその有効性を評価します。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- AWS Partner DevOps コンピテンシーパートナー - モニタリング、ログ記録、およびパフォーマンス
- AWS オブザーバビリティツール
- 大規模なクラウド移行における重要業績評価指標 (KPI) の重要性
- How to track your cost optimization KPIs with the KPI Dashboard
- X-Ray ドキュメント
- Amazon CloudWatch ダッシュボードの使用
- Amazon QuickSight の KPI

関連動画:

- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health
- AWS re:Invent 2023 - Performance & efficiency at Pinterest: Optimizing the latest instances
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Creating an Effective Metrics Strategy for Your Business | AWS Events

関連する例:

- Creating a dashboard with Amazon QuickSight

PERF05-BP02 モニタリングソリューションを活用して、パフォーマンスが最も重要な分野について把握する

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。

一般的なアンチパターン:

- パフォーマンスの問題を検出するには、CPU 使用率やメモリプレッシャーなどの標準的なコンピューティングメトリクスで十分であると考えている。
- 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を活用し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。

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。この領域を調査したら、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。

実装手順

- 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。以下は、AWS のモニタリングソリューションの一部です。

サービス	使用する場所
Amazon CloudWatch リアルユーザーモニタリング (RUM)	実際のユーザークライアントサイド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	アプリケーションレイヤー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、ワークロードコンポーネント間の関係性とレイテンシーを確認します。
Amazon Relational Database Service Performance Insights	データベースのパフォーマンスメトリクスを表示し、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	データベース OS のパフォーマンスメトリクスを表示します。

サービス	使用する場所
Amazon DevOps Guru	顧客に影響が及ぶ前に運用上の問題を特定できるように、異常な運用パターンを検出します。

- 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。テストの実行方法の例として、次のようなものがあります。
  - CloudWatch Synthetic Canaries をセットアップして、Linux の cron ジョブまたは rate 式を使用してブラウザベースのユーザーアクティビティをプログラムで模倣するように設定し、長期にわたって一貫したメトリクスを生成します。
  - AWS での分散負荷テストのソリューションを使用して、ピークトラフィックを生成するか、予想される増加率でワークロードをテストします。
- 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。
- 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。一例として、CloudWatch Evidently を使用することで、ワークロードへの新たな改善とパフォーマンスへの影響をテストできます。

リソース

関連ドキュメント:

- What's new in AWS Observability at re:Invent 2023
- The Amazon Builders' Library
- X-Ray ドキュメント
- Amazon CloudWatch RUM
- Amazon DevOps Guru

関連動画:

- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2023 - Building an effective observability strategy

- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 years of Amazon operational excellence
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Visual Monitoring of Applications with Amazon CloudWatch Synthetics

関連する例:

- Amazon CloudWatch Synthetics を使用してページのロード時間を測定する
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Python
- AWS での分散負荷テスト

PERF05-BP03 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。例えば、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。

一般的なアンチパターン:

- 現在のアーキテクチャが静的であり、今後更新されないと考えている。
- 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、ワークロード設計に経時的な影響を与えるために、収集されたデータを使用できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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。

ワークロードの重要なパフォーマンス上の制約を特定します。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。

実装手順

- KPI を特定する: 「PERF05-BP01 ワークロードの状態とパフォーマンスを測定するための主要業績評価指標 (KPI) を設定する」にあるように、ワークロードのパフォーマンス KPI を特定してワークロードのベースラインを定めます。
- モニタリングを実装する: AWS オブザーバビリティツールを使用してパフォーマンスメトリクスを収集し、KPI を測定します。
- 分析を行う: 「PERF05-BP02 モニタリングソリューションを活用して、パフォーマンスが最も重要な分野について把握する」にあるように、詳細な分析を行って、、ワークロード内でパフォーマンスが低い領域 (設定やアプリケーションコードなど) を特定します。分析ツールとパフォーマンスツールを使用して、パフォーマンス改善戦略を特定します。
- 改善を検証する: サンドボックス環境または本番前環境を使用して、戦略の有効性を検証します。
- 変更を実装する: 変更を本番環境に実装し、ワークロードのパフォーマンスを継続的にモニタリングします。改善点を文書化し、利害関係者に変更点を報告します。
- 見直してブラッシュアップする: パフォーマンス改善プロセスを定期的に見直し、さらに改善できる部分がないか見きわめます。

リソース

関連ドキュメント:

- AWSブログ
- AWS の最新情報
- AWS スキルビルダー

関連動画:

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance

関連する例:

- AWS GitHub

PERF05-BP04 ワークロードの負荷テストを実施する

ワークロードの負荷テストを実施して、本番環境の負荷に対応できることを確認し、パフォーマンスのボトルネックを特定します。

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。
- 負荷テストを、Amazon EC2 Testing Policy を実施したりシミュレーションイベント送信フォームを送信したりすることなく実行しています。これは、サービス妨害イベントとみなされ、テストの実行の失敗につながります。

このベストプラクティスを活用するメリット: 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を活用しない場合のリスクレベル: 低

実装のガイダンス

クラウドでの負荷テストは、予想されるユーザー負荷を考慮して、現実的な条件下でクラウドワークロードのパフォーマンスを測定するプロセスです。このプロセスでは、本番環境と同様のクラウド環境をプロビジョニングし、負荷テストツールを使用して負荷を生成したうえで、メトリクスを分析してワークロードが現実的な負荷に対応できるかを評価します。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。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。このプロセスにより、必要なパフォーマンスを継続的に達成できます。

実装手順

- テスト目標を決める: スループットや応答時間など、ワークロードで評価対象とするパフォーマンスの側面を特定します。
- テストツールを選択する: ワークロードに適した負荷テストツールを選択して設定します。
- 環境を設定する: 本番環境に基づいてテスト環境を設定します。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。
- モニタリングを実装する: Amazon CloudWatch などのモニタリングツールを使用して、アーキテクチャ内のリソース全体でメトリクスを収集します。カスタムメトリクスを収集して発行することもできます。
- シナリオを定義する: 負荷テストのシナリオとパラメータ (テスト期間やユーザー数など) を定義します。
- 負荷テストを実施する: テストシナリオを大規模に実施します。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。
- 結果を分析する: 結果を分析して、パフォーマンスのボトルネックおよび改善が必要な領域を特定します。
- 結果を文書化して共有する: 結果と推奨事項を文書化して報告します。この情報を利害関係者と共有して、パフォーマンスの最適化戦略に関して十分な情報に基づいた意思決定を行えるようにします。
- 継続的に実行する: 負荷テストは定期的に実行する必要があります。特に更新によりシステムが変更された後は必ず実行します。

リソース

関連ドキュメント:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

関連動画:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing

- AWS re:Invent 2022 - Scaling on AWS for your first 10 million users
- Solving with AWS Solutions: Distributed Load Testing

- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

関連する例:

- Distributed Load Testing on AWS

PERF05-BP05 自動化でパフォーマンス関連の問題をプロアクティブに修正する

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。

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

アラームを使用して、可能な場合には自動的に問題を修正するアクションを呼び出します。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。例えば、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。

実装手順

- 修正ワークフローを特定する: 自動的に修正できるパフォーマンスの問題を特定して把握します。Amazon CloudWatch や AWS X-Ray など、AWS のモニタリングソリューションを使用することで、問題の根本原因をよりよく理解できるようになります。
- オートメーションプロセスを定義する: 問題の自動修正に使用できるステップバイステップの修正計画とプロセスを作成します。
- 開始イベントを設定する: 修正プロセスを自動的に開始するようにイベントを設定します。例えば、CPU 使用率が特定のしきい値に達したときにインスタンスを自動的に再起動するトリガーを定義できます。
- 修正を自動化する: AWS のサービスとテクノロジーを使用して修正プロセスを自動化します。例えば、AWS Systems Manager Automation を使用すると、安全かつスケーラブルに修正プロセスを自動化できます。問題がうまく解決されない場合は、必ず自己修復ロジックを使用して変更を元に戻してください。
- ワークフローをテストする: 自動修正プロセスを本番前環境でテストします。
- ワークフローを実装する: 自動修正を本番環境に実装します。
- プレイブックを作成する: 開始イベント、修正ロジック、実行されたアクションなど、修正計画の手順を記したプレイブックを作成して文書化します。自動修正イベントに適切に対応できるように、必ず関係者へのトレーニングを行ってください。
- 見直してブラッシュアップする: 自動修正ワークフローの有効性を定期的に評価します。必要に応じて開始イベントと修正ロジックを調整します。

リソース

関連ドキュメント:

- CloudWatch ドキュメント
- AWS Partner Network DevOps コンピテンシーパートナー - モニタリング、ログ記録、およびパフォーマンス
- X-Ray ドキュメント
- CloudWatch でのアラームとアラームアクションの使用
- クラウドオートメーションプラクティスを構築して運用上の優秀性を実現する: AWS Managed Services 提供のベストプラクティス
- Automate your Amazon Redshift performance tuning with automatic table optimization

関連動画:

- AWS re:Invent 2023 - Strategies for automated scaling, remediation, and smart self-healing
- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2021 - Intelligently automating cloud operations
- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - Automating patch management and compliance using AWS
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS
- AWS re:Invent 2021 -{New Launch} Automatically detect and resolve issues with Amazon DevOps Guru
- AWS re:Invent 2023 - Centralize your operations

関連する例:

- CloudWatch Logs Customize Alarms

PERF05-BP06 ワークロードとサービスを最新の状態に保つ

新しいクラウドサービスと機能の最新情報を入手し、効率的な機能を取り入れ、問題を取り除き、ワークロードの全体的なパフォーマンス効率を向上させます。

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: 新しいサービスやオファリングに関する最新情報を入手するプロセスを確立することで、新しい機能を取り入れ、問題を解決し、ワークロードパフォーマンスを向上させることができます。

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

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、パフォーマンスを向上させる方法を検討します。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。

実装手順

- ワークロードをインベントリに登録する: 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。
- アップデートソースを特定する: 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。例えば、AWS の最新情報ブログをサブスクライブすれば、ワークロードのコンポーネントに合った製品を確認できます。RSS フィードをサブスクライブするかメールの購読を管理することで実行できます。
- 更新スケジュールを定義する: ワークロード用の新しいサービスと機能を評価するためのスケジュールを設定します。
  - AWS Systems Manager インベントリを使用すれば、Amazon EC2 インスタンスからオペレーティングシステム (OS)、アプリケーション、インスタンスのメタデータを収集し、どのインスタンスがソフトウェアポリシーで要求されるソフトウェアと設定を実行しているか、どのインスタンスがアップデートする必要があるかを迅速に把握することが可能です。
- 新しい更新を評価する: 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。クラウドの俊敏性を利用して、新しい機能によってワークロードがどのように改善するかをすばやくテストし、パフォーマンス効率を向上させます。
- 自動化を使用する: 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 を使用すると、AMI、コンテナイメージなど、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するとシステム更新のプロセスを自動化でき、AWS Systems Manager Maintenance Windows を使用するとアクティビティをスケジュールできます。
- プロセスを文書化する: 更新と新しいサービスとを評価するプロセスを文書化します。アップデートや新しいサービスを調査、テスト、実験、検証するために必要な時間と場所を所有者に提供しま

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。

リソース

関連ドキュメント:

- AWSブログ
- AWS の最新情報
- Implementing up-to-date images with automated EC2 Image Builder pipelines

関連動画:

- AWS re:Inforce 2022 - Automating patch management and compliance using AWS
- All Things Patch: AWS Systems Manager | AWS Events

関連する例:

- Inventory and Patch Management
- 1 つのオブザーバビリティワークショップ

PERF05-BP07 メトリクスを定期的に見直す

定期的なメンテナンスの一環として、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。この見直しを通じて、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で追跡すると問題の特定、対応、防止に役立つと思われるかを特定します。

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、問題について適切に識別、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。

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。

実装手順

- メトリクスを定義する: モニタリング対象となる主要なパフォーマンスメトリクス (応答時間やリソースの使用率などワークロード目標に沿ったもの) を定義します。
- ベースラインを設定する: 各メトリクスのベースラインと目標値を設定します。ベースラインの設定により、逸脱や異常を特定するための基準点が明確になります。
- 頻度を設定する: 重要なメトリクスをレビューする頻度 (毎週、毎月など) を設定します。
- パフォーマンス上の問題を特定する: 各レビューでは、傾向とベースライン値からの偏差を評価します。パフォーマンスのボトルネックや異常がないか調べます。特定された問題については、詳細な根本原因分析を実施して、問題の背後にある主な理由を把握します。
- 是正措置を特定する: 分析結果に基づいて是正措置を特定します。これには、パラメータの調整、バグの修正、リソースのスケーリングが含まれます。
- 結果を文書化する: 特定された問題、根本原因、是正措置など結果を文書化します。
- 反復して改善する: メトリクスのレビュープロセスを継続的に評価し改善します。前回のレビューで学んだ教訓を活かして、徐々にプロセスを強化します。

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メトリクス、ログ、トレースを収集する
- CloudWatch Metrics Insights を使用してメトリクスをクエリする
- AWS Partner Network DevOps コンピテンシーパートナー - モニタリング、ログ記録、およびパフォーマンス

- X-Ray ドキュメント

関連動画:

- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS

関連する例:

- Creating a dashboard with Amazon QuickSight
- CloudWatch Dashboards

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。コスト最適化の柱のホワイトペーパーでは、実装に関する規範的なガイダンスを確認できます。

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しますか?

COST 1. クラウド財務管理はどのように実装しますか?

クラウド財務管理を導入することで、組織はコストと使用量を最適化し、AWS でスケールしながら、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 コスト最適化の所有権を設定する
- 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 コスト最適化の所有権を設定する

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、Cloud Center of Excellence または FinOps チーム) を作成します。コスト最適化の所有者には、組織全体およびクラウド財務を理解している個人またはチーム (財務、テクノロジー、およびビジネスチームの人材が必要) を指定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

これは、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て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) 機能またはチームについての概要です。この機能は、組織内の個人でもチームでも構いません。組織全体から財務、テクノロジーなどの主な関係者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者と比較すると、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の開発など、複合的なアプローチが求められます。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。

- 集中型: FinOps チーム、クラウド財務管理 (CFM) チーム、クラウドビジネスオフィス (CBO)、Cloud Center of Excellence (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社内全体で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、コスト最適化を実行します。
- ハイブリッド: 集中型チームと分散型チームの両方が協力して、コスト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。

この担当者のためにエグゼクティブスポンサーシップを確保する必要があり、これが重要な成功要因になります。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、チームの考え方を上長にエスカレーションし、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にチームをサポートします。これを怠ると、ガイダンスが無視される可能性があり、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チームは、協力して組織のクラウド利用を効率化し、ビジネスバリューを実現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: コスト管理には、組織内のすべての関係部署が貢献し、関心を持つ必要があります。通常、一般的な組織内チームには、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、以下のスキルセットが必要です。
  - ソフトウェア開発: スクリプトと自動化を構築する場合。
  - インフラストラクチャエンジニアリング: スクリプトをデプロイし、プロセスを自動化して、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します。
  - 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、スケーリングすることで、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。

  価値ベースまたはコストベースの重要業績指標 (KPI) を定義できます。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要因に結び付け、AWS の費用の変化を合理化するうえで役立ちます。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) は定期的にミーティングを行い、目標とメトリクスを確認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の詳細を報告します。

  このような定期的なレビューにより、ワークロードの効率性 (コスト) とビジネス成果を確認できます。例えば、ワークロードのコストが 20% 増加した場合、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:

- Cloud 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、組織の目標やターゲット、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。

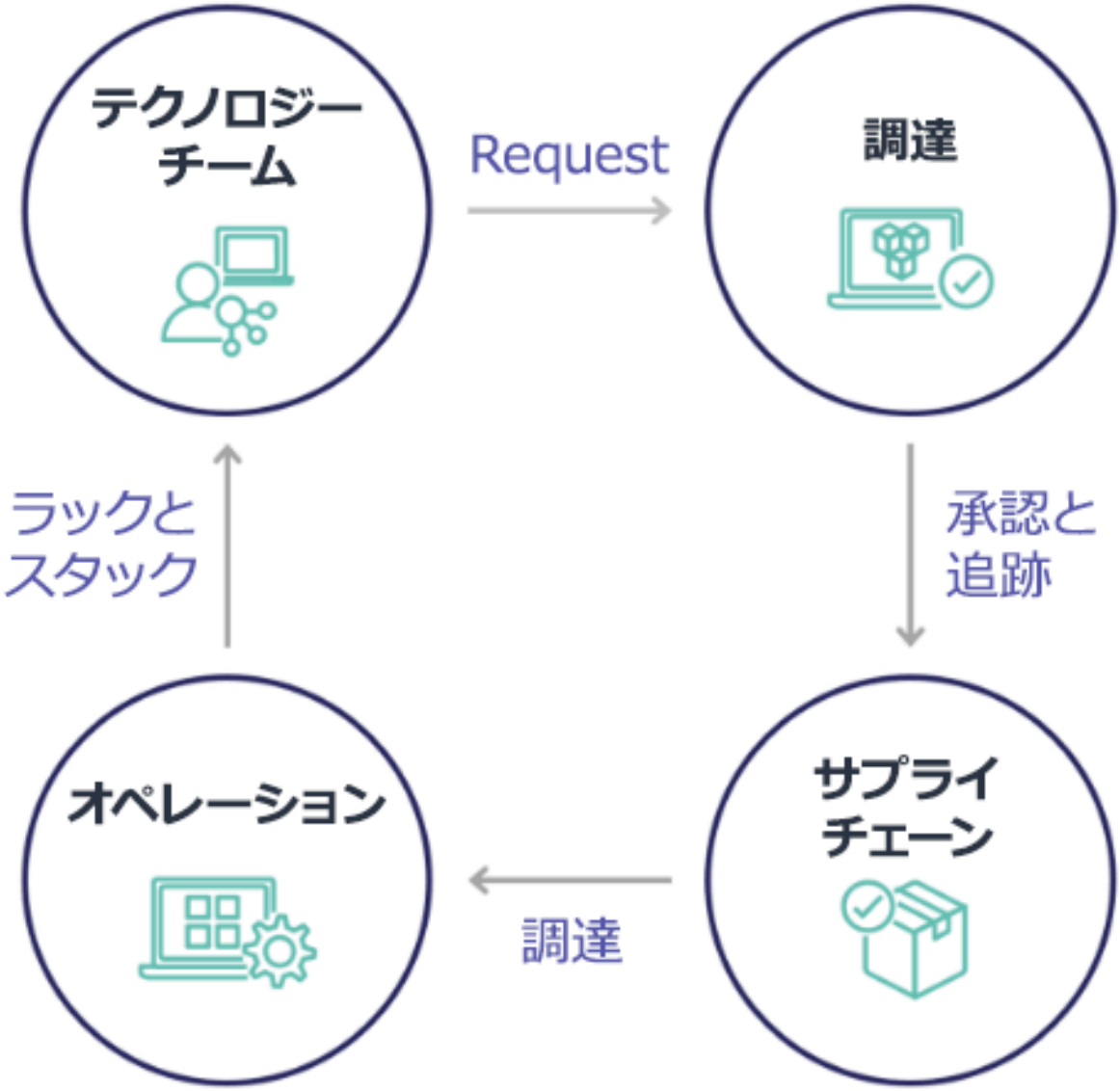
このベストプラクティスを活用しない場合のリスクレベル: 高

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化することができます。財務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幅に費やし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 1 つです。

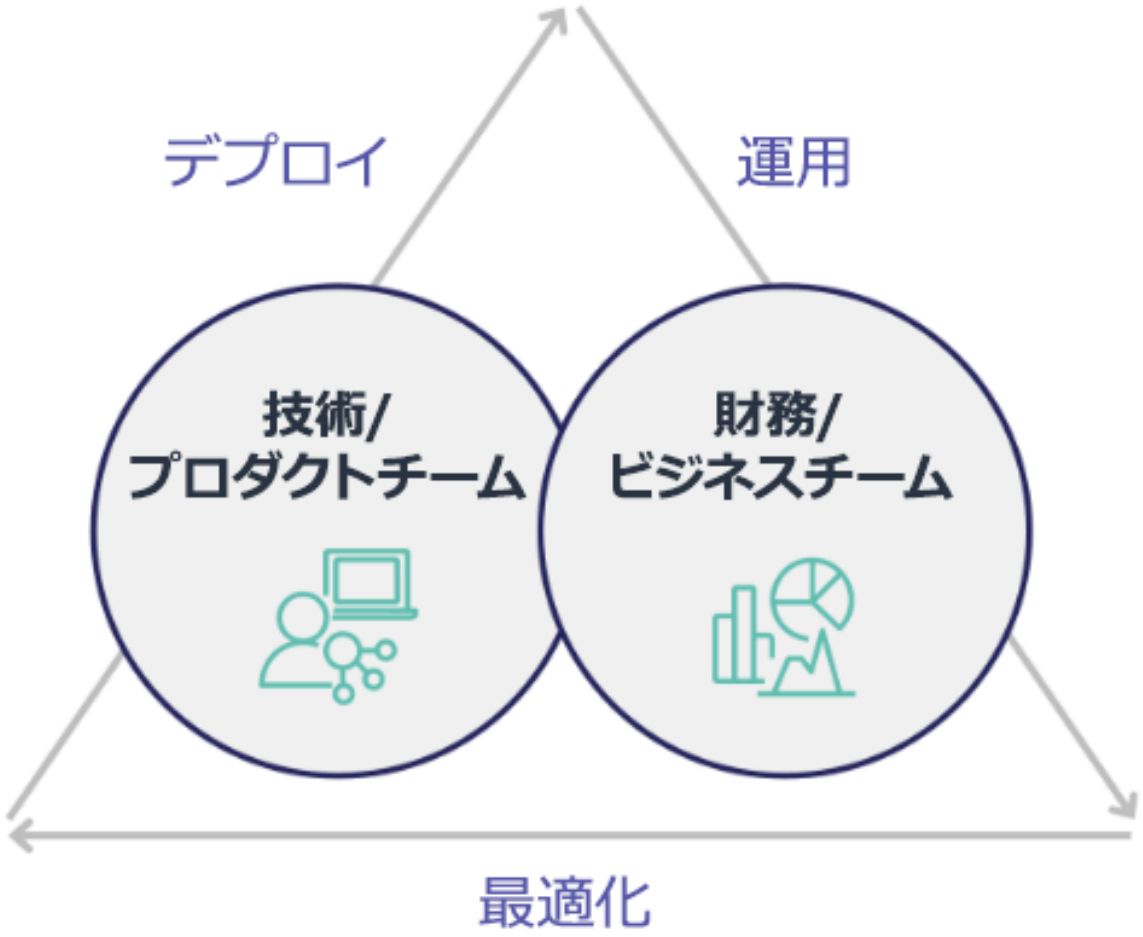
- 通常、エンジニアリングチームや IT チームは依頼主です。
- さまざまな財務チームが承認者、調達者として機能します。
- 運用チームは、すぐに使えるインフラストラクチャのラック、スタック、および引き渡しを行います。



クラウドの導入により、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活動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、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迅速に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どうし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、以下のとおりです。

- 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、テクノロジー関連の支出がビジネスの成果にどのように関連しているの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、テクノロジー関連の支出はコストではなく投資とみなされます。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、従量制料金やティア別料金、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、クラウドと共に使用されます。
- 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、組織内でかか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、製品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、動的なものに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

従来のオンプレミス IT 環境では、ピーク需要を満たすための新しい IT ハードウェアやサービスの購入など、まれにしか変化しない固定費を計画するという課題に直面することがあります。これとは対照的に、AWS クラウドでは、お客様が実際の IT ニーズやビジネスニーズに応じて、使用するリソースに対して料金を支払うという異なるアプローチを取っています。クラウド環境では、需要が月単位、日単位、さらには時間単位で変動する場合があります。

クラウドを使用すると効率性、スピード、俊敏性が向上し、その結果、コストと使用パターンの変動性も高まります。コストは、ワークロードの効率性の向上、または新しいワークロードや機能のデプロイによって減少することもあれば、場合によっては増加することもあります。拡大する顧客ベースに対応するためのワークロードのスケールに伴って、リソースのアクセス性が高まるため、クラウドの使用量とコストも相応に増加します。クラウドサービスにおけるこの柔軟性はコストと予測にも及び、一定の伸縮性がもたらされます。

こうしたビジネスニーズや需要ドライバーの変化に合わせて密に調整を行い、可能な限り正確な計画を目指すことが不可欠です。この変動に対応できるように、従来の組織の予算プロセスを適応させる必要があります。

新しいワークロードのコストを予測する際は、コストモデリングを検討します。コストモデリングによって、予想されるクラウドコストのベースラインを把握できるため、これを基に総保有コスト (TCO)、投資収益率 (ROI) およびその他の財務分析を実行し、関係者と共に目標と期待値を設定して、コスト最適化の機会を特定することができます。

組織は、コストの定義と承認されているグループ化を理解しておく必要があります。予測の詳細度の度合いは、組織の構造と社内ワークフローによって異なります。特定の要件と組織環境に適した詳細度を選択してください。どのレベルで予測を実行するかを理解しておくことが重要です。

- 管理アカウントまたは AWS Organizations レベル: 管理アカウントは、AWS Organizations の作成に使用するアカウントです。組織には、デフォルトで 1 つの管理アカウントがあります。
- 連結アカウントまたはメンバーアカウント: 組織のアカウントは標準の AWS アカウントで、AWS リソースと、それらのリソースにアクセスできる ID を含みます。

