AWS Well-Architected フレームワーク

2019 年 7 月

本書では AWS Well-Architected フレームワークについて説明しています。お客様はクラウドベースのアーキテクチャを評価および改善し、設計上の決定がビジネスに及ぼす影響をより良く理解できるようになります。ここでは Well-Architected フレームワークの柱とされる 5 つの概念領域における一般的な設計の原則と、特定のベストプラクティスおよびガイダンスを紹介します。

注意

お客様は、このドキュメント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。このドキュメントは、(a) 情報提供のみを目的としており、(b) AWS の現行製品とプラクティスを表したものであり、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義務や確約を意味するものではありません。AWS の製品やサービスは、明示または暗示を問わず、いかなる保証、表明、条件を伴うことなく「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本ドキュメントは、AWS とお客様の間で行われるいかなる契約の一部でもなく、そのような契約の内容を変更するものでもありません。

はじめに

AWS Well-Architected フレームワークの目的は、AWS でシステムを構築する際の選択肢の長所と短所をお客様が理解できるように支援することです。効率が良く、費用対効果が高く、安全で信頼のおけるクラウド対応システムを設計して運用するために、アーキテクチャに関するベストプラクティスをこのフレームワークに従って学ぶことができます。ベストプラクティスに照らして一貫した基準でアーキテクチャを評価し改善領域を見つけることができます。アーキテクチャのレビュープロセスは、アーキテクチャに関する決定についての前向きな話し合いであって、監査過程ではありません。システムを適切に設計することによってビジネスの成功の可能性が大いに高まると当社は確信しています。

AWS ソリューションアーキテクトはさまざまなビジネス分野やユースケースのソリューションを長年設計してきました。AWS に対応するアーキテクチャを設計し評価する多くのお客様を支援してきました。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 フレームワークに関するドキュメントには、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にうまく合致しているかどうかを理解するための基本的な質問が記載されています。このフレームワークは、現代のクラウドベースのシステムに期待する品質を評価するための一貫したアプローチと、その品質を達成するために必要な対応を提供します。AWS は進化し続け、お客様との共同作業で学ぶことも尽きないため、優れた設計の定義も改良され続けます。

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。本書にはクラウドのワークロードの設計と運用に適用する AWS のベストプラクティスと戦略が説明されており、導入の詳細とアーキテクチャパターンへのリンクが記載されています。詳細については、AWS Well-Architected homepage ホームページを参照してください。

AWS では、ワークロードを確認するための無料サービスも提供しています。AWS Well-Architected Tool (AWS WA Tool) は、AWS Well-Architected フレームワークを使ってアーキテクチャをレビューし、測定するための一貫したプロセスを提供する、クラウド内のサービスです。AWS WA Tool は、ワークロードを信頼性が高く、よりセキュアかつ効率的で、コスト効率性に優れたものにするためのレコメンデーションを提供します。

ベストプラクティスの適用をサポートするため、AWS Well-Architected Labs を作成しました。このラボでは、コードとドキュメントのリポジトリを使用して、ベストプラクティスの実装を実践的に体験することができます。また、AWS Well-Architected パートナープログラムのメンバーである AWS パートナーネットワーク (APN) パートナーと提携しています。これらの APN パートナーは AWS に関する深い知識を持っており、ワークロードの確認と改善をサポートします。

定義

AWS のエキスパートは、日々、お客様がクラウド上でベストプラクティスの利点を活かせるようなシステムを構築できるようにお手伝いしています。私たちは、設計が進化するにつれて発生するアーキテクチャとのトレードオフをお客様とともに考えてきました。お客様がライブ環境にシステムをデプロイするたびに、当社はそのシステムの実際のパフォーマンスやトレードオフの結果を学びます。

当社は学んできたことに基づいて、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には、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集が含まれています。

AWS Well-Architected フレームワークは、運用性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化という 5 本の柱を基本としています。

表1 AWS Well-Architected フレームワークの柱

名前	説明
運用上の優秀性	ビジネス価値を提供し、サポートのプロセスと手順を継続的に向上させるためにシステムを稼働およびモニタリングする能力
セキュリティ	リスク評価とリスク軽減の戦略を通してビジネスに価値をもたらす、情報、システム、アセットのセキュリティ保護機能
信頼性	インフラストラクチャやサービスの中断から復旧し、需要に適したコンピューティングリソースを動的に獲得し、誤設定や一時的なネットワークの問題といった中断の影響を緩和する能力
パフォーマンス効率	システムの要件を満たすためにコンピューティングリソースを効率的に使用し、要求の変化とテクノロジーの進化に対してもその効率性を維持する能力です
コスト最適化	最も低い価格でシステムを運用してビジネス価値を実現する能力

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、要件に対して提供されるコード、設定、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から分離されます。
- ワークロードという用語は、ビジネス価値を提供する一連のコンポーネントを識別するために使用します。ワークロードの詳細レベルは通常、ビジネスリーダーとテクノロジーリーダーが話し合う場合のレベルになります。

- マイルストーンは、アーキテクチャが設計、テスト、サービス開始、および運用という製品ライフサイクル全体を通じて進化するにあたり、アーキテクチャにおける重要な変更を記録します。
- アーキテクチャとは、コンポーネントがワークロードで連携する方法であると考えられます。コンポーネントが通信や対話を行う方法は、アーキテクチャ図の中心となることがよくあります。
- 企業内のテクノロジーポートフォリオとは、ビジネスの運営に必要なワークロードの集合体です。

ワークロードを設計するときには、ビジネスの状況に応じて 5 本の柱の間でトレードオフを行います。これらのビジネス上の決定は、エンジニアリング面での優先事項を推進させることができます。開発環境では信頼性を犠牲にすることでコストを削減するという最適化を行う場合や、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。e コマースソリューションでは、パフォーマンスが収益と顧客の購買傾向に影響することがあります。セキュリティと運用性は、通常、他の柱とトレードオフされることはありません。

アーキテクチャ

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。大抵のテクノロジーアーキテクチャチームは、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で構成されます。テクノロジーアーキテクチャチームでエンタープライズアーキテクチャ機能の一部としてよく使用されるのが、TOGAF または Zachman Framework です。

AWS では 1 つの中心チームに職能を持たせるのではなく、その職能を複数のチームに分散することが望まれています。決定権限を分散することには、複数のチームを内部標準に準拠させるといった点でリスクがあります。当社はそのようなリスクを 2 つの方法で軽減します。第 1 に、AWS には各チームにその機能を持たせることに焦点を当てたプラクティス[1]があり、チームが満たすべき基準のバーを上げることを確実にするためにエキスパートを配置しています。第 2 に、基準を確実に満たすための自動チェックを実行するメカニズム[2]を導入しています。この分散型アプローチは、Amazon のリーダーシッププリンシプルによって支持されており、お客様を起点に物事を考える[3]という文化を、すべてのロールにわたって確立します。お客様のことを真剣に考えているチームが、お客様の要件に応じて商品を開発します。

[1]方法、プロセス、標準、受け入れられている基準です。
[2]「意志だけでは絶対にうまくいかない。成し遂げるには適切なメカニズムが必要です。」Jeff Bezos。つまり、人間の努力に置き換えるメカニズム (多くの場合、自動化) がルールやプロセスに遵守しているか確認することです。
[3]Working backwards(お客様を起点に物事を考える) は当社のイノベーションプロセスの基本となる部分です。AWS では、お客様とその要望を起点に当社の取り組みを定義して進めます。

それをアーキテクチャに当てはめて、各チームがアーキテクチャを構築してベストプラクティスに準拠できるようにします。新しいチームがそれらの職能を獲得したり、既存のチームが水準を高めたりすることができるように、それらのチームがプリンシパルエンジニアの仮想コミュニティを利用して、エンジニアに設計を評価してもらったり、AWS のベストプラクティスを理解するのを助けてもらったりすることができるようにします。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。その方法の例としては、ベストプラクティスを実例に適用することについてランチタイムトークで取り上げます。その話を録音して、新しいチームメンバー向けのオンボーディング教材として使用できます。

AWS のベストプラクティスはインターネットの規模で多く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を定義する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。新しいベストプラクティスが顕在してくると、チームがそれらに従うことができるようにプリンシパルエンジニアがコミュニティとして活動します。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施しているものであり、ソリューションアーキテクチャといったフィールド職や内部エンジニアリングチームでのプリンシパルエンジニアリングの考えが体系化されています。Well-Architected フレームワークは当社が学んだことをお客様に活用してもらうためのメカニズムであり、拡張可能です。

プリンシパルエンジニアリングコミュニティが取り組んでいるアーキテクチャの分散所有に従うことによって、お客様の要件に基づいて優れた設計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優れた設計の評価を実施することで、お客様のテクノロジーポートフォリオのリスクがよくわかるようになります。この方法ですべてのチームに関わる課題を特定して、その課題に取り組むことができます。プリンシパルエンジニアはメカニズムや、トレーニング、ランチタイムトークを活用して、チーム間の特定の領域について考えを伝えることができます。

一般的な設計の原則

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

- 必要キャパシティーの推測が不要に: 必要なインフラストラクチャキャパシティーを予測する必要がなくなります。システムをデプロイする前にキャパシティーを決定すると、高価な余剰リソースが生まれて、キャパシティーの制約によるパフォーマンスへの影響に対処することになる可能性があります。クラウドコンピューティングにはこのような問題はありません。必要な分のみキャパシティーを使用し、自動的にスケールアップまたはスケールダウンできます。
- 本稼働スケールでシステムをテストする: クラウド上では、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。
- 自動化によってアーキテクチャでの実験を容易にする: 自動化によって低コストでシステムを作成およびレプリケートして、手動作業の支出を回避できます。自動化に対する変更を追跡し、影響を監査して、必要な場合は以前のパラメータに戻すことができます。
- 発展するアーキテクチャが可能に: アーキテクチャを発展させることができます。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施されることが多く、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ます。ビジネスとその状況が変化し続けるにつれて、当初の決定が原因で、変化するビジネス要件にシステムが対応できなくなる可能性があります。クラウド上では、自動化し、オンデマンドでテストできるので、設計変更によって生じる影響のリスクを軽減できます。そのため、イノベーションを標準プラクティスとしてビジネスで活用できるように、システムを時間とともに進化させることができます。
- データに基づいてアーキテクチャを進化させる: クラウド上ではアーキテクチャに関する選択がワークロードの動作に与える影響についての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ですので、そのデータに基づいてアーキテクチャに関する選択と改善を徐々に進めることができます。
- ゲームデーを利用して改善する: ゲームデーを定期的にスケジュールして、本稼働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 5 本の柱

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、ビルの健全性や機能を損なう構造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場合、運用性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化の 5 本の柱を疎かにすると、意図したとおりに要件に従って稼働するシステムの構築が難しくなるでしょう。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。

運用上の優秀性

運用上の優秀性の柱には、ビジネス価値を提供し、サポートのプロセスと手順を継続的に向上させるためにシステムを稼働およびモニタリングする能力 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」のホワイトペーパーを参照してください。

設計原則

クラウドでの運用上の優秀性には、6 つの設計原則があります。

- 運用をコードとして実行する: クラウドでは、アプリケーションコードに使用しているものと同じエンジニアリング原理を環境全体に適用できます。ワークロード全体 (アプリケーション、インフラストラクチャ) をコードとして定義し、コードを使用して更新できます。運用手順をコードとして実装し、イベントに対応してそのコードをトリガーすることで自動的に実行できます。運用をコードとして実行することで、人為的なミスを抑制し、イベントへの一貫性のある対応を実現できます。
- ドキュメントに注釈を付ける: オンプレミス環境では、ドキュメントは手動で作成し、ユーザーが使用します。そのため、変更のたびに同期することは困難です。クラウドでは、各ビルド後の注釈を付けたドキュメントの作成を自動化できます (または、手動で作成したドキュメントに自動的に注釈を付けることができます)。注釈を付けたドキュメントは、ユーザーまたはシステムが使用できます。注釈は運用コードへの入力として使用します。
- 定期的に、小規模な、元に戻すことができる変更を適用する: コンポーネントを定期的に更新できるようにワークロードを設計します。変更は、失敗した場合に元に戻すことができるように小規模に行います (可能な場合は、顧客に影響がないようにします)。
- 運用手順を定期的に改善する: 運用手順を実施するときに、改善の機会を探します。ワークロードを改良するときに、手順もそれに応じて改良します。定期的なゲーム

デーを計画し、すべての手順が効果的で、チームがその手順を熟知していることを確認および検証します。

- 障害を予想する: 障害の考えられる原因を除去または軽減できるように、原因を特定する「プレモータム」演習を実施します。障害シナリオをテストし、その影響に関する理解を検証します。対応手順をテストし、手順が効果的で、チームが手順の実行を十分に理解していることを確認します。定期的なゲームデーを計画し、ワークロードと、シミュレートされたイベントに対するチームの応答をテストします。
- 運用上のすべての障害から学ぶ: 運用上のすべてのイベントと障害から教訓を学び、改善を促進します。チーム間と組織全体で教訓を共有します。

定義

クラウドでの運用上の優秀性には、3 つのベストプラクティスの分野があります。

- 準備
- 運用
- 進化

運用チームは、ビジネスのニーズと顧客のニーズを理解し、ビジネスの成果を達成できるように効果的かつ効率的に支援する必要があります。運用では、運用上のイベントに応答するための手順を作成および使用し、その効果を検証してビジネスのニーズに対応します。運用では、目的とするビジネスの成果の達成度を測定するために使用するメトリクスを収集します。ビジネスの状況、ビジネス上の優先事項、顧客のニーズなど、すべては変化し続けます。そのため、時間の経過に伴う変化に対応した進化を促進する運用を設計し、運用のパフォーマンスから学んだ教訓を取り入れることが重要です。

ベストプラクティス

準備

運用上の優秀性を達成するには、効果的な準備が必要です。ビジネスの成功は、ビジネス、開発、運用の全体で目標と理解を共有することで実現できます。共通の基準が持つと、ワークロードの設計と管理を簡素化することができ、運用上の成功をもたらします。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のコンポーネント、およびカスタマーエクスペリエンスと顧客の行動をモニタリングし、インサイトを取得するメカニズムを使用してワークロードを設計します。

ワークロードまたは変更が本番環境に移行できる状態であり、運用上、サポートされていることを検証するためのメカニズムを作成します。運用準備状態はチェックリストを使用して検証し、ワークロードが規定の標準を満たしていることと、必要な手順がランブックとプレイブックに適切に記載されていることを確認します。ワークロードを効果

的にサポートするために、十分にトレーニングを受けた担当者がいることを確認します。移行前に、運用上のイベントと障害への応答をテストします。障害の投入やゲームデーイベントによって、サポートされる環境で応答を練習します。

AWS を使用すると、クラウド上でコードとしての運用が可能になるため、安全な実験、運用手順の開発、障害に備えた練習を実施できるようになります。AWS CloudFormation を使用すると、テンプレート化された整合性のあるサンドボックスの開発環境、テスト環境、本番環境を構築することができ、運用制御のレベルを向上できます。AWS では、さまざまなログ収集機能とモニタリング機能を使用してすべてのレイヤーでワークロードを可視化することができます。Amazon CloudWatch、AWS CloudTrail、VPC フローログを使用して、リソース使用に関するデータ、アプリケーションプログラミングインターフェイス (API)、ネットワークフローのログを収集できます。collectd プラグイン、つまり CloudWatch Logs エージェントを使用して、オペレーティングシステムに関する情報を CloudWatch に収集できます。

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

OPS 1: 優先順位はどのように決定すればよいでしょうか?
だれもが、ビジネスを成功させるうえで自分が果たす役割を理解する必要があります。リソースの優先順位を設定するため、共通の目標を設定してください。これにより、取り組みから得られるメリットが最大化されます。
OPS 2: ワークロードの状態を理解できるようにするには、ワークロードをどう設計すればよいでしょうか?
ワークロードを設計する際には、すべてのコンポーネントにわたって内部状態 (メトリクス、ログ、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。これにより、適時に必要な応答を提供できるようになります。
OPS 3: 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、バグ修正を可能にし、本番環境への変更のフローを改善するアプローチを採用します。これらにより、本番環境に採用される有益な変更を加速させ、デプロイされた問題を制限できます。またデプロイアクティビティを通じて挿入された問題をすばやく特定し、修復できます。
OPS 4: どのようにデプロイのリスクを軽減しますか?
品質に関する迅速なフィードバックを提供し、望ましい結果をもたらさない変更から迅速に復旧できるようにするアプローチを採用します。このような手法を使用すると、変更のデプロイによって生じる問題の影響を軽減できます。
OPS 5: ワークロードをサポートする準備が整っていることはどうすれば確認できるでしょうか?
ワークロード、プロセス、手順、従業員の運用準備状況を評価し、ワークロードに関連する運用上のリスクを理解するようにします。

最小限のアーキテクチャ標準をワークロードに実装します。標準を実装するコストと、ワークロードが得られる利点と運用にかかる負荷のバランスを取ります。サポートされ

ている標準の数を減らすことで、許容範囲に達しない標準が間違って適用されることがないようにします。多くの場合、運用担当者のリソースは限られています。

運用アクティビティをコードとして実装することに投資することにより、運用担当者の生産性を最大に引き上げ、エラーの発生を最小限に抑え、自動応答を可能にします。クラウドの伸縮性を活用したデプロイ方法を導入し、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。

運用

ワークロードの運用の成功は、ビジネスの成果と顧客の成果の達成度によって評価されます。予想される成果を定義し、成功を評価する方法を決定します。また、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ワークロードと運用のメトリクスを特定します。ワークロードの状態と、そのワークロードで実行する運用の状態と成功 (デプロイとインシデント対応など) の両方を含めて、運用の状態を検討します。運用の向上または低下を特定するための基準を確立し、メトリクスを収集および分析します。次に、運用の成功に関する理解と、時間の経過とともにどのように変化するかについて検証します。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。これには、計画した運用上のイベントと計画外の運用上のイベントの両方が含まれます。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、その他のイベントの解決にはプレイブックを使用します。ビジネスと顧客への影響に基づいてイベントへの応答に優先順位を付けます。イベントへの応答でアラートが発生する場合、実行する関連プロセスがあり、所有者が具体的に指名されていることを確認します。イベントを解決する担当者を事前に決めておき、影響 (期間、規模、範囲) に基づいて、必要に応じて他の担当者を関与させるためにエスカレーションするトリガーを含めます。以前に処理したことがないイベント応答によってビジネスに影響が及ぶ場合は、アクションの方針を決定する権限を持つ担当者を特定し、関与させます。

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

計画外のイベント、および計画したイベントからの想定外の影響について根本原因を特定します。この情報を使用して、将来イベントが発生した場合に問題を緩和できるように手順を更新します。必要に応じて影響するコミュニティに根本原因を知らせます。

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のアプリケーションを活用し、運用アクティビティについて、ビジネス、ワークロード、運用レベルのビューをまとめて表示できます。AWS では、AWS X-Ray、CloudWatch、CloudTrail、VPC フローログなどのログ機能によって、ワークロードを深く理解することができます。この機能を使用すると、ワークロードの問題を特定することができ、根本原因の分析や修正に役立ちます。

以下の質問は、運用上の優秀性に関するこれらの考慮事項に焦点を当てています。

OPS 6: ワークロードの正常性をどのように把握しますか?