- 環境: 環境は、アプリケーションバージョンを実行する AWS リソースのコレクションです。環境は、複数の連結アカウントまたはメンバーアカウントで構成できます。
- プロジェクト: プロジェクトは、一定の期間内に達成すべき目標またはタスクの組み合わせです。予測する際は、プロジェクトのライフサイクルを考慮することが重要です。
- AWS サービス: 予測のために AWS サービスをグループ化できるコンピューティングサービスやストレージサービスなどのグループまたはカテゴリ。
- カスタムグループ化: ビジネスユニット、コストセンター、チーム、コスト配分タグ、コストカテゴリ、連結アカウント、これらの組み合わせなど、組織のニーズに基づいてカスタムグループを作成できます。

使用コストに影響を与える可能性のあるビジネスドライバーを特定し、それぞれについて個別に予測して、予想される使用量を事前に計算します。組織内の IT チームや製品チームに関連するドライバーもあります。マーケティングイベント、プロモーション、地理的拡大、合併、買収など、その他のビジネスドライバーについては、営業、マーケティング、ビジネス部門のリーダーが把握しているため、これらの関係者と協力して、これらすべての需要ドライバーについても考慮することが重要です。

AWS Cost Explorer を使用すると、過去の支出に基づいて、定義された将来の時間範囲におけるトレンドベースの予測を行うことができます。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を組み合わせて使用し、すべての料金タイプにわたる費用を個別に予測します。

予測プロセスを確立しモデルを構築した後、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。予算は通常 1 年単位で設定され、固定されるため、関係者全員の厳守が求められます。一方、予測はより柔軟であり、年間を通じて再調整が可能で、1 年、2 年、または 3 年の期間にわたる動的な予測が可能です。予算と予測はどちらも、テクノロジーやビジネスのさまざまなステークホルダーの間で財務上の期待事項を確立するうえで重要な役割を果たします。正確な予測と実装は、第一にコストのプロビジョニングに直接責任を負うステークホルダーに説明責任をもたらし、全体的なコスト意識を高めることにもつながります。

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、定期的に AWS Budgets レポートを作成して自分とステークホルダーに E メールで送信されるようにスケジュールします。また、実際のコストに基づいて AWS Budgets アラートを作成することもできます (これは反応型です)。または、予測コストに基づいて作成することも可能で、この場合は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。コストや使用量が一定レベルを実際に超えた場合、または予算額を超えると予測された場合にアラートを受け取ることができます。

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) と、動的で支出が変動する環境に最適な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、ワークロードの新しい環境など) を使用して、既存の予算編成と予測のプロセスをより動的なものになるよう調整します。Cost Explorer またはその他のツールを使用してトレンドベースの予測を決定したら、AWS Pricing Calculator を使用し、予想される使用量 (トラフィック、1 秒あたりのリクエスト数、または必要な Amazon EC2 インスタンス) に基づいて AWS ユースケースと将来のコストを見積もります。

予算はこうした予測計算と見積もりに基づいて設定する必要があるため、その予測の正確性を追跡します。統合されたクラウドコスト予測の正確性と有効性を監視します。実際の支出を予測と比較して定期的に見直し、必要に応じて調整して予測の精度を向上させます。予測の差異を追跡し、報告された差異について根本原因分析を実行して、措置を講じ、予測を調整します。

COST01-BP02 財務とテクノロジーの連携を確立する で説明しているように、IT、財務、その他の関係者が一貫性を保つために同じツールやプロセスを使用し、パートナーとなって連携することが重要です。予算を変更する必要がある場合は、ミーティングの回数を増やし、こうした変更により迅速に対応できるようにします。

実装手順

- 組織内のコスト言語を定義する: 複数のディメンションとグループ化を使用して、組織内に共通の AWS コスト言語を作成します。ステークホルダーが予測の詳細度、料金モデル、およびコスト予測のレベルを理解していることを確認します。
- トレンドベースの予測を分析する: AWS Cost Explorer や Amazon Forecast などのトレンドベースの予測ツールを使用します。サービス、アカウント、タグ、コストカテゴリなど、複数のディメンションで使用コストを分析します。高度な予測が必要な場合は、AWS のコストと使用状況 (CUR) データを Amazon Forecast にインポートします (これにより、機械学習の一形式として線形回帰が適用され、予測が行われます)。
- ドライバーベースの予測を分析する: ビジネスドライバーがクラウドの使用に与える影響を特定し、それぞれについて個別に予測して、予想される使用コストを事前に計算します。ビジネスユニットのオーナーやステークホルダーと緊密に連携して新しいドライバーへの影響を把握し、予想されるコストの変化を計算して正確な予算を決定します。
- 既存の予測および予算編成プロセスを更新する: トレンドベース、ビジネスドライバーベースなど、採用されている予測方法、または両方の予測方法の組み合わせに基づいて、予測および予算編成プロセスを定義します。予算は計算された現実的なもので、予測に基づいている必要があります。
- アラートと通知を設定する: AWS Budgets アラートとコスト異常検出を使用して、アラートと通知を取得します。

- 主要関係者と定期的なレビューを行う: 例えば、IT、財務、プラットフォームの各チーム、およびその他のビジネス分野の関係者と、ビジネスの方向性や使用状況の変化について調整します。

リソース

関連ドキュメント:

- AWS Cost Explorer
- AWS Cost and Usage Report
- Cost Explorer で予測する
- Amazon QuickSight 予測
- Amazon Forecast
- AWS Budgets

関連動画:

- 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?
- AWS コスト最適化シリーズ: AWS Budgets

関連する例:

- ドライバーベースの予測を理解し構築する
- 予測の文化を醸成する方法
- クラウドコスト予測の改善方法
- 適切なツールによるクラウドコスト予測

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。

- 変更による財務への影響を数値化するコスト測定が変更管理に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管理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を含め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。
- AWS Cost Anomaly Detection、AWS Budgets、AWS Budgets レポートのような無料の AWS ネイティブツールを利用します。

組織がクラウド財務管理 (CFM) プラクティスを一貫して採用すると、それらの行動は仕事や意思決定の方法に根付いていきます。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開発者から、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に至るまで、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特定します。リソースの作成や終了につながるすべてのプロセスをレビュー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意識の支援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。これにより、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。
- コストを意識したプロセスに更新する: 各プロセスをコストが意識されるよう変更します。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

関連ドキュメント:

- AWS 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
- 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

クラウド予算を設定して、使用量の異常を検出するメカニズムを設定します。関連ツールで、事前定義済み目標に対するコストと使用量に関連するアラートを設定し、使用量がそれらの目標を超えた場合に通知を受け取るようにします。定期的にミーティングを開催して、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。コストパフォーマンスについて話し合う専用セッションの運用や、ワークロードの定期的な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、コストパフォーマンスを定期的にモニタリングし、コスト節減機会を実現します。

AWS Cost Explorer を使用して、複数のフィルターと詳細度でコストと使用状況を確認できます。これにより、サービス別またはアカウント別のコスト、日次コスト、マーケットプレイスコストなどのダッシュボードとレポートが表示されます。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 レポートを使用します。

AWS Budgets を使用して、コストと使用状況を追跡し、しきい値を超えた場合に E メールまたは Amazon Simple Notification Service (Amazon SNS) 通知から受信したアラートにすばやく対応できるようにカスタム予算を設定します。優先予算期間を日次、月次、四半期、または年次に設定し、特定の予算制限を作成して、実際のコストまたは予測コストと使用状況が予算しきい値に対してどのように推移するかを常に把握します。アラートとそれらのアラートに対するアクションを設定し、自動的に実行することも、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もできます。

コストと使用状況に関する通知を実装して、想定外の場合にコストと使用状況の変化を迅速に処理できるようにします。AWS Cost Anomaly Detection を使用すると、イノベーションを遅らせることなく想定外のコストを減らし、制御を強化できます。AWS Cost Anomaly Detection は異常な支出と根本原因を特定し、想定外の請求のリスクを減らすために役立ちます。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。

Amazon QuickSight と AWS Cost and Usage Report (CUR) データを使用して、より詳細なデータによる高度にカスタマイズされたレポートを受け取ることもできます。Amazon QuickSight では、レポートのスケジュール設定や、過去のコストと使用状況、またはコスト削減の機会に関する定期的なコストレポート E メールの受信が可能です。Amazon QuickSight 上に構築された Cost Intelligence Dashboard (CID) ソリューションを確認して、高度な可視性を取得してください。

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検証するためのガイダンスが提供されます。

視覚的なグラフで、詳細なコストと使用量に関する Savings Plans のレコメンデーションを確認できます。時間単位のグラフには、オンデマンドの支出と推奨される Savings Plans のコミットメントが表示され、推定削減額、Savings Plans の適用範囲、Savings Plans 使用率に関するインサイトが得られます。これにより、組織は支出分析モデルの構築に時間とリソースを費やすことなく、毎時の支出にどのように Savings Plans が適用されているかを理解できます。

Savings Plans、リザーブドインスタンス、および 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削減するか、アカウントチームやパートナーにクラウドコストの最適化と節約の支援を依頼します。

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む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実現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: コストと使用に関するガバナンス
- コスト最適化について報告する: 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共に、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と所有者、財務、およびクラウド支出に関する主要な意思決定者の代表者が参加する必要があります。

リソース

関連ドキュメント:

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS Cost and Usage Report
- AWS Budgets のベストプラクティス
- Amazon S3 分析

関連する例:

- Well-Architected ラボ: コストと使用に関するガバナンス
- 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: 中

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するダッシュボードにより、主な担当者が必要な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。

AWS Cost Explorer または Amazon QuickSight など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するための、日次または頻繁なルーチンを作成します。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、予想どお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。AWS ソリューションアーキテクトによって構築された Amazon QuickSight ソリューションである Cost Intelligence Dashboard を使用して独自のレポートを作成し、予算を実際のコストや使用状況と比較することもできます。

実装手順

- コスト最適化について報告する: 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と所有者、財務、経営の代表者が参加する必要があります。

- コストと使用量の日次詳細度 AWS Budgets を作成してアクティブ化し、潜在的なコスト超過を防ぐためのアクションをタイムリーに実行する: AWS Budgets ではアラート通知を設定できるため、予算タイプが事前設定されたしきい値から外れても常に把握できます。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。それにより、予算を超えたものは使い過ぎとみなすことができます。
- コストモニタとして AWS Cost Anomaly Detection を作成する: AWS Cost Anomaly Detection は、高度な機械学習テクノロジーによっ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 を使用して、または AWS Cost and Usage Report (CUR) データを Amazon QuickSight ダッシュボードに統合して、組織のコストを可視化する: AWS Cost Explorer には、経時的に AWS コストと使用状況を確認し、理解して管理することを可能にする使いやすいインターフェイスがあります。カスタマイズ可能でアクセスしやすい Cost Intelligence Dashboard が、独自のコスト管理と最適化ツールの基盤作成を支援します。

リソース

関連ドキュメント:

- 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: 中

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情報については、「AWS Cost Management」、「AWS ニュースブログ」、「AWS コスト管理ブログ」、「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にサブスクライブします。通信設定ページで E メールアドレスを使用してサインアップできます。
- AWS ニュースをサブスクライブする: 新しいサービスと機能のリリース情報については、「AWS ニュースブログ」および「AWS の最新情報」を定期的に参照してください。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です。AWS は 2006 年以降、134 回値下げしています (2023 年 9 月 20 日現在)。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) インスタンスを含む過去の値下げの取り組みについては、「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します。

リソース

関連ドキュメント:

- 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、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: 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、料金モデルを意識することと併せ、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意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ことにあります (例えば、新しいワークロードを設計する場合や既存のワークロードを変更する場合、コストは機能要件でないとみなされます)。

この文化のわずかな変化が、現在および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、コスト面にどのような影響を与え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 (ワークロードあたりのコストや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対して要求された変更がコストに及ぼす影響を探求して、必要な分だけを支払うようにします。
- 変更計画者は、想定される変更のコストへの影響を意識し、高いコスト効率で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を意識するためのトレーニングを受けられるようにします。
- 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。
- ブログをサブスクライブする: AWS ブログページに移動し、最新情報ブログやその他の関連ブログにサブスクライブして、AWS が共有する新しいリリース、実装、例、変更をチェックします。

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 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により、将来のコスト最適化投資に対して関係者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成果を測定するためのフレームワークを取得できます。

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

ビジネス価値の数値化とは、企業の行動や決断がもたらす利益を測定することです。ビジネス価値は、有形 (経費の削減や利益の向上など) の場合もあれば、無形 (ブランドの評判や顧客満足度の向上など) の場合もあります。

コスト最適化によるビジネス価値の数値化とは、支出を効率化する取り組みがもたらす価値や利益を判断することです。例えば、ある企業が AWS へのワークロードのデプロイに 100,000 USD を費やし、後日最適化した結果、品質や成果を損なうことなく、わずか 80,000 USD にコストダウンしたとします。この場合、コスト最適化がもたらすビジネス価値を数値化すると、20,000 USD の節約になります。しかし、単純な節約額だけでなく、納期の短縮や顧客満足度の向上、コスト最適化の取り組みの成果が現れたその他の指標を踏まえて、価値を数値化することもできます。関係者は、コスト最適化の潜在的価値、ワークロードの最適化にかかるコスト、投資収益率を判断する必要があります。

コスト最適化によって節約した金額を報告することに加えて、実現した付加価値も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 を購入すると、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 インスタンスを削除した場合や、アタッチされていない Amazon Elastic Block Store (Amazon EBS) ボリュームを削除した場合は、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

- ビジネス上のメリットを評価する: これは、AWS クラウドコストを分析、調整して、支出から得られるメリットを最大化するプロセスです。ビジネス価値を顧みずコスト削減にばかり着目するのではなく、コスト最適化がもたらすビジネス上の利点と投資収益率を検討してください。支出した金額からもっと価値を引き出せるようになります。賢く支出し、最大の収益率を見込める分野に投資および出費することが大切です。
- 予測 AWS コストを分析する: 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

関連ドキュメント:

- AWS クラウドエコノミクス
- 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
- AWS Cost Explorer

関連動画:

- AWS で Windows を使用してビジネス価値を実現する

関連する例:

- CUSTOMER 360 のビジネス価値を測定し最大化する
- Amazon Web Services マネージドデータベースを採用することのビジネス価値
- 独立系ソフトウェアベンダーが Amazon Web Services を採用することのビジネス価値
- クラウドモダナイゼーションのビジネス価値
- Amazon Web Services に移行することのビジネス価値

経費支出と使用量の認識

Questions

- COST 2. どのように使用状況を管理するのですか?
- COST 3. コストと使用量はどのように監視すればよいでしょうか?
- COST 4. リソースはどのように廃止するのですか?

COST 2. どのように使用状況を管理するのですか?

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取り入れることによって、無駄なコストを費やすことなく革新することができます。

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、廃止を含む、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て、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で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。使用量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策定することです。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、廃止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、(チーム間で誤解が生じないように) 具体的である必要があります。さらに、お客様のビジネス状況や優先順位が変わると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ミディアムにする)、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、期待される成果について説明します。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。
- 範囲: 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ます。例えば、DevOps X Team が X 環境 (本番環境または非本番環境) で us-east 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) 本番環境で停止した Amazon EC2 インスタンスを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、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、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て、クラウド財務管理チームに通知します。
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します。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。

12AWS 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。

手順: 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ために、さまざまなサードパーティー製ツールや 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て、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めます。
- 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、同意したポリシーと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ます。
- 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してグループ化する: 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。類似する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: 特定済みの場所、リソース、およびユーザーを使用して、ワークロードのライフタイム (開発、運用、廃止) にわたって成果を得るために、それぞれが必要とするアクションを定義します。各場所で、グループ内の個々の要素ではなく、グループに基づいてアクションを特定します。開始時には読み取りまたは書き込みを幅広く設定し、それぞれのサービスについて、特定のアクションへと絞り込んでいきます。
- レビュー期間を定義する: ワークロードと組織の要件は時間の経過とともに変化する可能性があります。ワークロードのレビュースケジュールを定義して、組織の優先順位に合わせた状態を維持します。
- ポリシーを文書化する: 定義されたポリシーが、必要に応じて組織で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

関連ドキュメント:

- クラウドにおける変更管理
- ジョブ機能の AWS 管理ポリシー
- AWS 複数アカウントの請求戦略
- AWS のサービスのアクション、リソース、および条件キー
- AWS 管理とガバナンス
- IAM ポリシーを使用して AWS リージョンへのアクセスを制御する
- 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

関連する例:

- 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、期待される成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示します。

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。これらの目標や主要業績評価指標 (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。事業運営において重要な効率向上を測定できる測定可能で達成可能な目標を設定します。目標は、組織に期待される成果に関するガイダンスと方向性をもたらします。

ターゲットは、具体的かつ測定可能な達成すべき成果をもたらします。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。適切なメトリクスを使用し、組織に合わせて計算された KPI を設定してください。基本的な KPI から始めて、後でビジネスニーズに応じて発展させていくことができます。

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。すべてのワークロードにこの目標を設定するのと併せて、6 か月 ~ 1 年ごとに効率を 5% 高めるなどのターゲットを設定します。クラウドの場合、これは、コスト最適化能力の構築と、新しいサービスや機能のリリースによって達成できます。

ターゲットは、目標達成のために到達を目指す数値化可能なベンチマークであり、このベンチマークによって実際の結果をターゲットと比較します。コンピューティングサービス (スポット導入、Graviton 導入、最新のインスタンスタイプ、オンデマンドカバレッジなど)、ストレージサービス (EBS GP3 導入、古い EBS スナップショット、Amazon S3 Standard ストレージなど)、またはデータベースサービスの使用量 (RDS オープンソースエンジン、Graviton 導入、オンデマンドカバ

レッジなど) のユニットあたりのコストについて、KPI を使用してベンチマークを設定します。これらのベンチマークと KPI により、最も費用対効果の高い方法で AWS のサービスを利用していることを確認することができます。

次の表は、参照用の標準の AWS メトリクスの一覧です。これらの KPI の目標値は、組織によって異なります。

カテゴリ	KPI (%)	説明
コンピューティング	EC2 使用量カバレッジ	SP + RI + スポットを使用した EC2 インスタンス (コストまたは時間単位) と EC2 インスタンスの合計 (コストまたは時間単位) の比較
コンピューティング	コンピューティング SP/RI 使用率	利用可能な SP または RI 時間の合計に対する SP または RI の使用時間
コンピューティング	EC2/時間コスト	EC2 コストをその時間に実行されている EC2 インスタンスの数で割った値
コンピューティング	vCPU コスト	すべてのインスタンスの vCPU あたりのコスト
コンピューティング	最新のインスタンス生成	Graviton (またはその他の最新世代のインスタンスタイプ) のインスタンスの割合
データベース	RDS カバレッジ	RI を使用した RDS インスタンス (コストまたは時間単位) と RDS インスタンスの合計 (コストまたは時間単位) の比較
データベース	RDS 使用率	利用可能な RI 時間の合計に対する使用された RI 時間

カテゴリ	KPI (%)	説明
データベース	RDS の稼働時間	RDS コストをその時間に実行されている RDS インスタンスの数で割った値
データベース	最新のインスタンス生成	Graviton (またはその他の最新インスタンスタイプ) のインスタンスの割合
ストレージ	ストレージの使用率	最適化ストレージコスト (Glacier、ディープアーカイブ、低頻度アクセスなど) を合計ストレージコストで割ったもの
タグ付け	タグ付けされていないリソース	Cost Explorer: 1. クレジット、割引、税金、返金、マーケットプレイスを除外し、最新の月額コストをコピーします。 2. Cost Explorer で [タグ付けされていないリソースのみ表示] を選択します。 3. タグ付けされていないリソースの額を月額コストで割ります。

この表を使用して、組織の目標に基づいて算出した目標値またはベンチマーク値を含めます。正確かつ現実的な KPI を定義するには、ビジネスの特定のメトリクスを測定し、そのワークロードのビジネス成果を理解する必要があります。組織内のパフォーマンスメトリクスを評価する場合は、個別の目的を果たすさまざまな種類のメトリクスを識別します。これらのメトリクスは、ビジネス全体への影響を直接測定するものではなく、技術的なインフラストラクチャのパフォーマンスと効率を主に測定するものです。例えば、サーバーの応答時間、ネットワークレイテンシー、システムの稼働時間などを追跡します。これらのメトリクスは、インフラストラクチャが組織の技術面での運用をどの程度

サポートしているかを評価するうえで非常に重要です。ただし、顧客満足度、収益増加、市場シェアなど、より広範なビジネス目標に関する直接的なインサイトは得られません。ビジネスパフォーマンスを包括的に理解するには、ビジネスの成果と直接相関する戦略的なビジネスメトリクスを、これらの効率性メトリクスの補完として使用します。

KPI と関連するコスト削減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します。KPI 目標の定義と追跡を始める際は、クラウドインテリジェンスダッシュボード (CID) 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、コストと使用状況レポート (CUR) から取得したデータに基づいた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す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のステークホルダーによって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、およびより広範の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、季節的な要因による増加やマーケティングキャンペーンによって何が変化する可能性がある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すこと、データ転送を増やすこと、または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することが必要な場合があります。こうした主要ポイントごとにコストを特定し、使用量の変化に伴うコストの変化を予測します。
- 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、使用量とコスト、および使用量とコストの関係を考慮したものにする必要があります。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一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、目標とターゲットでの損失原因となりうるコストを具体的に定義したり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改善が目標である場合、ターゲットでは、改善の量 (通常は 1 USD あたりのビジネス成果) と、その改善をいつ達成すべきかを数値化します。例えば、過剰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とします。この場合、ターゲットとしては、実稼働ワー

クロードの最初の階層でコンピューティングの過剰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% 以下に抑える、さらに 2 つ目のターゲットとして、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過剰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、といったような内容が考えられます。

リソース

関連ドキュメント:

- ジョブ機能の AWS 管理ポリシー
- AWS 複数アカウントの請求戦略
- IAM ポリシーを使用して AWS リージョンへのアクセスを制御する
- S.M.A.R.T. 目標
- CID KPI ダッシュボードでコスト最適化 KPI を追跡する方法

関連動画:

- Well-Architected ラボ: 目標とターゲット (レベル 100)

関連する例:

- 単位メトリクスとは
- ビジネスをサポートする単位メトリクスの選択
- 実際の単位メトリクス: 得た教訓
- 単位メトリクスがビジネス機能間の調整にどのように役立つか
- Well-Architected ラボ: リソースを廃止する (目標とターゲット)
- 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を活用しない場合のリスクレベル: 高

実装のガイダンス

AWS Organizations では、ワークロードを AWS でスケール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、「管理アカウントに関する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、設計したアカウント構造に基づき、メンバーアカウントとして新しい 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。

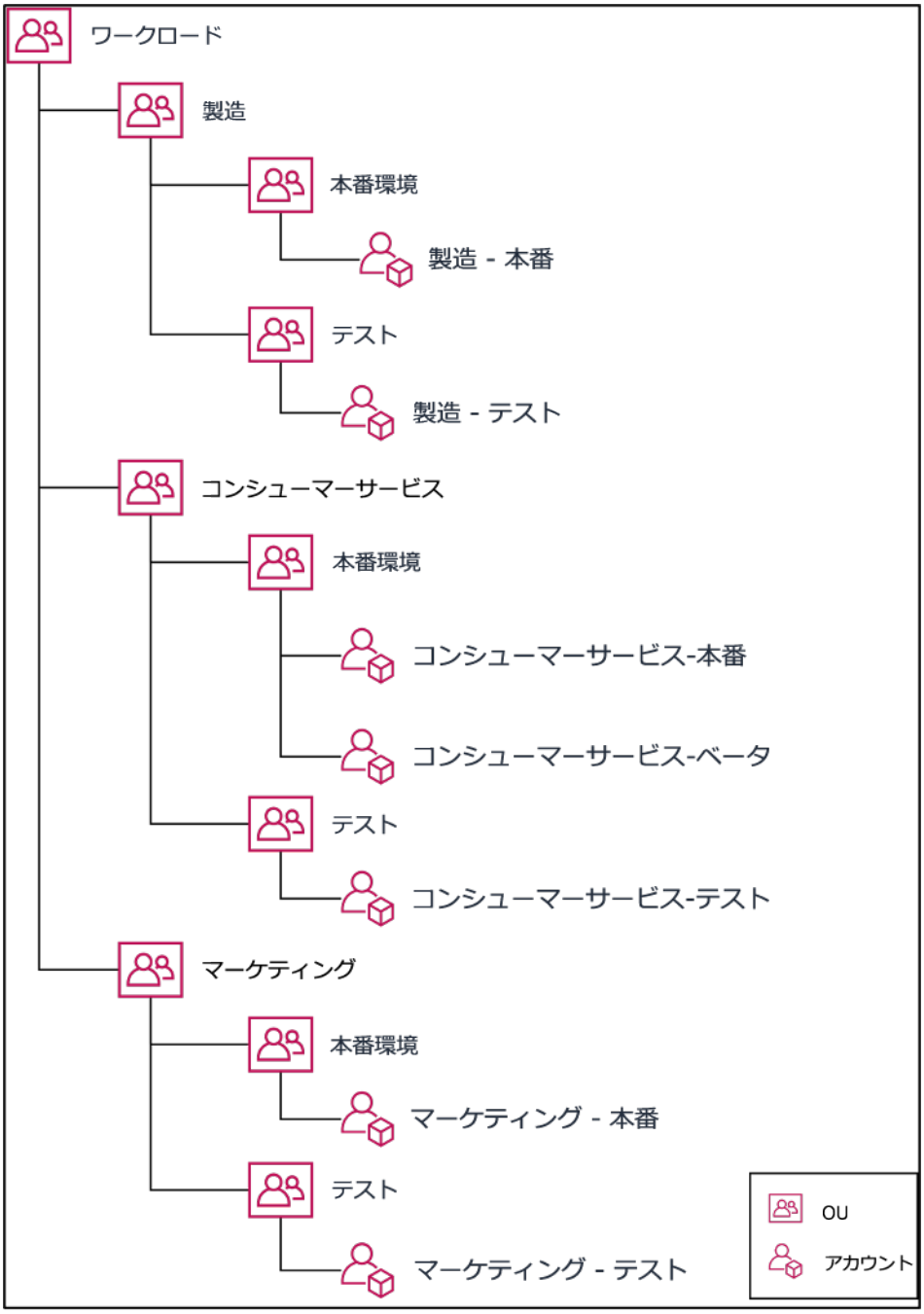
組織の規模や使用状況にかかわらず、少なくとも 1 つの管理アカウントとそれに紐づく 1 つのメンバーアカウントを常に持つことをお勧めします。すべてのワークロードリソースはメンバーアカウント内にのみ存在する必要があります。管理アカウントにはリソースを作成しないでください。AWS アカウントをいくつ持つべきかについて、一律の答えはありません。現在と将来の運用モデルとコストモデルを評価し、AWS アカウントの構造が組織の目標を反映するようにします。ビジネス上の理由から複数の AWS アカウントを作成する企業もあります。次に例を示します。