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

OPS 7: オペレーションの正常性をどのように把握しますか?

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

OPS 8: ワークロードと運用イベントはどのように管理しますか?

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

日常的な運用や計画外のイベントへの応答は自動化する必要があります。デプロイ、リリース管理、変更、ロールバックについて、手作業のプロセスは避ける必要があります。リリースは、低頻度で行われる大規模なバッチ処理にしないようにします。変更が大きいと、ロールバックが難しくなります。ロールバック計画や障害による影響を軽減する機能がないと、運用の継続性を妨げることになります。応答が効果的にビジネスの継続性を維持するように、メトリクスをビジネスのニーズに合わせます。手動応答による 1 回限りの分散化されたメトリクスは、計画外のイベント中に運用を大きく中断させることになります。

進化

運用上の優秀性を維持するには運用の進化が必要です。継続的かつ段階的な改善を行うために専用の作業サイクルを作成します。ワークロードと運用手順の両方について、改善の機会 (機能のリクエスト、問題の修正、コンプライアンス要件など) を定期的に評価し、優先順位を付けます。手順にフィードバックループを取り入れ、改善が必要な分野をすばやく特定し、実際に運用して教訓を学びます。

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

AWS 開発者用ツールを使用すると、継続的デリバリーのビルド、テスト、デプロイのアクティビティを実装できます。そのアクティビティは、AWS とサードパーティから提供されるソースコード、ビルド、テスト、デプロイ用のさまざまなツールと連携します。デプロイアクティビティの結果を使用して、デプロイと開発の両方について改善の機会を特定できます。運用とデプロイのアクティビティデータをまとめたメトリクスデータを分析し、そのアクティビティがビジネスの成果と顧客の成果に与える影響を分析できます。このデータは、チーム間の遡及的分析で使用して、改善の機会とその方法を特定できます。

以下の質問は、運用上の優秀性に関するこれらの考慮事項に焦点を当てています。

OPS 9: オペレーションを進化させる方法

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

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

主要な AWS のサービス

運用上の優秀性に不可欠なAWS のサービスは AWS CloudFormation であり、ベストプラクティスに基づいたテンプレートを作成できます。これにより、開発環境から本番環境まで規則的で一貫した方法でリソースをプロビジョンできます。以下のサービスと機能では、運用上の優秀性の 3 つの分野がサポートされます。

- 準備: AWS Config と AWS Config のルールを使用して、ワークロードの標準を作成し、本番稼働の前に環境がその標準に準拠しているかどうかを確認できます。
- 運用: Amazon CloudWatch では、ワークロードの運用状態をモニタリングできます。
- 進化: Amazon Elasticsearch Service (Amazon ES) を使用すると、ログデータを分析し、実用的なインサイトをすばやく確実に取得できます。

リソース

運用上の優秀性に関する AWS 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 and AWS

ホワイトペーパー

- Operational Excellence Pillar

動画

- DevOps at Amazon

セキュリティ

セキュリティの柱には、リスク評価とリスク軽減の戦略を通してビジネスに価値をもたらす、情報、システム、アセットのセキュリティ保護機能 が含まれます。

このセキュリティの柱では、設計原則、ベストプラクティス、質問の概要を説明します。実装に関する規範的なガイダンスについては、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。

設計原則

クラウドでのセキュリティには、7 つの設計原則があります。

- 強力なアイデンティティ基盤の実装: 最小権限の原則を実装し、役割分担を徹底させ、各 AWS リソースとの通信において適切な認証を実行します。権限の管理を一元化し、長期的な認証情報への依存を軽減あるいは解消します。
- トレーサビリティの実現: ご使用の環境に対して、リアルタイムで監視、アラート、監査のアクションと変更を行うことができます。システムにログとメトリクスを統合し、自動で応答してアクションをとります。
- 全レイヤーへのセキュリティの適用: 外部に接する単一のレイヤーを保護することだけに重点を置くのではなく、深層防御をその他のレイヤーにも適用してセキュリティをコントロールします。すべてのレイヤー (エッジネットワーク、VPC、サブネット、ロードバランサー、すべてのインスタンス、オペレーティングシステム、アプリケーションなど) に適用します。
- 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、スケール機能を改善して、安全に、より速く、より費用対効果の高いスケールが可能になります。バージョン管理されているテンプレートにおいてコードとして定義および管理されるコントロールを実装するなど、セキュアなアーキテクチャを作成します。
- 伝送中および保管中のデータの保護: データを機密性レベルに分類し、暗号化、トークン分割、アクセスコントロールなどのメカニズムを適宜使用します。
- データに人の手を入れない: データに直接アクセスしたりデータを手動で処理したりする必要を減らす、あるいは無くすメカニズムとツールを作成します。これにより、機密性の高いデータを扱う際のデータの損失、変更、ヒューマンエラーのリスクを軽減します。
- セキュリティイベントへの備え: ご所属の組織の要件に合わせたインシデント管理プロセスにより、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、自動化されたツールを使用して、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、5 つのベストプラクティスの分野があります。

- アイデンティティ管理とアクセス管理
- 発見的統制
- インフラストラクチャ保護
- データ保護
- インシデント対応

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

AWS の責任共有モデルにより、このクラウドを導入した組織はセキュリティとコンプライアンスの目標を達成することができます。AWS がこのクラウドサービスの基盤となるインフラストラクチャを物理的に保護しているため、AWS のお客様はサービスを使用して自分たちの目標を達成することだけに集中できます。また、AWS クラウドは、より優れたセキュリティデータへのアクセスと、セキュリティイベントに対応する自動化された手段を提供します。

ベストプラクティス

アイデンティティ管理とアクセス管理

アイデンティティ管理とアクセス管理は情報セキュリティプログラムの重要な要素であり、これによりお客様が意図した仕方で、承認され認証されたユーザーのみがリソースにアクセスできます。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とるユーザー、グループ、サービス、ロール) を定義し、これらのプリンシパルに合わせたポリシーを構築し、強力な認証情報管理を実装できます。これらの権限管理機能は認証と承認の中枢となっています。

AWS では、権限管理は主に AWS Identity and Access Management (IAM) サービスによってサポートされ、このサービスがユーザーの管理や、AWS のサービスとリソースへのプログラムによるアクセスを可能にしています。詳細なポリシーを適用し、ユーザー、グループ、ロール、またはリソースに権限を割り当てることができます。また、複雑性レベル、再利用禁止、多要素認証 (MFA) の強制など、強力なパスワード設定をする機能があります。また既存のディレクトリサービスでフェデレーションを使用することもで

きます。AWS へのアクセス権を持つシステムを必要とするワークロードの場合、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。

以下の質問は、セキュリティに関するこれらの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、およびベストプラクティスの一覧については、付録を参照してください。).

SEC 1: 認証情報と認証をどのように管理していますか?
認証情報と認証メカニズムには、ワークロードにおいてアクセス権を直接的または間接的に付与するパスワード、トークン、キーなどが含まれます。適切なメカニズムで認証情報を保護することで、不注意または悪意による不正利用のリスクを減らすことができます。
SEC 2: 人為的なアクセスをどのように制御していますか?
定義されたビジネス要件に合致するコントロールを実装することで人為的なアクセスを制御し、リスクと不正アクセスの影響を軽減します。これは特権ユーザーと AWS アカウントの管理者に適用されます。また、アプリケーションのエンドユーザーにも適用されます。
SEC 3: プログラムによるアクセスをどのように制御していますか?
適切に定義、制限、分離されたアクセス権によってプログラムによるアクセス、または自動アクセスを制御することにより、不正アクセスのリスクを軽減します。プログラムによるアクセスには、ワークロード内部でのアクセスや、AWS 関連リソースへのアクセスが含まれます。

すべてのユーザーやシステムが認証情報を共有してはいけません。ユーザーアクセス権は、パスワード要件や MFA の強制などのベストプラクティスを実践した上で、最小権限で与えられるべきです。AWS のサービスに対する API コールなど、プログラムによるアクセスを、AWS Security Token Service などが発行する、権限が制限された一時的な認証情報を使用して実行できます。

AWS では、Identity and Access Management に役立つリソースを提供しています。ベストプラクティスを学ぶには、Managing Credentials & Authentication、Control Human Access、Control Programmatic Access のハンズオンラボを参照してください。

発見的統制

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

AWS では、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、発見的統制を実装できます。CloudTrail ログ、AWS API

コール、CloudWatch により、メトリクスのモニタリングとアラーム設定を行い、AWS Config で設定履歴を確認できます。Amazon GuardDuty はマネージド型の脅威検出サービスです。悪意のある動作や不正な動作を継続的にモニタリングし、お客様が AWS のアカウントとワークロードを保護できるようにします。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のログをとることができます。

以下の質問は、セキュリティに関するこれらの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をどのように検出し、調査していますか？
ログやメトリクスからイベントを可視化して把握し、分析します。セキュリティイベントや潜在的な脅威に対して措置をとることで、ワークロードを保護します。
SEC 5: 新しいセキュリティ脅威に対してどのように防御していますか?
AWS と業界のベストプラクティスおよび脅威インテリジェンスに関する最新情報を入手すれば、新しいリスクを認識するのに役立ちます。これにより、脅威モデルを作成し、ワークロードの保護に役立つ適切なコントロールを特定、優先順位付け、および実装することが可能になります。

ログ管理は、セキュリティやフォレンジックから規制要件や法的要件まで十分に対応できる、優れたアーキテクチャを設計するために重要です。潜在的なセキュリティインシデントを特定するために、ログの分析とそれに対する対応は特に重要です。AWS には、データ保持期間を定義したり、データを保持、アーカイブ、または最終的に削除する場所を定義したりする機能があるため、これによりログ管理を簡単に実装できます。予測可能で信頼性の高いデータ処理が、さらに簡単かつ費用対効果の高いものになります。

インフラストラクチャ保護

インフラストラクチャ保護には、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、深層防御などの制御手段が含まれています。これらの手段を用いることは、クラウドやオンプレミスの環境で滞りなく運用していくために特に重要です。

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

以下の質問は、セキュリティに関するこれらの考慮事項に焦点を当てています。

SEC 6: ネットワークをどのように保護していますか?
パブリックおよびプライベートネットワークでは、内部および外部の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 7: コンピューティングリソースをどのように保護していますか?
ワークロード内のコンピューティングリソースを内外の脅威から守るには、複数の防御レイヤーを設ける必要があります。コンピューティングリソースには、EC2 インスタンス、コンテナ、AWS Lambda 関数、データベースサービス、IoT デバイスなどがあります。

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

AWS ユーザーは、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon EC2 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 Machine Image (AMI) を作成できます。そして、この AMI を使用して起動するすべての新しい仮想サーバー (インスタンス) は、Auto Scaling でトリガーするか手動で起動して、その強化した設定を引き継ぐことができます。

データ保護

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

AWS では、以下のプラクティスによりデータの保護を支援します。

- AWS のお客様は、お客様のデータの完全なコントロールを維持します。
- AWS により、データを暗号化したり、キーの定期的なローテーションなどによってキーを管理したりすることが容易になり、そうした作業を AWS により自動化したり、お客様がメンテナンスしたりすることができます。
- ファイルのアクセスや変更などの重要な内容を含む詳細なログを記録できます。
- AWS には優れた弾力性を持つストレージシステムがあります。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により、間違って上書きしたり削除したりしてデータが損なわれることを防ぎます。
- AWS ではリージョン間のデータの移動は発生しません。1 つのリージョンにあるコンテンツは、リージョン間の移動を可能にする機能を明示的に有効にしたり、その機能を提供するサービスを使用したりしない限りは、そのリージョンにとどまります。

以下の質問は、セキュリティに関するこれらの考慮事項に焦点を当てています。

SEC 8: データをどのように分類していますか?

分類とは、機密レベルに応じてデータを分けることです。これにより、どのような管理手法を用いてデータを保護および保持すべきかを判断できます。

SEC 9: 保管中のデータをどのように保護していますか?

要件を定義して暗号化など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、不正アクセスや喪失のリスクを軽減できます。

SEC 10: 伝送中のデータをどのように保護していますか?

要件を定義し、暗号化などのコントロールを実装して伝送中のデータを保護すれば、不正アクセスや漏洩のリスクを軽減できます。

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

インシデント対応

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

AWS では、以下のプラクティスにより効果的なインシデント対応を支援します。

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

以下の質問は、セキュリティに関するこれらの考慮事項に焦点を当てています。

SEC 11: セキュリティインシデントにどのように対応していますか?
セキュリティインシデントを適切なタイミングで調査および対応し、組織の中断を最小限に抑えられるようにするには、準備が重要です。

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

主要な AWS のサービス

セキュリティに不可欠なAWS のサービスは AWS Identity and Access Management (IAM) であり、これにより、AWS のサービスとリソースへのユーザーアクセスを保護して管理できます。以下のサービスと機能では、セキュリティの 5 つの分野がサポートされます。

- アイデンティティ管理とアクセス管理: IAM により、AWS のサービスとリソースへのアクセスを保護して管理できます。MFA は、ユーザーアクセスに保護レイヤーを追加します。AWS Organizations により、複数の AWS アカウントのポリシーを一元的に管理・強制できます。
- 発見的統制: AWS CloudTrail は AWS API コールを記録し、AWS Config は AWS のリソースと設定の詳細なインベントリを提供します。Amazon GuardDuty は、悪意のある動作や不正な動作を継続的にモニタリングする、マネージド型の脅威検出サービスです。Amazon CloudWatch は、CloudWatch Events をトリガーしてセキュリティ対応を自動化できる、AWS リソースのモニタリングサービスです。
- インフラストラクチャ保護: Amazon Virtual Private Cloud (Amazon VPC) を使用して、お客様が定義した仮想ネットワーク内で AWS リソースを起動できます。Amazon CloudFront は、DDoS を緩和する AWS Shield に統合されたビューワーに対して、データ、動画、アプリケーション、API を安全に提供する、グローバルコンテンツ配信ネットワークです。AWS WAF は、ウェブの一般的な脆弱性からウェブアプリケーションを保護するために役立つ、Amazon CloudFront または Application Load Balancer にデプロイされたウェブアプリケーションファイアウォールです。
- データ保護: ELB、Amazon Elastic Block Store (Amazon EBS)、Amazon S3、Amazon Relational Database Service (Amazon RDS) などのサービスには、伝送中および保管中のデータを保護する暗号化機能が含まれています。Amazon Macie は、機密性の高いデータを自動で発見し、分類し、保護します。AWS Key Management Service (AWS KMS) は、暗号化に使用するキーの作成と管理を容易にします。
- インシデント対応: IAM は、インシデント対応チームと対応ツールに適切な承認を与えるために使用します。AWS CloudFormation は、調査を実施する際の信頼できる環境やクリーンルームを作成するために使用できます。Amazon CloudWatch Events を使用すれば、AWS Lambda などの自動化されたレスポンスをトリガーするルールを作成できます。

リソース

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

ドキュメント

- AWS Cloud Security
- AWS Compliance
- AWS Security Blog

ホワイトペーパー

- Security Pillar
- AWS Security Overview
- AWS Security Best Practices
- AWS Risk and Compliance

動画

- AWS Security State of the Union
- Shared Responsibility Overview

信頼性

信頼性の柱には、インフラストラクチャやサービスの中断から復旧し、需要に適したコンピューティングリソースを動的に獲得し、誤設定や一時的なネットワークの問題といった中断の影響を緩和する能力 が含まれます。

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。

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。

- 復旧手順をテストする: オンプレミス環境においては、多くの場合、システムが特定のシナリオで動作することを実証するためにテストが実施されます。復旧戦略を検証するためにテストを実施することはあまりありません。クラウドでは、どのようにシステムの障害が発生するかをテストでき、復旧の手順も検証できます。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。これによって障害が発生する経路が明らかとなり、実際の障害が発生する前にテストおよび修正できるため、テストしたことがないコンポーネントに障害が発生するリスクを軽減できます。

- 障害から自動的に復旧する: システムで主要業績評価指標 (KPI) をモニタリングすることで、しきい値を超えた場合にオートメーションをトリガーできます。これによって障害発生時の自動通知と追跡が可能になり、障害に対処する、または障害を修正するための復旧プロセスを自動化できます。オートメーションをさらに高機能にすれば、障害が実際に発生する前に予想して修正することも可能です。
- 水平方向にスケールしてシステム全体の可用性を高める: 1 つの大規模なリソースを複数の小規模なリソースに置き換えることで、単一の障害がシステム全体に与える影響を軽減します。リクエストを複数の小規模なリソースに分散させることで、共通の障害点を共有しないようにします。
- キャパシティーを推測しない: オンプレミスのシステムにおける一般的な障害原因の 1 つは、リソースの飽和状態です。これは、システムに対する需要がシステムのキャパシティーを超えるときに生じます (この状態はしばしばサービス拒否攻撃の対象になります)。クラウドでは、需要とシステムの使用率をモニタリングしてリソースの追加や削除を自動化し、需要を満たすために最適なレベルを維持できます。プロビジョニングが超過または不足することはありません。
- オートメーションで変更を管理する: インフラストラクチャに対する変更は、オートメーションを使用して実行する必要があります。管理が必要な変更は、オートメーションに対する変更です。

定義

クラウドでの信頼性には、3 つのベストプラクティスの分野があります。

- 基盤
- 変更管理
- 障害の管理

高い信頼性を達成するため、システムの基盤について十分に計画し、モニタリングを実施する必要があります。需要や要件の変更に対応するためのメカニズムも必要です。障害を検出し、自動的に修復できるシステムを設計することが必要です。

ベストプラクティス

基盤

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。このよう

な要件は、単一のプロジェクトの範囲を超えているため、無視される場合があります。この点を無視すると、システムの信頼性の達成に多大な影響が及びます。オンプレミス環境では、依存関係により、このような要件が長いリードタイムの原因となる可能性があるため、初期計画に組み込んでおく必要があります。

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おり、必要に応じて変更できます。クラウドは基本的に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という要件を満たすことは AWS の責任であり、お客様はストレージデバイスのサイズといった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、信頼性に関するこれらの考慮事項に焦点を当てています。(信頼性に関する質問、およびベストプラクティスの一覧については、付録を参照してください。).

REL 1: サービス制限をどのように管理していますか？
必要以上のリソースが偶発的にプロビジョニングされることを防止するため、デフォルトのサービス制限が設けられています。また、不正利用などからサービスを保護するため、API オペレーションを呼びだす頻度も制限されています。AWS Direct Connect を使用している場合、接続ごとに転送できるデータの量にも制限があります。AWS Marketplace のアプリケーションを使用している場合、アプリケーションの制限を理解する必要があります。サードパーティーのウェブサービスやサービスとしてのソフトウェアを使用している場合も、これらのサービスに設けられている制限を認識しておく必要があります。
REL 2: ネットワークトポロジをどのように管理していますか?
アプリケーションは、既存のデータセンターインフラストラクチャ、公開されているパブリッククラウドインフラストラクチャ、またはプライベートアドレスが割り当てられたパブリッククラウドインフラストラクチャなど、1 つまたは複数の環境に存在可能です。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、プライベートアドレスの管理、名前解決などのネットワークの考慮点は、クラウド上のリソースを利用する上で基本的なものです。

AWS では、お客様が意図せずにリソースを過剰にプロビジョニングすることのないよう、サービスの制限を設定しています (チームが要求できる各リソースの数の上限)。ビジネスのニーズに合わせて制限をモニタリングし、変更するためのガバナンスとプロセスを用意することが必要になります。クラウドを導入する際に、既存のオンプレミスリソースと統合する計画が必要になる場合があります (ハイブリッドアプローチ)。ハイブリッドモデルでは、時間をかけて徐々にオールインクラウドに移行できます。そのため、AWS とオンプレミスのリソースがネットワークトポロジとしてどのように作用し合うかを考えて設計することが重要です。

変更管理

変更がシステムに及ぼす影響に注意していれば、事前に計画できるようになります。また、モニタリングによってキャパシティーの問題や SLA 違反につながりかねない傾向をすばやく識別できます。従来の環境では、変更管理プロセスはしばしば手動で行われ、誰がいつ変更するかを効果的に制御するには、監査との注意深い連携が必要です。

AWS を使用すると、システム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、システムを使用するユーザーが増えた場合には、サーバーを自動で追加できます。システム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、信頼性に関するこれらの考慮事項に焦点を当てています。

REL 3: システムが需要の変化にどのように対応していますか？
スケーラブルなシステムはリソースを自動的に追加したり削除したりできる伸縮性を備えているため、いつでも最新の需要に厳密に適合します。
REL 4: リソースをどのようにモニタリングしていますか？
ログとメトリクスは、ワークロードの状態に関する洞察を得るための強力なツールです。ワークロードは、しきい値を超えたり重大なイベントが発生したりしたときに、ログとメトリクスがモニタリングされて通知が送信されるように構成できます。ワークロードは、低パフォーマンスのしきい値を超えたり障害が発生したりしたときに、自動的に自己回復したり、スケールするように設計されているのが理想です。
REL 5: 変更をどのように実施していますか？
環境の変更を管理しないと、変更の影響を予測することが難しくなります。プロビジョニングされた AWS リソースとアプリケーションの変更制御は、アプリケーションと運用環境で既知のソフトウェアが実行されており、予測できる方法でパッチを適用または置換できることを確認するために必要です。

需要の変動に対応してリソースの追加や削除を自動で行うシステムを設計しておけば、信頼性が高まることに加え、ビジネスの成功が負担に変わってしまうことを避けられます。モニタリングを実行しておけば、KPI が予測された基準から逸脱すると、アラートが自動的にチームに送られます。環境の変更は自動的にログに記録されるため、アクションを監査して、信頼性に影響を与える可能性のあるアクションを特定できます。変更管理をコントロールすることで、必要な信頼性を実現するためのルールに効力を持たせることができます。

障害の管理

どのようなシステムでも、ある程度複雑になると障害が発生することが予想されます。そのため、それらの障害をどのように検出して対応し、再発を防止するかが問題です。

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

以下の質問は、信頼性に関するこれらの考慮事項に焦点を当てています。

REL 6: データをどうバックアップするか?
データ、アプリケーション、動作環境 (アプリケーションで設定したオペレーティングシステムとして定義) をバックアップして、平均復旧時間 (MTTR) と目標復旧地点 (RPO) の要件を満たします。
REL 7: どのようにしてシステムがコンポーネントのエラーに耐えるか?
ワークロードに関して顕在または潜在する条件として、高い可用性と短い平均復旧時間 (MTTR) を必要とする場合は、ワークロードを弾力性のある設計にし、障害に耐えるようワークロードを分散します。
REL 8: 弾力性をどのようにテストしていますか?
ワークロードの弾力性をテストして、本番環境でしか表面化しない潜伏バグを見つけられるようにします。こういったテストを定期的に行います。
REL 9: 災害対策をどのように計画していますか?
災害対策 (DR) は、バックアップからデータを復元する際に不可欠です。RTO と RPO の目標と一致するように目標、リソース、場所、復元するデータの機能を定義し、実行する必要があります。

定期的にデータをバックアップして、バックアップファイルをテストすることで、論理的および物理的なエラーから確実に復旧できるようになります。障害の管理で重要なのは、システムに対し自動化されたテストを頻繁に実施して障害を発生させ、どのように復旧するかを確認することです。そのためには、このようなテストは定期的なスケジュールでも大きなシステム変更後にトリガーされます。KPI を積極的に追跡し、目標復旧時間 (RTO)、目標復旧時点 (RPO)、システムの回復力を評価する (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、システム復旧プロセスを徹底的にテストして、問題が持続する場合でも、すべてのデータを復旧し、サービスをお客様に提供し続けられることを確認することです。復旧プロセスも、通常の本番プロセスと同じように実施する必要があります。

主要な AWS のサービス

信頼性に不可欠なAWS のサービスは Amazon CloudWatch であり、はランタイムメトリクスをモニタリングします。以下のサービスと機能では、信頼性の 3 つの分野がサポートされます。

- 基盤: AWS IAM により、AWS のサービスとリソースへのアクセスを保護して管理できます。Amazon VPC では、AWS クラウドのプライベートで孤立したセクションをプロビジョニングできます。ここでは、お客様が定義する仮想ネットワークで AWS リソースを起動できます。サービスの制限は AWS Trusted Advisor で可視化できます。AWS Shield はマネージド型分散サービス妨害攻撃 (DDoS) 防止サービスで、AWS で実行されるウェブアプリケーションを保護します。
- 変更管理: AWS CloudTrail ではアカウントの AWS API コールが記録され、監査のためにログファイルが送られます。AWS Config は AWS のリソースと設定の詳細なイ

ンベントリを提供し、設定内容の変更を継続的に記録します。Amazon Auto Scaling はデプロイされたワークロードに対して需要管理の自動化を実現するサービスです。Amazon CloudWatch には、カスタムメトリクスなどのメトリクスに基づいてアラートを送る機能があります。また、Amazon CloudWatch にはロギング機能があり、複数のリソースのログファイルを集約することができます。

- 障害の管理: AWS CloudFormation には、AWS リソースを作成し、予測可能な方法で順番にプロビジョニングするためのテンプレートが用意されています。Amazon S3 はバックアップの保存先として耐久性の高いサービスです。Amazon Glacier には耐久性の高いアーカイブを保存できます。AWS KMS は信頼性の高いキー管理システムで、AWS のサービスの多くと統合されています。

リソース

信頼性に関する AWS 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

- Service Limits
- Service Limits Reports Blog
- Amazon Virtual Private Cloud
- AWS Shield
- Amazon CloudWatch
- Amazon S3
- AWS KMS

ホワイトペーパー

- Reliability Pillar
- Backup Archive and Restore Approach Using AWS
- Managing your AWS Infrastructure at Scale
- AWS Disaster Recovery
- AWS Amazon VPC Connectivity Options

動画

- How do I manage my AWS service limits?
- Embracing Failure: Fault-Injection and Service Reliability

製品

- AWS Premium Support
- Trusted Advisor

パフォーマンス効率

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

設計原則

クラウドでのパフォーマンス効率には、5 つの設計原則があります。

- 最新テクノロジーの標準化: 実装が難しいテクノロジーでも、クラウドベンダーが必要な知識を提供して複雑さを解消すれば、簡単に利用できるようになります。IT チームは、新しいテクノロジーをホストおよび実行する方法を学習するのではなく、単にサービスとしてそのテクノロジーを利用できます。例えば、NoSQL データベース、メディア変換、機械学習はいずれも専門知識が必要なテクノロジーであり、その専門知識は技術者のコミュニティに均等に普及しているわけではありません。クラウドでこのようなテクノロジーは IT チームが利用できるようサービス化されており、IT チームはリソースのプロビジョニングや管理ではなく製品の開発に集中できます。
- 数分でグローバルに展開: わずか数回のクリックで、世界中の複数のリージョンにシステムを簡単にデプロイできます。これにより、最低限のコストでレイテンシーを短縮し、お客様へのサービスを向上させることができます。
- サーバーレスアーキテクチャを使用: クラウドでは、サーバーレスのアーキテクチャによって、従来のコンピューティングアクティビティを実行するためにサーバーの運用や管理を行う必要はありません。例えば、ストレージサービスが静的ウェブサイトとして動作するため、ウェブサーバーは必要なくなり、イベントサービスによってユーザーのコードをホストできます。これにより、ウェブサーバーを管理するための運用負担がなくなり、トランザクションコストを削減できます。このようなマネージド型サービスはクラウドスケールで運用されるためです。
- より頻繁に実験可能: 仮想的で自動化できるリソースを使うことで、異なるタイプのインスタンス、ストレージ、設定を使用した比較テストを簡単に実施できます。
- システムを深く理解: 実現しようとすることに最も適した技術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について考慮します。

定義

クラウドでのパフォーマンス効率には、4 つのベストプラクティスの分野があります。

- 選択
- レビュー
- モニタリング
- トレードオフ

高性能なアーキテクチャを選択する際は、データ駆動型のアプローチを選択します。ハイレベルな設計から、リソースタイプの選択と設定に至るまで、アーキテクチャのあらゆる側面に関するデータを収集してください。選択した内容を定期的にレビューすることで、絶えず進化し続けている AWS クラウドを活用できているかを確認できます。モニタリングを実施すれば、予想したパフォーマンスからのずれを把握し、その対策を講じることができます。さらに、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ることができます。

ベストプラクティス

選択

システムにとって最適なソリューションは、お客様のワークロードの種類に応じて異なります。多くの場合、複数のアプローチを組み合わせて使用します。優れた設計のシステムでは複数のソリューションが使用され、さまざまな機能によってパフォーマンスが改善されます。

AWS では、リソースは仮想化され、さまざまな種類や設定を持つリソースとして使用できます。これにより、お客様のニーズに密接に適合したアプローチを見つけやすくなるとともに、オンプレミスのインフラストラクチャでは簡単に実現できないオプションを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これらの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、およびベストプラクティスの一覧については、付録を参照してください。).

PERF 1: 最も良いパフォーマンスのアーキテクチャをどのように選択していますか？
多くの場合、ワークロード全体で最適なパフォーマンスを得るには、複数のアプローチが必要です。Well-architected のシステムでは複数のソリューションが使用され、さまざまな機能によってパフォーマンスが改善されます。

アーキテクチャに導入するパターンと実装を選択する場合は、データ駆動型のアプローチを使用して最適なソリューションを見つけてください。AWS ソリューションアーキテクト、AWS リファレンスアーキテクチャ、および AWS パートナーネットワーク (APN) のパートナーは、学んだことに基づいて、お客様がアーキテクチャを選択できるようサポートしますが、お客様のアーキテクチャを最適化するには、ベンチマークや負荷テストで得られたデータが必要になります。

アーキテクチャでは通常、アーキテクチャに関するさまざまなアプローチが組み合わされて使用されます (イベント駆動型、ETL、パイプラインなど)。アーキテクチャを実装する場合は、アーキテクチャのパフォーマンスを最適化するための AWS のサービスが使用されます。以下のセクションでは、お客様が検討すべき 4 つの主なリソースタイプ (コンピューティング、ストレージ、データベース、ネットワーク) について説明します。

コンピューティング

システムにとって最適なコンピューティングソリューションは、アプリケーションの設計、使用パターン、設定に応じて異なります。各アーキテクチャでは、コンポーネントごとに異なるコンピューティングソリューションが使用されることもあり、パフォーマンスを向上させるための機能も異なります。アーキテクチャに対して適切でない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場合があります。

AWS では、インスタンス、コンテナ、ファンクションの形式でコンピューティングが利用できます。

- インスタンスは仮想化されたサーバーなので、ボタンをクリックしたり、API コールしたりすることで機能を変更できます。クラウドでは、どのリソースを使用するかは固定されていないため、さまざまな種類のサーバーを実験できます。AWS では、さまざまなファミリーとサイズの仮想サーバーインスタンスが用意されており、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) といったさまざまな機能が提供されています。
- コンテナは、リソースが分離されたプロセスで、アプリケーションとその依存関係を実行できる、オペレーティングシステムを仮想化する方法です。

- ファンクションは、実行するコードに基づいて、実行環境を抽象化します。例えば、AWS Lambda を使用すれば、インスタンスを実行することなくコードを実行できます。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 2: コンピューティングソリューションをどのように選択していますか?

システムにとって最適なコンピューティングソリューションは、アプリケーションの設計、使用パターン、設定に応じて異なります。各アーキテクチャでは、コンポーネントごとに異なるコンピューティングソリューションが使用される可能性があるため、パフォーマンスを向上させるための機能も異なります。アーキテクチャで使用されるコンピューティングソリューションを正しく選択しないと、パフォーマンス効率が低下するおそれがあります。

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

ストレージ

システムにとって最適なストレージソリューションは、アクセス方法 (ブロック、ファイル、オブジェクト)、アクセスパターン (ランダム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。優れた設計のシステムでは複数のストレージソリューションが使用され、さまざまな機能によってパフォーマンスが改善されます。

AWS では、ストレージは仮想化され、さまざまな種類を持つストレージとして使用できます。これにより、ストレージ手法をより簡単かつより密接にお客様のニーズに適合させることができるとともに、オンプレミスのインフラストラクチャでは簡単に実現できないストレージオプションを提供できます。例えば Amazon S3 は、99.999999999% (イレブンナイン) の耐久性を実現するよう設計されています。また、磁気ハードディスク (HDD) から SSD に変更することもできます。さらには、あるインスタンスから別のインスタンスに仮想ドライブを秒単位で簡単に移動することもできます。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 3: ストレージソリューションをどのように選択していますか?