- 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分離し、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、一括請求を使用してこれらの連結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) で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。



組織単位の下に複数の AWS アカウントをグループ化する例。

AWS Control Tower では、複数の AWS アカウントのセットアップと構成をすばやく行い、ガバナンスが組織の要件に適合していることを確認できます。

実装手順

- 分離要件を定義する: 分離の要件は、セキュリティ、信頼性、財務構造など、複数の要因の組み合わせです。各要因を順番に確認し、ワークロードまたはワークロード環境を他のワークロードから

分離するかどうかを指定します。セキュリティは、アクセス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、実稼働ワークロードとテストワークロードを別々のアカウントで実行することや、組織内の個々の事業部門や部署、またはアカウントを所有する関係者に請求書と請求データを提供できるように別のアカウントを使用することなどが挙げられます。

- 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。例として、1 つ以上のワークロードから複数のテスト環境または開発環境をグループ化することが挙げられます。
- アカウント構造を定義する: これらの分離およびグループ化を使用して、各グループのアカウントを指定し、分離要件が維持されるようにします。これらのアカウントは、メンバーアカウントまたは連結アカウントです。これらのメンバーアカウントを単一の管理アカウントまたは支払者アカウントでグループ化することで使用量が合算されるため、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発行されます。請求データを分離し、各メンバーアカウントに請求データの個別のビューを表示することができます。メンバーアカウントが使用量や請求データを他のアカウントに表示してはならない場合、または AWS から別々の請求書を必要とする場合は、複数の管理アカウントまたは支払者アカウントを定義します。この場合、各メンバーアカウントは独自の管理アカウントまたは支払者アカウントを持つことになります。リソースは常にメンバーアカウントまたは連結アカウントに配置する必要があります。管理アカウントまたは支払者アカウントは、管理のためにのみ使用してください。

リソース

関連ドキュメント:

- コスト配分タグの使用
- ジョブ機能の AWS 管理ポリシー
- AWS 複数アカウントの請求戦略
- IAM ポリシーを使用して AWS リージョンへのアクセスを制御する
- AWS Control Tower
- AWS Organizations
- 管理アカウントとメンバーアカウントのベストプラクティス

- 複数のアカウントで AWS 環境を構成する
- 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:

- AWS Organizations のご紹介
- 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 環境を設定する

関連する例:

- Well-Architected ラボ: AWS 組織の作成 (レベル 100)
- AWS Cost and Usage Report の分割とアクセスの共有
- 通信会社の AWS マルチアカウント戦略の定義
- AWS アカウントを最適化するためのベストプラクティス
- AWS Organizations での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、廃止できるユーザーを管理します。例えば、開発、テスト、本番グループを実装します。これは、AWS のサービスやサードパーティーのソリューションに適用されます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

ユーザーのロールとグループは、安全で効率的なシステムを設計および実装するうえで基本となる構成要素です。ロールとグループは、組織が統制の必要性と柔軟性や生産性の要件とのバランスを図るうえで役立ち、最終的には組織の目標とユーザーのニーズの実現を助けます。AWS Well-Architected

フレームワークの「セキュリティの柱」の「ID とアクセス管理」セクションで推奨されているように、適切な条件下で適切なリソースへのアクセスを提供するために、堅牢な ID 管理とアクセス許可が必要です。ユーザーには、それぞれの業務を完遂するために必要なアクセス権のみを与えます。そうすることで、不正アクセスや誤用に伴うリスクが最小限に抑えられます。

ポリシーを作成した後で、組織内の論理グループとユーザーロールを作成できます。アクセス許可を割り当て、使用量を制御できるようになり、強固なアクセス制御メカニズムの実装を助け、機密情報への不正アクセスも防止できます。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、ビジネスアナリストなど) と合致します。グループによって、類似したタスクに従事し、類似したアクセス権を必要とするユーザーを分類します。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。個々のユーザー単位ではなく、グループやロール単位でアクセス許可を管理する方が簡単です。ロールとグループを通じてユーザー全員に一貫して体系的にアクセス許可を割り当てることで、ミスや不整合を防ぐことができます。

ユーザーのロールが変更された場合、管理者は個々のユーザーアカウントを設定し直さなくても、ロールまたはグループのレベルでアクセス権を調整できます。例えば、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、認証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「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、AWS Well-Architected フレームワークの「セキュリティの柱」を参照してください。

リソース

関連ドキュメント:

- ジョブ機能の AWS 管理ポリシー
- AWS 複数アカウントの請求戦略
- セキュリティの柱 - AWS Well-Architected Framework
- AWS Identity and Access Management (IAM)

- AWS Identity and Access Management ポリシー

関連動画:

- アイデンティティ管理とアクセス管理を使用する理由

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
- IAM ポリシーを使用して AWS リージョンへのアクセスを制御する
- クラウド財務管理ジャーニーの開始: クラウドコストオペレーション

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を活用しない場合のリスクレベル: 中

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制限を理解したら、ガバナンスを適用できます。AWS Budgets では、AWS コスト、使用量、コミットメント割引 (Savings Plans とリザーブドインスタンス) の通知を設定し、月額予算を定義できます。予算は、集計コストのレベル (例えば、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 で予算制限を設定したら、AWS Cost Anomaly Detection を使用して予期しない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モニタリングし、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 Identity and Access Management および AWS Organizations サービスコントロールポリシー (SCP) を使用して AWS にガバナンスポリシーを適用します。IAM により、AWS のサービスとリソースへのアクセスを安全に管理できます。IAM を使用すると、AWS のリソースを作成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、IAM ポリシーを割り当てて正しい使用法を適用します。SCP は、組織内のすべてのアカウントで利用可能な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。アクセス管理の実装の詳細については、Well-Architected のセキュリティの柱についてのホワイトペーパーを参照してください。

AWS Service Quotas を管理することで、ガバナンスを導入することもできます。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 (リソースの作成と廃止の両方) を理解し、クォータ変更をどれだけすばやく実装できるかを考慮する必要があります。Service Quotas を使用して、必要に応じてクォータを増加させることができます。

実装手順

- 支出に関する通知を実装する: 定義した組織のポリシーを使用して、AWS Budgets を作成し、支出がポリシーを外れた場合に通知を提供するようにします。アカウントごとに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、自動的に開始するか、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、組織内の予想外のコストを削減し、異常とみられる支出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

関連ドキュメント:

- ジョブ機能の AWS 管理ポリシー
- AWS 複数アカウントの請求戦略
- IAM ポリシーを使用して AWS リージョンへのアクセスを制御する
- AWS Budgets
- AWS Cost Anomaly Detection
- AWS コストを管理する

関連動画:

- 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:

- 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 予算アクション
- タグを使用して Amazon EC2 リソースへのアクセスを制御する IAM ポリシーを作成する方法を教えてください。
- 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ことはできますか?
- Amazon EC2 の使用をファミリー別に制限する IAM ポリシーを作成する

- Well-Architected ラボ: コストと使用に関するガバナンス (レベル 100)
- Well-Architected ラボ: コストと使用に関するガバナンス (レベル 200)
- AWS Chatbot を使用したコスト異常検出のための Slack 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

プロジェクトのライフサイクルを効果的に追跡することで、組織は計画、管理、リソースの最適化を改善し、コスト管理を強化できます。追跡で得られたインサイトは、意思決定に役立つ貴重な情報となり、コスト効率性やプロジェクト全体の成功に寄与します。

ワークロードのライフサイクル全体を追跡すれば、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時点でわかります。既存のワークロードとコンポーネントが使用中のように見える場合がありますが、AWS が新しいサービスや機能をリリースした時点で、廃止または刷新される可能性があります。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、以前の環境は廃止するか、再び必要になるまでキャパシティを大幅に削減することができます。

リソースに期間やリマインダーのタグを付けて、ワークロードがレビューされた時点をマークしておくことができます。例えば、開発環境の前回のレビューから数か月経っている場合は、再度レビューを行って、新しいサービスを導入できるか、環境が使用中かを調査する適切なタイミングである可能性があります。アプリケーションを AWS の myApplications でグループ化してタグ付けし、重要度、環境、最終レビュー、コストセンターなどのメタデータを管理および追跡できます。ワークロードのライフサイクルを追跡すると共に、アプリケーションのコスト、状態、セキュリティ体制、パフォーマンスをモニタリングおよび管理できます。

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さまざまな管理およびガバナンスサービスが用意されています。AWS Config または AWS Systems Manager を使用して、AWS リソースと設定の詳細なインベントリを入手できます。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が推奨されます。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。

アプリケーションライフサイクル管理 (ALM) と同様に、プロジェクトのライフサイクルを追跡するには、設計と開発、テスト、本番稼働、サポート、ワークロードの冗長性など、複数のプロセス、ツール、チームが連携する必要があります。

プロジェクトのライフサイクルの各段階を注意深く監視することで、組織は重要なインサイトを得て管理を強化し、プロジェクトを計画から実施、完遂に至るまで円滑に進めることができます。入念な監視下で、プロジェクトは品質基準を満たすだけでなく、納期どおりに予算内で完了し、全体的なコスト効率が向上します。

エンティティライフサイクル追跡の実装の詳細については、AWS Well-Architected 運用上の優秀性の柱についてのホワイトペーパーを参照してください。

実装手順

- プロジェクトのライフサイクルモニタリングプロセスを確立する: Cloud Center of Excellence チームは、プロジェクトのライフサイクルモニタリングプロセスを確立する必要があります。ワークロードを監視するための構造的かつ体系的なアプローチを確立し、プロジェクトの管理、可視性、パフォーマンスを高めます。監視プロセスの効果と価値を最大限に引き出すために、プロセスの透明性と協調性を高め、継続的に改善していきます。
- ワークロードレビューを実行する: 組織のポリシーに従い、定期的に既存のプロジェクトを監査し、ワークロードレビューを実施します。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廃止など、ワークロードの調整が必要です。

リソース

関連ドキュメント:

- AWS でのタグ付けのガイダンス
- ALM (アプリケーションライフサイクル管理) とは何ですか?
- ジョブ機能の AWS 管理ポリシー

関連する例:

- IAM ポリシーを使用して AWS リージョンへのアクセスを制御する

関連ツール

- AWS Config
- AWS Systems Manager
- AWS Budgets
- AWS Organizations
- AWS CloudFormation

COST 3. コストと使用量はどのように監視すればよいでしょうか?

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

コスト管理ツールとレポートツールを設定して、コストと使用状況に関するデータの分析と透明性を改善します。コストと使用量の追跡と区別を容易にするログエントリを作成するようにワークロードを設定します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

時間単位の粒度など、コスト管理ツールの詳細な請求情報により、組織は消費量をさらに詳細に追跡でき、コスト増加の原因を特定する手助けとなります。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。

AWS Data Exports を使用して AWS Cost and Usage Report (CUR) 2.0 のエクスポートを作成できます。これは、AWS から詳細なコストと使用状況の詳細を取得するための新しい推奨方法です。こ

れにより、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精度による使用量、レート、コスト、使用属性 (CUR と同じ情報) が提供されます。また、いくつかの改善点も提示されます。CUR では、タグ付け、場所、リソース属性、アカウント ID など想定可能なあらゆる側面からレポートを作成できます。

目的のエクスポートのタイプに応じて、標準データエクスポート、Amazon QuickSight 統合によるコストと使用状況ダッシュボードへのエクスポート、レガシーデータエクスポートの 3 種類のエクスポートタイプがあります。

- [標準データエクスポート]: Amazon S3 に定期的に配信されるテーブルのカスタマイズされたエクスポート。
- [コストと使用状況ダッシュボード]: Amazon QuickSight へのエクスポートと統合で、事前に構築されたコストと使用状況のダッシュボードを展開します。
- [レガシーデータのエクスポート]: レガシー AWS Cost and Usage Report (CUR) のエクスポートです。

次のカスタマイズを行ったデータエクスポートを作成できます。

- リソース ID の包含
- 分割コスト配分データ
- 時間単位の詳細
- バージョニング
- 圧縮タイプとファイル形式

Amazon ECS または Amazon EKS でコンテナを実行するワークロードの場合、分割コスト配分データを有効にすると、コンテナワークロードによる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、個々のビジネスユニットやチームにコンテナコストを配分できます。分割コスト配分データにより、新しいコンテナレベルのリソースのコストと使用状況データが AWS Cost and Usage Report に導入されます。分割コスト配分データは、クラスターで実行されている個々の ECS サービスとタスクのコストを計算することで算出されます。

コストと使用状況ダッシュボードは、コストと使用状況ダッシュボードテーブルを定期的に S3 バケットにエクスポートし、事前構築済みのコストと使用状況ダッシュボードを Amazon QuickSight にデプロイします。コストと使用状況データのダッシュボードをすぐにデプロイしたい場合は、このオプションを使用します (カスタマイズはできません)。

必要に応じて、レガシーモードで CUR をエクスポートできます。AWS Glue など他の処理サービスを統合して分析用にデータを準備して、SQL でデータをクエリして Amazon Athena でデータを分析したりできます。

実装手順

- データエクスポートを作成する: 必要なデータを使用してカスタマイズされたエクスポートを作成し、エクスポートのスキーマを制御します。基本的な SQL を使用して請求とコスト管理データのエクスポートを作成し、Amazon QuickSight と連携して請求とコスト管理データを可視化します。また、標準モードでデータをエクスポートして、Amazon Athena などの他の処理ツールでデータを分析することもできます。
- 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 で時間単位の詳細度を設定する: 過去 14 日間の時間単位の粒度でコストと使用状況データにアクセスするには、請求コンソールで時間単位とリソースレベルのデータを有効にすることを検討してください。
- アプリケーションログ記録を有効にする: 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についてのホワイトペーパーを参照してください。

リソース

関連ドキュメント:

- AWS Data Exports
- AWS Glue
- Amazon QuickSight
- AWS コスト管理の料金
- AWS リソースのタグ付け
- Cost Explorer によるコストの分析
- AWS Cost and Usage Reportの管理
- Well-Architected 運用上の優秀性の柱

関連する例:

- AWS アカウントのセットアップ
- AWS Billing and Cost Management のデータエクスポート
- AWS Cost Explorer の一般的なユースケース

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 でタグ付けを実装して、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。キーと値のペアの一例としては、キーが `Environment` で、値は `Production` となります。本稼働環境のすべてのリソースには、キーと値のペアがあります。タグ付けにより、関連性の高い組織情報を使用して、コストを分類、追跡できます。組織のカテゴリ (コストセンター、アプリケーション名、プロジェクト、オーナーなど) を表すタグを適用し、ワークロードやワークロードの特性 (テストや本番など) を識別して、組織全体のコストと使用状況の帰属先を付与できます。

AWS リソース (Amazon Elastic Compute Cloud インスタンスや Amazon Simple Storage Service バケットなど) にタグを付け、そのタグをアクティブ化すると、AWS はこの情報をコストと使用状況レポートに追加します。タグ付けされたリソースとタグ付けされていないリソースに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。タグポリシーを AWS Organizations で使用して、ルールを定義します。ルールは、AWS Organizations のアカウントの AWS リソースに対してタグをどのように使用できるかを定めたものです。タグポリシーを使用すると、AWS リソースにタグを付ける標準アプローチを簡単に導入できます。

AWS 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、検索結果からそのリソースのタグを管理します。

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、より高い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

- タグスキーマを定義する: すべての利害関係者をビジネス全体から集めて、スキーマを定義します。これには通常、技術、財務、および管理ロールの担当者が含まれます。すべてのリソースに必要なタグのリストと、リソースに必要なタグのリストを定義します。タグの名前と値が組織全体で一貫していることを確認します。
- リソースをタグ付けする: 定義したコスト帰属カテゴリを使用して、カテゴリに従ってワークロードのすべてのリソースにタグを付けます。効率を高めるには、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: タグ付けを実装しなくても Cost Categories 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タグ付けの高いレベルを維持していることを確認するには、タグ付けを自動化して、リソースの作成時に自動的にタグ付けされるようにします。AWS CloudFormation などのサービスを使用して、リソースの作成時にタグ付けされていることを確認します。Lambda 関数を使用して自動的にタグ付けするカスタムソリューションや、ワークロードを定期的にスキャンし、タグ付けされていないリソースをすべて削除するマイクロサービスを作成することもできます。これは、テスト環境および開発環境に最適です。
- タグ付けをモニタリング、レポートする: 組織全体でタグ付けの高いレベルを維持していることを確認するには、ワークロード全体でタグをレポートおよびモニタリングします。AWS Cost Explorer を使用して、タグ付けされたリソースとタグ付けされていないリソースのコストを表示したり、タグエディタなどのサービスを使用したりできます。タグ付けされていないリソースの数を定期的に確認し、必要なレベルのタグ付けになるまでタグを追加するアクションを実行します。

リソース

関連ドキュメント:

- タグ付けのベストプラクティス
- AWS CloudFormation リソースタグ
- AWS Cost Categories
- AWS リソースのタグ付け
- AWS Budgets によるコストの分析
- Cost Explorer によるコストの分析
- AWS コストと使用状況レポートの管理

関連動画:

- コストセンターまたはプロジェクトによる請求を分割するための AWS リソースをどのようにタグ付けすればよいか教えてください
- AWS リソースのタグ付け

COST03-BP03 コスト属性カテゴリを特定する

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ます。こうしたカテゴリを活用して、支出の説明責任の徹底、コスト意識の向上、効果的な消費行動の促進を図ります。

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

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、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。また、クラウド支出を配分することで、組織は一元化されたクラウドガバナンスのベストプラクティスをさらに採用できるようになります。

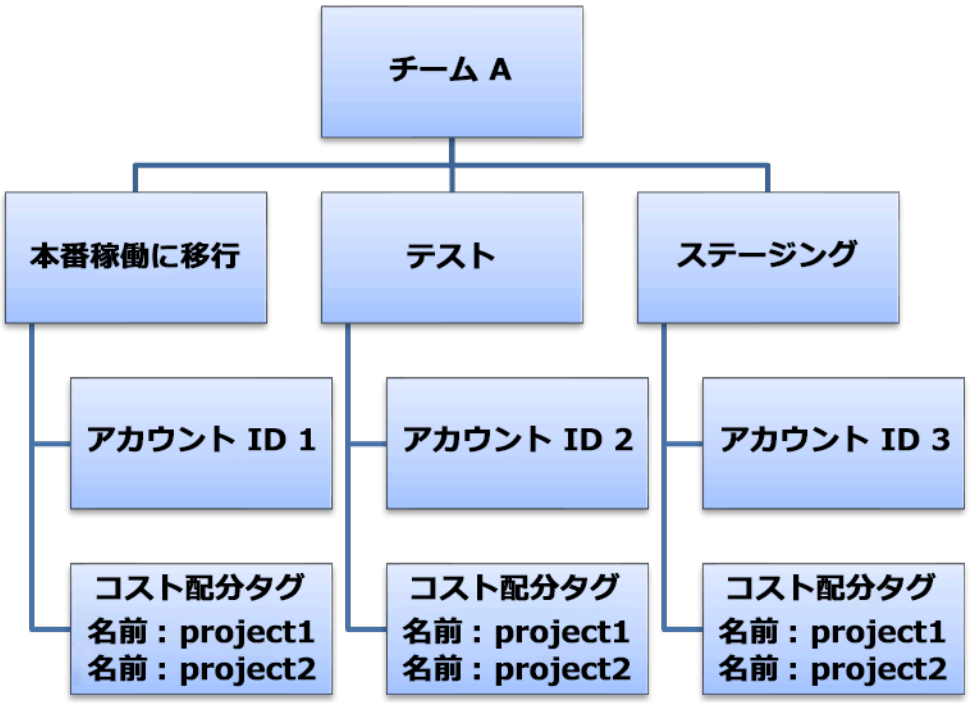
定期的なミーティングで、財務チームやその他の関係者と協力し、組織内でコストを配分する方法の要件を理解します。ワークロードのコストは、開発、テスト、本稼働、廃止などライフサイクル全体にわたって配分する必要があります。学習、スタッフ育成、アイデア創出に要したコストが、どのよ

うに組織に帰属するかを理解します。これは、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うえで役立ちます。

組織内のステークホルダーとコスト属性カテゴリを定義したら、AWS Cost Categories を使用して、コストと使用状況の情報を特定のプロジェクトのコストや部門やビジネスユニットの AWS アカウントなど、AWS クラウドの有意義なカテゴリ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。Cost Categories を使うと、ルールベースのエンジンを使用してコストを分類できます。ルールを設定することで、コストをカテゴリ別に分類します。ルール内では、特定の AWS アカウント、AWS サービス、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 コンソールで使用できます。これには AWS Cost Explorer、AWS Budgets、AWS Cost and Usage Report、および AWS Cost Anomaly Detection があります。

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)、複数の環境 (ルール)、そして複数のリソースまたはアセットを持つ各環境 (ディメンション) にグループができます。



コストと使用状況の組織図

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、AWS Cost Explorer、AWS Budgets、AWS Cost and Usage Report、および AWS Cost Anomaly Detection に表示されます。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。これを使用して、特定のコストカテゴリ値でフィルタリングしたり、コストカテゴリ別にグループ化したりできます。

実装手順

- 組織のカテゴリを定義する: 社内の関係者およびビジネスユニットとミーティングを行い、組織の構造と要件を反映したカテゴリを定義します。これらのカテゴリは、ビジネスユニット、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。
- 機能を反映したカテゴリを定義する: 社内の関係者およびビジネスユニットとミーティングを行い、企業内の機能を反映した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。
- AWS Cost Categories を定義する: AWS Cost Categories の使用に対してコストと使用状況情報を整理するコストカテゴリを作成して、AWS コストと使用状況を有意義なカテゴリにマッピン

グします。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、AWS Cost Categories を使用したカテゴリ化された構造内でコストを管理できるようになります。

リソース

関連ドキュメント:

- AWS リソースのタグ付け
- コスト配分タグの使用
- AWS Budgets によるコストの分析
- Cost Explorer によるコストの分析
- AWS Cost and Usage Report 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 の機能

関連する例:

- 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
- Well-Architected ラボ: コストと使用状況の可視化
- Well-Architected ラボ: Cost Categories

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を活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードのアウトプットがビジネスの成功に対してどのように測定されるかを理解します。通常、各ワークロードには、パフォーマンスを示す主な成果の小さな組み合わせがあります。多数のコンポーネントを含む高度なワークロードがある場合は、リストに優先順位を付けるか、各コンポーネントのメトリクスを定義して追跡できます。チームと協力して、どのメトリクスを使用するか理解します。この単位は、ワークロードの効率または各ビジネス成果のコストを把握するために使用されます。

実装手順

- ワークロードの成果を定義する: ビジネスの利害関係者とミーティングをして、ワークロードの成果を定義します。これらは顧客の使用状況の主要な測定指標であり、技術的メトリクスではなく、ビジネスメトリクスである必要があります。ワークロードごとに少数の概要的なメトリクス (5 つ未満) が存在する必要があります。ワークロードが異なるユースケースで複数の成果を生成する場合は、それらを単一のメトリクスにグループ化してください。
- ワークロードコンポーネントの成果を定義する: 必要に応じて、大規模で複雑なワークロードがある場合、または明確に定義された入出力を使用してワークロードをコンポーネント (マイクロサービスなど) に簡単に分割できる場合は、各コンポーネントのメトリクスを定義します。この作業では、コンポーネントの価値とコストを反映する必要があります。最大のコンポーネントから開始し、大きさ順で、最小のコンポーネントまで作業します。

リソース

関連ドキュメント:

- AWS リソースのタグ付け
- AWS Budgets によるコストの分析
- Cost Explorer によるコストの分析
- 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、組織のポリシーに合ったコスト管理ツールを設定します。これには、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の制御の強化、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コスト削減を行うサービス、ツール、リソースが含まれます。

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: 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します。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る可能性があります。

クラウド支出の説明責任を推進するには、コストと使用状況の可視化が可能なツールへのアクセスをユーザーに許可します。AWS では、以下の目的に合わせてすべてのワークロードとチーム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。AWS Control Tower や AWS Organizations などのツールを使用して複数の AWS アカウントを作成します。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、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で追跡します。適切なステークホルダー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、サービスコントロールポリシー (SCP)、タグポリシー、IAM ポリシー、予算アラートを使用する際の想定外のシナリオを回避します。例えば、チームが効果的な制御メカニズムを使用する場合のみ目的のリージョンで推奨リソースを作成できるようにしたり、特定のタグ (cost-center など) がないとリソースを作成できないようにしたりすることができます。
- 現状確認: 現在のコストと使用量を示すダッシュボードを設定します。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。データをエクスポートし、AWS Cost Optimization Hub のコストと使用状況ダッシュボードまたは任意のサポート対象製品を使用することで、このような可視性が可能になります。ペルソナごとに別々のダッシュボードを作成しなければならない場合があります。例えば、マネージャーのダッシュボードはエンジニアリングのダッシュボードとは異なる場合があります。
- 通知: コストまたは使用量が定義された制限を超え、AWS Budgets または AWS コスト異常検出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量の情報を要約します。詳細で帰属先が特定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。レポートを使用するチームと関連性があり、推奨事項を含めたレポートを作成します。