システムにとって最適なストレージソリューションは、アクセス方法 (ブロック、ファイル、オブジェクト)、アクセスパターン (ランダム、シーケンシャル)、必要なスループットやアクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、可用性と耐久性に関する制約によって異なります。優れた設計のシステムでは、複数のストレージソリューションを使用し、さまざまな機能を有効にしてパフォーマンスとリソースの使用効率を高めています。

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

データベース

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

Amazon RDS では、フルマネージド型のリレーショナルデータベースが用意されています。Amazon RDS を使用すると、データベースのコンピューティングリソースとストレージリソースをダウンタイムなしでスケールできます。Amazon DynamoDB は、あらゆる規模で 10 ミリ秒未満のレイテンシーを実現するフルマネージド型 NoSQL データベースです。Amazon Redshift は、ペタバイト規模のマネージド型データウェアハウスで、パフォーマンスや容量のニーズの変化に応じてノードの数や種類を変更できます。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 4: データベースソリューションをどのように選択していますか。

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

ワークロードのデータベースアプローチ (RDBMS、NoSQL) はパフォーマンス効率に大きく影響しませんが、データ駆動型のアプローチよりも、組織の方針に基づいて選択されることが多くあります。ストレージと同様、ワークロードのアクセスパターンを検討することが重要です。また、データベースではないソリューション (検索エンジンやデータウェアハウスなど) を使用してより効率的に問題を解決できないか検討することも重要です。

ネットワーク

システムにとって最適なネットワークソリューションは、レイテンシーやスループットなどの要件によって異なります。場所のオプションはユーザーリソースやオンプレミスリソースなどの物理的制約の影響を受けますが、最新技術を使用することやリソースを置き換えることで、こうした影響を軽減できます。

AWS では、ネットワークは仮想化され、さまざまな種類や構成を持つネットワークとして使用できます。これにより、ネットワーク手法をより簡単かつより密接にお客様のニーズに適合させることができます。AWS では、拡張ネットワーク、Amazon EBS 最適化インスタンス、Amazon S3 Transfer Acceleration、動的 Amazon CloudFront など、ネットワークトラフィックを最適化するための製品機能を提供しています。また、Amazon Route 53 のレイテンシールーティング、Amazon VPC エンドポイン

ト、AWS Direct Connect など、ネットワーク距離を短縮しジッターを削減するネットワーキング機能も提供しています。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 5: ネットワークソリューションをどのように選択していますか？

システムにとって最適なネットワークソリューションは、レイテンシー、スループット要件などによって異なります。場所のオプションはユーザーリソースやオンプレミスリソースなどの物理的な制約の影響を受けますが、エッジ技術やリソースプレイスメントを使うことでカバーできます。

ネットワークソリューションを選択する際は、場所を検討する必要があります。AWS を使用すれば、使用される場所に近い所にリソースを配置できるため、ネットワークの距離を縮めることができます。リージョンや、プレイスメントグループ、エッジロケーションを活用すれば、パフォーマンスを大幅に向上させることができます。

レビュー

ソリューションを設計するときは、選択できるオプションが限られていても、時間が経つにつれ、アーキテクチャのパフォーマンスを向上させることができる新しいテクノロジーやアプローチが利用できるようになります。

AWS を使用すれば、お客様のニーズに応じて絶えず進化している革新的なテクノロジーを利用できます。AWS では、新しいリージョンやエッジロケーション、サービス、機能を定期的にリリースしています。これらを利用すれば、お客様のアーキテクチャのパフォーマンス効率を向上させることができます。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 6: ワークロードを進化させるためにどのように新機能を取り込んでいますか？

ワークロードを設計する際に選択できるオプションには限りがありますが、時間が経つにつれ、ワークロードのパフォーマンスの向上に役立つ新しいテクノロジーやアプローチを利用できるようになります。

アーキテクチャのどの部分でパフォーマンスが制約されているかを把握することで、そうした制約を緩和できるリリースを見つけることができます。

モニタリング

アーキテクチャの実装後は、お客様が発見する前に問題を修正できるよう、アーキテクチャのパフォーマンスをモニタリングする必要があります。モニタリングメトリクスを使用して、メトリクスがしきい値を超えたときにアラームを発生させるようにします。アラームを使用すれば、正しく動作していないコンポーネントが見つかったときに、そのコンポーネントを回避するための自動アクションをトリガーできます。

Amazon CloudWatch では、通知アラームのモニタリングと送信を行う機能が用意されています。自動化機能を使用すれば、Amazon Kinesis、Amazon Simple Queue Service

(Amazon SQS)、AWS Lambda を通じてアクションをトリガーでき、パフォーマンスの問題を回避できます。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 7: リソースが正常に動作していることを確認するためにどのようにモニタリングしていますか？
システムのパフォーマンスは徐々に低下することがあります。システムのパフォーマンスをモニタリングして低下の兆候を見つけ、オペレーティングシステムやアプリケーション負荷などの内部的および外部的な要素を修正します。

効果的にモニタリングするには、大量の誤検出を発生させたり、大量のデータに振り回されたりしないことが重要です。自動化されたトリガーを使用すれば、ヒューマンエラーを防ぐことができ、問題解決までの時間を短縮できます。アラームソリューションをテストするシミュレーションを本番環境で実行するゲームデーを計画し、そのソリューションが問題を正しく認識するか確認してください。

トレードオフ

ソリューションのアーキテクチャを設計する場合は、トレードオフを考慮して最適なアプローチを選択します。より高いパフォーマンスを実現できるように、整合性、耐久性、容量を重視するのか、時間またはレイテンシーを重視するのかを、お客様の状況に応じてトレードオフしてください。

AWS を使用すれば、数分で世界各地にリソースを展開して、エンドユーザーに近い場所にリソースをデプロイできます。また、データベースシステムなどの情報ストアに読み取り専用のレプリカを動的に追加できるため、プライマリデータベースの負荷を減らすことができます。AWS では、Amazon ElastiCache などのキャッシュソリューションも用意されています。このソリューションでは、インメモリデータストア (またはインメモリキャッシュ) と Amazon CloudFront が用意されており、お客様の静的コンテンツをエンドユーザーに近い場所にキャッシュできます。Amazon DynamoDB Accelerator (DAX) では、Amazon DynamoDB の前面にリードスルー/ライトスルー分散キャッシュ層が設けられています。また、DAX では、Amazon DynamoDB と同じ API がサポートされていますが、キャッシュ内にあるエンティティに対して 1 ミリ秒未満のレイテンシーが実現されています。

以下の質問は、パフォーマンス効率に関するこれらの考慮事項に焦点を当てています。

PERF 8: パフォーマンスを向上させるために、トレードオフをどのように利用していますか？

アーキテクチャの設計にあたって、最適なアプローチとなるトレードオフについて積極的に考慮します。多くの場合、整合性、耐久性、時間とレイテンシーと引き換えに、パフォーマンスを向上することが可能です。

トレードオフを行うことで、アーキテクチャが複雑になる可能性があります。また、トレードオフを行った場合は、ロードテストを実施して、トレードオフによって目に見える効果が得られたか確認する必要があります。

主要な AWS のサービス

パフォーマンス効率に不可欠なAWS のサービスは Amazon CloudWatch であり、リソースとシステムをモニタリングし、全体的なパフォーマンスと動作状況を可視化します。以下のサービスと機能では、パフォーマンス効率の 4 つの分野がサポートされます。

- 選択
  - コンピューティング: 需要に対応し、応答性を維持するための十分なインスタンスを確保するには、Auto Scaling が重要です。
  - ストレージ: Amazon EBS では、ユースケースを最適化できる幅広いストレージオプション (SSD、プロビジョニングされた 1 秒あたりの入力/出力オペレーション数 (PIOPS) など) をご利用いただけます。Amazon S3 では、サーバーレスのコンテンツ配信機能が用意されています。また、Amazon S3 Transfer Acceleration を使用すれば、すばやく簡単かつセキュアにファイルを長距離転送できます。
  - データベース: Amazon RDS では、ユースケースを最適化できる幅広いデータベース機能が用意されています (PIOPS やリードレプリカなど)。Amazon DynamoDB は、あらゆる規模で 10 ミリ秒未満のレイテンシーを実現します。
  - ネットワーク: Amazon Route 53 では、レイテンシーベースのルーティングが用意されています。Amazon VPC エンドポイントおよび AWS Direct Connect を使用すれば、ネットワークの距離を縮めたり、ジッターを減らしたりできます。
- レビュー: AWS ブログと AWS ウェブサイトの「最新情報」セクションで、新たに利用できるようになった機能とサービスを確認できます。
- モニタリング: Amazon CloudWatch で提供されているメトリクスやアラーム、通知をお客様の既存のモニタリングソリューションに組み込んだり、AWS Lambda と組み合わせてアクションをトリガーしたりできます。
- トレードオフ: Amazon ElastiCache、Amazon CloudFront、AWS Snowball は、パフォーマンスの向上を可能にするサービスです。Amazon RDS でレプリカを読み込めば、読み込み負荷が高いワークロードをスケーリングできます。

リソース

パフォーマンス効率に関する AWS 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

- Amazon S3 Performance Optimization
- Amazon EBS Volume Performance

ホワイトペーパー

- Performance Efficiency Pillar

動画

- AWS re:Invent 2016: Scaling Up to Your First 10 Million Users (ARC201)
- AWS re:Invent 2017: Deep Dive on Amazon EC2 Instances

コスト最適化

コスト最適化の柱には、最も低い価格でシステムを運用してビジネス価値を実現する能力 が含まれます。

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

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。

- 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費用のみを支払い、複雑な予測をすることなく、ビジネス要件に応じて使用量を増減できます。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かかったコストを測定します。この測定値を使用して、生産性を向上し、コストを削減することで得られる利点を把握します。
- データセンター運用のための費用を排除する: AWS を使用すると、サーバーの設置、積み上げ、電力供給といった手間のかかる作業が不要になるため、IT インフラストラクチャではなく顧客と組織のプロジェクトに焦点を当てることができます。

- 費用を分析し、帰結させる: クラウドでは、システムの使用状況とコストを正確に特定し、IT コストと各ワークロードの所有者との帰属関係を簡単に透明にできます。これによって投資収益率 (ROI) を把握でき、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。
- アプリケーションレベルのマネージドサービスを使用して所有コストを削減する: クラウドでは、アプリケーションレベルのマネージドサービスを使用することで、E メールの送信やデータベースの管理といったタスクのためにサーバーを維持する運用負担がなくなります。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ごとのコストを削減できます。

定義

クラウドでのコスト最適化には、4 つのベストプラクティスの分野があります。

- 費用認識
- 費用対効果の高いリソース
- 需要と供給を一致させる
- 長期的な最適化

他の柱と同様、検討する必要があるトレードオフも存在します。たとえば、優先するのは、市場投入までのスピードですか、それともコストですか? 市場投入スピードを短縮する、新しい機能を導入する、締め切りに間に合わせるといったケースでは、前払いコストの投資を最適化するよりも、スピードを優先した方が望ましい結果となります。長期的にコストを最適化できるワークロードのベンチマークの選定に時間を掛けるよりも、「万一の場合」の備えを過度に重視してしまう傾向が常にあるため、設計についての決定が実際のデータに基づかずに、性急に行われることがあります。その結果として、多くの場合、非常に過剰なプロビジョニングで、最適化の不十分なデプロイを行い、ライフサイクルが終わるまでその問題が残ってしまうのです。以下のセクションでは、デプロイにおける初期段階でのコスト最適化と継続的なコスト最適化について、そのテクニックと戦略的ガイダンスを紹介します。

ベストプラクティス

費用認識

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

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

AWS では、Cost Explorer を利用し、費用を追跡して実際の内訳を把握できます。AWS Budgets を使用すると、使用量やコストが予想と一致しない場合に通知を送ることができます。リソースへのタグ付けを使用すると、使用状況やコストにビジネスや組織の情報を付けることができ、組織の観点から最適化についてさらに深く理解できます。

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

COST 1: 使用状況をどのように管理しますか?
発生コストを適正な範囲内に抑えつつ、目的を確実に達成するためのポリシーとメカニズムを設定します。チェックアンドバランスのアプローチを採用することで、無駄なコストを費やすことなくイノベーションが可能です。
COST 2: 使用状況とコストをどのようにモニタリングしますか?
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 3: 不要なリソースをどのように削除しますか？
プロジェクトの開始から終了まで変更管理とリソース管理を実装します。これにより、使用されていないリソースをシャットダウンまたは終了して、無駄を減らします。

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

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。請求アラートを設定して、予測される過剰なコストの通知を受けることができます。また、AWS 簡易見積りツールを使用して、データ転送コストを計算できます。

費用対効果の高いリソース

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

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

ニーズに最も適した方法で 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が良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最小コミットメントの定めがなく、コンピューティングキャパシティーに対して 1 時間単位で料金が発生します。リザーブドインスタンスキャパシティーを予約することで、最大 75% のコストを削減できます。スポットインスタンスでは、使用されていない Amazon EC2 キャパシティーを活用し、オンデマンド料金と比べて最大 90% 節約できます。スポットインスタンスステートレスなウェブサーバー、バッチ処理など、サーバー群の中で個々のサーバーが動的に追加または削除されることを許容するシステムや、HPC とビッグデータの使用に適しています。

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

以下の質問は、コスト最適化に関するこれらの考慮事項に焦点を当てています。

COST 4: サービスを選択するとき、どのようにコストを評価しますか?
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、管理や運用によって発生するオーバーヘッドを削減またはゼロにでき、アプリケーション開発やビジネス上の他の活動に注力できるようになります。
COST 5: リソースタイプとサイズを選択する際、どうすればコスト目標を達成できるでしょうか?
目の前にあるタスクに合わせて適切なリソースのサイズを選択するようにします。最もコスト効率の高いタイプとサイズを選択することで、無駄を最小限に抑えます。
COST 6: コストを削減するには、料金モデルをどのように使用したらよいでしょうか?
リソースのコストを最小限に抑えるのに最も適した料金モデルを使用します。
COST 7: データ転送料金についてどのように計画していますか？
データ転送料金を計画し、モニタリングすることで、これらのコストを最小化するためのアーキテクチャ上の決定を下すことができます。小規模でも効果的なアーキテクチャ変更により、長期的な運用コストを大幅に削減できる場合があります。

サービスの選択時にコストを考慮に入れ、Cost Explorer や AWS Trusted Advisor などのツールを使って定期的に AWS の使用状況を確認すると、使用状況をアクティブにモニタリングしてデプロイを適宜調整できます。

需要と供給を一致させる

需要と供給を最適に一致させることでワークロードのコストを最低限に抑えることができますが、プロビジョニングの時間と個々のリソースの障害に備えて十分に余裕を持った供給も必要です。需要が一定の場合も変動する場合もあり、管理に多大なコストがかからないようにメトリクスと自動化が必要です。

AWS では、需要に合わせてリソースを自動的にプロビジョンできます。Auto Scaling を使用すると、需要ベース、バッファーベース、時間ベースのアプローチによって、必要に応じたリソースの追加と削除が可能です。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。

以下の質問は、コスト最適化に関するこれらの考慮事項に焦点を当てています。

COST 8: リソースの供給と顧客の需要をどのように一致させていますか?

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、費用を掛けたすべてのものが活用されるようにし、使用率が著しく低いインスタンスが生じるのを回避します。利用が過剰でも過少でも偏りが生じると、運用コスト (利用過剰によるパフォーマンスの低下) または無駄な AWS 費用 (過剰なプロビジョニング) のいずれかで、組織に悪影響が及びます。

需要と供給を一致させる設計をする場合、使用パターンと新しいリソースのプロビジョニングにかかる時間をよく検討する必要があります。

長期的な最適化

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。

AWS のマネージドサービスによってワークロードを大幅に最適化できます。新しいマネージドサービスと機能が利用可能になったときに、こうしたサービスに気付くことが不可欠です。たとえば、Amazon RDS データベースを運用すると、Amazon EC2 で自分のデータベースを運用するより安価になる場合があります。

以下の質問は、コスト最適化に関するこれらの考慮事項に焦点を当てています。

COST 9: 新しいサービスをどのように評価していますか？

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選択をレビューし、現在でもコスト効率が最も優れているかどうかを確認することがベストプラクティスです。

デプロイを定期的に見直すときには、新しいサービスでどのようにコストを節約できるか評価します。たとえば、RDS で Amazon Aurora を使用すると、リレーショナルデータベースのコストを削減できます。

主要な AWS のサービス

コスト最適化に不可欠なツールは Cost Explorer であり、ワークロードと組織の全体にわたり、使用量を可視化して深く理解するのに役立ちます。以下のサービスと機能では、コスト最適化の 4 つの分野がサポートされます。

- 費用認識: AWS Cost Explorer を使用すると、詳細な使用状況の確認と追跡が行えます。AWS Budgets では、使用量や費用が実際の、または事前の予算範囲を超えたときに通知が送付されます。
- 費用対効果の高いリソース: リザーブドインスタンスのレコメンデーションに Cost Explorer を利用すると、これまでの AWS リソースに対する支払いパターンを確認できます。Amazon CloudWatch と Trusted Advisor を使用して、リソースを適切にサイジングします。RDS で Amazon Aurora を使用すると、データベースのライセンスコストを排除できます。AWS Direct Connect と Amazon CloudFront を、データ転送の最適化に使用できます。
- 需要と供給を一致させる: Auto Scaling を使用すると、浪費することなく、需要に対応するようにリソースの追加や削除を行うことができます。
- 長期的な最適化: AWS ニュースブログと AWS ウェブサイトの最新情報セクションには、新しくリリースされた機能やサービスについてご案内するリソースがあります。AWS Trusted Advisor で AWS 環境を検査すると、未使用のリソースやアイドル状態のリソースを削除することや、リザーブドインスタンスのキャパシティーを活用することで費用を節約する方法を見つけることができます。

リソース

コスト最適化に関する AWS 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

- Analyzing Your Costs with Cost Explorer
- AWS Cloud Economics Center
- AWS Detailed Billing Reports

ホワイトペーパー

- Cost Optimization Pillar

動画

- Cost Optimization on AWS

ツール

- AWS Total Cost of Ownership (TCO) Calculators
- AWS Simple Monthly Calculator

レビュープロセス

アーキテクチャの評価は非難せずに掘り下げることができる一貫した方法で実施される必要があります。これは何日もかけずに数時間で終わる軽いプロセスである必要があります。話し合いであり、監査ではありません。アーキテクチャレビューの目的は、対策を必要とする重大な問題や改善可能な領域を特定することです。レビュー結果は、お客様のワークロードの扱いやすさを改善する対策です。

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

AWS Well-Architectedは、AWS のシステムとサービスに関する内部評価方法と合致しています。これは設計方法を左右する設計原則と、根本原因の分析 (RCA) でよく取り上げられる領域が軽視されないようにするための質問に基づきます。内部システム、AWS のサービス、またはお客様に重大な問題があるときに、当社は RCA を検討し、使用している評価プロセスを改善できるかどうかを判断します。

変更が困難な一方通行のドア (のような決定) [1]を避けるため、設計の初期段階におけるレビューを実施します。また、本番運用前にもレビューを行います。本稼働開始後に新しい機能を追加したり、テクノロジーの実装を変更したりするにつれて、ワークロードは変化し続けます。ワークロードのアーキテクチャは時間とともに変わります。アーキテクチャを変えるにつれてアーキテクチャの特性が劣化しないように適切な予防策を取る必要があります。アーキテクチャを大幅に変更するときには、Well-Architected レビューを含めて、予防プロセスに従う必要があります。

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

会議を進行するために次のアイテムをお勧めします。

- ホワイトボードのある会議室
- 印刷した構成図や設計ノート

[1] 多くの決定は取り消しが可能な双方向の扉です。こうした決定には簡易なプロセスを適用します。一方通行のドア (のような決定) の場合は、取り消しが困難または不可能であるため、決定を下す前により詳細な検証が必要です。

- 回答に別途調査が必要な質問のアクションリスト (例えば「暗号化を有効にしましたか ?」)

レビューを完了すると、問題のリストが出来上がります。ビジネスの状況に応じてその優先順位を決めることができます。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。課題に早く対処することによって、繰り返す課題の解決にではなく、ビジネス価値の創造に費やす時間ができます。課題に対処しながらレビューを更新することで、アーキテクチャがどのように改善しているのかを知ることができます。

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

- 「忙しすぎる」 (チームが大きな仕事を始める準備をしているときによくある発言)
  - 大きな仕事を始める準備をしているならば、それが円滑に進む必要があります。レビューを実施することによって、見逃していたかもしれない問題を把握できます。
  - 製品ライフサイクルの早い段階でレビューを実施して、リスクを洗い出し、機能提供ロードマップに沿ったリスク軽減計画を立てることをお勧めします。
- 「結果が出ても対策をとる時間がない」 (スーパーボ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。 すべての課題に対策を講じることができないとしても、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。
- 「We don’t want others to know the secrets of our solution implementation!」
  - Well-Architected フレームワークの質問を示せば、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何度かレビューを実施すると、主題となる課題が見つかるかもしれません。例えば、特定の柱または主題に関して多くの課題を抱えているチームが複数あるかもしれません。すべてのレビューを総合的に検討し、それらの主題に関する課題に対処するのに役立つメカニズム、トレーニング、またはプリンシパルエンジニアリングの説明を見つける必要があります。

まとめ

AWS Well-Architected フレームワークの目的は、効率が良く、費用対効果が高く、安全で信頼のおけるクラウド対応システムを設計して運用するための 5 本柱のアーキテクチャに関するベストプラクティスを提供することです。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あります。このフレームワークをアーキテクチャに適用し、安定した効率のよいシステムを構築することにより、機能面の要件に注力できます。

寄稿者

本ドキュメントの執筆にあたり、次の人物および組織が寄稿しました。

- "Fitz" Philip Fitzsimons: Well-Architectedシニアマネージャー、アマゾン ウェブ サービス
- Brian Carlson: Well-Architected オペレーションズリード、アマゾン ウェブ サービス
- Ben Potter: Well-Architected セキュリティリード、アマゾン ウェブ サービス
- Rodney Lester: Well-Architected 信頼性リード、アマゾン ウェブ サービス
- John Ewart: Well-Architected パフォーマンスリード、アマゾン ウェブ サービス
- Nathan Besh: Well-Architected コストリード、アマゾン ウェブ サービス
- Jon Steele: Well-Architectedテクニカルアカウントマネージャー、アマゾン ウェブ サービス
- Ryan King: テクニカルプログラムマネージャー、アマゾン ウェブ サービス
- Erin Rifkin: シニアプロダクトマネージャー、アマゾン ウェブ サービス
- Max Ramsay: プリンシパルセキュリティソリューションズアーキテクト、アマゾン ウェブ サービス
- Scott Paddock: セキュリティソリューションズアーキテクト、アマゾン ウェブ サービス
- Callum Hughes: ソリューションズアーキテクト、アマゾン ウェブ サービス

その他の資料

AWS Well-Architected Partner program

AWS Well-Architected Tool

AWS Well-Architected homepage

Cost Optimization Pillar whitepaper

Operational Excellence Pillar whitepaper

Performance Efficiency Pillar whitepaper

Reliability Pillar whitepaper

Security Pillar whitepaper

ドキュメントの改訂

表2

主な改訂

日付	説明
2019 年 7 月	AWS Well-Architected Tool の追加、AWS Well-Architected Labs へのリンク、AWS Well-Architected パートナー、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。
2018 年 11 月	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 6 月	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。
2017 年 11 月	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。
2016 年 11 月	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。
2015 年 11 月	最新の Amazon CloudWatch Logs の情報に基づいて付録を更新しました。
2015 年 10 月	初版発行。

付録: 質問とベストプラクティス

運用上の優秀性

準備

OPS 1 優先順位はどのように決定すればよいでしょうか?

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

ベストプラクティス:

- 外部顧客のニーズを評価する: ビジネス、開発、運用チームを含む主要関係者と協力して、外部顧客のニーズに対するオペレーションの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。
- 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オペレーションの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。
- コンプライアンス要件を評価する: 法令遵守の要件や業界標準などの外的要因を評価して、特定の重点領域の必須化や重視が必要となる可能性のあるガイドラインや義務についてしっかりと認識してください。コンプライアンス要件が特定されていない場合は、必ずこの決定にデューデリジェンスを適用してください。
- 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、オペレーションの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。
- トレードオフを評価する: 競合する利益間のトレードオフの影響を評価し、オペレーションの重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、新しい機能を市場に出す速度を上げることが、コストの最適化より重視されることがあります。
- メリットとリスクを管理する: メリットとリスクを管理し、オペレーションの重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、重要な新機能を顧客に公開できるように、未解決の問題を記録するシステムをデプロイしておくと便利な場合があります。

OPS 2 ワークロードの状態を理解できるようにするには、ワークロードをどう設計すればよいでしょうか?

ワークロードを設計する際には、すべてのコンポーネントにわたって内部状態 (メトリクス、ログ、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。これにより、適時に必要な応答を提供できるようになります。

ベストプラクティス:

- アプリケーションテレメトリーを実装する: 内部状態、ステータス、およびビジネス成果の達成に関する情報が送出されるよう、アプリケーションコードをインストルメント化します。例えば、キューの長さ、エラーメッセージ、応答時間などの情報です。この情報を使用して、応答が必要とされるタイミングを特定します。
- ワークロードテレメトリーを実装して設定する: 内部状態や現在のステータスに関する情報が送出されるよう、ワークロードを設計および設定します。例えば、API 呼び出しボリューム、http ステータスコード、スケーリングイベントなどの情報です。この情報を使用して、応答が必要とされるタイミングを特定します。
- ユーザーアクティビティテレメトリーを実装する: ユーザーアクティビティに関する情報が送出されるよう、アプリケーションコードをインストルメント化します。例えば、クリックストリームのほか、開始、放棄、完了済みトランザクションなどの情報です。この情報を使用して、アプリケーションの使用のされ方や使用パターンを理解したり、応答が必要とされるタイミングを特定したりできます。
- 依存関係のテレメトリーを実装する: ワークロードが依存するリソースのステータスに関する情報が送出されるよう、ワークロードを設計および設定します。例えば、外部データベース、DNS、およびネットワーク接続性などの情報です。この情報を使用して、応答が必要とされるタイミングを特定します。
- トランザクショントレーサビリティを実装する: ワークロード全体のトランザクションフローに関する情報が送出されるよう、アプリケーションコードを実装し、ワークロードコンポーネントを設定します。この情報を使用して、応答が必要とされるタイミングを特定し、問題の原因の特定に役立てます。

OPS 3 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、バグ修正を可能にし、本番環境への変更のフローを改善するアプローチを採用します。これらにより、本番環境に採用される有益な変更を加速させ、デプロイされた問題を制限できます。またデプロイアクティビティを通じて挿入された問題をすばやく特定し、修復できます。

ベストプラクティス:

- 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。
- 変更をテストし、検証する: エラーの制限と検出に役立てるため、変更をテスト、検証します。手動プロセスによって発生するエラーと、テストにかかわる労力を減らすためにテストを自動化します。
- 構成管理システムを使用する: 設定の変更を行い、追跡するには、設定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。
- 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。
- パッチ管理を実行する: パッチ管理を実行し、問題を解決して、ガバナンスに準拠するようにします。パッチ管理の自動化により、手動プロセスによって発生するエラーと、パッチにかかわる労力を減らすことができます。
- 設計標準を共有する: チーム全体でベストプラクティスを共有し、デプロイ作業における利点の認識を高め、それを最大限にします。
- 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。たとえば、テスト駆動型デプロイ、コードレビュー、標準の導入などがあります。
- 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。コントロールレベルを上げて、ワークロードが意図したとおりに運用するように自信を付けます。
- 頻繁に、小さく、可逆的な変更を行う: 頻繁に、小さく、可逆的な変更を行うことで、変更の範囲と影響を減らします。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また変更を元に戻すこともできます。
- 統合とデプロイを完全自動化する: ワークロードの構築、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。

OPS 4 どのようにデプロイのリスクを軽減しますか?

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

ベストプラクティス:

- 変更の失敗に備える: 変更が望ましい結果をもたらさない場合に、既知の良好な状態に戻すか、本番環境で修正を行うことを計画します。この準備を行うことで、迅速な対応によって復旧時間を短縮できます。
- 変更をテストし、検証する: あらゆるライフサイクルステージで変更をテストし、その結果を検証することで、新しい機能を確認するとともに、デプロイの失敗のリスクと影響を最小限に抑えます。
- デプロイ管理システムを使用する: デプロイ管理システムを使用して変更を追跡および実装します。これにより、手動プロセスによって発生するエラーと、変更をデプロイする労力を減らすことができます。
- 限定的なデプロイを使用してテストする: 完全なデプロイを行う前に、既存のシステムと並行して限定的なデプロイを実施してテストを行い、望ましい結果が得られるかどうか確認します。例えば、デプロイカナリアテストまたはワンボックスデプロイを使用します。
- 並列環境でデプロイする: 並列環境に変更を実装し、その後、新しい環境に移行します。デプロイの成功を確認するまで、以前の環境を維持します。こうすることで、以前の環境へのロールバックが可能になり、復旧時間を最小限に抑えることができます。
- 小規模で可逆的な変更を頻繁にデプロイする: 小規模で可逆的な変更を頻繁に行うことで、変更の範囲を減らします。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また、変更をロールバックすることもできます。
- 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。
- テストとロールバックを自動化する: デプロイした環境のテストを自動化し、望ましい結果が得られるかどうか確認します。結果が達成されない場合に以前の既知で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ともに、手動プロセスによるエラーを減らします。

OPS 5 ワークロードをサポートする準備が整っていることはどうすれば確認できるでしょうか?

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

ベストプラクティス:

- 従業員の対応力の確保: 運用上のニーズに対応できるようにトレーニングを受けた、適切な人数の従業員が配置されていることを検証するメカニズムを導入します。効果的なサポートを継続できるように必要に応じて従業員のトレーニングを実施し、従業員の対応力を調整します。
- 運用準備状況の一貫した確認を実現する: ワークロードの運用に関する準備状況の一貫して確認するようにしてください。確認には、チームとワークロードに関する運用準備状況、またセキュリティ上の考慮事項を必ず含める必要があります。必要に応じて確認アクティビティをコードで実装し、イベントに対応して自動確認をトリガーし、一貫性、実行スピードを確認して、手動プロセスによって発生するエラーを減らします。
- ランブックを使用して手順を実行する: ランブックは、具体的な成果を達成するための文書化された手順です。ランブックに手順を文書化することにより、一貫性を保ち、汎用イベントにすみやかに対応できるようになります。必要に応じてランブックをコードとして実装し、イベントに対応してランブックの実行をトリガーし、一貫性、対応スピードを確認して、手動プロセスによって発生するエラーを減らします。
- プレイブックを使用して問題を特定する: 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。調査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。プレイブックをコードとして実装し、必要に応じてプレイブックの実行をトリガーし、一貫性、対応スピードを確認して、手動プロセスによって発生するエラーを減らします。
- システムや変更をデプロイするために十分な情報に基づいて決定を下す: ワークロードと、ワークロードのガバナンスとのコンプライアンスをサポートするためにチームの能力を評価します。システムを移行するか、本番環境に変更するかどうかを判断する際に、これらをデプロイの利点に対して評価します。メリットとリスクを理解し、十分な情報に基づく決定を下します。

運用

OPS 6 ワークロードの正常性をどのように把握しますか?

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

ベストプラクティス:

- 主要業績評価指標（KPI）を特定する: ビジネスと顧客が求める成果に基づいて、主要業績評価指標 (KPI) を特定します。KPI を評価して、ワークロードの成功を判別します。
- ワークロードのメトリクスを定義する: KPI の達成度を測定するため、ワークロードのメトリクスを定義します。ワークロードの正常性を測定するため、ワークロードのメトリクスを定義します。メトリクスを評価して、ワークロードが必要な成果に達しているかを判定し、ワークロードの正常性を把握します。
- ワークロードメトリクスを収集および分析する: メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を決定できます。
- ワークロードメトリクスの基準値を設定する: コンポーネントのパフォーマンスを比較し、過不足を特定する基準となる期待値として、メトリクスに対する基準値を設定します。
- ワークロードに対して予想されるアクティビティのパターンを知る: ワークロードアクティビティのパターンを確立すると、行動が期待値から外れるタイミングを把握し、必要に応じて適切に対応できるようになります。
- ワークロードの結果にリスクがある場合に警告する: ワークロードの結果にリスクがある場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
- ワークロードの異常が検出された場合に警告する: ワークロードの異常が検出された場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
- KPI とメトリクスの成果の達成度と有効性を検証する : ワークロードオペレーションに対するビジネスレベルの視点を確立すると、ニーズを満足しているかどうかを判断したり、ビジネス目標を達成するために改善が必要な領域を特定したりできます。KPI とメトリクスの有効性を検証し、必要に応じて修正します。

OPS 7 オペレーションの正常性をどのように把握しますか?

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

ベストプラクティス:

- 主要業績評価指標（KPI）を特定する: ビジネスおよび顧客が求める成果に基づいて、主要業績評価指標 (KPI) を特定します。KPI を評価して、オペレーションの成功を判別します。
- オペレーションのメトリクスを定義する: KPI の達成度を測定するため、オペレーションのメトリクスを定義します。オペレーションの正常性を測定するため、オペレーションのメトリクスを定義します。メトリクスを評価して、オペレーションが必要な成果に達しているかを判定し、オペレーションの正常性を把握します。
- オペレーションメトリクスを収集し、分析する: メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。
- オペレーションメトリクスの基準値を設定する: プロセスのパフォーマンスを比較し、過不足を特定する基準となる期待値として、メトリクスに対する基準値を設定します。
- オペレーションに対して予想されるアクティビティのパターンを知る: 比較基準となる期待値として、メトリクスの基準値を設定します。
- オペレーションの結果にリスクがある場合に警告する: オペレーションの結果にリスクがある場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
- オペレーションの異常が検出された場合に警告する: オペレーションの異常が検出された場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
- KPI とメトリクスの成果の達成度と有効性を検証する : オペレーションアクティビティに対するビジネスレベルの視点を確立すると、ニーズを満足しているかどうかを判断したり、ビジネス目標を達成するために改善が必要な領域を特定したりできます。KPI とメトリクスの有効性を検証し、必要に応じて修正します。

OPS 8 ワークロードと運用イベントはどのように管理しますか?

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

ベストプラクティス:

- イベント、インシデント、問題に対する管理プロセスを使用する: イベント、介入の必要なイベント (インシデント)、介入が必要であり、かつ再度発生するまたは現時点で解決できないイベント (問題) が見つかったときに対応するためのプロセスを用意しておきます。これらのプロセスを利用することで、適切な時点に適切な対応を取ることが可能になり、イベントがビジネスや顧客に与える影響を緩和できます。
- 根本原因の分析のプロセスを使用する: イベントの根本原因を特定してドキュメント化するプロセスを用意しておき、再発を抑制または防止する緩和策と、迅速で効果的な対応手順を展開できるようにしておきます。必要に応じ、対象者に合わせて根本原因を通知します。
- アラートごとのプロセスを使用する: アラートを発生させるイベントすべてに対して具体的な対応策 (ランブックやプレイブック) を定め、所有者を明確に指定しておくようにします。こうすることで、運用上のイベントに対する効果的で迅速な対応が可能になり、アクションの必要なイベントが重要度の低い通知に埋もれてしまうことを避けられます。
- ビジネスへの影響に基づき、運用上のイベントを優先します。: 介入を必要とする複数のイベントが発生したときに、ビジネスにとって最重要なものから対応できるようにしておきます。影響の例として、死亡や傷害、経済的損失、評判や信用の低下などがあります。
- エスカレーション経路を決定する: ランブックとプレイブックで、エスカレーションをトリガーするものとエスカレーションの手順を含むエスカレーション経路を決定します。特に、各アクションの所有者を特定し、運用イベントに効果的かつすばやく対応できるようにします。
- プッシュ通知を有効にする: ユーザーの使用するサービスに影響が生じる場合や、通常運用状態に復帰する場合に、ユーザーと直接通信し (E メールや SMS など)、ユーザーが適切な対応アクションを起こせるようにします。
- ダッシュボードでステータスを知らせる: 対象となる利用者 (内部技術チーム、指導部、顧客など) に合わせたダッシュボードを用意して、現在の業務の運用状況と、相手が関心を持つメトリクスを知らせます。
- イベントへの対応を自動化する: イベントへの対応を自動化し、手動プロセスによって発生するエラーを減らして、迅速かつ一貫した対応を実現します。

進化

OPS 9 オペレーションを進化させる方法

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

ベストプラクティス:

- 継続的改善のプロセスを用意する: 最も大きな利益をもたらす取り組みに集中できるように、改善の機会を定期的に評価し、優先順位を設定します。
- フィードバックループを実装する: 手順とワークロードにフィードバックループを組み込むことで、改善を必要としている問題や領域を特定できます。
- 改善の推進要因を定義する: 機会を評価して優先順位を設定できるよう、改善の推進要因を特定します。
- 洞察を検証する: 分析結果を確認してクロスな役割を持つチームやビジネスオーナーで応答します。これらのレビューに基づいて共通の理解を確立し、追加的な影響を特定するとともに、一連のアクションを決定します。必要に応じて対応を調整してください。
- オペレーションメトリクスのレビューを実行する: ビジネスのさまざまな分野のチームメンバー間でオペレーションメトリクスの遡及分析を定期的に実施します。これらのレビューに基づいて、改善の機会と取り得る一連のアクションを特定するとともに、教訓を共有します。
- 教訓を文書化して共有する: オペレーションアクティビティの実行から学習した教訓を文書化して共有し、社内とチーム全体で共有できるようにします。
- 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。

セキュリティ

アイデンティティ管理とアクセス管理

SEC 1 認証情報と認証をどのように管理していますか?