- 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します。
- 分析: チームメンバーは、さまざまなフィルター (リソース、アカウント、タグなど) を使用して、時間単位、日単位、または月単位でカスタム分析とディープ分析を実行できます。
- 検査: リソースのデプロイとコスト最適化の機会を最新の状態に保ちます。Amazon CloudWatch、Amazon SNS、または Amazon SES を使用して、組織レベルでのリソースデプロイに関する通知を受け取ります。AWS Trusted Advisor または AWS Compute Optimizer を使用してコスト最適化の推奨事項を確認します。
- トレンドレポート: 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。

AWS Cost Optimization Hub を使用して、統合された潜在的なコスト削減の機会を一元的な場所から理解し、Amazon Athena と統合するためのデータエクスポートを作成できます。また、AWS Cost Optimization Hub を使用してコストと使用状況ダッシュボードをデプロイすることもできます。このダッシュボードでは、Amazon QuickSight を使用してインタラクティブなコスト分析を行ったり、コストに関するインサイトを安全に共有したりできます。

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)、または AWS パートナーを利用できます。サードパーティーのツールを利用することもできますが、利用に際しては必ず価値提案を検証するようにしてください。

実装手順

- ツールへの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、各自の使用を追跡できるようになります。
- コストタグとカテゴリを整理する: チーム、ビジネスユニット、アプリケーション、環境、プロジェクト全体でコストを整理します。リソースタグを使用して、コスト配分タグごとにコストを整理します。タグ、アカウント、サービスなどを使用してディメンションに基づいて Cost Categories を作成し、コストをマッピングします。
- AWS Budgets を設定する: ワークロードのすべてのアカウントでAWS Budgets を設定します。タグとコストカテゴリを使用して、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。予算額を超えたときや、推定コストが予算を超えるときにアラートを受信するよう、AWS Budgets の通知を設定します。

- AWS コスト異常検出を設定する: AWS コスト異常検出を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 SNS 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。
- コスト分析ツールを使用する: 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。
- コスト削減分析ツールを使用する: AWS Cost Optimization Hub を使用して、未使用リソースの削除、適切なサイズ設定、Savings Plans、予約、Compute Optimizer の推奨事項など、カスタマイズされた推奨事項でコスト削減の機会を特定します。
- 高度なツールを設定する: 任意でビジュアルを作成して、インタラクティブな分析やコストインサイトの共有を支援できます。AWS Cost Optimization Hub でデータエクスポートを使用すると、Amazon QuickSight を活用したコストと使用状況ダッシュボードを組織に合わせて作成します。このダッシュボードでは、さらなる詳細と粒度が得られます。また、Amazon Athena でデータエクスポートを使用して高度な分析機能を実装することで高度なクエリを実施したり、Amazon QuickSight でダッシュボードを作成したりできます。AWS パートナーと協力して、統合されたクラウド請求書のモニタリングと最適化のためのクラウド管理ソリューションを導入できます。

リソース

関連ドキュメント:

- AWS Billing and Cost Management とは
- ベストプラクティスの AWS 環境を確立する
- AWS リソースのタグ付けのベストプラクティス
- AWS リソースのタグ付け
- AWS Cost Categories
- AWS Budgets によるコストの分析
- AWS Cost Explorer によるコストの分析
- AWS データエクスポートとは

関連動画:

- クラウドインテリジェンスダッシュボードのデプロイ
- 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

関連する例:

- Amazon QuickSight によって提供されるコストと使用状況ダッシュボード
- AWS コストと使用状況ガバナンスワークショップ

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

コスト最適化とは、最低の価格点でビジネス成果を達成するということです。ワークロードメトリクス (ワークロードの効率で測定) に基づいて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 のコストを確認することで取得できます。この分析を時間単位で実行します。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) でリクエストレートが変化する (使用量のピークが午前 9 時から午後 5 時で、夜間のリクエストはほとんどない、など) 場合、通常、効率性は変化します。静的コストと変動コストの関係を理解しておくと、最適化アクティビティの焦点を絞ることができます。

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) や Amazon API Gateway のコンテナ化されたアプリケーションのような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。

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 QuickSight などの分析サービスを使用して、ワークロード全体や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

関連ドキュメント:

- AWS リソースのタグ付け
- AWS Budgets によるコストの分析
- Cost Explorer によるコストの分析
- AWS コストと使用状況レポートの管理

関連する例:

- 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. リソースはどのように廃止するのですか?

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。

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 廃止プロセスを実装する
- COST04-BP03 リソースを廃止する
- COST04-BP04 自動的にリソースを廃止する
- 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を活用し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。もう 1 つの例は、削除されるタグキーの名前や値などのリソースに `LifeSpan` または `TTL` を使用して、廃止の期間や特定の時間を定義するものです。

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 のタグ付けの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に対してワークロードスループット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まで下がる場合は、時間要因を組み込みます。未使用または十分に活用されていないリソースの詳細については、「AWS Trusted Advisor コスト最適化チェック」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 を使用すると、同じ AWS リージョンにある AWS リソースを整理し管理することができます。ほとんどのリソースにタグを追加して、組織内のリソースを識別および並べ替えることができます。サポートされているリソースに一括でタグを追加するときはタグエディタを使用します。承認済み製品のポートフォリオを作成、管理し、エンドユーザーに配布して、製品ライフサイクルを管理するときは、AWS Service Catalog の使用を検討してください。

リソース

関連ドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor コスト最適化チェック
- AWS リソースのタグ付け
- カスタムメトリクスをパブリッシュする

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にはどうすればよいですか?
- AWS Trusted Advisor を使用してコストを最適化する方法を教えてください。

COST04-BP02 廃止プロセスを実装する

未使用のリソースを特定して廃止するためのプロセスを実装します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

組織全体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

- 廃止プロセスを作成、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廃止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。リソースを廃止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。廃止プロセスが開始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、以下の廃止手順を使用してください。

- 廃止されるリソースを特定する: AWS クラウドで廃止の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、廃止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。
- 調整とコミュニケーションをする: ワークロードの所有者と協力して、廃止されるリソースを確認します。
- メタデータを記録してバックアップを作成する: 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) を記録し、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はバックアップ (Amazon Elastic Block Store スナップショットの作成、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 の検証をする: リソースが AWS 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、必要に応じて再デプロイできるようにします。
- アクセスの防止をする: リソースが必要かどうかを判断する間にリソースの使用を防ぐため、制限付きコントロールを一定期間適用します。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。
- 内部廃止プロセスを遵守する: 組織ドメインからのリソースの削除、DNS レコードの廃止、または設定管理ツール、モニタリングツール、自動化ツール、およびセキュリティツールからのリソース削除など、組織の管理タスクと廃止プロセスに従います。

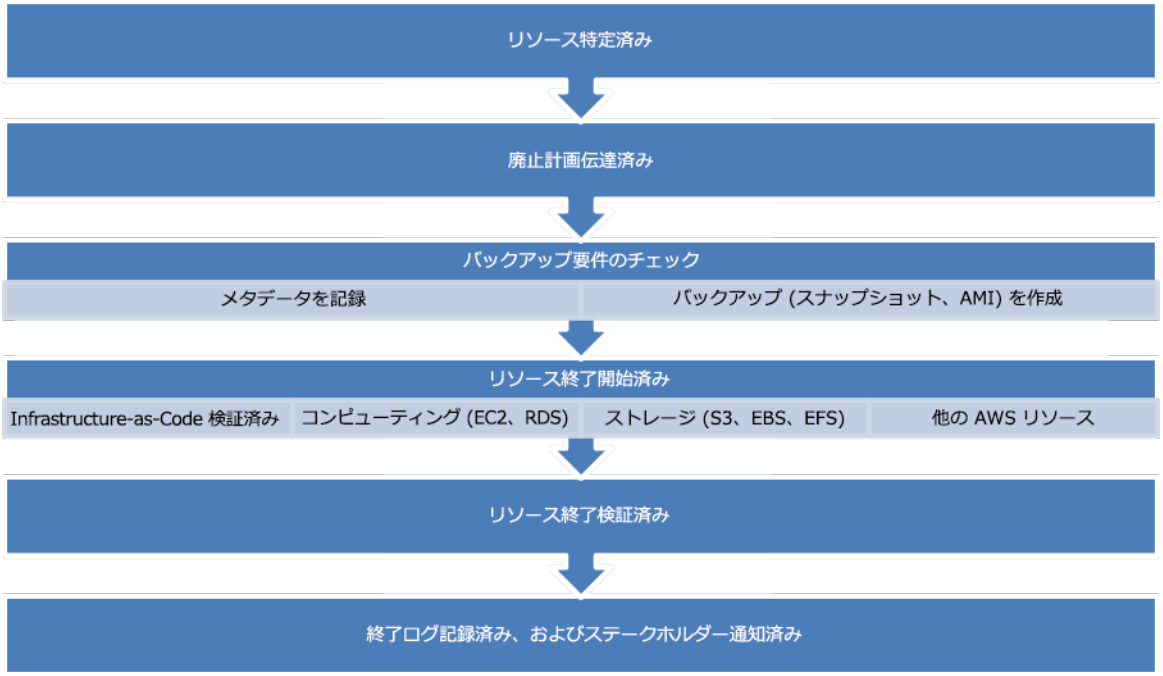
リソースが Amazon EC2 インスタンスである場合は、次のリストを参照してください。詳細については、「Amazon EC2 リソースを削除するか、終了する方法を教えてください。」を参照してください。

- すべての Amazon EC2 インスタンスとロードバランサーを停止または終了します。Amazon EC2 インスタンスは、終了後しばらくの間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する
- すべての専有ホストを解放します。
- Amazon EBS ボリュームと Amazon EBS スナップショットをすべて削除します。
- すべての Elastic IP アドレスを解放します。
- すべての Amazon マシンイメージ (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 S3 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 S3 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 ストレージクラスのパフォーマンスを参照してください。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。

次の簡単な廃止プロセスのフローチャートは、廃止手順を概説しています。リソースを廃止する前に、廃止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。



リソース廃止フローです。

リソース

関連ドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

関連動画:

- CloudFormation スタックを削除するがいくつかのリソースを保持する
- Amazon EC2 インスタンスを起動したユーザーを確認するには

関連する例:

- Amazon EC2 リソースを削除するか、終了する方法を教えてください。
- 自分のアカウントで EC2 インスタンスを起動したユーザーを確認するにはどうすればよいですか?

COST04-BP03 リソースを廃止する

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廃止します。通常、廃止は定期的に行われ、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活用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開始されます。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開始される場合もあります。

実装手順

- リソースを廃止する: これは、不要になった AWS リソースの廃止段階またはライセンス契約の終了段階です。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、廃止段階に移行してリソースを廃止する前に完了したすべての最終チェックを完了します。廃止プロセスを使用して、未使用と識別された各リソースを廃止します。

リソース

関連ドキュメント:

- AWS Auto Scaling
- AWS Trusted Advisor

関連する例:

- Well-Architected ラボ: リソースの廃止 (レベル 100)

COST04-BP04 自動的にリソースを廃止する

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廃止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を活用し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、廃止プロセスの関連コストを削減または削除します。自動廃止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。Amazon EC2 Auto Scaling または Application Auto Scaling を使用して、廃止プロセスを実行できます。API または SDK でカスタムコードを実装し、ワークロードリソースを自動的に廃止することもできます。

モダンアプリケーションはサーバーレスファースト、つまりサーバーレスサービスの採用を優先するように構築されています。AWS は、コンピューティング、インテグレーション、データストアというスタックの 3 つのレイヤー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

- Amazon EC2 Auto Scaling または Application Auto Scaling を実装する: サポートされているリソースは Amazon EC2 Auto Scaling または Application Auto Scaling で設定します。これらのサービスは、AWS サービス利用時の使用率とコスト効率の最適化に役立ちます。これらのサービスは、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。
- インスタンスを終了するように CloudWatch を設定する: CloudWatch アラーム を使用してインスタンスを終了するように設定できます。廃止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廃止できます。AWS と統合し、使用されなくなったリソースを終了または削除するコードをアプリケーション内に実装します。
- サーバーレスサービスを使用する: アプリケーションを構築および実行する際はサーバーレスアーキテクチャとイベント駆動型アーキテクチャを AWS で構築することを優先します。AWS は、自動的に最適化されたリソース使用率と自動廃止 (スケールインとスケールアウト) を提供するサーバーレステクノロジーサービスを提供します。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

関連ドキュメント:

- Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling の使用を開始する
- Application Auto Scaling
- AWS Trusted Advisor
- AWS でのサーバーレス
- EC2 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 CloudWatch アラームへの終了アクションの追加

関連する例:

- AWS CloudFormation スタックの自動削除のスケジュール
- Well-Architected ラボ: リソースを自動的に廃止する (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の削除を処理します。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活用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-backed 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。また、API または SDK を使用してカスタムコードを実装することで、オブジェクトを自動的に削除するライフサイクルポリシーとポリシールールを作成することもできます。

実装手順

- Amazon Data Lifecycle Manager を使用する: Amazon Data Lifecycle Manager の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-backed 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いて Amazon S3 がオブジェクトのライフサイクル中に実行するアクションおよびオブジェクトライフサイクルの終了時の削除アクションを定義します。

リソース

関連ドキュメント:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- Amazon S3 のバケットのライフサイクル設定を行う方法

関連動画:

- Amazon Data Lifecycle Manager を使用した Amazon EBS スナップショット管理の自動化
- 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にはどうすればよいですか?

関連する例:

- 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にはどうすればよいですか?
- Well-Architected ラボ: リソースを自動的に廃止する (レベル 100)

費用対効果の高いリソース

Questions

- COST 5. サービスを選択するときは、どのようにコストを評価するのですか?
- COST 6. コストターゲットに合わせて、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には、どうすればよいですか?
- COST 7. 料金モデルは、コスト削減のためにどのように使用するのですか?
- COST 8. データ転送料金はどのように計画するのですか?

COST 5. サービスを選択するときは、どのようにコストを評価するのですか?

Amazon EC2、Amazon EBS、Amazon S3 は、基盤となる AWS のサービスです。Amazon RDS や Amazon DynamoDB などのマネージドサービスは、高レベルまたは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、管理または運用のオーバーヘッドを大幅に削減または排除することができ、アプリケーションとビジネス関連の活動に専念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 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-BP05 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

ほとんどの組織で、情報技術 (IT) 部門は複数の小さなチームから構成されており、抱える議題や注力分野は、所属メンバーの専門分野とスキルを反映してそれぞれに異なります。組織全体の目標、優先事項、目的を把握し、各部門や各プロジェクトがそうした目標にどのように貢献しているかを理解する必要があります。人員、設備、技術、資材、外部サービスなど、すべての重要なリソースを分類しておくことは、組織の目標を達成し、包括的な予算計画を立てるうえで不可欠です。こうした体系的なアプローチでコストを特定し、把握することが、組織にとって現実的で健全なコスト計画を立てるための基本です。

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。コスト最適化と、AWS Well-Architected フレームワークの他の柱 (パフォーマンスや信頼性など) とのバランス

を図ります。このプロセスは、組織の目標、市況、業務のダイナミクスの変化を反映するために、体系的かつ定期的に実施する必要があります。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。製品、ビジネス、技術、財務など、組織内のすべてのチームと会合し、情報を収集します。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。

例えば、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。

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ます。柱を順番に並べたリストを作成してください。また、それぞれの柱に重みを付け、その柱が他の柱よりどの程度重視されているかや、2 つの柱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。
- 技術的負債に対処し、文書化する: ワークロードのレビュー中に、技術的負債に対処します。バックログ項目を文書化して、後日、リファクタリングやリアーキテクティングで最適化を進めることを目標に、ワークロードを保持します。生じたトレードオフを他のステークホルダーに明確に伝えることが大切です。

リソース

関連するベストプラクティス:

- 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
- OPS01-BP06 トレードオフを評価する

関連ドキュメント:

- 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

COST05-BP02 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

組織にビジネス価値をもたらすべく設計されたワークロードコンポーネントには、さまざまなサービスが含まれる場合があります。コンポーネントごとに、ビジネスニーズに対応する特定の AWS クラウドサービスを選択できます。何が選択されるかは、それらのサービスに関する知識や使用経験などの要因によって違ってきます。

「COST05-BP01 組織のコスト要件を特定する」で説明されているように組織の要件を特定したら、ワークロード内のすべてのコンポーネントを徹底的に分析します。現時点と今後予測されるコストとサイズを考慮して、各コンポーネントを分析します。分析のコストを、ワークロードのライフサイクル全体で削減が見込まれる額と比較検討してください。対象となるワークロードのコンポーネントをすべて分析するための労力が、そのコンポーネントの最適化により見込まれる節約額や改善に見合っていなければなりません。例えば、提案されたリソースのコストが月額 10 USD で、予測負荷が月額 15 USD を超えない場合に、コストを 50% (月額 5 USD) 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。例えば、最初は EMR Serverless の使用が適しているかもしれません。ただし、そのサービスの使用量が増えてきたら、EC2 の EMR に移行することで、ワークロードの該当コンポーネントのコストを削減できる可能性があります。

AWS Cost Explorer および AWS Cost and Usage Report (CUR) では、概念実証 (PoC) または実行中の環境のコストを分析できます。AWS Pricing Calculator を使用してワークロードのコストを見積もることもできます。

技術チームがワークロードを見直すためのワークフローを作成します。このワークフローはシンプルなものにし、必要なステップをすべて網羅することで、チームがワークロードの各コンポーネントと

その料金を理解できるようにします。組織はこのワークフローに従い、またそれを各チームの特定のニーズに基づいてカスタマイズできます。

1. ワークロードに使用されている各サービスを一覧表示する: これは良い出発点です。現在使用されているすべてのサービスと、コストの発生源を特定します。
2. これらのサービスの料金体系を理解する: 各サービスの料金モデルを理解します。AWS の各サービスには、使用量、データ転送、機能に固有の料金などの要因に基づくさまざまな料金モデルがあります。
3. 予期しないワークロードコストがあり、予想される使用量やビジネス成果と一致しないサービスに着目する: AWS Cost Explorer または AWS Cost and Usage Report の使用量と価値に対してコストが比例しない例外やサービスを特定します。最適化にかける労力に優先順位を付けるには、コストとビジネス成果を相関付けることが重要です。
4. AWS Cost Explorer、CloudWatch Logs、VPC フローログ、Amazon S3 ストレージレンズを使用して、これらの高コストの根本原因を把握する: これらのツールは、高コストの診断に役立ちます。各サービスは、使用量とコストの確認と分析に役立つ異なる観点を提供します。例えば、Cost Explorer は全体的なコスト傾向の特定に役立ち、CloudWatch Logs は運用に関するインサイトを提供します。また、VPC フローログは IP トラフィックを表示し、Amazon S3 ストレージレンズはストレージ分析に有用です。
5. AWS Budgets を使用して、サービスまたはアカウントの特定金額の予算を設定する: 予算の設定は、積極的なコスト管理方法です。AWS Budgets を使用して、カスタム予算しきい値を設定し、コストがそのしきい値を超えたときにアラートを受け取ります。
6. 請求および使用状況アラートを送信するように Amazon CloudWatch アラームを設定する: コストと使用状況メトリクスのモニタリングとアラートを設定します。CloudWatch アラームを使用すると特定のしきい値を超えたときに通知できるため、介入の応答時間を短縮できます。

現在の属性に関係なく、すべてのワークロードコンポーネントを戦略的に見直すことで、時間の経過に伴う着実な強化とコスト削減を促進します。このレビュープロセスに費やす労力は、それに見合う効果が得られるか慎重に検討したうえで、決める必要があります。

実装手順

- ワークロードコンポーネントを一覧する: ワークロードのコンポーネントのリストを作成します。このリストを使用して、各コンポーネントが分析されたことを確認します。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度に見合うものにする必要があります。効率向上のために、リソースを機能別にグループ化します (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など)。

- コンポーネントリストを優先順位付けする: 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

関連ドキュメント:

- AWS Pricing Calculator
- AWS Cost Explorer
- Amazon S3 ストレージクラス
- AWS クラウド 製品

関連動画:

- AWS コスト最適化シリーズ: CloudWatch

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を活用し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、アプリケーションを AWS Elastic Beanstalk などのフルマネージドプラットフォームに移行することで、データベースインスタンスの管理に要する時間を短縮したいと考えているとします。

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、余剰キャパシティを最小限に抑え、パフォーマンスを最大化する必要があります。AWS Management Console や AWS API および SDK を使用して AWS Managed Services の属性を変更し、需要の変化に合わせてリソースのニーズを調整できます。例えば、Amazon EMR クラスター (または Amazon Redshift クラスター) のノード数を増減して、規模をスケールアウトまたはスケールインできます。

また、AWS リソースの複数のインスタンスを圧縮して、高密度での使用を有効に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 つを専用の 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ています。Amazon Athena、Amazon EMR、および 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、オンゴーイングの管理コストを削減できます。

実装手順

- 徹底分析を実行する: 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。

- マネージドリソースと非マネージドリソースを比較する: 管理するリソースの運用コストを考慮して、AWS マネージド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) を使用した場合と比較したり、Amazon EMR を、Amazon EC2 で Apache Spark を実行する場合と比較したりします。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。考慮すべき最も重要な 3 つの要因は、使用するマネージドサービスのタイプ、データの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?
- What is Amazon EMR and how can I use it for processing data?