認証情報と認証メカニズムには、ワークロードにおいてアクセス権を直接的または間接的に付与するパスワード、トークン、キーなどが含まれます。適切なメカニズムで認証情報を保護することで、不注意または悪意による不正利用のリスクを減らすことができます。

ベストプラクティス:

- アイデンティティおよびアクセス管理要件を定義する: 組織要件、法律要件、コンプライアンス要件を満たすアイデンティティおよびアクセス管理設定を定義する必要があります。
- AWS ルートユーザーを保護する: AWS ルートユーザーを、アクセスキーを使用せずに MFA で保護し、その使用を制限して AWS アカウントを保護します。
- Multi-Factor Authentication (MFA) の使用を義務化する: ソフトウェアまたはハードウェアメカニズムを使用した Multi-Factor Authentication (MFA) を義務化することで、アクセスコントロールの機能を強化できます。
- アクセスコントロールの義務化を自動化する: 自動化されたツールを使用し、違反を報告することで、アクセスコントロールを義務化します。これは認証情報の管理要件を維持するのに役立ちます。
- 一元化されたフェデレーションプロバイダーと統合する: フェデレーテッド ID プロバイダーまたはディレクトリサービスと統合し、すべてのユーザーを一元的に認証します。これにより、複数の認証情報を使用する必要がなくなり、管理の複雑さも軽減されます。
- パスワード要件を義務化する: パスワードの最小文字数、複雑さ、再利用のポリシーを義務化することで、総当たり攻撃などのパスワードを対象とした攻撃に対する保護を強化します。
- 認証情報を定期的にローテーションする: 認証情報を定期的にローテーションすることで、権限のないシステムやユーザーによって古い認証情報が使用されるリスクを減らすことができます。
- 認証情報を定期的に監査する: 認証情報の監査を行い、定義されているコントロール (MFA など) が義務化されているか、定期的にローテーションされているか、アクセスレベルが適切かどうかを確認します。

SEC 2 人為的なアクセスをどのように制御していますか?

定義されたビジネス要件に合致するコントロールを実装することで人為的なアクセスを制御し、リスクと不正アクセスの影響を軽減します。これは特権ユーザーと AWS アカウントの管理者に適用されます。また、アプリケーションのエンドユーザーにも適用されます。

ベストプラクティス:

- 人為的なアクセス要件を定義する: 不要な権限によるリスクを低減するため、職務機能に基づいてユーザーのアクセス要件を明確に定義します。
- 最小限の権限を付与する: 不正アクセスのリスクを低減するため、定義した最小限の権限のみをユーザーに付与します。
- 各個人に一意の認証情報を割り当てる: ユーザー間の分離や追跡可能性のために、ユーザー間で認証情報は共有されません。
- ユーザーライフサイクルに基づいて認証情報を管理する: アクセス管理とユーザーライフサイクルを統合します。たとえば、ユーザーが退職するか、役割を変えたときにユーザーを削除し、使用されていない不要な認証情報を取り消します。
- 認証情報管理を自動化する: 認証情報管理を自動化し、最小権限を強制して、使用されていない認証情報を無効化します。ユーザーのライフサイクルの監査、レポート、管理を自動化します。
- ロールまたはフェデレーションを通じてアクセス権を付与する: IAM ユーザーまたは静的アクセスキーの代わりに IAM ロールを使用して、安全なクロスアカウントアクセスとフェデレーティッドユーザーを許可します。

SEC 3 プログラムによるアクセスをどのように制御していますか?

適切に定義、制限、分離されたアクセス権によってプログラムによるアクセス、または自動アクセスを制御することにより、不正アクセスのリスクを軽減します。プログラムによるアクセスには、ワークロード内部でのアクセスや、AWS 関連リソースへのアクセスが含まれます。

ベストプラクティス:

- プログラムによるアクセス要件を定義する: 不要な権限によるリスクを低減するため、自動アクセスまたはプログラムによるアクセスに対するアクセス要件を明確に定義します。
- 最小限の権限を付与する: 不正アクセスのリスクを低減するため、自動アクセスまたはプログラムによるアクセスに対して、最小限の権限のみを付与します。
- 認証情報管理を自動化する: 認証情報管理を自動化し、最小権限を強制して、使用されていない認証情報を無効化します。動的認証の監査、レポート、および管理を自動化します。
- 各コンポーネントに一意の認証情報を割り当てる: 分離や追跡可能性のために、コンポーネント間で認証情報は共有されません。例えば、AWS Lambra 機能と EC2 インスタンスに対しては、異なる IAM ロールを使用します。
- ロールまたはフェデレーションを通じてアクセス権を付与する: IAM ユーザーまたは静的アクセスキーの代わりに IAM ロールまたはフェデレーションを使用して、安全なプログラムによるアクセスを許可します。
- 動的認証を実装する: 認証情報は動的に取得され、サービスまたはシステムにより頻繁にローテーションされます。

発見的統制

SEC 4 セキュリティイベントをどのように検出し、調査していますか？

ログやメトリクスからイベントを可視化して把握し、分析します。セキュリティイベントや潜在的な脅威に対して措置をとることで、ワークロードを保護します。

ベストプラクティス:

- ログの要件を定義する: 組織上、法律上、コンプライアンス上の要件を満たすよう、ログの保管とアクセスコントロールに関する要件を定義します。
- メトリクスの要件を定義する: メトリクスを収集し、ベースラインを定義することで、潜在的なセキュリティ上の脅威に関する知見を得ることができます。
- アラートの要件を定義する: だれがアラートを受信するか、受信者はアラートを受け取ったら何をすべきかについて定義します。
- サービスとアプリケーションのログ記録を設定する: アプリケーションログ、AWS のサービスログ、リソースログを含め、ワークロード全体でログ記録を設定します。
- ログを一元的に分析する: 悪意のあるアクティビティや侵害の兆候、ならびに異常を検出するため、すべてのログが一元的に収集され、自動的に分析されるようにする必要があります。
- 主要な指標に関するアラートを自動化する: セキュリティに関連するメトリクスやイベントを含む主要な指標をモニタリングし、しきい値に基づいてアラートが自動的にトリガーされるようにする必要があります。
- 調査プロセスを開発する: インシデント対応プロセスのエスカレーションパスを含め、さまざまなタイプのイベントを調査するプロセスを開発します。

SEC 5 新しいセキュリティ脅威に対してどのように防御していますか?

AWS と業界のベストプラクティスおよび脅威インテリジェンスに関する最新情報を入手すれば、新しいリスクを認識するのに役立ちます。これにより、脅威モデルを作成し、ワークロードの保護に役立つ適切なコントロールを特定、優先順位付け、および実装することが可能になります。

ベストプラクティス:

- 組織要件、法的要件、コンプライアンス要件に関する最新情報を入手する: 組織要件、法的要件、コンプライアンス要件に関する最新情報を入手し、セキュリティ体制を調整してそれらの要件を満たせるようにします。
- セキュリティのベストプラクティスに関する最新情報を入手する: AWS と業界の両方のセキュリティベストプラクティスに関する最新情報を入手し、ワークロードの保護を進化させます。
- セキュリティ脅威に関する最新情報を入手する: セキュリティ脅威に関する最新情報を入手して攻撃ベクトルを理解します。これにより、発見的統制、および予防的統制を実装できます。
- 新しいセキュリティサービスとセキュリティ機能を定期的に評価する: AWS および APN パートナーのセキュリティサービス (新機能など) を評価し、脅威のリスクを抑制します。
- 脅威モデルを使用してリスクを定義し優先順位付けする: 脅威モデルを使用して潜在的な脅威を特定し、その登録を最新の状態に維持します。脅威に優先順位を付けて、それに対応できるようセキュリティ体制を調整します。
- 新しいセキュリティサービスとセキュリティ機能を実装する: セキュリティサービスとセキュリティ機能を導入し、ワークロードを保護するのに役立つコントロールを実装します。

インフラストラクチャ保護

SEC 6 ネットワークをどのように保護していますか?

パブリックおよびプライベートネットワークでは、内部および外部の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:

- ネットワーク保護要件を定義する: 組織要件、法的要件、コンプライアンス要件を満たすため、ネットワークの保護コントロールを定義します。
- 露出を制限する: 最低限必要なアクセスのみを許可することにより、ワークロードの露出をインターネットと内部ネットワークに制限します。
- 設定管理を自動化する: 設定管理サービスやツールを使うことで、自動的に安全性の高い設定を適用および検証し、ヒューマンエラーを減らすことができます。
- ネットワーク保護を自動化する: 保護メカニズムを自動化し、脅威インテリジェンスと異常検出に基づく自己防御型ネットワークを提供します。
- 検査および保護を実装する: アプリケーションレベルでトラフィックを検査してフィルタリングします。たとえば、ウェブアプリケーションファイアウォールを使用して脅威から保護します。
- すべてのレイヤーでトラフィックをコントロールする: 送受信トラフィックを制御するためのコントロール (データ損失防止など) を適用します。Amazon Virtual Private Cloud (VPC) の場合、これにはセキュリティグループ、ネットワーク ACL、サブネットが含まれます。AWS Lambda の場合は、トラフィックを制御するためにプライベート VPC で実行することを検討してください。

SEC 7 コンピューティングリソースをどのように保護していますか?

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

ベストプラクティス:

- コンピューティング保護要件を定義する: 組織上、法律上、コンプライアンス上の条件を満たすため、コンピューティングリソースに対する保護コントロールを定義します。
- 脆弱性をスキャンし、パッチを適用する: 新しい脅威から保護するためには、コードベースやインフラストラクチャに脆弱性がないか頻繁にスキャンし、パッチを適用します。
- 設定管理を自動化する: 設定管理サービスやツールを使うことで、自動的に安全性の高い設定を適用および検証し、人為的なエラーを減らすことができます。
- コンピューティング保護を自動化する: 未知の脅威に対する保護能力を含め、侵入防止策で防御を自動化します (仮想パッチを使用するなど)。
- 攻撃領域を削減する: EC2 オペレーティングシステムを強化したり、コンテナやサーバーレスリソースを構成したりするなど、公開範囲を限定して攻撃領域を削減します。
- マネージドサービスを活用する: セキュリティ保全タスクを減らすため、Amazon RDS、AWS Lambda、Amazon ECS といったリソースを管理するサービスを実装します。

データ保護

SEC 8 データをどのように分類していますか?

分類とは、機密レベルに応じてデータを分けることです。これにより、どのような管理手法を用いてデータを保護および保持すべきかを判断できます。

ベストプラクティス:

- データ分類要件を定義する: 組織要件、法的要件、コンプライアンス要件を満たすデータ分類要件を定義します。
- データ保護コントロールを定義する: 分類レベルに基づいてデータを保護します。例えば、公開データはベストプラクティスを用いて保護し、機密データは追加のコントロールで保護します。
- データの識別を実行する: 容易に識別可能なインジケータでデータを分類します。例えば、バケット内のデータを分類する Amazon S3 のバケットおよびオブジェクトのタグを使用します。
- 識別および分類を実装する: データの識別および分類を自動化し、ヒューマンエラーのリスクを軽減します。
- データのタイプを特定する: ワークロード内のデータのタイプを認識し、組織要件、法的要件、コンプライアンス要件を満たすコントロールを実装できるようにします。

SEC 9 保管中のデータをどのように保護していますか?

要件を定義して暗号化など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、不正アクセスや喪失のリスクを軽減できます。

ベストプラクティス:

- 保管中のデータの管理と保護に関する要件を定義する: 暗号化やデータ保持など、保管中のデータの管理と保護に関する要件を定義することで、組織上、法律上、コンプライアンス上の要件を満たすことができます。
- 安全なキー管理を実装する: 暗号化キーは、AWS Key Management Service といったキー管理サービスを使用して、安全に保管し、厳格なアクセスコントロールの下にローテーションする必要があります。異なるデータ分類レベルや保管要件の区別のため、キーを別々にすることを検討します。
- 保管中に暗号化を適用する: 最新の標準やベストプラクティスに基づき、定義済みの暗号化要件を適用することは、保管中のデータの保護に役立ちます。
- アクセスコントロールを適用する: 最低限の権限によるアクセスコントロールや、バックアップ、分離、バージョニングなどのメカニズムを適用することは、保管中のデータの保護に役立ちます。所有しているデータのうち、どのデータが公開されているかを把握してください。
- 人をデータから遠ざけるメカニズムを提供する: すべてのユーザーが機密データに直接アクセスできないようにします。たとえば、データストアに直接アクセスさせる代わりにダッシュボードを提供したり、間接的にデータを管理するためのツールを提供したりします。

SEC 10 伝送中のデータをどのように保護していますか?

要件を定義し、暗号化などのコントロールを実装して伝送中のデータを保護すれば、不正アクセスや漏洩のリスクを軽減できます。

ベストプラクティス:

- 伝送中のデータの保護に関する要件を定義する: データ分類に基づいて伝送中のデータの保護に関する要件 (暗号化標準など) を定義し、組織要件、法的要件、コンプライアンス要件を満たします。ベストプラクティスは、すべてのトラフィックを暗号化および認証し、最新の標準と暗号を適用することです。
- 安全な鍵および証明書管理を実装する: AWS Certificate Manager などの証明書管理サービスを使用して暗号化キーと証明書を安全に保管し、厳格なアクセスコントロールの下でローテーションします。
- 伝送中に暗号化を適用する: 最新の標準やベストプラクティスに基づいて定義済みの暗号化要件を適用すれば、組織要件、法的要件、コンプライアンス要件を満たすのに役立ちます。
- データ漏洩の検出を自動化する: ツールまたは検出メカニズムを使用し、定義された境界の外側にデータを移動する試みを自動的に検出します。たとえば、不明なホストにデータをコピーしているデータベースシステムを検出します。
- ネットワーク通信を認証する: Transport Layer Security (TLS) や IPsec などのプロトコルを使用して通信のアイデンティティを確認し、データの改ざんや損失のリスクを軽減します。

インシデント対応

SEC 11 セキュリティインシデントにどのように対応していますか?

セキュリティインシデントを適切なタイミングで調査および対応し、組織の中断を最小限に抑えられるようにするには、準備が重要です。

ベストプラクティス:

- 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に役立つ内部および外部の担当者とリソースを特定します。
- ツールを特定する: 組織のインシデント対応に役立つ AWS、パートナー、オープンソースツールを特定します。
- インシデント対応計画を策定する: インシデント対応計画を作成します。ワークロードと組織にとって最も可能性が高いシナリオから始めます。内部および外部に伝達およびエスカレーションする方法を含めます。
- 封じ込め機能を自動化する: インシデントの封じ込めを自動化し、対応時間を短縮するとともに組織的影響を軽減します。
- フォレンジック機能を確認する: 外部のスペシャリストを含む、利用可能なフォレンジック調査機能を確認します。
- アクセスを事前準備する: セキュリティ担当者がインシデントに適切に対応できるように、セキュリティ担当者に AWS への適切なアクセス権が事前に設定されていることを確認します。
- ツールを事前デプロイする: セキュリティ担当者がインシデントに適切に対応できるように、セキュリティ担当者に適切なツールが事前デプロイされていることを確認します。
- ゲームデーを実施する: インシデント対応ゲームデー (シミュレーション) で定期的に訓練し、そこで得られた教訓を計画に組み込み、対応と計画を継続的に改善します。

信頼性

基盤

REL 1 サービス制限をどのように管理していますか？

必要以上のリソースが偶発的にプロビジョニングされることを防止するため、デフォルトのサービス制限が設けられています。また、不正利用などからサービスを保護するため、API オペレーションを呼びだす頻度も制限されています。AWS Direct Connect を使用している場合、接続ごとに転送できるデータの量にも制限があります。AWS Marketplace のアプリケーションを使用している場合、アプリケーションの制限を理解する必要があります。サードパーティーのウェブサービスやサービスとしてのソフトウェアを使用している場合も、これらのサービスに設けられている制限を認識しておく必要があります。

ベストプラクティス:

- 制限があることは認識しているが、制限を追跡していない: 制限があることは認識しているが、現在の制限を追跡していません。
- 制限の監視と管理: 潜在的な使用量を評価し、リージョンごとの上限を適切に上げて、計画的な使用量の増加を可能にします。
- 制限の自動モニタリングと管理を使用する: 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。制限緩和のリクエストを自動化できるようになるまでは、担当グループに警告する配信メカニズムを使用してもよいでしょう。
- アーキテクチャを通じて、固定されたサービス制限に対応する: 変更できないサービスの制限を知り、それらを念頭に置いて設計してください。
- 現在のサービスの制限とフェールオーバーに対応できる最大使用量の間に十分な余裕を設けるようにする: リソースでエラーが発生したときには、リソースが正常に停止されるまで、制限にカウントされます。エラーが生じたリソースが停止されるまで、エラーが生じたすべてのリソースと代替リソースの合計リソース数が制限内に収まるようにします。このギャップを計算する際、アベイラビリティーゾーンの不具合を考慮する必要があります
- 関連するすべてのアカウントとリージョンでサービス制限を管理する: 複数の AWS アカウントまたは AWS リージョンを使用している場合は、本番ワークロードを実行するすべての環境で必ず同じ制限をリクエストしてください。

REL 2 ネットワークトポロジをどのように管理していますか?