関連する例:

- マネージドデータベースに移行すべき理由
- Consolidate data from identical SQL Server databases into a single Amazon RDS for SQL Server database using AWS DMS
- Amazon Managed Streaming for Apache Kafka (Amazon MSK) にデータを大規模に提供する
- Migrate an ASP.NET web application to AWS Elastic Beanstalk

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を活用しない場合のリスクレベル: 低

実装のガイダンス

「オープンソース」は、ソフトウェア開発の文脈で生まれた用語であり、ソフトウェアが特定の無料配布基準に準拠しているという意味です。オープンソースソフトウェアは、誰でも検査、変更、拡張できるソースコードで構成されています。組織はビジネス要件、エンジニアのスキル、予測される使用状況、その他の技術的な依存関係を踏まえて、ライセンスコストを最小限に抑えるため、AWS でオープンソースソフトウェアを使用することを検討できます。言い換えれば、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で得られる効果を、ワークロードの最適化にかかる総コストに照らして測定してください。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。

例えば、Linux オペレーティングシステムを搭載した Amazon EC2 インスタンスを us-east-1 で運用する場合、Windows で実行する別の Amazon EC2 インスタンスを運用する場合と比較して約 45% のコスト削減になります。

AWS Pricing Calculator では、Amazon RDS インスタンスや各種データベースエンジンなど、ライセンスオプションが異なるさまざまなリソースのコストを包括的に比較できます。さらに、AWS Cost Explorer では、既存のワークロード (特にライセンスがさまざまに異なるワークロード) のコストを有益な視点で検討できます。ライセンス管理のために、AWS License Manager はソフトウェアライセンスを監督および処理するための合理化された方法を提供します。お客様は、AWS クラウドでお好みのオープンソースソフトウェアをデプロイして運用できます。

実装手順

- ライセンスオプションを検討する: 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調べます。好条件があれば、ソフトウェアのコストに見合う利点が得られます。
- 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

関連ドキュメント:

- Open Source at AWS
- 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

関連する例:

- オープンソースブログ
- AWS オープンソースブログ
- 最適化とライセンス評価

COST05-BP05 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、支出を最小限に抑えます。

このベストプラクティスを活用しない場合のリスクレベル: 中

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、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である、Amazon Relational Database Service (Amazon RDS)、Amazon DynamoDB、Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Email Service (Amazon SES) を使用して組織の全体的なコストを削減することが含まれます。

コンピューティングにはサーバーレスやコンテナを使用します。AWS Lambda や静的ウェブサイト用の Amazon Simple Storage Service (Amazon S3) などです。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。

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。

イベント駆動型アーキテクチャは、サーバーレスサービスで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 サーバーレスアプリケーションレンズのホワイトペーパーを参照してください。

実装手順

- 各サービスを選択してコストを最適化する: 優先順位リストと分析を使用して、組織の優先順位に最も合致する各オプションを選択します。需要に合わせてキャパシティ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見直す必要がある場合、インスタンスサイズを増やす、または 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ことを検討します。

- 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
- イベント駆動型アーキテクチャとは
- Amazon S3 ストレージクラス
- AWS クラウド製品
- Amazon ElastiCache (Redis OSS)

関連する例:

- イベント駆動型アーキテクチャの導入
- イベント駆動型アーキテクチャ
- Amazon ElastiCache (Redis OSS) を使用して 100 倍効率よく Statsig を実行する方法
- 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 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。レビューの開始要因には、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。

組織のステークホルダーと話し合い、AWS Cost Explorer の予測機能を使用してサービス変更の潜在的な影響を予測し、時間の経過とともにさまざまな使用状況のコスト分析を行います。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き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、そ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
- Amazon EC2 Auto Scaling
- クラウドへのデータ移行
- AWS Snow Family

関連動画:

- AWS OpsHub for Snow Family

COST 6. コストターゲットに合わせて、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-BP04 共有リソースの使用を検討する

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。予測されたさまざまな負荷のワークロードに対して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釣り合っているなどです。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロード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。

予測されたさまざまな負荷のワークロードに対して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で構成されているワークロードのコストモデリングを作成するときは、AWS Compute Optimizer がワークロード実行のためのコストモデリングに役立ちます。使用履歴に基づき、コンピューティングリソースの正しいサイズ設定に関するレコメンデーションを提供します。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適正化のために、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は、AWS Trusted Advisor、Amazon CloudWatch、Amazon CloudWatch Logs など複数あります。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 AWS Trusted Advisor は、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とおりです。

- 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。例えば、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

関連ドキュメント:

- AWS Auto Scaling
- 適切なサイジングのための機会の特定
- Amazon CloudWatch の特徴
- コスト最適化: Amazon EC2 の適切なサイジング
- AWS Compute Optimizer
- AWS Pricing Calculator

関連する例:

- Perform a Data-Driven Cost Modelling
- 計画している AWS リソース構成のコストを見積もる方法を教えてください。
- Choose the right AWS tools

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を活用しない場合のリスクレベル: 中

実装のガイダンス

Amazon EC2 では、さまざまなユースケースに合わせて、CPU、メモリ、ストレージ、ネットワークキャパシティのレベルが異なる、幅広いインスタンスタイプを選択肢として用意しています。インスタンスタイプごとに CPU、メモリ、ストレージ、ネットワーク機能の組み合わせが異なるため、プロジェクトに適したリソースの組み合わせを柔軟に選択できます。どのインスタンスタイプにもサイズが複数用意されており、ワークロードの需要に基づいてリソースを調整できます。必要なインスタンスタイプを判断するには、インスタンスで実行する予定のアプリケーションまたはソフトウェアのシステム要件に関する詳細情報を収集する必要があります。これらの詳細には、次の内容を含める必要があります。

- オペレーティングシステム
- CPU コア数
- GPU コア
- システムメモリ (RAM) の容量
- ストレージタイプとスペース
- ネットワーク帯域幅要件

コンピューティング要件の目的と必要なインスタンスを明らかにしたうえで、さまざまな Amazon EC2 インスタンスファミリーを検討します。次のインスタンスタイプファミリーが提供されています。

- 汎用
- コンピューティング最適化
- メモリを最適化
- ストレージの最適化
- 高速コンピューティング
- HPC 最適化

特定の Amazon EC2 インスタンスファミリーが達成できる特定の目的とユースケースの詳細については、「AWS インスタンスタイプ」を参照してください。

お客様のニーズに最適な特定のインスタンスファミリーとインスタンスタイプを選択するには、システム要件の収集が不可欠です。インスタンスタイプ名は、ファミリー名とインスタンスサイズで構成されます。例えば、t2.micro インスタンスは T2 ファミリーに属するマイクロサイズです。

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。AWS 料金見積りツールやコスト管理ツールを使用すれば、十分な判断材料を基にインスタンスのタイプ、サイズ、構成を決定できます。

実装手順

- データに基づいてリソースを選択する: コストモデリングのデータを使用して、予測されるワークロードの使用レベルを選択し、指定されたリソースタイプとサイズを選択します。コストモデリングデータに基づいて、インスタンスに求められるデータ転送速度を考慮しつつ、仮想 CPU の数、総メモリ (GiB)、ローカルインスタンスストアボリューム (GB)、Amazon EBS ボリューム、ネットワークパフォーマンスレベルを決定します。常に詳細な分析と正確なデータに裏付けられた選択を行い、パフォーマンスの最適化とコスト管理の効率化を両立させましょう。

リソース

関連ドキュメント:

- AWS インスタンスタイプ
- AWS Auto Scaling
- Amazon CloudWatch の特徴
- EC2 Right Sizing によるコスト最適化

関連動画:

- Selecting the right Amazon EC2 instance for your workloads
- Right size your service

関連する例:

- It just got easier to discover and compare Amazon EC2 instance types

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を活用しない場合のリスクレベル: 低

実装のガイダンス

ワークロード内に、実行中のワークロードのアクティブなメトリクスを使用してそのワークロードを変更するフィードバックループを作成します。適切なサイジング操作を実行するように設定した AWS Auto Scaling などのマネージドサービスを使用できます。AWS には、API、SDK、最小限の労力でリソースを変更することができる機能も用意されています。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で選べる機能が組み込まれているものもあります。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オブザーバビリティを高める: 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。検索する主なメトリクスは、CPU 使用率とメモリ使用率です (例えば、「Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled」で説明されているように、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では、Amazon S3 ストレージレンズを使用できます。これにより、さまざまなカテゴリにわたる 28 のメトリクスを

バケットレベルで表示でき、デフォルトでダッシュボードに 14 日間の履歴データを表示できます。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推奨事項を表示する: AWS Compute Optimizer の適切なサイジングの推奨事項を使用するか、コスト管理コンソールで 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でワークロードを調整する AWS Trusted Advisor を確認します。さまざまなリソースの適切なサイジングを行うときは、適切なツールを使用して、Amazon EC2 インスタンス、AWS ストレージクラス、Amazon RDS インスタンスタイプのいずれであれ、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取得した、状況に応じた推奨事項を使用することで、ストレージを最適化する手順をすぐに実行することができます。
- 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で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てオートスケールできます。スポットインスタンスの使用で割引を受けるだけでなく、リザーブドインスタンスまたは Savings Plan を使用して、通常のオンデマンドインスタンス コストの割引料金を受け取ることができます。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判断できます。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。新世代のインスタンスタイプがリリースされたら自動的にこれを使用し、Amazon EC2 スポットインスタンスにより、これまでよりも広い範囲のキャパシティにアクセスすることができます。Amazon EC2 Fleet 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では、Amazon S3 Intelligent-Tiering および Amazon EFS Infrequent Access 機能を使用できます。この機能を使用すると、データアクセスパターンが変更されても、パフォーマンスに影響を与えたり、運用オーバーヘッドを発生させたりすることなく、ストレージクラスを自動的に選択してストレージコストを自動的に削減できます。

リソース

関連ドキュメント:

- AWS Auto Scaling
- AWS での適切なサイジング
- AWS Compute Optimizer
- Amazon CloudWatch の特徴
- CloudWatch のセットアップ
- CloudWatch カスタムメトリクスをパブリッシュする
- Amazon EC2 Auto Scaling の使用を開始する
- Amazon S3 Storage Lens
- Amazon S3 の新しいストレージクラス、S3 Intelligent-Tiering を発表
- Amazon EFS 低頻度アクセス
- SDK で Amazon EC2 インスタンスを起動する

関連動画:

- Right Size Your Services

関連する例:

- Amazon EC2 Fleet 用自動スケーリングでの属性ベースのインスタンスタイプ選択
- スケジュールされたスケーリングを使用して Amazon Elastic Container Service を最適化しコストを削減する
- Amazon EC2 Auto Scaling での予測スケーリング
- Amazon S3 ストレージレンズでコストを最適化し、使用状況を可視化する
- Well-Architected ラボ: 適切なサイジングの推奨事項 (レベル 100)

COST06-BP04 共有リソースの使用を検討する

複数のビジネスユニットに組織レベルでデプロイ済みのサービスについては、リソースの使用率を高め、総保有コスト (TCO) を削減するために共有リソースの使用を検討してください。共有リソースの使用は、既存のソリューションを使用するか、コンポーネントを共有する、あるいはその両方を行

うことで管理とコストを一元化できるコスト効率の高いオプションです。アカウント境界の内側、または専用のアカウントでモニタリング、バックアップ、接続性などの一般的な機能を管理します。また、標準化の実装、重複の削減、複雑さの軽減もコストの削減につながります。

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

複数のワークロードが同じ機能を果たす場合は、既存のソリューションと共有コンポーネントを使用して管理を改善し、コストを最適化します。セキュリティのベストプラクティスと組織の規制に従うことでクラウドコストを軽減できるように、本稼働の対象外となるデータベースサーバーやディレクトリサービスなどの既存のリソース (特に共有リソース) を使用することを検討してください。最適な価値実現と効率化を実現するには、消費が活発なビジネスの適正分野にコストを配分することが重要です (ショーバックとチャージバックを使用する) 。

ショーバックとは、消費者、ビジネスユニット、総勘定元帳勘定、責任を負うその他のエンティティなどの帰属カテゴリにクラウドコストを分類するレポートを指します。ショーバックの目的は、チーム、ビジネスユニット、または個人に、各自が消費したクラウドリソースのコストを示すことです。

チャージバックとは、特定の財務管理プロセスに適した戦略に基づいて、中央のサービスの支出を各コストユニットに割り当てることです。お客様の場合、チャージバックは、1 つの共有サービスアカウントで発生したコストを、お客様の報告プロセスに適したさまざまな財務コストカテゴリに請求します。チャージバックメカニズムを確立することで、さまざまなビジネスユニット、製品、チームで発生したコストを報告できます。

ワークロードは、重大または重大ではないに分類できます。この分類に基づいて、重大度の低いワークロードには一般的な設定の共有リソースを使用します。コストをさらに最適化するには、専有サーバーを重大なワークロード専用に確保します。複数のアカウント間でリソースの共有やプロビジョニングを行うことで、リソースを効率的に管理できます。開発環境、テスト環境、本番環境が分かれていても、安全に共有を行うことが可能であり、組織構造を損なうことはありません。

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の理解を深めて最適化するには、分割コスト配分データを使用します。これにより、アプリケーションによる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。分割コスト配分データを使用することで、Amazon Elastic Container Service (Amazon ECS) または Amazon Elastic Kubernetes Service (Amazon EKS) で実行されているコンテナワークロードでタスクレベルのショーバックとチャージバックを実現できます。

分散型アーキテクチャの場合は、共有サービス VPC を構築します。これにより、各 VPC のワークロードで必要な共有サービスに一元的にアクセスできます。これらの共有サービスには、ディレクト

リサービスや VPC エンドポイントなどのリソースを含めることができます。管理オーバーヘッドとコストを削減するには、各 VPC にリソースを構築する代わりに、一元的な場所からリソースを共有します。

共有リソースを使用すると、運用コストの節約、リソース使用率の最大化、一貫性の向上につながります。マルチアカウント設計では、一部の AWS のサービスを一元的にホストし、1 つのハブ内で複数のアプリケーションとアカウントを使用してアクセスすることでコストを節約できます。AWS Resource Access Manager (AWS RAM) を使用することで、VPC サブネットや AWS Transit Gateway アタッチメント、AWS Network Firewall、Amazon SageMaker Pipelines など、他の一般的なリソースを共有することができます。マルチアカウント環境の場合、AWS RAM でリソースを作成したら、それを他のアカウントと共有します。

組織は、共有コストに効果的にタグ付けし、コストの大部分がタグ付けされていない、または配分されていないといったことがないよう確認する必要があります。共有コストが効果的に配分されず、共有コストの管理責任を誰も負わない場合、共有クラウドのコストは悪循環に陥る可能性があります。リソース、ワークロード、チーム、または組織レベルのどこでコストが発生しているのかを把握しておく必要があります。これを把握することで、コスト発生場所での実際の価値を達成したビジネス成果と比較して理解することができます。最終的に、組織はクラウドインフラストラクチャの共有によってコスト削減のメリットを享受します。クラウド支出を最適化するために、共有クラウドリソースのコスト配分を奨励してください。

実装手順

- 既存のリソースを評価する: ワークロードに類似のサービスを使用する既存のワークロードを確認します。ワークロードのコンポーネントに応じて、ビジネスロジックや技術的要件で許容される場合は、既存のプラットフォームを検討します。
- AWS RAM でリソース共有を使用し、適宜制限を適用する: AWS RAM を使用して、組織内の他の AWS アカウントとリソースを共有します。リソースを共有する場合、複数のアカウントでリソースを重複する必要がないため、リソースメンテナンスの運用負担を最小限に抑えることができます。またこのプロセスにより、作成したリソースをアカウント内のロールやユーザーの他、他の AWS アカウントとも安全に共有することができます。
- リソースにタグを付ける: コストレポートの対象候補となるリソースにタグを付け、コストカテゴリ内で分類します。コスト配分用にこうしたコスト関連のリソースタグを有効にすると、AWS リソースの使用状況を可視化できます。コストと使用状況の可視性に関して適切な度合いの詳細度を定めることに重点を置き、コスト配分レポートと KPI 追跡によってクラウドの消費行動に影響を与えます。

リソース

関連するベストプラクティス:

- SEC03-BP08 組織内でリソースを安全に共有する

関連ドキュメント:

- What is AWS Resource Access Manager?
- AWS Organizations で使用できる AWS サービス
- 共有可能な AWS リソース
- AWS コストと使用状況レポート (CUR) クエリ

関連動画:

- AWS Resource Access Manager - granular access control with managed permissions
- How to design your AWS cost allocation strategy
- AWS Cost Categories

関連する例:

- 共有サービスのチャージバック方法: An AWS Transit Gateway の例
- How to build a chargeback/showback model for Savings Plans using the CUR
- Using VPC Sharing for a Cost-Effective Multi-Account Microservice Architecture
- Improve cost visibility of Amazon EKS with AWS Split Cost Allocation Data
- 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 7. 料金モデルは、コスト削減のためにどのように使用するのですか?

費用を最小化するために、リソースロードに最適な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を長期間実行するか (コミットメント割引)、動的に短期間実行するか (スポットまたはオンデマンド) 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 には複数の料金モデルがあり、組織のニーズに合い、製品に応じ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。可用性に基づいて決定すると、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では、実行しているインスタンスに応じて、コンピューティングまたはデータベース容量に対して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。長期契約や前払いは不要です。

Savings Plans は、1 年または 3 年の期間で一定の使用量 (USD/時間、で計算) をコミットメントする代わりに、Amazon EC2、Lambda、AWS Fargate を低料金で利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、予備のコンピューティング容量を時間単価の割引価格 (オンデマンド価格の最大 90% オフ) でリクエストできる Amazon EC2 の料金の仕組みです。前払いのコミットメントはありません。

リザーブドインスタンスでは、容量に対する前払いにより、最大 75% の割引を受けることができます。詳細については、「予約によるコストの最適化」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。また、サンドボックスリソースは必要なときにのみ電源が入るため、その環境内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、Compute 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、Saving Plans による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ている場合、あるいは組織内でコスト配分を実施している場合、今後も当面の間は Amazon EC2 リザーブドインスタンスをご利用いただけま

す。ただし、より柔軟にコストを節約するため、いずれは Savings Plans を使用する戦略に切り替えることが推奨されます。AWS Cost Management の Savings Plans (SP) レコメンデーションは、更新することでいつでも新しい 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) を使用すると、Amazon RDS、Amazon Redshift、Amazon ElastiCache、Amazon OpenSearch Service のコストを削減できます。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、定期的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、フォールトトレラントで柔軟性がある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) にオフにできるかを分析します。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% 以上のコストを削減できます。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。インスタンスをリザーブドキャパシティで実行しているかどうかにかかわらず、オンデマンドの場合と同等の料金がキャパシティ予約に課金されます。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 は不要に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化を調べます。短期間のインスタンスはスポットインスタンスまたはスポットフリートの候補です。
  - Well-Architected ラボ: Cost Explorer
  - Well-Architected ラボ: コストの可視化
- 既存の料金契約を見直す: 現在の契約やコミットメントを長期間のニーズの観点から見直します。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、ニーズに最

適な契約を調整できます。長期コミットメントの場合は、リザーブド料金割引、特定のインスタンスタイプに対するリザーブドインスタンスまたは Savings Plans、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するには、Well-Architected ラボに従ってください。

リソース

関連ドキュメント:

- 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

関連する例:

- Well-Architected ラボ: Cost Explorer
- Well-Architected ラボ: コストの可視化
- Well-Architected ラボ: 料金モデル

COST07-BP02 コストに基づいてリージョンを選択する

リソースの料金は各リージョンで異なる場合があります。リージョン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、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

AWS クラウドインフラストラクチャはグローバルで、世界中の複数の場所でホストされ、AWS リージョン、アベイラビリティーゾーン、ローカルゾーン、AWS Outpost、Wavelength Zone 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ています。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。AWS 見積りツールを使用して、ロケーションタイプ (リージョン、Wavelength ゾーン、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安価な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 アジアパシフィック (シドニー)(ap-southwest-2) 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 インスタンスを米国東部 (バージニア北部)(us-east-1) 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表

前述のマトリックス表から、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 6 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ときは、データ転送コストを考慮します。データは顧客とリソースの近くに置いてください。データ転送が最小限でデータの流れがよい、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応じて、Amazon CloudFront、AWS PrivateLink、AWS Direct Connect、AWS Virtual Private Network を使用することで、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 の料金
- インスタンス購入オプション
- リージョン表

関連動画:

- Save up to 90% and run production workloads on Spot

関連する例:

- 一般的なアーキテクチャでのデータ転送コストの概要
- グローバルデプロイにおけるコストの考慮事項
- 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: リージョンごとにサービスの使用を制限する (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ルする契約と料金を選択します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

クラウド環境のコスト管理に役立つさまざまな製品が流通しています。こうした製品は、ターゲットとなる顧客の要件に応じて、コストガバナンスやコスト可視化を重視したものや、コスト最適化を重視したものなど、機能面で違いが見られる場合があります。効果的なコスト最適化とガバナンスの鍵を握る要因の 1 つは、料金モデルが適正で、必要な機能が揃った適切なツールを使用することです。製品ごとに料金モデルは異なります。毎月の請求総額の一定の割合を支払うものもあれば、実際の節約額の一定の割合を支払うものもあります。必要な分だけ支払う従量課金制が理想的です。

クラウドでサードパーティーのソリューションやサービスを利用する場合は、期待する成果に合わせて料金体系を選ぶことが重要です。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。例えば、実際に節約できたコストの一部を支払う成功報酬型ソフトウェアの場合、節約率 (成果) が上がるほど、請求額も高くなります。支出負担が増えるに従い、支払う金額も増えるライセンス契約は、コスト最適化の点で必ずしも最適解とは限りません。ただし、請求書のあらゆる項目で優遇を受けられる場合、こうした変動料金が妥当なケースもあるかもしれません。

例えば、Amazon EC2 のレコメンデーションを提供し、請求総額の一定割合を課金するソリューションでは、優遇なしの他のサービスを利用すると、割高になる場合があります。もう 1 つの例は、管理対象となるリソースのコストを一定の割合で支払うマネージドサービスです。インスタンスサイズが大きくなっても必ずしも管理の負担が増えるわけではありませんが、請求額は高くなります。こうしたサービス料金設定に、コスト最適化のプログラムや、効率を向上するサービス機能が含まれていることを確認してください。

顧客は市場に流通しているこれらの製品を比較的高度である、または使いやすいと感じるかもしれません。こうした製品のコストを考慮し、長期的に見てコスト最適化の可能性があるかどうかを考える必要があります。

実装手順

- サードパーティーの契約と条件を分析する: サードパーティーの契約における料金を確認します。さまざまな使用レベルに応じ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

関連ドキュメント:

- リザーブドインスタンスのレコメンデーションへのアクセス
- インスタンス購入オプション

関連動画:

- Save up to 90% and run production workloads on Spot

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、リザーブドキャパシティに対して長時間稼働しない、中断することのできない短期ワークロードに対してのみ使用します (リソースタイプに応じて、期間の 25% から 75%)。

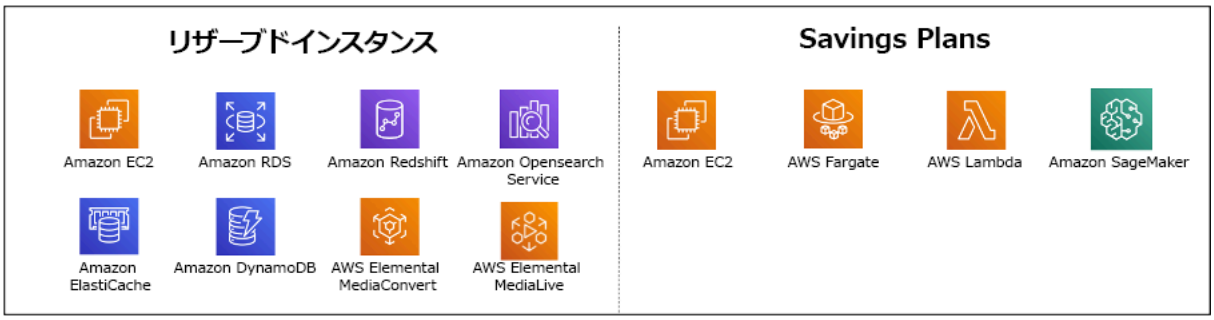
このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

コスト効率を上げるために、AWS は過去の使用状況に基づいて確約利用 (コミットメント) のレコメンデーションをいくつか提示します。これらのレコメンデーションを参考にして、実際に節約できるコストと、そのコミットメントの活用法を理解できます。これらのサービスをオンデマンドまたはスポットで利用することも、一定期間の利用を確約してリザーブドインスタンス (RI) や Savings Plans (SP) でオンデマンドコストを削減することもできます。ワークロードを最適化するには、各ワークロードコンポーネントや複数の AWS サービスだけでなく、該当するサービスのコミットメント割引、購入オプション、スポットインスタンスについても理解する必要があります。

ワークロードのコンポーネントの要件を検討し、これらのサービスのさまざまな料金モデルを理解しましょう。これらの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、経時的に必要となるワークロード要件を確認します。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。

例えば、AWS のこのウェブアプリケーションアーキテクチャを検討してみましょう。このワークロードのサンプルは、Amazon Route 53、AWS WAF、Amazon CloudFront、Amazon EC2 インスタンス、Amazon RDS インスタンス、ロードバランサー、Amazon S3 ストレージ、Amazon Elastic File System (Amazon EFS) など、複数の AWS サービスで構成されています。これらのサービスをそれぞれ見直し、さまざまな料金モデルでコストをどれくらい削減できるのかを確認する必要があります。RI または SP を利用できるものもあれば、オンデマンドでしか利用できないものもあります。次の図からわかるように、AWS の一部のサービスは利用を確約し、RI または SP を使用できます。



リザーブドインスタンスと Savings Plans を使用してコミットされた AWS サービス

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 またはリザーブドインスタンスの購入、スポットインスタンスの実装を行います。コミットメントの初回購入時には、リストの上位 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。このプロセスは AWS Cost Management Console が案内してくれます。コンソールから RI または SP のレコメンデーションを確認し、その内容 (タイプ、支払い、期間) をカスタマイズし、時間単位の確約利用料 (1 時間あたり 20 USD など) を確認して、カートに追加します。割引は、対象となる使用量に自動的に適用されます。コミットメント割引で定期的に少量を購入します (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。最後に、Amazon EC2 オンデマンドインスタンスを選択し、残りの要件にリソースを割り当てます。
- 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カバレッジを達成したら、部分的に (数か月ごと)、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

関連ドキュメント:

- Understanding your Savings Plans recommendations
- 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
- 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
- AWS の他のサービスの予約モデル
- Savings Plans Supported Services

関連動画:

- Save up to 90% and run production workloads on Spot

関連する例:

- Savings Plans を購入する前に考慮すべきことは何ですか?
- Cost Explorer で使用率とコストを分析する方法を教えてください。

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全体を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る場合があります。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。

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 (RI) または Savings Plans (SP) を持つ AWS 組織内のすべてのアカウントの使用量を考慮して計算されます。また、割引共有が有効になったときに計算され、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推奨事項は、独立した各アカウントでの削減を最大化するために、個人アカウントレベルで計算されます。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。

1. ゾーン RI
2. 標準 RI

3. コンバーティブル RI
4. Instance Savings Plan
5. Compute Savings Plan

管理アカウントレベルで SP を購入した場合、割引率の高い順に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に節約を適用します。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、まず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。これは共有が有効になっている場合にのみ発生します。Savings Plans は、削減率が最も高いものがまず適用されます。削減率が等しい使用が複数ある場合、Savings Plans は、Savings Plans の割合が最も低い使用にまず適用されます。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。残りの使用はオンデマンド価格で課金されます。AWS コスト管理の 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つけます。

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、月次費用の見積もりと月次節約額の見積もりを行っていることを確認します。管理アカウントレベルでのレコメンデーションを確認します。レコメンデーションは、アカウント間で最大限の節約を実現するために、RI または Savings Plans の割引共有が有効になっている AWS 組織内におけるすべてのメンバーアカウントの使用量を考慮して計算されます。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

関連ドキュメント:

- AWS の料金のしく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
- Savings Plans の概要
- 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
- Saving Plans 推奨事項を理解する
- AWS の使用に Savings Plans が適用される仕組み
- 一括請求を利用した Saving 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

関連する例:

- AWS Well-Architected ラボ: 料金モデル (レベル 200)
- AWS Well-Architected ラボ: 料金モデルの分析 (レベル 200)
- Savings Plans を購入する前に考慮すべきことは何ですか?
- 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
- スポットインスタンスを使用するタイミング

COST 8. データ転送料金はどのように計画するのですか?

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、時間と共に運用コストを大幅に削減できます。

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

クラウドでソリューションを設計する際、習慣的にオンプレミスのデータセンターを使用してアーキテクチャを設計してしまったり、知識が不足していたりするせいで、データ転送料金を見落としがちです。AWS のデータ転送料金は、送信元、送信先、トラフィック量によって決まります。設計段階からこれらの料金を織り込めば、コスト削減につながる可能性があります。総保有コスト (TCO) を正確に見積もるには、ワークロードにおけるデータ転送の発生箇所、転送コスト、関連するメリットを把握することがきわめて重要です。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。例えば、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。

ワークロードでデータを転送するサービスコンポーネントをモデリングし、これが、求められる信頼性と耐障害性を実現するために許容されるコストであるか (両方のアベイラビリティーゾーンのコンピューティングとストレージに支払うのと同様であるか) を判断します。さまざまな使用量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的に変化します。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。

データ転送をモデリングする際は、取り込まれるデータの量と転送元を考慮します。また、処理されるデータ量と、必要なストレージやコンピューティングのキャパシティについても検討してください。モデリング中は、ワークロードのアーキテクチャに則したネットワークのベストプラクティスに従い、見込まれるデータ転送コストを最適化します。

AWS Pricing Calculator を使用して、特定の AWS サービスのコストの見積りと、予想されるデータ転送を確認できます。ワークロードを (テスト目的で、または実稼働前の環境で) 既に実行している場合は、AWS Cost Explorer または AWS Cost and Usage Report (CUR) を使用してデータ転送コストを把握し、モデル化します。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

- 要件を特定する: 送信元と送信先の間で予定されているデータ転送の、主な目標とビジネス要件は何ですか? 最終的にどのようなビジネス成果を期待していますか? ビジネス要件を収集し、期待される成果を定義します。

- 送信元と送信先を特定する: データ転送の送信元と送信先はどこですか (AWS リージョン内の転送、AWS サービスへの転送、インターネットへの転送など)?
  - AWS リージョン内のデータ転送
  - AWS リージョン 間のデータ転送
  - インターネットへのデータ転送
- データ分類を特定する: 転送されるデータはどのように分類されますか? データの種類は? データの大きさは? データ転送の頻度は? 機密データですか?
- 使用する AWS サービスまたはツールを特定する: このデータ転送にはどの AWS サービスを使用しますか? プロビジョニング済みのサービスを別のワークロードに使用できますか?
- データ転送コストを計算する: 以前に作成したデータ転送モデリングの AWS 料金を使用して、ワークロードのデータ転送コストを計算します。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、アベイラビリティーゾーン間の転送であれば冗長性のためかもしれません。
- データ転送モデルを作成する: すべての情報を収集したら、複数のユースケースやさまざまなワークロードの基準となる、データ転送の概念モデルを作成します。

リソース

関連ドキュメント:

- AWS キャッシュソリューション
- AWS の料金
- 「Amazon EC2 の料金」
- Amazon VPC の料金
- Understanding data transfer charges

関連動画:

- Monitoring and Optimizing Your Data Transfer Costs
- S3 Transfer Acceleration

関連する例:

- 一般的なアーキテクチャでのデータ転送コストの概要
- AWS 規範ガイダンス

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を選択し、データ転送コストを低減するようにアーキテクチャを設計します。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ーゾーン (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: 中

実装のガイダンス

データ転送を念頭に置いたアーキテクチャでは、データ転送コストを最小限に抑えることができます。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を使用したりする場合があります。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。

AWS クラウドとの間やその内部でデータを転送する場合、データ転送を最適化する適切な AWS サービスを選択するために、さまざまなユースケース、データの性質、利用可能なネットワークリソースに基づいて転送先を把握することが不可欠です。AWS は、多様なデータ移行要件に対応する幅広いデータ転送サービスを提供しています。組織内のビジネスニーズに基づいて、適切なデータストレージとデータ転送オプションを選択します。

ワークロードアーキテクチャを計画または確認するときは、次の点を考慮してください。

- AWS 内の VPC エンドポイントを使用する: VPCエンドポイントにより、VPC とサポートされている AWS サービス間のプライベート接続が可能になります。これにより、データ転送コストが発生する可能性のある公開インターネットの使用を回避できます。
- NAT ゲートウェイを使用する: NAT ゲートウェイを使用して、プライベートサブネット内のインスタンスがインターネットまたは VPC 外のサービスに接続できるようにします。NAT ゲートウェイの背後にあるリソースのうち、最大量のトラフィックを送信しているリソースのアベイラビリティーゾーンが NAT ゲートウェイと同じかどうかを確認します。違う場合は、そのリソースと同じアベイラビリティーゾーンに新しい NAT ゲートウェイを作成し、AZ 間のデータ転送料金を削減します。

- AWS Direct Connect を使用する: AWS Direct Connect は公開インターネットをバイパスして、オンプレミスネットワークと AWS との間にプライベート接続を直接確立します。インターネットを介して大量のデータを転送するよりも、この方が高コスト効率で確実です。
- リージョンの境界をまたぐデータ転送は回避する: 通常、(特定のリージョンから別のリージョンへの) AWS リージョン 間のデータ転送には、料金が発生します。複数のリージョンをまたいで転送する場合は、慎重に検討したうえで決断してください。詳細については、「複数リージョンのシナリオ」を参照してください。
- データ転送をモニタリングする: Amazon CloudWatch と VPC フローログを使用して、データ転送とネットワークの使用状況に関する詳細情報をキャプチャします。VPC 内でネットワークインターフェイスとの間を行き来するネットワークトラフィックについて、IP アドレスや範囲などの、キャプチャされた情報を分析します。
- ネットワーク使用状況を分析する: AWS Cost Explorer、CUDOS Dashboard、CloudWatch など、測定とレポートのためのツールを使用して、ワークロードのデータ転送コストを把握します。

実装手順

- データ転送用のコンポーネントを選択する: COST08-BP01 データ転送モデリングを実行する で説明した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 (またはコストを削減) 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-BP03 データ転送コストを削減するサービスを実装する

関連ドキュメント:

- クラウドへのデータ移行
- AWS キャッシュソリューション
- Amazon CloudFront でコンテンツ提供を高速化する

関連する例:

- 一般的なアーキテクチャでのデータ転送コストの概要
- AWS Network Optimization Tips
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format

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、エッジロケーションや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、アプリケーションサーバーまたはデータベースの前にキャッシュレイヤーを構築する、クラウドへの接続に VPN ではなく専用ネットワーク接続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: 中

実装のガイダンス

ネットワークデータ転送の使用量を最適化するのに役立つさまざまな AWS サービスがあります。ワークロードのコンポーネント、種類、クラウドアーキテクチャにもよりますが、これらのサービスはクラウド上でのトラフィックの圧縮、キャッシュ、共有、分散に役立ちます。

- Amazon CloudFront は、低レイテンシーかつ高速の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 を使用することで、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。経時的に使用量を増やす予定がある場合、Security Savings Bundle を使用すると、CloudFront の使用量を最大 30% 節約できます。
- AWS Direct Connect により、AWS への専用ネットワーク接続を確立できます。このサービスにより、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実現します。
- AWS VPN を使用すると、プライベートネットワークと AWS グローバルネットワークとの間に安全なプライベート接続を確立できます。シンプル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た AWS 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 S3 と Amazon DynamoDB がサポートされています。インターフェイス VPC エンドポイントは AWS PrivateLink により提供され、時間単位の料金と GB あたりの使用料が発生します。

- NAT ゲートウェイにはスケーリングと管理機能が組み込まれており、スタンドアロンの NAT インスタンスとは異なりコストを節約できます。NAT ゲートウェイはトラフィックの多いインスタンスと同じアベイラビリティーゾーンに配置し、Amazon DynamoDB または Amazon S3 にアクセスが必要なインスタンスでは VPC エンドポイントを使用して、データ転送とデータ処理コストを削減することを検討してください。
- エッジでデータを収集および処理するときは、コンピューティングリソースを備えた AWS Snow Family デバイスを使用します。AWS Snow Family デバイス (Snowcone、Snowball、Snowmobile) を使用すると、ペタバイト規模のデータをコスト効率よく、オフラインで AWS クラウドに移動できます。

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、VPC フローログを確認して、サービスとワークロードタイプに基づいて適切な AWS ネットワークサービスを選択します。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

関連ドキュメント:

- AWS Direct Connect
- AWS の製品を見る
- AWS キャッシュソリューション
- Amazon CloudFront
- AWS Snow Family
- Amazon CloudFront Security Savings Bundle

関連動画:

- Monitoring and Optimizing Your Data Transfer Costs
- AWS コスト最適化シリーズ: CloudFront
- NAT ゲートウェイのデータ転送料金を削減するにはどうすればよいですか?

関連する例:

- 共有サービスのチャージバック方法: An AWS Transit Gateway の例
- Athena クエリと QuickSight を使用してコストと使用状況レポートから AWS データ転送の詳細を深く理解する
- 一般的なアーキテクチャでのデータ転送コストの概要
- AWS Cost Explorer でデータ転送コストを分析する
- Amazon CloudFront の各種機能で AWS アーキテクチャのコストを最適化する
- NAT ゲートウェイのデータ転送料金を削減するにはどうすればよいですか?

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、リソースを供給しますか?

COST 9. どのように需要を管理し、リソースを供給しますか?

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、費用をかけたすべてのものが活用されるようにして、使用率が著しく低いインスタンスの発生を回避します。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-BP02 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

クラウドコンピューティングのワークロード需要を分析するには、クラウド環境で開始されるコンピューティングタスクのパターンと特性を理解する必要があります。この分析により、ユーザーはリソース割り当てを最適化し、コストを管理し、パフォーマンスが必要なレベルを満たしていることを確認することができます。

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を変更する必要があ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るようにします。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。

クラウドコンピューティングのワークロード需要分析を行う際に考慮すべき重要な点は、次のとおりです。

1. リソースの使用状況とパフォーマンメトリクス: AWS リソースの使用状況を経時的に分析します。ピーク時とオフピーク時の使用パターンを特定して、リソースの割り当てとスケーリング戦略を最適化します。応答時間、レイテンシー、スループット、エラー率などのパフォーマンスメトリクスをモニタリングします。これらのメトリクスは、クラウドインフラストラクチャの全体的な状態と効率を評価するのに役立ちます。
2. ユーザーとアプリケーションのスケーリング動作: ユーザーの行動と、その行動がワークロードの需要にどのように影響するかを理解します。ユーザートラフィックのパターンを調べることは、コンテンツの配信とアプリケーションの応答性を向上させるうえで役立ちます。需要の増加に伴ってワークロードがどのようにスケールするかを分析します。ロードの変動に対応するために、自動スケーリングパラメータが適切かつ効果的に設定されているかどうかを判断します。
3. ワークロードのタイプ: バッチ処理、リアルタイムデータ処理、ウェブアプリケーション、データベース、機械学習など、クラウドで実行されているさまざまなタイプのワークロードを特定します。ワークロードのタイプごとに、リソース要件とパフォーマンスプロファイルが異なる場合があります。
4. サービスレベルアグリーメント (SLA): 実際のパフォーマンスを SLA と比較して、コンプライアンスを確保し、改善が必要な領域を特定します。

Amazon CloudWatch を使用して、メトリクスの収集とトラッキング、ログファイルの収集とモニタリング、アラームの設定、AWS リソースの変更への自動対応を行うことができます。Amazon CloudWatch を使用して、リソースの使用率、アプリケーションのパフォーマンス、運用の状況をシステム全体で把握できます。

AWS Trusted Advisor では、ベストプラクティスに従ってリソースをプロビジョニングすることで、システムのパフォーマンスと信頼性を向上させ、セキュリティを強化し、コスト節減の機会を探すことができます。また、本番環境以外のインスタンスをオフにして、需要の増減に合わせて Amazon CloudWatch や自動スケーリングを使用することもできます。

最後に、AWS Cost Explorer または Amazon QuickSight を AWS Cost and Usage Report (CUR) ファイルまたはアプリケーションログと共に使用して、ワークロード需要の高度な分析を実行できます。

全体的には、包括的なワークロード需要分析により、組織はリソースのプロビジョニング、スケーリング、最適化について情報に基づいた意思決定が可能になり、パフォーマンス、コスト効率、ユーザー満足度の向上につながります。

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。Amazon CloudWatch、ログファイルとモニタリングデータを使用して、ワークロードの使用状況についてインサイトを得ます。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。
- 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

関連ドキュメント:

- Amazon CloudWatch
- AWS Trusted Advisor

- AWS X-Ray
- AWS Auto Scaling
- AWS での Instance Scheduler
- Amazon SQS の開始方法
- AWS Cost Explorer
- Amazon QuickSight

関連動画:

関連する例:

- コスト最適化のためのモニタリング、追跡、分析
- CloudWatch でのログの検索および分析

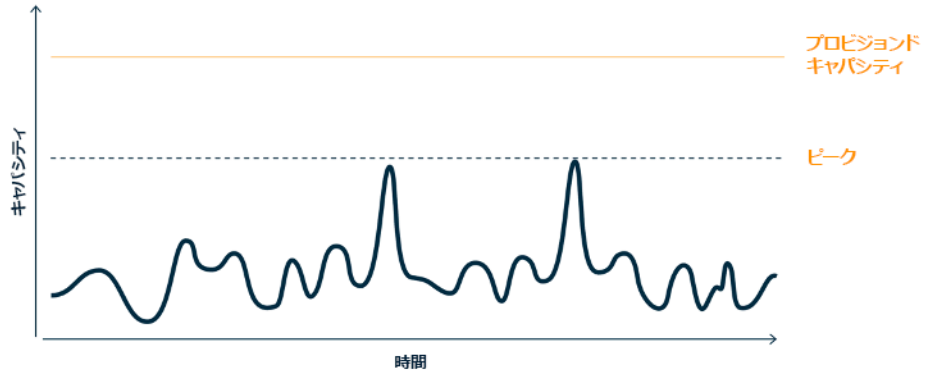
COST09-BP02 需要を管理するためのバッファまたはスロットルを実装する

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。バッファリングを実装して、リクエストを保存し、処理を延期できます。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: 中

実装のガイダンス

クラウドコンピューティングでは、需要を管理し、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するために、バッファまたはスロットリングの実装が不可欠です。パフォーマンスを最適化するには、ピークを含む総需要、リクエストの変化のペース、必要な応答時間を測定することが重要です。クライアントにリクエストの再送機能がある場合は、スロットリングの適用が現実的です。逆に、クライアントに再試行の機能がなければ、バッファソリューションの実装が理想的なアプローチです。バッファは、入ってくるリクエストの交通整理を行い、動作速度がさまざまに異なるアプリケーションとの通信を最適化します。



高いプロビジョンドキャパシティを必要とする 2 つの異なるピークの需要曲線

上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。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。ピークをならすには、スロットリングまたはバッファリングのソリューションの実装を検討してください。

理解を深めるために、スロットリングとバッファリングについて見ていきましょう。

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。スロットリングでは、その時点で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需要側に通知されます。需要側は一定時間待ってから、リクエストを再試行します。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。

バッファベース: バッファベースのアプローチでは、プロデューサー (キューにメッセージを送信するコンポーネント)、コンシューマー (キューからメッセージを受信するコンポーネント)、およびキュー (メッセージを保持) を使用してメッセージを保存します。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。バッファを中心にした方法を採用することで、プロデューサーが送信したメッセージはキューまたはストリームに蓄えられ、コンシューマーがそれぞれの運用上の需要に応じたペースでアクセスできるようになります。

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、単独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。

バッファリングとスロットリングは、ワークロードの需要を変化させ、ピークを滑らかにします。クライアントがアクションを再試行する場合はスロットリングを使用し、リクエストを保留して後で処理する場合はバッファリングを使用します。バッファベースのアプローチを採用する場合は、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。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。

実装手順

- クライアント要件を分析する: 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スロットリングを提供できます。

リソース

関連するベストプラクティス:

- 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
- REL05-BP02 リクエストのスロットル

関連ドキュメント:

- AWS Auto Scaling
- AWS での Instance Scheduler
- Amazon API Gateway
- Amazon Simple Queue Service
- Amazon SQS の開始方法
- Amazon Kinesis

関連動画:

- 分散アプリに適したメッセージングサービスの選択

関連する例:

- ワークロードでの API スロットリングの管理とモニタリング
- API Gateway を使用した階層型マルチテナント REST API を大規模にスロットリングする
- Amazon API Gateway を使用したマルチテナント Amazon EKS SaaS ソリューションで階層化とスロットリングを有効にする
- キューとメッセージを使用したアプリケーション統合

COST09-BP03 リソースを動的に供給する

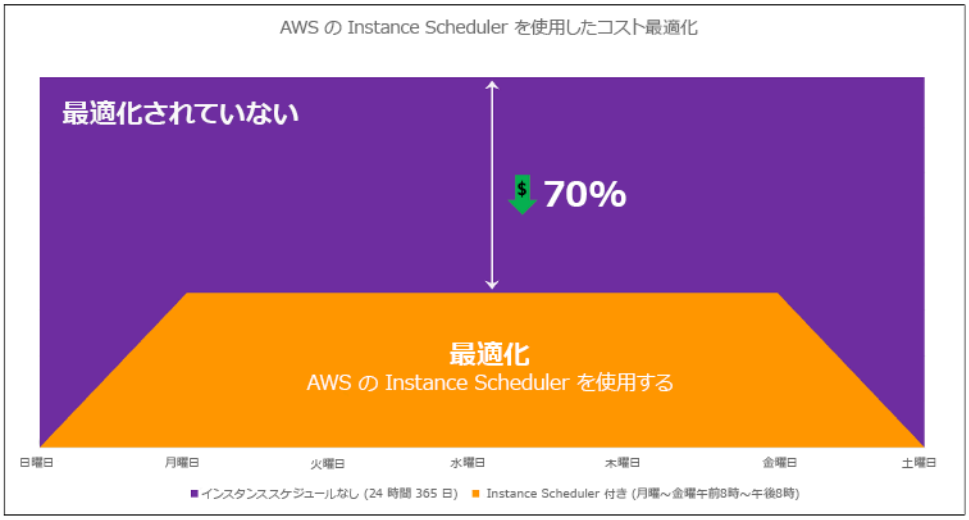
リソースを計画的にプロビジョニングします。これは、自動スケーリングなどの需要ベース、または需要が予測可能でリソースが時間に基づいて提供される時間ベースで行います。これらの手法を使用すると、過剰プロビジョニングやプロビジョニング不足を最小限に抑えることができます。

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。その 1 つが AWS Instance Scheduler を使用した方法です。これにより、Amazon Elastic Compute Cloud (Amazon EC2) と Amazon Relational Database Service (Amazon RDS) インスタンスの起動および停止を自動化します。もう 1 つの方法は 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ルできます。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 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 Quick Setup 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、または Amazon Redshift や Amazon OpenSearch Service などのサービスを管理するインスタンスを停止/開始することはできません。Auto Scaling グループには、グループ内のインスタンスに対して独自のスケジューリングがあり、このスケジュールに基づいて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 インスタンスおよびスポットフリート、Amazon ECS、Amazon DynamoDB、Amazon Aurora と統合するアプリケーションの容量をスケールするためのフルマネージドで無料のサービスです。自動スケーリングでは、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。

- 現在のインスタンスレベルの常時維持
- 手動でスケールする
- スケジュールに基づくスケーリング
- 需要に基づくスケーリング

- 予測スケーリングの使用

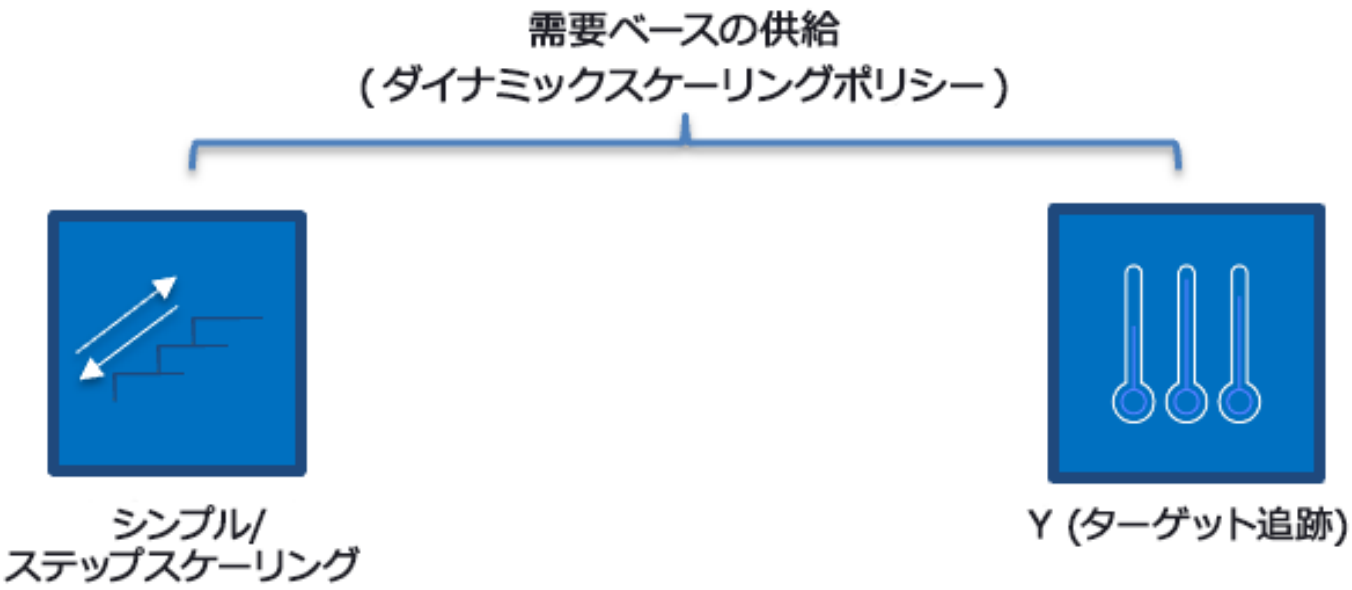
自動スケーリングポリシーは異なり、動的スケーリングポリシーとスケジュールスケーリングポリシーに分類できます。動的ポリシーには、手動または動的スケーリング、スケジュールスケーリングまたは予測スケーリングがあります。スケーリングポリシーは、動的スケーリング、スケジュールスケーリング、予測スケーリングに使用できます。Amazon CloudWatch のメトリクスとアラームを使用して、ワークロードのスケーリングイベントをトリガーすることもできます。最新の機能や改善点にアクセスできるように、起動テンプレートを使用することをお勧めします。起動設定を使用する場合、すべての自動スケーリング機能を使用できるわけではありません。たとえば、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 Auto Scaling グループや、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ン管理では、パラメータのフルセットのサブセットを作成できます。その後、再使用して、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 でコードにスケーリングを実装できます。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させることでスケーリングに役立ちます。ASG と Elastic Load Balancing を使用して、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、自動スケーリング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をスケールしたり、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したりできます。



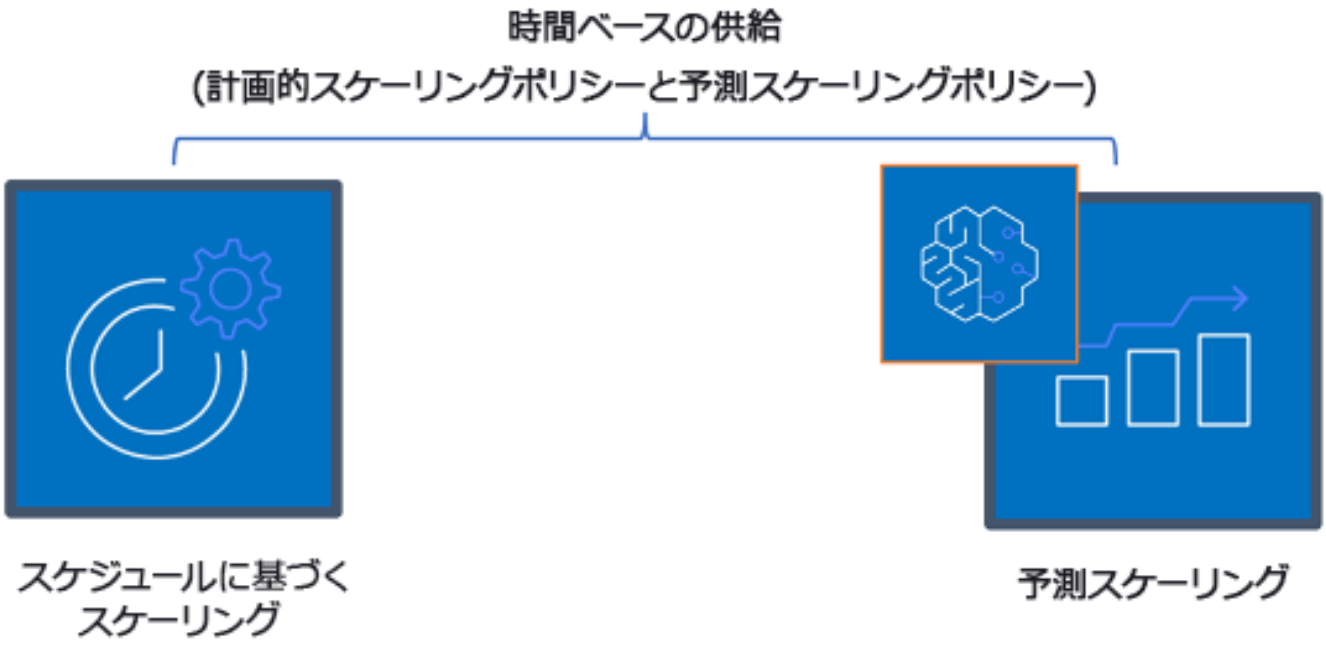
需要ベースの動的スケーリングポリシー

- 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、ユーザ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、メトリクスをユーザ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自動スケーリングまたは予測自動スケーリングを使用して、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。また、Auto Scaling グループ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。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 Fleet 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。

AWS API と SDK と AWS CloudFormation を活用して、必要に応じて環境全体を自動でプロビジョニングおよび廃止することもできます。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。第 1 に、使用パターンの一貫性について、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

- スケジュール済みのスケーリングを設定する: 需要の変化を予測できるため、時間ベースのスケーリングは適切な数のリソースを適時に提供できます。また、リソースの作成と設定が、需要の変化に対応するのに十分ではない場合にも役立ちます。ワークロード分析を活用して、AWS Auto Scaling を使用してスケジュールに基づくスケーリングを設定します。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して、トラフィックフローの日次および週次のパターンに先立って Auto Scaling グループ内の Amazon EC2 インスタンスの数を増やします。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、自動スケーリングを使用します。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自動スケーリングを設定し、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てオートスケールできます。スポットインスタンスの使用で割引を受けるだけでなく、リザーブドインスタンスまたは Savings Plan を使用して、通常のオンデマンドインスタンス コストの割引料金を受け取ることができます。これらのすべての要素を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つつ、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得ることができます。

リソース

関連ドキュメント:

- 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
- Amazon EC2 Auto Scaling の使用を開始する
- 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:

- 自動スケーリングのターゲットトラッキングスケーリングポリシー
- AWS での Instance Scheduler

関連する例:

- Amazon EC2 Fleet 用自動スケーリングでの属性ベースのインスタンスタイプ選択
- スケジュールされたスケーリングを使用して Amazon Elastic Container Service を最適化しコストを削減する
- Amazon EC2 Auto Scaling での予測スケーリング
- AWS 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

Questions

- COST 10. どのように新しいサービスを評価するのですか?
- COST 11. 労力コストを評価する方法

COST 10. どのように新しいサービスを評価するのですか?