アプリケーションは、既存のデータセンターインフラストラクチャ、公開されているパブリッククラウドインフラストラクチャ、またはプライベートアドレスが割り当てられたパブリッククラウドインフラストラクチャなど、1 つまたは複数の環境に存在可能です。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、プライベートアドレスの管理、名前解決などのネットワークの考慮点は、クラウド上のリソースを利用する上で基本的なものです。

ベストプラクティス:

- パブリッククラウドとオンプレミス環境のプライベートアドレス間で可用性の高い接続を活用する: 個別にデプロイされたプライベート IP アドレス空間との間で、複数の AWS Direct Connect (DX) 回路と複数の VPN トンネルを使用します。高可用性のために複数の DX ロケーションを使用します。複数の AWS リージョンを使用する場合は、少なくとも 2 つのリージョンに複数の DX ロケーションを設けることも必要です。VPN を終了する AWS Marketplace アプライアンスを評価することができます。AWS Marketplace アプライアンスを利用する場合は、さまざまなアベイラビリティーゾーンでの高可用性のために冗長インスタンスをデプロイします。
- ワークロードのユーザー向けに可用性の高いネットワーク接続を活用する: 高可用性 DNS、CloudFront、API ゲートウェイ、負荷分散、またはリバースプロキシを、アプリケーションのパブリック用エンドポイントとして利用します。負荷分散やプロキシ用の AWS Marketplace アプライアンスを評価することができます。
- 接続されている複数のプライベートアドレス空間において、重複しないプライベート IP アドレス範囲を指定します。: VPN 経由でピアリングされるか接続される場合、各 VPC の IP 範囲が競合しないようにする必要があります。オンプレミス環境と他のクラウドプロバイダーへのプライベート接続についても同様です。プライベート IP 範囲を必要に応じて割り当てる方法を用意する必要があります。
- 拡張や可用性のために割り当てる IP サブネットを確保する: 個別の 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、アプリケーションの要件を満たすために十分な大きさが必要です。これには、ロードバランサー、AWS Lambda 関数、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。また、将来の拡張の可能性を考えて利用できる IP アドレスを確保しておきます。

変更管理

REL 3 システムが需要の変化にどのように対応していますか？

スケーラブルなシステムはリソースを自動的に追加したり削除したりできる伸縮性を備えているため、いつでも最新の需要に厳密に適合します。

ベストプラクティス:

- ワークロードを拡大または縮小する際、自動的にリソースを調達する: Amazon S3、Amazon CloudFront、Amazon Auto Scaling、AWS Lambda などの自動的にスケーリングするサービスを使用します。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。
- ワークロード内のサービス不足が検出された時点でリソースを調達する: 可用性に影響が及ぶ場合、リソースを手動でスケーリングします。
- ワークロードにリソースがさらに必要であると検出された時点で手動でリソースを調達する: 需要が予測された時点で、コンピューティングとストレージの規模を手動で調整します。
- ワークロードの負荷テスト: 負荷テスト手法を採用して、規模の拡大や縮小がワークロード要件を満たすかどうかを測定します。

REL 4 リソースをどのようにモニタリングしていますか？

ログとメトリクスは、ワークロードの状態に関する洞察を得るための強力なツールです。ワークロードは、しきい値を超えたり重大なイベントが発生したりしたときに、ログとメトリクスがモニタリングされて通知が送信されるように構成できます。ワークロードは、低パフォーマンスのしきい値を超えたり障害が発生したりしたときに、自動的に自己回復したり、スケールするように設計されているのが理想です。

ベストプラクティス:

- ワークロードにおける全ての層をモニタリングする: ワークロードの各階層は、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 のサービスは、Personal Health Dashboard を使ってモニタリングします。
- モニタリングに基づいて通知を送信する: 重要なイベントが発生すると、把握しておく必要のある組織が通知を受信します。
- イベントに対して自動化された応答を実行する: 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 (たとえば、障害が発生したコンポーネントを交換します)。
- 定期的にレビューを実施する: アーキテクチャと実装を評価するため、重要なイベントと変更に基づいてワークロードのモニタリングを頻繁にレビューします。

REL 5 変更をどのように実施していますか？

環境の変更を管理しないと、変更の影響を予測することが難しくなります。プロビジョニングされた AWS リソースとアプリケーションの変更制御は、アプリケーションと運用環境で既知のソフトウェアが実行されており、予測できる方法でパッチを適用または置換できることを確認するために必要です。

ベストプラクティス:

- 計画的に変更をデプロイする: デプロイとパッチ適用は文書化されたプロセスに従って行います。
- 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的に行います。

障害の管理

REL 6 データをどうバックアップするか?

データ、アプリケーション、動作環境 (アプリケーションで設定したオペレーティングシステムとして定義) をバックアップして、平均復旧時間 (MTTR) と目標復旧地点 (RPO) の要件を満たします。

ベストプラクティス:

- バックアップする必要があるデータをすべて特定し、バックアップやソースからのデータの複製を実行する: Amazon S3、Amazon EBS スナップショット、サードパーティー製ソフトウェアを使用して重要なデータをバックアップします。また、データをソースから複製して RPO を満たせる場合は、バックアップが必要ない場合があります。
- データのバックアップまたはソースからのデータの複製を自動的に実行する: AWS の機能 (たとえば、Amazon RDS と Amazon EBS のスナップショット、Amazon S3 の複数バージョンなど)、AWS Marketplace ソリューション、サードパーティーのソリューションを使ってバックアップまたはソースからの複製を自動化します。
- 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: 復旧テストを行って、バックアッププロセスの設定が目標復旧時間と目標復旧地点を満たしていることを検証します。
- バックアップを保護または暗号化するか、データが安全なソースから複製できることを確認する: AWS IAM などの認証と許可によってアクセスを検出し、暗号化を使用してデータの整合性の侵害を検出します。

REL 7 どのようにしてシステムがコンポーネントのエラーに耐えるか?

ワークロードに関して顕在または潜在する条件として、高い可用性と短い平均復旧時間 (MTTR) を必要とする場合は、ワークロードを弾力性のある設計にし、障害に耐えるようワークロードを分散します。

ベストプラクティス:

- ワークロードのすべてのレイヤーをモニタリングしてエラーを検知する: システムの状態を継続的にモニタリングして、状態の悪化や完全なエラーをレポートします。
- 疎結合の依存関係を実装する: キューイングシステム、ストリーミングシステム、ワークフロー、ロードバランサーなどの依存関係は、緩やかに結合しています。
- 該当するハードな依存関係をソフトな依存関係に変換するため、グレイスフルデグラデーションを実装する: コンポーネントの依存関係に異常が生じても、コンポーネント自体が異常をレポートすることはありません。そのコンポーネントは異常状態でリクエストを処理します。
- 単一の場所を求める技術的な制約がワークロードの一部または全体に存在するため、復旧全体を自動化する: ワークロードの要素は、1 つのアベイラビリティーゾーンまたは 1 つのデータセンターでのみ実行可能であるため、定義された復旧目標を使用して、ワークロードの完全な再構築を実行する必要があります。
- 複数の場所にワークロードをデプロイする: 複数のアベイラビリティーゾーンや AWS リージョンにワークロードを分散します (例えば DNS、ELB、Application Load Balancer、API ゲートウェイ)。これらの場所は必要に応じて多様化できます。
- すべてのレイヤーの修復を自動化する: エラー検知時に自動機能で修正アクションを実行します。
- 可用性に影響を及ぼすイベントが発生したら通知を送信する: 重大なイベントが検知されたら、問題が自動的に修復されたとしても、通知が送信されます。

REL 8 弾力性をどのようにテストしていますか?

ワークロードの弾力性をテストして、本番環境でしか表面化しない潜伏バグを見つけられるようにします。こういったテストを定期的に行います。

ベストプラクティス:

- 予期しない障害に関するプレイブックを使用する: 予期しない障害シナリオに関するプレイブックを使用することで、原因を特定し、防止や軽減のための戦略を立てることができます。
- 根本原因の分析 (RCA) を実施し、結果を共有する: 重要なイベントに基づいてシステム障害のレビューを行い、アーキテクチャを評価し根本原因を特定します。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。
- 障害を生成して回復力をテストする: 意図的な障害を定期的にテストして、障害経路の範囲を確認します。
- 定期的にゲームデーを実施する: ゲームデーを利用して、実際の障害シナリオに関与する予定の人々と共に、障害手順を定期的に訓練します。

REL 9 災害対策をどのように計画していますか?

災害対策 (DR) は、バックアップからデータを復元する際に不可欠です。RTO と RPO の目標と一致するように目標、リソース、場所、復元するデータの機能を定義し、実行する必要があります。

ベストプラクティス:

- ダウンタイムやデータ消失に関する復旧目標を定義する: ワークロードには、目標復旧時間 (RTO) と目標復旧時点 (RTO) が定義されます。
- 復旧目標を満たすため、定義された復旧戦略を活用する: 目標を達成するために災害対策 (DR) 戦略が定義されています。
- 災害対策の実装をテストし、検証する: DR へのフェイルオーバーを定期的にテストし、RTO と RPO が満たされていることを確認します。
- すべての変更について構成のずれを管理する: AMI とシステム構成の状態および AWS のサービスに対する制限が DR サイトやリージョンで最新の状態であることを確認します。
- 復旧を自動化する: AWS またはサードパーティーのツールを使用して、システムの復旧を自動化します。

パフォーマンス効率

選択

PERF 1 最も良いパフォーマンスのアーキテクチャをどのように選択していますか？

多くの場合、ワークロード全体で最適なパフォーマンスを得るには、複数のアプローチが必要です。Well-architected のシステムでは複数のソリューションが使用され、さまざまな機能によってパフォーマンスが改善されます。

ベストプラクティス:

- 利用可能なサービスやリソースを理解する: AWS で利用できる幅広いサービスやリソースに関する情報を取得し、その内容を理解します。自分のワークロードに関連するサービスと設定オプションを特定し、最適なパフォーマンスをあげるためにそのオプションの利用方法を学びます。
- アーキテクチャにかかわる選択プロセスを決める: AWS 内での過去の経験と知識、または公開されたユースケース、関連ドキュメント、ホワイトペーパーなどの外部リソースを利用して、リソースとサービスの選定プロセスを決定します。例えば、ワークロードで使用される可能性があるサービスの実験やベンチマークを促進するプロセスです。
- コストや予算を決断に反映させる: ワークロードにはその範囲内での運用を義務付ける予算が設定されることが多く、予算は効率的な運用の重要な要素となっています。内部コスト統制を利用し、リソースのタイプとサイズを選択する際は予測されたリソースのニーズに基づいた予算を考慮します。
- ポリシーやリファレンスアーキテクチャを使用する: 内部ポリシーまたは既存のリファレンスアーキテクチャを使用してワークロードに最適なアーキテクチャを選択します。パフォーマンスと効率性を最大化する、ワークロードに最適なサービスと設定を見極めます。
- AWS または APN パートナーからのガイダンスを利用する: 決断をする際は、ソリューションアーキテクトなどの AWS リソースや APN パートナーを参考にします。これらのリソースは、最適なパフォーマンスを達成するために必要なアーキテクチャの改良点を見直し、提案するのに役立ちます。
- 既存のワークロードのベンチマークを実施する: 既存のワークロードのパフォーマンスにベンチマーク結果を参考に、AWS での実行状況を把握します。こうしたベンチマークから収集されるデータを使用して、アーキテクチャに関する決断を下しやすくします。
- ワークロードの負荷テストを実施する: 種類やサイズの異なるリソースを使用して、最新版システムを AWS にデプロイし、モニタリングしながらボトルネックや過剰なキャパシティを判別するパフォーマンスメトリクスを把握します。この情報を利用して、アーキテクチャを設計または改良するとともに、実行状況に基づいてリソースを選定します。

PERF 2 コンピューティングソリューションをどのように選択していますか?

システムにとって最適なコンピューティングソリューションは、アプリケーションの設計、使用パターン、設定に応じて異なります。各アーキテクチャでは、コンポーネントごとに異なるコンピューティングソリューションが使用される可能性があるため、パフォーマンスを向上させるための機能も異なります。アーキテクチャで使用されるコンピューティングソリューションを正しく選択しないと、パフォーマンス効率が低下するおそれがあります。

ベストプラクティス:

- 使用可能なコンピューティングオプションを評価する: 利用可能なコンピューティング関連のオプションのパフォーマンス特性を調べて理解します。インスタンス、コンテナ、関数の動作とそれらがワークロードにもたらすメリットとデメリットを把握します。
- 利用可能なコンピューティング設定オプションについて理解する: さまざまなオプションがワークロードをどのように補完し、どの設定がお使いのシステムに最適かを理解します。これらのオプションの例として、インスタンスファミリー、サイズ、機能 (GPU、I/O)、機能サイズ、コンテナインスタンス、シングルテナントとマルチテナントなどがあります。
- コンピューティング関連のメトリクスを収集する: システムのパフォーマンスを理解する最良の方法の 1 つは、各種リソースの実際の使用率を記録して追跡することです。その後、このデータは、リソース要件をより正確に決定するためにフィードバックされます。
- 適切なサイジングによって必要な設定を決定する: ワークロードのさまざまなパフォーマンス特性とそれらの特性とメモリ、ネットワーク、CPU 使用率との関連を分析します。このデータは、ワークロードのプロファイルに最も合うリソースを選択する時に使用します。例えば、メモリ負荷の高いワークロード (データベースなど) の場合は、インスタンスの R ファミリーが最も適している可能性があります。一方、バースト性のあるワークロードの場合は、Amazon Elastic Container Service などの伸縮自在なコンテナシステムを使用することで、より大きなメリットが得られる可能性があります。
- 利用可能な伸縮自在なリソースを使用する: AWS では、さまざまなメカニズム (AWS Auto Scaling、Amazon Elastic Container Service、AWS Lambda など) を使用してリソースを動的に拡張または縮小し、需要の変化に対応する柔軟性が得られます。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、ワークロードがこのような変化に自動的に対応し、最適なリソースセットを使用して目標を達成できるようになります。
- メトリクスに基づいてコンピューティングニーズを再評価する: システムレベルのメトリクスを使用して、ワークロードの経時的な動作と要件を特定します。利用可能なリソースとこれらの要件を比較することでワークロードのニーズを評価し、ワークロードのプロファイルに最も合致するようにコンピューティング環境に変更を加えます。例えば、システムは、時間の経過とともに、当初想定していたよりもメモリ負荷が高くなる場合があります。そのため、別のインスタンスファミリーまたはインスタンスサイズに移行すると、パフォーマンスと効率性の両方が向上する可能性があります。

PERF 3 ストレージソリューションをどのように選択していますか?

システムにとって最適なストレージソリューションは、アクセス方法 (ブロック、ファイル、オブジェクト)、アクセスパターン (ランダム、シーケンシャル)、必要なスループットやアクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、可用性と耐久性に関する制約によって異なります。優れた設計のシステムでは、複数のストレージソリューションを使用し、さまざまな機能を有効にしてパフォーマンスとリソースの使用効率を高めています。

ベストプラクティス:

- ストレージ特性と要件を理解する: ワークロードに最適なサービス (Amazon S3、Amazon EBS、Amazon Elastic File System (Amazon EFS)、Amazon EC2 インスタンスストアなど) を選択するうえで必要な特性 (共有可能性、ファイルサイズ、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、データの永続性など) を理解してください。
- 利用可能な設定オプションを評価する: さまざまな特性や設定オプションとそれらがストレージにどのように関連するかを評価します。PIOPS、SSD、磁気ストレージ、Amazon S3、Amazon Glacier、エフェメラルストレージを使用する状況と方法を理解し、ワークロードのストレージ容量とパフォーマンスを最適化します。
- アクセスパターンとメトリクスに基づいて意思決定を行う: ワークロードがデータにアクセスする方法を考慮したうえで、ストレージシステムの選択と設定を行います。アクセスパターンに最も合致するキャッシュサービスまたはインスタンスを選択する、Amazon S3 または DynamoDBにデータを保存する際に最適なキー分散を使用する、ストレージボリュームをストライピングする、システムの測定に基づいてデータをパーティション化するなどして、パフォーマンスを高めます。Amazon S3 などのオブジェクトストレージや Amazon Elastic Block Store などのブロックストレージを選択してストレージ効率を高めます。選択したストレージオプションを、データのアクセスパターンに合致するように設定します。

PERF 4 データベースソリューションをどのように選択していますか。

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

ベストプラクティス:

- データの特性を理解する: ワークロード内のデータのさまざまな特性について理解します。ワークロードにトランザクションが必要かどうか、ワークロードでデータをどのように操作するか、そのパフォーマンス要件はどのようなものかを確認します。このデータを使用して、ワークロードににとって最適なデータベースアプローチ (リレーショナルデータベース、No SQL、データウェアハウス、インメモリストレージなど) を選択します。
- 使用可能なオプションを評価する: ワークロードのストレージメカニズムの一部として利用できるサービスおよびストレージオプションを評価します。特定のサービスやシステムをデータストレージにいつ、どのように使用するかを理解します。データベースのパフォーマンスや効率性をさらに最適化するために使用できる設定オプション (PIOP、メモリおよびコンピューティングリソース、キャッシングなど) を確認します。
- データベースのパフォーマンスメトリクスを収集して記録する: データ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。例えば、1 秒あたりのトランザクション数、実行速度の遅いクエリ、データベースにアクセスする際に生じるシステムレイテンシーなどを測定します。このデータを使用して、データベースシステムのパフォーマンスを把握します。
- アクセスパターンに基づいてデータストレージを選択する: ワークロードのアクセスパターンを使用して、使用するサービスとテクノロジーを決定します。例えば、トランザクションを必要とするワークロードにリレーショナルデータベースを使用したり、スループットは高いものの、必要に応じて結果整合性になるキーバリューストアを使用したりできます。
- アクセスパターンとメトリクスに基づいてデータストレージを最適化する: データの保存方法とクエリ方法を最適化して可能な限り最高のパフォーマンスを達成するパフォーマンス特性とアクセスパターンを使用します。インデックス作成、キー分散、データウェアハウス設計、キャッシング戦略などの最適化がシステムパフォーマンスや全体的な効率性に与える影響を測定します。

PERF 5 ネットワークソリューションをどのように選択していますか？

システムにとって最適なネットワークソリューションは、レイテンシー、スループット要件などによって異なります。場所のオプションはユーザーリソースやオンプレミスリソースなどの物理的な制約の影響を受けますが、エッジ技術やリソースプレイスメントを使うことでカバーできます。

ベストプラクティス:

- ネットワーキングがパフォーマンスに与える影響を理解する: ネットワーク関連の意思決定がワークロードのパフォーマンスに与える影響を分析し、理解します。例えば、ネットワークレイテンシーは一般にユーザーエクスペリエンスに影響し、誤ったプロトコルを使用すると、過剰なオーバーヘッドによってネットワークキャパシティーが枯渇する可能性があります。
- 使用可能な製品オプションを理解する: ネットワーク関連のパフォーマンスを最適化するために使用できるサービスレベル機能を把握します。例えば、EC2 インスタンスのネットワーク機能、拡張ネットワーク、Amazon EBS 最適化インスタンス、Amazon S3 Transfer Acceleration、Amazon CloudFront による動的なコンテンツ配信などがあります。
- 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 AWS 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、ジッターを軽減するために使用できるネットワークレベルの機能 (Amazon Route 53 のレイテンシーベースルーティング、Amazon VPC のエンドポイント、AWS Direct Connect など) を活用できます。
- 最小限のネットワーク ACL を使用する: ネットワーク要件を満たしながら、ACL の数を最小限に抑えるようにネットワークを設計します。ネットワーク ACL が多すぎると、ネットワークパフォーマンスに悪影響が及び、システムパフォーマンスや効率性が低下する可能性があります。
- 暗号化オフロードと負荷分散を活用する: 負荷分散を使用して暗号化ターミネーション (TLS) をオフロードし、パフォーマンスを高めるとともに、トラフィックを効率的に管理およびルーティングします。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 AWS の伸縮性を活用できるようにします。
- 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: システムとネットワーク間の通信プロトコルを選択する際には、それらのプロトコルがワークロードのパフォーマンスに与える影響に基づいて意思決定を行います。
- ネットワーク要件に基づいてロケーションを選択する: 利用可能なロケーションオプション (AWS リージョン、アベイラビリティゾーン、配置グループ、エッジロケーションなど) を使用して、ネットワークレイテンシーを軽減するか、スループットを向上させます。
- 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。それらの変更の影響を測定し、その影響測定値を将来の意思決定に使用します。

レビュー

PERF 6 ワークロードを進化させるためにどのように新機能を取り込んでいますか？

ワークロードを設計する際に選択できるオプションには限りがありますが、時間が経つにつれ、ワークロードのパフォーマンスの向上に役立つ新しいテクノロジーやアプローチを利用できるようになります。

ベストプラクティス:

- リソースとサービスに関する情報を常に最新に保つ: 新しいサービス、設計パターン、製品が利用可能になったら、パフォーマンスの向上方法を検討します。こうした新しい要素がワークロードのパフォーマンス向上や効率性改善にどのように役立つかを、都度評価したり、外部分析などで検討します。
- 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、新規のインスタンスに対して、既存のパフォーマンステストを実行し、この製品を使用することで得られるパフォーマンスや効率性の改善点を判断します。
- ワークロードのパフォーマンスを時間の経過とともに進化させる: 組織として、評価プロセスを通じて収集した情報を使用し、新しく利用可能となったサービスやリソースの導入を積極的に推し進めて、ワークロードのパフォーマンス向上や効率性の改善を図ります。

モニタリング

PERF 7 リソースが正常に動作していることを確認するためにどのようにモニタリングしていますか？

システムのパフォーマンスは徐々に低下することがあります。システムのパフォーマンスをモニタリングして低下の兆候を見つけ、オペレーティングシステムやアプリケーション負荷などの内部的および外部的な要素を修正します。

ベストプラクティス:

- パフォーマンスに関連するメトリクスを記録する: Amazon CloudWatch、サードパーティーサービス、またはセルフマネージモニタリングツールを使用して、パフォーマンスに関連するメトリクスを記録します。例えば、データベーストランザクション、実行速度の遅いクエリ、I/O レイテンシー、HTTP リクエストのスループット、サービスレイテンシー、その他の重要なデータを記録します。
- イベントやインシデントが発生したときにメトリクスを分析する: イベントやインシデントが発生した後 (または発生中) に、モニタリングダッシュボードやレポートを使用してその影響を把握して診断します。これらのビューを使用すると、ワークロードのどの部分が予想されたレベルで動作していないかを把握できます。
- KPI を確立してワークロードのパフォーマンスを測定する: システムが意図したとおりに動作しているかどうかを示す KPI を特定します。例えば、API ベースのワークロードでは、全体的なレスポンスレイテンシーを全体的なパフォーマンスの指標として使用できます。また、e コマースサイトでは、購入数を KPI として使用できます。
- モニタリングを使用してアラームベースの通知を生成する: モニタリングシステムを使用し、定義したパフォーマンス関連の KPI の測定値が予想された境界線の外側にある場合に自動的にアラームを生成します。
- メトリクスを定期的に見直す: 定期的なメンテナンスとして、またはイベントやインシデントに応じて、収集対象のメトリクスを見直します。このような見直しでは、問題解決の鍵となったメトリクスや、追跡していた場合には、問題を特定、解決、または防止するのに役立ったと考えられる追加のメトリクスを特定します。
- モニタリングしてプロアクティブに警告する: KPI とモニタリングおよびアラートシステムを組み合わせて、パフォーマンス関連の問題にプロアクティブに対処します。アラームを使用して、問題を修正する自動化されたアクションをトリガーします (可能な場合)。自動的な対応が不可能な場合は、対応可能な担当者にアラームをエスカレーションします。例えば、予想される KPI 値を予測し、それらが特定のしきい値に違反した場合にアラームを送信できるシステムや、KPI が予想された値の範囲外にある場合に自動的にデプロイを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。

トレードオフ

PERF 8 パフォーマンスを向上させるために、トレードオフをどのように利用していますか？

アーキテクチャの設計にあたって、最適なアプローチとなるトレードオフについて積極的に考慮します。多くの場合、整合性、耐久性、時間とレイテンシーと引き換えに、パフォーマンスを向上することが可能です。

ベストプラクティス:

- パフォーマンスが最も重要な分野を理解する: ワークロードのパフォーマンスの向上が効率性やカスタマーエクスペリエンスにプラスの影響を与える分野を理解し、特定します。例えば、多くの顧客インタラクションが行われるウェブサイトでは、Amazon CloudFront などエッジサービスを使用してコンテンツ配信を顧客の近くに移動することでメリットが得られます。
- デザインパターンとサービスについて理解する: ワークロードのパフォーマンスの向上に役立つさまざまなデザインパターンとサービスについて調査し、理解します。分析の一環として、パフォーマンスの向上と引き換えになる可能性があるものを特定します。例えば、Amazon ElastiCache を使用すると、データベースシステムにかける負荷を低減できます。ただし、安全なキャッシングを実装するためのエンジニアリングを行う必要があり、場合によっては一部の領域で結果整合性が生じる可能性があります。
- トレードオフが顧客と効率性にどのように影響するかを明らかにする: パフォーマンス関連の向上を評価する場合、それらの選択肢が顧客とワークロードの効率性にどのように影響するかを検討してください。例えば、Amazon DynamoDB などのキーバリューストレージを使用すると、システムのパフォーマンスが大幅に向上しますが、Amazon DynamoDB の結果整合性が顧客に与える影響を評価することも重要になります。
- パフォーマンス向上の影響を測定する: パフォーマンスを向上させるための変更を加えた後、収集されたメトリクスとデータを評価し、パフォーマンスの向上がワークロード、そのコンポーネント、顧客に与えた影響を判断します。この測定により、トレードオフから得られた改善を理解するとともに、副次的な悪影響が生じたかどうかを確認することができます。
- さまざまなパフォーマンス関連戦略を使用する: 適用可能な場合は、いくつかの戦略を活用してパフォーマンスを高めてください。例えば、データをキャッシュして過度のネットワークまたはデータベース呼び出しを防ぐ、といった戦略を使用できます。データベースエンジンにリードレプリカを使用して読み取り速度を高める、可能な場合にデータをシャーディングまたは圧縮してデータボリュームを減らす、利用可能な結果をバッファに入れてストリーム配信する ことでブロックを回避するなどの戦略を使用できます。

コスト最適化

費用認識

COST 1 使用状況をどのように管理しますか?

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

ベストプラクティス:

- 組織の要件に基づいてポリシーを策定する: 組織のリソースの管理方法を定義するポリシーを策定します。ポリシーでは、リソースのライフサイクル全体にわたる作成、変更、削除を含む、リソースとワークロードのコスト面をカバーする必要があります。また、ワークロードのコスト目標も策定します。
- 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。
- グループとロールを実装する: ポリシーに適合するグループとロールを実装し、各グループ (開発、テスト、本番グループなど) でインスタンスとリソースを作成、変更、または削除できるユーザーをコントロールします。これは、AWS のサービスやサードパーティのソリューションに適用されます。
- コストコントロールを実装する: 組織のポリシーと定義済みのグループおよびロールに基づいてコントロールを実装します。これにより、コストが組織の要件で定義されているとおりに発生することが保証されます。例えば、IAM ポリシーでリージョンまたはリソースタイプへのアクセスをコントロールできます。
- 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。

COST 2 使用状況とコストをどのようにモニタリングしますか?

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

ベストプラクティス:

- AWS のコストと使用状況レポートを設定する: AWS のコストと使用状況レポートを設定して、使用状況と請求に関する詳細な情報を把握します。
- コスト属性カテゴリを特定する: 組織内でのコストの配分に使用可能な組織カテゴリを特定します。
- 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、お客様向けに作成された顧客レポートやお客様に提供されるウェブページが挙げられます。
- タグ付けを定義、実行する: 組織、ワークロード属性、コスト配分カテゴリに基づいてタグ付けスキーマを定義し、すべてのリソースにタグを付けます。
- 請求およびコスト管理ツールを設定する: AWS Cost Explorer と AWS Budgets を組織のポリシーに沿って設定します。
- コスト最適化に関して報告、通知する: コストと使用量を目標と比較して通知するように AWS Budgets を設定します。定期会議を開催して、このワークロードのコスト効率を分析し、社内にコスト意識を浸透させます。
- コストをプロアクティブにモニタリングする: この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するために、ツールとダッシュボードを導入します。通知を受信したときに、コストやカテゴリのみに目を向けないようにすることで、改善傾向を認識し、こうした傾向を組織全体で推進することができます。
- ワークロードメトリクスに基づいてコストを配分する: メトリクスや業績に基づいてこの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。洞察力とチャージバック機能を備えるAmazon Athena を使用してAWS のコストと使用状況レポートを分析するプロセスを実装します。

COST 3 不要なリソースをどのように削除しますか？

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:

- 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。
- 削除プロセスを実装する: 孤立したリソースを特定して削除するためのプロセスを実装します。
- 計画外の形でリソースを削除する: 計画外ベースでリソースを削除します。これは特に定期監査といったイベントによってトリガーされ、通常は手動で実行されます。
- 自動的にリソースを削除する: 重要度が低いリソース、不要なリソース、使用率が低いリソースを特定して削除する作業を適切に行えるようにワークロードを設計します。

費用対効果の高いリソース

COST 4 サービスを選択するとき、どのようにコストを評価しますか?

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、管理や運用によって発生するオーバーヘッドを削減またはゼロにでき、アプリケーション開発やビジネス上の他の活動に注力できるようになります。

ベストプラクティス:

- 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: 現在のサイズや現在のコストに関係なく、必ずすべてのワークロードを分析します。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。
- 各コンポーネントの詳細な分析を実行する: 各コンポーネントの、組織にとっての全体的なコストを調べます。運用および管理のコスト、特にマネージドサービスを使用するコストを考慮して総所有コストを調べます。見直しを行う際には、分析に費やされた時間がコンポーネントのコストに比例しているなどの潜在的利益を織り込む必要があります。
- 組織の優先順位に従ってコストが最適化されるようにこのワークロードのコンポーネントを選択する: すべてのコンポーネントを選択したときのコストを考慮します。これには、Amazon RDS、Amazon DynamoDB、Amazon SNS、Amazon SES などのアプリケーションレベルのサービスとマネージドサービス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。AWS Lambda 、静的ウェブサイト用のAmazon S3、Amazon ECSなどのサーバーレスやコンテナをコンピューティングに使用します。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を使用してライセンスコストを最小限に抑えます。例えば、Amazon Linux を計算ワークロードに使用したり、データベースを Amazon Aurora に移行したりします。
- 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とともに変化する可能性があり、サービスや機能の中には、使用量によってコスト効率性が向上するものがあります。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、このワークロードのコスト効率性をそのライフタイム全体にわたって維持できます。

COST 5 リソースタイプとサイズを選択する際、どうすればコスト目標を達成できるでしょうか?

目の前にあるタスクに合わせて適切なリソースのサイズを選択するようにします。最もコスト効率の高いタイプとサイズを選択することで、無駄を最小限に抑えます。

ベストプラクティス:

- コストモデリングの実行: 組織の要件を特定し、ワークロードとその各コンポーネントのコストモデリングを実行します。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。モデリングを行う際には、費やされた時間がコンポーネントのコストに比例しているなどの潜在的利益を織り込む必要があります。
- 見積もりに基づいてリソースのタイプやサイズを選択する: ワークロードとリソースの特性に基づいて、リソースのサイズやタイプを見積ります。コンピューティング、メモリ、スループット、書き込み頻度などについて検討します。この見積りは通常、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を用いて行います。
- メトリクスに基づいてリソースのタイプやサイズを選択する: 現在実行しているワークロードからのメトリックスを用いて、コストを最適化する適切なサイズやタイプを選択します。Amazon EC2、Amazon DynamoDB、Amazon EBS (PIOPS)、Amazon RDS、Amazon EMR、ネットワークなどのサービスに、適切なスループット、サイジング、ストレージをプロビジョンします。これは、自動スケーリングといったフィードバックループまたは手動によるサイズ変更プロセスによって行います。

COST 6 コストを削減するには、料金モデルをどのように使用したらよいでしょうか?

リソースのコストを最小限に抑えるのに最も適した料金モデルを使用します。

ベストプラクティス:

- 料金モデルの分析を実行する: AWS Cost Explorer のリザーブドインスタンスの推奨機能を使用して、ワークロードに関する分析を実行します。
- 低カバレッジでさまざまな料金モデルを実装する: 全体的な推奨範囲の 80% 未満という低カバレッジで、ワークロードにリザーブドキャパシティー、スポットインスタンス、スポットブロック、スポットフリートを実装します。
- コストに基づいてリージョンを選択する: リソースの料金は各リージョンで異なる場合があります。リージョンコストを織り込むことで、このワークロードに対して支払う料金の合計を最低限に抑えることができます。
- この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します。: 永続的に実行されるリソースには、推奨範囲の 80% 以上という高カバレッジでリザーブドキャパシティーが実装されます。短期的な使用には、スポットインスタンス、スポットブロック、スポットフリートを使用するように設定します。オンデマンドは、中断することのできない、かつリザーブドキャパシティーに対して実行時間の長さが十分ではない短期ワークロードに対してのみ使用されます (通常はリソースタイプに応じて年間 25% から 75%)。

COST 7 データ転送料金についてどのように計画していますか？

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

ベストプラクティス:

- データ転送モデリングの実行: 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。
- データ転送コストを最適化するコンポーネントを選択する: すべてのコンポーネントが選択され、データ転送コストを低減するようアーキテクチャが設計されます。これには、WAN 最適化やマルチ AZ 構成といったコンポーネントの使用が含まれます。
- データ転送コストを削減するサービスを実装する: データ転送を減らすためのサービスを実装します。たとえば、Amazon CloudFront をはじめとする CDN を使用してエンドユーザーにコンテンツを配信し、AWS への接続のために VPN の代わりに Amazon ElastiCache または AWS Direct Connect を使用してレイヤーをキャッシュします。

需要と供給を一致させる

COST 8 リソースの供給と顧客の需要をどのように一致させていますか?

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、費用を掛けたすべてのものが活用されるようにし、使用率が著しく低いインスタンスが生じるのを回避します。利用が過剰でも過少でも偏りが生じると、運用コスト (利用過剰によるパフォーマンスの低下) または無駄な AWS 費用 (過剰なプロビジョニング) のいずれかで、組織に悪影響が及びます。

ベストプラクティス:

- 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。この分析では、季節的傾向を考慮に入れ、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させてください。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。
- リソースを反応的に、または計画外にプロビジョニングする: リソースレベルは需要に応じて変化しますが、プロビジョニングは計画外のやり方で (通常は手動で) 行われ、有害事象やワークロードの変化によってトリガーされます。リソースの変更は時間がかかり、通常は過剰プロビジョニングまたはプロビジョニング不足になります。
- リソースを動的にプロビジョニングする: リソースを計画的なやり方でプロビジョニングします。これは、自動スケーリングなどの需要ベース、時間の経過とともに需要が拡散し、全体的なリソース使用率が低下するバッファーベース、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。

長期的な最適化

COST 9 新しいサービスをどのように評価していますか？

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選択をレビューし、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:

- コスト最適化担当を設定する: 組織全体のコストと使用状況を定期的に確認するチームを作ります。
- ワークロードレビュープロセスを開発する: ワークロードレビューの基準とプロセスを定義するプロセスを開発します。レビューを行う際には、潜在的利益を織り込む必要があります。例えば、請求の10%以上の価値を持つコアワークロードは四半期ごとにレビューし、10%未満のワークロード年に 1 回レビューするなどです。
- サービスを予定外でレビューして導入する: 新しいサービスを予定外で導入します。
- このワークロードを定期的にレビューして分析する: 既存のワークロードは、定義済みのプロセスに従って定期的にレビューされます。
- 新しいサービスリリースに関する最新情報を入手する: 専門家や APN パートナーに定期的に相談し、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。