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。

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を活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードの費用対効果を最大にするには、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。例えば、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。例えば、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と信頼性の柱のレビュープロセスと同様に、最適化および改善のアクティビティを特定、検証、優先順位付けし、これを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 サポートのプロアクティブサービス
- SAP ワークロードの定期的なレビューを計画する

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レ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 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」では、AWS の新機能について詳しく説明し、リリースされた AWS サービス、機能、リージョン拡大に関して概説します。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。他のお客様のアーキテクチャ設計、課題、ソリューションについては、AWS の「This is My Architecture」ビデオをご覧ください。「All-In series」で、実際の AWS サービスとカスタマーストーリーをご覧ください。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、調査、および分類する「Back to Basics」ビデオシリーズを視聴することもできます。もう 1 つのソースは、「How to Build This」動画です。この動画は、AWS サービスを使用して実用最小限の製品 (MVP) を実現する方法について、大きな構

想を持つユーザー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、「入門ガイド」の資料を参照できます。ステップバイステップのチュートリアルが含まれています。

レビュープロセスを開始する前に、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。
- 新しいサービスを実装する: 分析の結果、変更を実施する場合は、まずワークロードのベースラインを実行し、各アウトプットの現在のコストを把握します。変更を実施し、分析を実行して、各アウトプットの新しいコストを確認します。

リソース

関連ドキュメント:

- AWS ニュースブログ
- AWS の最新情報
- AWS ドキュメント
- AWS 使用開始
- AWS 一般的なリソース

関連動画:

- AWS - This is My Architecture
- AWS - Back to Basics
- AWS - All-In シリーズ
- 構築する方法

COST 11. 労力コストを評価する方法

ベストプラクティス

- COST11-BP01 運用のオートメーションを実行する

COST11-BP01 運用のオートメーションを実行する

管理タスク、デプロイ、人的エラーのリスク低減、コンプライアンス、運用において、オートメーションによって達成可能な時間と労力の節約を数値化することに重点を置いて、クラウドの運用コストを評価します。運用作業に必要な時間と関連コストを評価し、管理タスクを自動化することで、可能な限り手作業を最小限に抑えます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

運用を自動化することで、手動タスクの頻度が減り、効率が高まると共に、ワークロードのデプロイ、管理、運用において安定した信頼性の高いエクスペリエンスを提供できるため顧客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、リソース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。そのソリューションは、安全で、高速で、費用対効果が高く、リスクを最小限に抑え、最大限の信頼性を備えている必要があります。

運用コスト全体を確認して、必要な労力に基づいて運用作業に優先順位を付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。反復的で価値が高く、時間のかかる複雑なアクティビティの自動化を優先します。通常、高価値で、人的エラーの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 (運用チームの追加作業時間など) の発生につながることが多いためです。

AWS Systems Manager または AWS Config などのオートメーションツールを使用して、運用、コンプライアンス、モニタリング、ライフサイクル、終了のプロセスを合理化します。AWS のサービス、ツール、およびサードパーティー製品を使用して、実装するオートメーションを特定の要件に合

わせてカスタマイズできます。次の表は、AWS のサービスを使用して管理と運用を自動化することで実現できる主な運用の機能の一部を示しています。

- AWS Audit Manager: AWS の使用状況を継続的に監査して、リスクとコンプライアンスの評価を簡素化する
- AWS Backup: データ保護を一元的に管理し自動化します。
- AWS Config: コンピューティングリソース、評価、監査、設定の評価、リソースインベントリを設定します。
- AWS CloudFormation: Infrastructure as Code を使用して高可用性リソースを起動します。
- AWS CloudTrail: IT の変更管理、コンプライアンス、制御。
- Amazon EventBridge は、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: イベントによりトリガーするか、AWS EventBridge で固定スケジュールにより実行して、反復的な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、オペレーティングシステムへのパッチ適用、設定の自動化、継続的な管理。
- 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: テンプレートの使用、コンプライアンスと制御を備えた Infrastructure as Code。

AWS の製品やサービスを使用してすぐ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は、AWS Managed Services (AMS)、AWS プロフェッショナルサービス、AWS パートナーのいずれかにご連絡いただければ、オートメーションの導入数を増やしクラウドでのオペレーショナルエクセレンスを高めることができます。

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、オンゴーイングの管理コストを削減できます。

また、AWS プロフェッショナルサービスは、AWS を使用して目的のビジネス成果を達成し、運用を自動化できるようサポートします。自動化された堅牢かつ俊敏な IT 運用と、クラウドに最適化されたガバナンス機能のデプロイについてお客様を支援します。モニタリング例の詳細と推奨されるベストプラクティスについては、運用上の優秀性の柱についてのホワイトペーパーを参照してください。

実装手順

- 1 回の構築で多数のデプロイ: CloudFormation、AWS SDK、AWS CLI などの Infrastructure-as-code を使用して、1 回のデプロイで、同様の環境やディザスタリカバリシナリオ向けに何回も使用します。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変更、設定、デプロイの方法をガイドします。Service Catalog を使用することもできます。こちらを使用すると、承認済みの Infrastructure as Code テンプレートを作成し管理して AWS で使用でき、承認済みのセルフサービス型クラウドリソースを誰でも見つけることができます。
- 継続的なコンプライアンスを自動化する: 記録済みの設定を、事前定義された基準に照らして自動的に評価および修正することを検討します。AWS Organizations を AWS Config および AWS CloudFormation の機能と組み合わせることで、多数のメンバーアカウントの、設定コンプライアンスの大規模な管理および自動化を効率的に実行できます。設定の変更や、AWS リソース間の関係を確認して、リソース設定の履歴を詳しく調べることができます。
- モニタリングタスクを自動化する: AWS には、サービスのモニタリングに使用できるさまざまなツールが用意されています。これらのツールを設定して、モニタリングタスクを自動化できます。ワークロードのすべての部分からモニタリングデータを収集するモニタリング計画を作成して実装すると、マルチポイント障害が発生した場合のデバッグがより簡単になります。例えば、自動モニタリングツールを使用して、Amazon EC2 を観察し、システムステータスチェック、インスタンスステータスチェック、および Amazon CloudWatch アラームで問題が検出された場合に報告を受けることができます。
- メンテナンスとオペレーションを自動化する: 日常的なオペレーションを自動化して人による介入をなくします。AWS サービスとツールを使用して、実装する AWS オートメーションを選択し、特定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たり、AWS SSM を使用して EC2 インスタンスにパッチを適用したりするなどです。必要なアクションを AWS のサービスで実行できない場合、またはリソースのフィルタリングを行う、より複雑なアクションを必要とする場合は、AWS Command Line Interface (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 では、AWS のサービスの制御および管理プロセス全体を、スクリプトを使用して、AWS Management Consoleを使用せずに自動化することができます。AWS のサービスとやり取りする 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。
- 自動化により継続的なライフサイクルを構築する: 規制や冗長性のためだけでなく、コスト最適化を実現するためにも、確固たるライフサイクルポリシーを確立してこれを維持することが重要

です。AWS Backup を使用して、バケット、ボリューム、データベース、ファイルシステムなどのデータストアのデータ保護を一元的に管理および自動化できます。Amazon Data Lifecycle Manager を使用して、EBS スナップショットと EBS-backed AMI の作成、保持、削除を自動化することもできます。

- 不要なリソースを削除する: 未使用のリソースが、サンドボックスや開発用の AWS アカウントに溜まることがよくあります。開発者は通常の開発サイクルの中でさまざまなサービスやリソースを構築して実験し、不要になってもこうしたリソースを削除しないためです。未使用のリソースは、組織にとって不要なコスト、場合によっては高額なコストをもたらす場合があります。これらのリソースを削除することで、これらの環境の運用コストを削減できます。データが不要であることを確認し、不要かどうか不明な場合はバックアップ済みであることを確認します。AWS CloudFormation を使用してデプロイ済みのスタックをクリーンアップできます。これにより、テンプレートで定義されているリソースの大部分が自動的に削除されます。あるいは、aws-nuke のようなツールを使用すると、AWS リソースを削除するための自動化を作成することができます。

リソース

関連ドキュメント:

- AWS クラウドでのオペレーションのモダナイズ
- オートメーション向けの AWS サービス
- インフラストラクチャとオートメーション
- AWS Systems Manager Automation
- 自動モニタリングと手動モニタリング
- SAP 管理および運営のための AWS 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

関連動画:

- AWS での大規模な継続的コンプライアンスの自動化
- AWS Backup Demo: Cross-Account & Cross-Region Backup
- Patching for your Amazon EC2 Instances

関連する例:

- 自動運用の改革 (パート I)
- 自動運用の改革 (パート II)
- aws-nuke による AWS リソースの削除の自動化
- AWS Config および AWS SSM を使用して未使用の Amazon EBS ボリュームを削除
- AWS での大規模な継続的コンプライアンスの自動化
- AWS Lambda による IT オートメーション

持続可能性

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活用するメリット: ワークロードを Amazon 再生可能エネルギープロジェクトまたは公開されている炭素強度の低いリージョンの近くに配置すると、クラウドワークロードのカーボンフットプリントを減らすのに役立ちます。

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: 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がリージョンに備えられているかどうかを確認します。
  - AWS Pricing Calculator 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 (市場ベースおよびロケーションベースの方法) に基づいて、毎年のカーボンフットプリントを追跡して比較します。
- 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

関連ドキュメント:

- 炭素排出量の推定の理解
- 世界中の Amazon
- 再生可能エネルギーの方法論
- 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:

- AWS re:Invent 2023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

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。継続的に需要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さずに、容量を増加させたままにする。

このベストプラクティスを活用するメリット: ワークロードの伸縮性を設定およびテストすることで、需要とクラウドリソースの供給を効率的に一致させ、容量の過剰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で、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備えます。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。

需要は、一定の場合も変動する場合もあり、管理面の負担を軽減するには、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垂直方向に変更し (スケー

ルアップ/スケールダウン)、インスタンス数を水平方向に変更して (スケールイン/スケールアウト)、またはこれらの組み合わせで調整します。

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モニタリングし、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: 日次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: ネイティブにスケーリングするサービス (サーバーレスなど) を設計によって選択するか、機能として自動スケーリングを提供します。

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することで、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以下に示します。

自動スケーリングメカニズム	使用する場所
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使用できる Amazon EC2 インスタンスの数が正しいことを検証するために使用します。
Application Auto Scaling	Lambda 関数や Amazon Elastic Container Service (Amazon ECS) サービスなど、Amazon EC2 を超えた個々の AWS のサービスのリソースを自動的にスケールするために使用します。

自動スケーリングメカニズム	使用する場所
Kubernetes Cluster Autoscaler	AWS の Kubernetes クラスターを自動的にスケールするために使用します。

- スケーリングは、一般的に、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議論されます。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の設定を検討してください。
- 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種類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する場合は、100% の CPU 使用率が予想されるため、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、スケーリングポリシーにカスタマイズされたメトリクス (メモリ使用率など) を使用できます。適切なメトリクスを選ぶには、Amazon EC2 の以下のガイダンスを考慮してください。
  - メトリクスは有効な利用率メトリクスでなければならず、インスタンスのどの程度ビジーかを記述する必要があります。
  - メトリクス値は Auto Scaling グループ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の代わりに、動的スケーリングを使用します。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して、Auto Scaling グループのスケーリングアクティビティを確認できます。
- 予測可能なパターンについてワークロードを評価し、予測および計画された需要の変化を想定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 の予測スケーリング」を参照してください。

リソース

関連ドキュメント:

- 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
- Introducing Native Support for Predictive Scaling with Amazon EC2 Auto Scaling
- Karpenter の概要 - オープンソースの高性能 Kubernetes Cluster Autoscaler
- Amazon ECS クラスターの Auto Scaling を深く探る

関連動画:

- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment
- AWS re:Invent 2022 - Scaling containers from one user to millions
- AWS re:Invent 2023 - Scaling FM inference to hundreds of models with Amazon SageMaker
- AWS re:Invent 2023 - Harness the power of Karpenter to scale, optimize & upgrade Kubernetes

関連する例:

- Autoscaling

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、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と整合すると、ビジネスニーズを満たすと同時にリソース使用を最適化できます。

このベストプラクティスを活用しない場合のリスクレベル: 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かかわります。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

- 持続可能性の目標を理解する: 炭素削減やリソース使用率の向上など、組織内の持続可能性の目標を特定します。
- SLA の確認: SLA を評価して、ビジネス要件をサポートしているかどうかを評価します。SLA を超えている場合は、さらに見直しを行います。
- トレードオフを理解する: ワークロードの複雑さ (大量の同時接続ユーザーなど)、パフォーマンス (レイテンシーなど)、持続可能性への影響 (必要なリソースなど) のトレードオフを理解します。通常、2 つの要素に優先順位を付けると、3 つ目の要素が犠牲になります。
- SLA を調整する: 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: 可用性の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向上させると、環境への影響が大きくなる可能性があります。
  - 持続可能性とセキュリティ: ワークロードが過度に安全であれば、環境への影響が大きくなる可能性があります。
- 可能であれば持続可能性 SLA を定義する: ワークロードに持続可能性 SLA を含めます。例えば、コンピューティングインスタンスの持続可能性に関する SLA として最小の使用率レベルを定義します。
- 効率的な設計パターンを使用する: 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 AWS 上のマイクロサービスなどの設計パターンを使用します。
- 説明責任を伝達して確立する: 開発チームや顧客を含む関連するすべての関係者と SLA を共有します。レポートを使用して SLA を追跡およびモニタリングします。SLA の持続可能性目標を達成するための説明責任を割り当てます。

- インセンティブと報酬を使用する: インセンティブと報酬を使用して、持続可能性の目標に沿った SLA を達成または超過します。
- 見直しと反復: SLA を定期的に見直して調整し、進化する持続可能性とパフォーマンスの目標に合致していることを確認します。

リソース

関連ドキュメント:

- 回復性のパターンとトレードオフを理解して、効率的なクラウド設計を実現
- SaaS プロバイダ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:

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

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: 未使用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

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

- 棚卸しを実施する: 包括的な棚卸しを実施して、ワークロード内のすべてのアセットを特定します。
- 使用率を分析する: 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。
- 未使用のアセットを削除する: 不要になったアセットを削除する計画を立てます。
  - 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。
  - 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。
- サードパーティーに伝達する: 不要になったアセットをお客様に代わって管理しているサードパーティーに、アセットの生成と保存を止めるように指示します。冗長アセットを統合するように依頼します。
- ライフサイクルポリシーを使用する: ライフサイクルポリシーを使用して、未使用のアセットを自動的に削除します。
  - Amazon S3 ライフサイクルを使用すると、オブジェクトのライフサイクル全体を管理できます。
  - Amazon Data Lifecycle Manager を使用して、EBS スナップショットと Amazon EBS-backed AMI の作成、保持、削除を自動化できます。
- 確認と最適化をする: 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

関連ドキュメント:

- 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:

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Preserving and maximizing the value of digital media assets using Amazon S3
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

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などのロケーションオプション、そして AWS Outposts、AWS Local Zones などのエッジロケーションから構成されます。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。
  - Amazon CloudWatch や AWS CloudTrail などのモニタリングツールを使用して、ネットワークアクティビティに関するデータを収集します。
  - 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。
  - サステナビリティの目標: 「リージョンの選択」を参照してください。
  - データがある場所: 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。
  - ユーザーがいる場所: ユーザー向けアプリケーションの場合は、ワークロードのユーザーに近いリージョン (1 つまたは複数) を選択します。
  - その他の制約:「ワークロードのリージョンを選択する際の考慮事項」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。
- 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データに使用すると、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	どのようなときに使うか
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。

- 以下のように、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	どのようなときに使うか
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 Functions	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

関連ドキュメント:

- 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 III : ネットワーキング
- Amazon ElastiCache のドキュメント
- Amazon CloudFront とは何ですか?
- Amazon CloudFront の主な特徴
- AWS グローバルインフラストラクチャ
- AWS Local Zones および AWS Outposts などエッジ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

関連動画:

- Demystifying data transfer on AWS
- Scaling network performance on next-gen Amazon EC2 instances
- AWS Local Zones 解説動画
- AWS Outposts: 概要と機能紹介
- AWS re:Invent 2023 - エッジワークロードとオンプレミスワークロードの移行戦略
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020 - AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Amazon CloudFront で低レイテンシーウェブサイトを構築
- AWS re:Invent 2022 - AWS Global Accelerator でパフォーマンスと可用性を向上
- AWS re:Invent 2022 - AWS で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 - Amazon Route 53 でグローバルなトラフィック管理を実現

関連する例:

- AWS ネットワーキングワークショップ
- 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

サービスを利用するためにチームメンバーが使用するリソース、その予想ライフサイクル、および経営と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

- 省エネ型ワークステーションを使用する: チームメンバーに省エネ型ワークステーションと周辺機器を支給します。これらのデバイスで効率的な電源管理機能 (低電力モードなど) を使用して、エネルギー使用量を削減します。
- 仮想化を利用する: 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、アップグレードの必要性とデバイス要件を軽減します。
- リモートコラボレーションを奨励する: Amazon Chime および AWS Wickr などのリモートコラボレーションツールを使用するようチームメンバーに奨励して、出張の必要性や出張に関連する炭素排出量を削減します。
- エネルギー効率が優れたソフトウェアを使用する: 不要な機能やプロセスを削除または無効にして、チームメンバーにエネルギー効率が優れたソフトウェアを支給します。
- ライフサイクルを管理する: 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。ワークステーションまたはソフトウェアの定期的なメンテナンスと更新を行って、効率性を維持、改善します。
- リモートデバイス管理を実施する: 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

関連ドキュメント:

- Amazon WorkSpaces とは
- Amazon WorkSpaces のコストオプティマイザー
- Amazon AppStream 2.0 のドキュメント
- NICE DCV

関連動画:

- Managing cost for Amazon WorkSpaces on AWS

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:

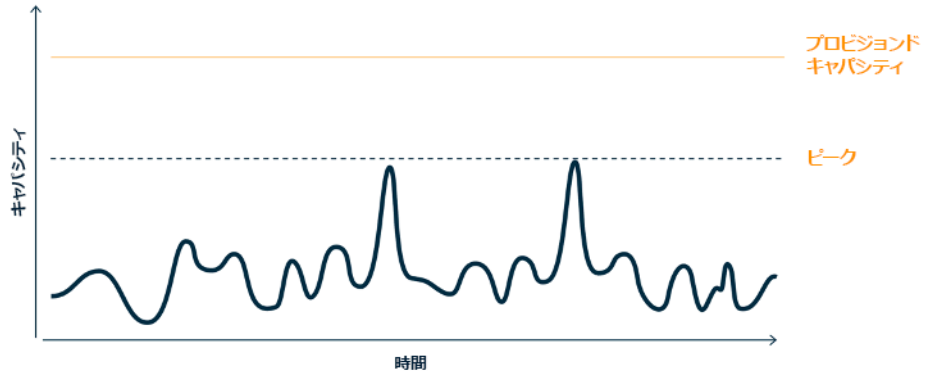
- 即時対応が不要なクライアントのリクエストを即時処理している。
- クライアントのリクエストの要件を分析していない。

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

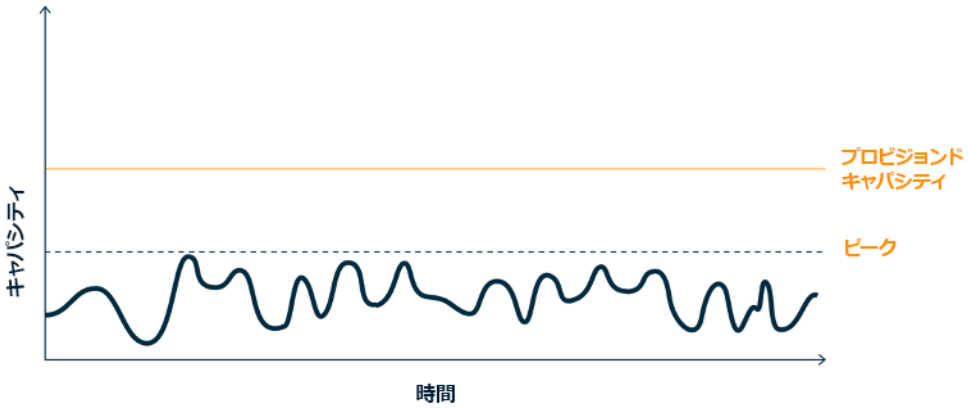
実装のガイダンス

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。



高いプロビジョンドキャパシティを必要とする 2 つの異なるピークの需要曲線

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。バッファリングを実装して、リクエストを保存し、処理を延期できます。



需要曲線とプロビジョニングされた容量に対するスロットリングの影響。

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。考慮すべき課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 を使用すると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、単独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

関連ドキュメント:

- Amazon SQS の開始方法
- キューとメッセージを使用したアプリケーション統合
- ワークロードでの API スロットリングの管理と監視
- 階層化されたマルチテナント REST API を API Gateway を使用して大規模にスロットリング
- 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:

- AWS re:Invent 2022 - Application integration patterns for microservices
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

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廃止しま

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。顧客がお客様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-BP05 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スケールできます。
- 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を活用しない場合のリスクレベル: 中

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と、コンポーネントの使用率が均等になり、ワークロードのオーバープロビジョニングを抑えます。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自動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。

キューイングメカニズム	説明
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 と 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を組み合わせると、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。

スケジューリングメカニズム	説明
Amazon EventBridge スケジューラ	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 Amazon EventBridge の機能です。

スケジューリングメカニズム	説明
AWS Glue 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 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、スケジュールに基づいて、または EventBridge イベントへの応答として、タスクを実行します。
Instance Scheduler	Amazon EC2 および Amazon Relational Database Service インスタンスの開始、停止スケジュールを設定します。

- 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、高効率のワークロードを構築します。
- AWS でサーバーレスを活用して、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。
  - または AWS Cost Explorer または AWS Compute Optimizer で適切なサイズ設定に関する推奨事項を使用して、適切なサイズ設定の機会を特定できます。
  - 詳細については、「適切なサイジング: ワークロードに適したインスタンスのプロビジョニング」を参照してください。

リソース

関連ドキュメント:

- Amazon Simple Queue Service とは
- Amazon MQ とは
- Amazon SQS に基づくスケーリング
- AWS Step Functions とは

- AWS Lambda とは
- Amazon SQS での AWS Lambda の使用
- Amazon EventBridge とは
- REST API を使用した非同期ワークフローの管理

関連動画:

- AWS re:Invent 2023 - Navigating the journey to serverless event-driven architecture
- AWS re:Invent 2023 - Using serverless for event-driven architecture & domain-driven design
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- Asynchronous Message Patterns | AWS Events

関連する例:

- AWS Graviton プロセッサと Amazon EC2 スポットインスタンスを使用したイベント駆動型アーキテクチャ

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 など AWS の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。

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、無駄が最小限に抑えられ、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: 中

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 関数のランタイムが大幅に低下すると、メモリサイズを小さくする必要があることを示す指標になります。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

- AWS リソースのインベントリを用意します。AWS では、AWS Resource Explorer を有効にして AWS リソースを探索および整理できます。詳細については、「AWS re:Invent 2022 - How to manage resources and applications at scale on AWS」を参照してください。
- ワークロードの重要なコンポーネント (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) の使用率メトリクスをモニタリング、キャプチャします。
- アーキテクチャ内の未使用のコンポーネントや使用率の低いコンポーネントを特定します。
  - 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チェックして、アイドル状態、未使用、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。
  - Amazon ECR における未使用イメージの自動クリーンアップ
  - AWS Config および AWS Systems Manager を使用して未使用の Amazon Elastic Block Store (Amazon EBS) ボリュームを削除
- 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代わりに、1 つの 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。
- ワークロードによってプロビジョニングされる、作業の単位を完了するために必要なリソースを理解します。

リソース

関連ドキュメント:

- AWS Trusted Advisor
- Amazon CloudWatch とは
- 適切なサイジング: ワークロードに適したインスタンスのプロビジョニング
- 適切なサイズ設定の推奨事項によるコストの最適化

関連動画:

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three

関連する例:

- 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:

- リソースの使用量に対して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: 効率的なコードを使用すると、リソースの使用量が最小限に抑えられ、パフォーマンスが向上します。

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: 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

- 効率的なプログラミング言語を使用する: ワークロードに効率的なオペレーティングシステムとプログラミング言語を使用します。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、「Rust での持続可能性」を参照してください。
- AI コーディングコンパニオンを使用する: Amazon CodeWhisperer などの AI コーディングコンパニオンを使用して、コードの効率的な記述を検討します。
- コードレビューを自動化する: 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。
  - Amazon CodeGuru Reviewer でのコードレビューの自動化
  - Amazon CodeGuru での同時実行バグの検出
  - 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
- コードプロファイラーを使用する: 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。
  - 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
  - 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
  - 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
- モニタリングと最適化をする: 継続的なモニタリングリソースを使用して、リソース要件が高い、または最適ではない構成のコンポーネントを特定します。
  - 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。
- コードのリファクタリングまたは変換を使用する: アプリケーションのメンテナンスとアップグレードに Amazon Q コード変換を検討します。
  - Amazon Q コード変換による言語バージョンのアップグレード
  - AWS re:Invent 2023 - Amazon Q コード変換を使用してアプリケーションのアップグレードとメンテナンスを自動化する

リソース

関連ドキュメント:

- Amazon CodeGuru Profiler とは
- FPGA インスタンス
- AWS の構築ツールの AWS SDK

関連動画:

- Improve Code Efficiency Using Amazon CodeGuru Profiler
- AWS re:Invent 2023 - Best practices for Amazon CodeWhisperer
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru

関連する例:

- Amazon CodeGuru によるコードの最適化

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:

- 顧客によって使用されるデバイスの環境に対する影響を無視する。
- 顧客によって使用され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: 顧客のデバイス用に最適化されたソフトウェアパターンと機能を実装することで、クラウドワークロードの全体的な環境への影響を軽減できます。

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: 中

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。

- 下位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

- 棚卸しを実施する: 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。
- エネルギー効率が優れたデバイスを使用する: エネルギー効率が優れたデバイスをアーキテクチャで使用することを検討してください。デバイスの電源管理設定を使用して、使用していないときは低電力モードに切り替えます。
- 効率的なアプリケーションを実行する: 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 または AWS Elemental MediaConvert を使用して、サイズが大きい高品質の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、およびネット接続したテレビで再生できる形式に変換できます。
  - 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、または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。
- サプライヤーを関与させる: 持続可能な資材を使用し、サプライチェーンと環境認定に透明性を持ったデバイスサプライヤーと連携します。

- 無線通信 (OTA) アップデートを使用する: 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。
  - CI/CD パイプラインを使用してモバイルアプリケーションを更新できます。
  - AWS IoT Device Management を使用して、接続されたデバイスを大規模にリモートで管理できます。
- マネージド型 Device Farm を使用する: 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 を使用して、サポート対象のデバイスを拡大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。
- モニタリングと改善を続ける: デバイスのエネルギー使用量を追跡して、改善が必要な分野を特定します。新しいテクノロジーやベストプラクティスを活用して、これらのデバイスの環境に配慮した取り組みを強化します。

リソース

関連ドキュメント:

- AWS Device Farm とは
- Amazon AppStream 2.0 のドキュメント
- NICE DCV
- FreeRTOS 実行デバイスでファームウェアを更新するための OTA チュートリアル
- 環境持続可能性のための IoT デバイスの最適化

関連動画:

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm

SUS03-BP05 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: データアクセスとストレージパターンに基づいてアーキテクチャを選択、最適化すると、開発の複雑さが軽減され、全体的な使用率が向上します。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ルできるようになります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

データ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、独自の分析ユースケースに最適化された専用サービスを使用できるようにする最新のデータアーキテクチャを AWS で使用します。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。
  - データ型: 構造、半構造、非構造
  - データの増加: 制限あり、無制限
  - 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、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。
  - データ永続化を有効にするパターン
  - Let's Architect! モダンデータアーキテクチャ
  - AWS のデータベース: 特定のジョブに最適なデータベースを
- 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。

  - Athena でサポートされる圧縮ファイル形式
  - AWS Glue での ETL 入力および出力の形式オプション
  - Amazon Redshift を使用して Simple Storage Service (Amazon S3) から圧縮されたデータファイルをロードする
- アーキテクチャでのデータ処理に専用の分析サービスを使用します。AWS 専用分析サービスの詳細については、「AWS re:Invent 2022 - Building modern data architectures on AWS」を参照してください。
- 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保します。詳細については、「AWS データベース」および「AWS re:Invent 2022 - Modernize apps with purpose-built databases」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

関連ドキュメント:

- 列データ形式の COPY
- Firehose での入力レコード形式の変換
- 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 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 ストレージクラス
- Amazon DynamoDB を使用して CQRS イベントストアを構築する

関連動画:

- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3

- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge

関連する例:

- AWS 目的別データベースワークショップ
- AWS モダンデータアーキテクチャ Immersion Day
- Build a Data Mesh on AWS

[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より効果的に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-BP05 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、データの最も省エネ的なストレージ階層を決定できます。

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

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

- データインベントリを実施する: 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。
- データをグループ分けする: 組織に対するリスクに基づいて、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、「データを分類してスタートアップ企業を保護するための 4 つの簡単なステップ」を参照してください。
- データ分類レベルとポリシーを定義する: データグループごとに、データ分類レベル (パブリックポリシーや機密ポリシーなど) と処理ポリシーを定義します。分類にそってデータにタグを付けます。データ分類カテゴリの詳細については、データ分類に関するホワイトペーパーを参照してください。
- 定期的にレビューする: タグ付けされていないデータや分類されていないデータがないか、環境を定期的にレビューして監査します。オートメーションを使用してこのデータを特定し、データを適

切に分類してタグ付けします。例として、「AWS Glue でのデータ検出とカタログ化」を参照してください。

- データカタログを作成する: 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。
- 文書化する: 各データクラスのデータ分類ポリシーと処理手順を文書化します。

リソース

関連ドキュメント:

- データ分類に AWS クラウドを活用
- AWS Organizations Tag policies

関連動画:

- AWS re:Invent 2022 - Enabling agility with data governance on AWS
- AWS re:Invent 2023 - Data protection and resilience with AWS storage

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良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。

実装手順

- データとアクセスパターンの特徴を評価する: 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。
  - データ型: 構造、半構造、非構造
  - データの増加: 制限あり、無制限
  - 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、頻度、急増、安定
- 適切なストレージ技術を選択する: 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 との間でオブジェクトを転送し、オブジェクトにアクセスするには、Transfer Acceleration やアクセスポイント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。

タイプ	テクノロジー	主な特徴
アーカイブストレージ	Amazon S3 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)	複数の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張および縮小し、一貫した低レイテンシーを実現するようにパフォーマンスが最適化されています。
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 のレイテンシー、スループット、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適したファイルシステムを選択する際には、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) のために設計された、スケーラブルな高性能ブロックストレージサービスです。Amazon EBS には、トランザクション、IOPS を多用するワークロード用の SSD ベースのストレージと、スループットを多用するワークロード用の HDD ベースのストレージが含まれています。
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。

- ストレージ割当を自動化する: 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 と 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。
- 適切なストレージクラスを選択する: データに適したストレージクラスを選択します。
  - 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ます。1 つのバケットには、すべてのストレージクラスに保存されているオブジェクトを含めることができます。
  - Amazon S3 ライフサイクルポリシーを使用して、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、アプリケーションに変更を必要としません。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

関連ドキュメント:

- Amazon EBS volume types
- Amazon EC2 インスタンスストア
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O の特性
- Amazon S3 ストレージクラスを使用する
- Amazon S3 Glacier とは

関連動画:

- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2022: Building modern data architectures on AWS
- AWS re:Invent 2022: Modernize apps with purpose-built databases
- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWSre:Invent 2023: Advanced data modeling with Amazon DynamoDB

関連する例:

- Amazon S3 の例
- AWS 目的別データベースワークショップ
- 開発者向けデータベース
- AWS モダンデータアーキテクチャ Immersion Day
- Build a Data Mesh on AWS

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と、ワークロード内の効率的なデータアクセスと保持が保証されます。

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: 中

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。

ストレージサービス	自動ライフサイクルポリシーを設定する方法
Amazon S3	Amazon S3 ライフサイクルを使用すると、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、変更中、または予測不可能な場合は、Amazon S3 Intelligent-Tiering を使用できます。これにより、アクセスパターンがモニタリングされ、アクセスされていないオブジェクトが低コストのアクセス階層に自動的に移動します。Amazon S3 ストレージレンズメトリクスを活用して、ライフサイクル管理の最適化の機会とギャップを特定できます。
Amazon Elastic Block Store	Amazon Data Lifecycle Manager を使用して、EBS スナップショットと Amazon EBS-backed AMI の作成、保持、削除を自動化できます。

ストレージサービス	自動ライフサイクルポリシーを設定する方法
Amazon Elastic File System	Amazon EFS のライフサイクル管理では、ファイルシステムのファイルストレージが自動的に管理されます。
Amazon Elastic Container Registry	Amazon ECR ライフサイクルポリシーでは、年数またはカウントに基づいたイメージの有効期限を使用してコンテナイメージのクリーンアップを自動化できます。
AWS Elemental MediaStore	オブジェクトの MediaStore コンテナ内保存期間を管理する、オブジェクトのライフサイクルポリシーを作成することができます。

- 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

関連ドキュメント:

- Amazon S3 ストレージクラス分析を使用して Amazon S3 ライフサイクルルールを最適化する
- Evaluating Resources with AWS Config ルール

関連動画:

- AWS re:Invent 2021 - Amazon S3 Lifecycle best practices to optimize your storage spend
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- Simplify Your Data Lifecycle and Optimize Storage Costs With Amazon S3 Lifecycle
- Reduce Your Storage Costs Using Amazon S3 Storage Lens

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オーバープロビジョニングを最小限に抑えると、アイドル状態のリソースが減少し、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: 中

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するようにします。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。
- 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。Amazon EBS Elastic Volumes では、EBS ボリュームのボリュームサイズの増加、ボリュームタイプの変更、パフォーマンスの調整を行うことができます。
- 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

関連ドキュメント:

- Extend the file system after resizing an EBS volume
- Modify a volume using Amazon EBS Elastic Volumes
- Amazon FSx Documentation
- Amazon Elastic File System とは

関連動画:

- Deep Dive on Amazon EBS Elastic Volumes
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings
- Optimizing Amazon EFS for cost and performance, using best practices

SUS04-BP05 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、またはまったく行わない。
- 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 バージョニングを有効にしている。

このベストプラクティスを活用するメリット: 不要なデータを削除すると、ワークロードに必要なストレージサイズとワークロードの環境への影響が軽減されます。

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: 中

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 AWS の Open Data で公開されている既存のデータセットを使用し、データの保存を回避できるかどうかを評価します。
- 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。

ストレージサービス	重複排除メカニズム
Amazon S3	AWS Lake Formation FindMatches の新しい FindMatches ML Transform を使用して、データセット (識別子のないレコードを含む) 間で一致するレコードを検索します。
Amazon FSx	Amazon FSx for Windows のデータ重複排除を使用します。
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、最後にスナップショットを作成した時点から、ボリューム上で変更のあるブロックだけが保存されます。

- 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。削除には、Amazon DynamoDB の有効期限や Amazon S3 Lifecycle、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
  - 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 (独立ディスクの冗長アレイ (RAID) 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

関連ドキュメント:

- Change log data retention in CloudWatch Logs
- Data deduplication on Amazon FSx for Windows File Server
- Features of Amazon FSx for ONTAP including data deduplication
- Amazon CloudFront のファイルを無効化する
- AWS Backup を使用してAmazon EFS ファイルシステムをバックアップおよび復元する
- What is Amazon CloudWatch Logs?
- バックアップの概要
- AWS Lake Formation を使用してデータセットの統合および重複の削除を実施

関連動画:

- Amazon Redshift Data Sharing Use Cases

関連する例:

- Amazon Athena で Amazon S3 サーバーアクセスログを分析する方法を教えてください。

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: 共有ファイルシステム、ストレージを使用すると、データをコピーしなくても 1 つ以上のコンシューマーに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を活用しない場合のリスクレベル: 中

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。

ストレージオプション	どのようなときに使うか
Amazon EBS マルチアタッチ	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

ストレージオプション	どのようなときに使うか
	ゾーン内の複数のインスタンスにアタッチできます。
Amazon EFS	「Amazon EFS を選択するタイミング」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 にバックアップされた Amazon FSx for Lustre ファイルシステムを作成し、処理ジョブ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」で説明されているように、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

関連ドキュメント:

- Linking your file system to an Amazon S3 bucket
- Using Amazon EFS for AWS Lambda in your serverless applications
- 新機能 – Amazon EFS Intelligent-Tiering がアクセスパターンの変化に応じてワークロードのコストを最適化
- 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
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - File storage for builders and data scientists on Amazon Elastic File System

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ときは、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。

- 詳細なデータ (レコードレベルの情報は不要) を集約します。
- 「Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing」を参照してください。
- AWS Lake Formation のクロスアカウントの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	どのようなときに使うか
Lambda@Edge	オブジェクトがキャッシュにないときに実行され、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 Functions	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

関連ドキュメント:

- 持続可能な AWS インフラストラクチャの最適化、第三部:ネットワーキング編
- AWS グローバルインフラストラクチャ
- Amazon CloudFront 特徴 (グローバルエッジネットワーク他)
- Compressing HTTP requests in Amazon OpenSearch Service
- Intermediate data compression with Amazon EMR
- 圧縮されたデータファイルを Amazon S3 からロードする
- 圧縮ファイルを供給する

関連動画:

- Demystifying data transfer on AWS

関連する例:

- Architecting for sustainability - Minimize data movement across networks

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: 重要度の低いデータのバックアップを回避することでワークロードに必要なストレージリソースを減らし、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を活用しない場合のリスクレベル: 中

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。

実装手順

- SUS04-BP01 データ分類ポリシーを実装する で解説しているとおりにデータ分類ポリシーを実装します。
- データの重要度区分を用いて、目標復旧時間 (RTO) と目標復旧時点 (RPO) に基づき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 はフルマネージド型のバックアップサービスであり、AWS のサービス、クラウド内、およびオンプレミス間で簡単に一元化およびデータ保護を自動化できます。AWS Backup を使用した自動バックアップの作成方法に関する実践的ガイダンスについては、「Well-Architected Labs - Testing Backup and Restore of Data」を参照してください。
- AWS Backup を使用して Amazon EFS のバックアップを自動化しバックアップコストを最適化します。

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

関連ドキュメント:

- AWS Backup を使用して Amazon EFS ファイルシステムをバックアップおよび復元する
- Amazon EBS スナップショット
- Amazon Relational Database Service でバックアップを操作する
- APN パートナー: バックアップの支援が可能なパートナー
- AWS Marketplace: バックアップに活用できる製品
- Amazon EFS 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 2023 - Backup and disaster recovery strategies for increased resilience
- AWS re:Invent 2023 - What's new with AWS Backup
- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS

関連する例:

- Well-Architected Lab - Backup data

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ことで、ワークロードによる環境への影響を減らし、費用を節約して、パフォーマンス基準を維持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: 中

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 クラウドでは、AWS Auto Scaling などさまざまなメカニズムによって、リソースの数を必要に応じて柔軟に拡張または縮小することができ、需要の変化に対応することができます。また、リソースの変更を最小限の労力で実行できる API と 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

- インスタンスタイプを選択する: ニーズに最適なインスタンスタイプを選びます。Amazon Elastic Compute Cloud インスタンスの選び方や、属性ベースのインスタンスの選択といったメカニズムの使用方法については、以下を参照してください。
  - ワークロードに適した 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 Fleet 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 グループを作成する
- スケールする: 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する: 複数のコンピューティング購入オプションを使用することで、インスタンスの柔軟性、スケーラビリティ、コスト削減の間のバランスを取ります。
  - Amazon EC2 オンデマンドインスタンスは、インスタンスタイプやロケーション、処理時間の柔軟性が低い、ステートフルでスパイクが発生しやすい新規のワークロードに最適です。
  - Amazon EC2 スポットインスタンスは、耐障害性と柔軟性を備えたアプリケーションに関して、他の方法を補完する優れた方法です。
  - 定常状態のワークロードには、Compute Savings Plans を活用すれば、ニーズの変化 (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など) に柔軟に対応できます。
- 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する: 多様なインスタンスやアベイラビリティーゾーンを使用することで、アプリケーションの可用性を最大化し余剰のキャパシティを活用することができます。

- インスタンスを適切なサイズに設定する: AWS ツールの適切なサイジング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。詳細については、「Optimizing your cost with Rightsizing Recommendations」と「適切なサイジング: ワークロードに適したインスタンスのプロビジョニング」を参照してください。
  - 適切なサイジングの機会を特定するには、AWS Cost Explorer の適切なサイジングのレコメンデーション、または AWS Compute Optimizer を使用します。
- サービスレベルアグリーメント (SLA) を見直す: 容量を一時的に減らせるように SLA を見直すと同時に、オートメーションを使用して代替のリソースをデプロイします。

リソース

関連ドキュメント:

- 持続可能な AWS インフラストラクチャの最適化、第一部:コンピュート編
- 新機能 — EC2 Auto Scaling と 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 Compute Optimizerドキュメント
- Operating Lambda: パフォーマンスの最適化 – Part 2
- Auto Scaling ドキュメント

関連動画:

- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2022 - Optimizing Amazon Elastic Kubernetes Service for performance and cost on AWS
- AWS re:Invent 2023 - Sustainable compute: reducing costs and carbon emissions with AWS

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 の設定で 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。
- AWS Compute Optimizer など、AWS の適切なサイジングツールのレコメンデーションを確認しない。

このベストプラクティスを活用するメリット: エネルギー効率に優れた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下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: 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

- インスタンスタイプを探して詳しく調べる: ワークロードによる環境への影響を減らすことができるインスタンスタイプを特定します。
  - AWS の最新情報を購読して、AWS のテクノロジーとインスタンスに関する最新情報を入手します。
  - 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 でのワットあたりのエネルギー使用量が最も優れた、AWS Graviton ベースのインスタンスの詳細については、「re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances」と「Deep dive into AWS Graviton3 and Amazon EC2 C7g instances」をご覧ください。
- 環境への影響が最少のインスタンスタイプを使用する: 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します。
  - 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。ワークロードを AWS Graviton に移行する方法の詳細については、「AWS Graviton Fast Start でイノベーションを加速する」と「Considerations when transitioning workloads to AWS Graviton-based Amazon Elastic Compute Cloud instances」を参照してください。
- AWS マネージドサービスを使用するときは、AWS Graviton の利用を検討します。
- 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、Amazon EC2 DL1 など、特定のワークロード専用のハードウェアを活用します。AWSInferentia インスタンス (Inf2 インスタンスなど) を使用することで、同等の Amazon EC2 インスタンスに比べ、ワットあたりのパフォーマンスが最大で 50% 向上します。
- ML 推論エンドポイントのサイズを適正化するときは、Amazon SageMaker Inference Recommender を使用します。
- スパイクが発生しやすいワークロード (追加の容量が必要になる頻度が低いワークロード) には、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あるワークロードには、Amazon EC2 スポットインスタンスを使用することで、クラウドの全体的な使用率を増やし、未使用のリソースが持続可能性に与える影響を減らします。

- 運用しながら最適化する: ワークロードインスタンスを運用し、最適化します。
  - エフェメラルなワークロードの場合は、インスタンスの Amazon CloudWatch メトリクス (`CPUUtilization` など) を測定し、インスタンスがアイドル状態か、またはあまり利用されていないかを特定します。
  - 安定したワークロードの場合は、AWS の適切なサイジングツール (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適切なサイジングの機会を特定します。その他の例と推奨事項については、以下のラボを参照してください。
    - Well-Architected Lab - Rightsizing Recommendations
    - Well-Architected Lab - Rightsizing with Compute Optimizer
    - Well-Architected Lab - Optimize Hardware Patterns and Observice Sustainability KPIs

リソース

関連ドキュメント:

- 持続可能な AWS インフラストラクチャの最適化、第一部:コンピュート編
- AWS Graviton プロセッサ
- Amazon EC2 DL1 インスタンス
- キャパシティ予約フリート
- EC2 フリートとスポットフリート
- Function Configuration
- Amazon EC2 Fleet の属性ベースのインスタンスタイプの選択
- 持続可能で、効率的かつコストが最適化されたアプリケーションを AWS で構築する
- How the Contino Sustainability Dashboard Helps Customers Optimize Their Carbon Footprint

関連動画:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 = What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

関連する例:

- Solution: Guidance for Optimizing Deep Learning Workloads for Sustainability on AWS
- Migrating Amazon Relational Database Service Databases to Graviton

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。当社は、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を活用しない場合のリスクレベル: 中

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 にはデータベースのマネージドサービスがあります。Amazon Athena、Amazon EMR 、Amazon OpenSearch Service には分析のマネージドサービスがあります。

実装手順

1. ワークロードのリストを作成する: 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. コンポーネントの候補を特定する: 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。

タスク	AWS で使用するもの
データベースのホスティング	独自の Amazon RDS インスタンスを Amazon Elastic Compute Cloud (Amazon EC2) で維持する代わりに、マネージド型の Amazon

タスク	AWS で使用するもの
	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代わりに、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	AWS Amplify Amplify ホスティングを、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、使用します。

3. 移行計画を作成する: 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提供するサービスです。移行の計画を立てる際に、充分な情報に基づいて意思決定を行えるようになります。
4. テストを行う: マネージドサービスに移行する前にサービスをテストします。
5. セルフホスト型のサービスを置き換える: 作成した移行計画に基づいて、セルフホスト型のサービスをマネージドサービスに置き換えます。
6. モニタリングし調整する: 移行の完了後は、サービスを継続的にモニタして、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:

- AWS re:Invent 2021 - Cloud operations at scale with AWS Managed Services
- AWS re:Invent 2023 - Best practices for operating on AWS

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

- 要件に対応できる高速コンピューティングインスタンスを特定します。
- 機械学習のワークロードには、AWS Trainium、AWS Inferentia、Amazon EC2 DL1 などのワークロード専用ハードウェアを活用してください。AWSInf2 インスタンスなどの 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。

- 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、「Amazon CloudWatch で NVIDIA GPU メトリクスを収集する」のように、CloudWatch エージェントを使用して GPU の `utilization_gpu` や `utilization_memory` などのメトリクスを収集できます。
- 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 設定の最適化
  - Deep Learning AMI での GPU のモニタリングと最適化
  - Amazon SageMaker でのディープラーニング学習時における、GPU パフォーマンスチューニングのための I/O 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
- ワークロードに適した 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 Instances
- Amazon SageMaker でコンピュータビジョン推論に最適な AI アクセラレータとモデルコンパイルを選択

関連動画:

- AWS re:Invent 2021 - How to select Amazon EC2 GPU instances for deep learning
- AWS Online Tech Talks - Deploying Cost-Effective Deep Learning Inference
- AWS re:Invent 2023 - Cutting-edge AI with AWS and NVIDIA
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium

- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

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を活用し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

- 組織の持続可能性目標の理解と周知: 二酸化炭素削減やウォータースチュワードシップなど、組織の持続可能性目標を理解します。これらの目標をクラウドワークロードの持続可能性要件に変換します。これらの要件を主なステークホルダーに伝えます。
- 持続可能性要件のバックログへの追加: 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。
- 反復と改善: 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。
- 実用最小限の製品 (MVP) を使用したテスト: 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、テストのコストと環境への影響を削減します。
- プロセスの合理化: 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。
- トレーニングと啓発: チームメンバーを対象にトレーニングプログラムを実施して、持続可能性、および活動が組織の持続可能性目標にどのように影響するかについて教育します。
- 評価と調整: 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

関連動画:

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint

- AWS re:Invent 2022 - Sustainability in AWS global infrastructure
- AWS re:Invent 2023 - What's new with AWS observability and operations

関連する例:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports

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を活用し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

- プロセスの定義: 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使用します。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。
- インベントリの作成: 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すれば、Amazon EC2 インスタンスからオペレーティングシステム (OS)、アプリケーション、インスタンスのメタデータを収集し、どのインスタンスがソフトウェアポリシーで要求されるソフトウェアと設定を実行しているか、どのインスタンスがアップデートする必要があるかを迅速に把握することが可能です。
- 更新手順の学習: 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。

ワークロードコンポーネント	更新方法
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あります。

- 自動化の使用: 更新を自動化して、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 を使用すると、AMI、コンテナイメージなど、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するとシステム更新のプロセスを自動化でき、AWS Systems Manager Maintenance Windows を使用するとアクティビティをスケジュールできます。

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 の最新情報
- AWS 開発者用ツール

関連動画:

- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance
- All Things Patch: AWS Systems Manager

関連する例:

- Well-Architected Labs - Inventory and Patch Management
- Lab: 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を活用し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 を使用する: 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する: 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。
- 使用率を最大化する: 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
- What is AWS CloudFormation?
- AWS CodeBuild とは
- AWS での Instance Scheduler

関連動画:

- AWS re:Invent 2023 - Continuous integration and delivery for AWS

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を活用しない場合のリスクレベル: 低

実装のガイダンス

マネージド型 Device Farm を使用すると、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

- テストの要件を定義する: 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。
- マネージド型 Device Farm を選択する: テスト要件に対応できるマネージド型 Device Farm を選択します。例えば、AWS Device Farm を使用すると、代表的なハードウェア一式における変更をテストし、その影響を理解することができます。

- オートメーションを使用する: 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
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services
- 見直して調整する: 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

関連ドキュメント:

- AWS Device Farm device list
- CloudWatch RUM ダッシュボードの表示

関連動画:

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM

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

注意

お客様は、本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、明示または黙示を問わず、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 契約によって規定されます。本書は、AWS とお客様と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 © 2023 Amazon Web Services, Inc. or its affiliates.

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。