AWS Well-Architected フレームワーク

2020 年 7 月

This paper has been archived.

The latest version is now available at:

https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/framework/welcome.html

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、この文書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的としており、(b) AWS の現行製品と慣行ついて説明していますが、予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤーまたはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または暗示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本ドキュメントは、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 による優れた設計の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	開発をサポートし、ワークロードを効率的に実行し、運用に関する洞察を得て、ビジネス価値をもたらすためのサポートプロセスと手順を継続的に改善する能力。
セキュリティ	このセキュリティの柱では、データ、システム、資産を保護して、クラウドテクノロジーを活用し、セキュリティを向上させる機能について説明します。
信頼性	期待されるタイミングで、意図した機能を正確かつ一貫して実行するワークロードの能力。これには、そ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機能が含まれます。
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。
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、ゴーライブ、および本番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。
- 組織内のテクノロジーポートフォリオは、ビジネスの運営に必要なワークロードの集合体です。

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。[1] 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 [2] 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 [3] を実行するメカニズム 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 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]

[2] 方法、プロセス、標準、受け入れられている基準です。

[3] 「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。

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、 開発をサポートし、ワークロードを効率的に実行し、運用に関する洞察を得て、ビジネス価値をもたらすためのサポートプロセスと手順を継続的に改善する能力。 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、5 つの設計原則があります。

- 運用をコードとして実行する: クラウドでは、アプリケーションコードに使用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。
- 小規模かつ可逆的な変更を頻繁に行う: コンポーネントを定期的に更新できるようにワークロードを設計します。変更は、失敗した場合に元に戻すことができるように小規模に行います (可能な場合は、顧客に影響がないようにします)。
- 運用手順を定期的に改善する: 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、4 つのベストプラクティスの分野があります。

- 組織
- 準備
- 運用
- 進化

組織のリーダーシップは、ビジネス目標を定義します。組織は、要件と優先順位を理解し、これらを使用してビジネスの成果を達成するための作業を整理し、指導する必要があります。ワークロードはサポートに必要な情報を送出する必要があります。ワークロードの統合、デプロイ、提供を可能にするサービスを実装することで、反復プロセスが自動化され、本番環境への有益な変化の流れを増やすことができます。

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

ベストプラクティス

組織

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

ビジネスに対する脅威 (たとえば、ビジネスリスクと負債や情報セキュリティの脅威) を評価し、この情報をリスクレジストリに保持します。リスクの影響を評価し、競合する利益のトレードオフや代替アプローチを評価します。たとえば、新しい機能の市場投入までの時間を短縮することは、コストの最適化よりも重視されます。または、非リレーショナルデータ用にリレーショナルデータベースを選択すれば、リファクタリングなしで、システムの移行が簡素化されます。メリットとリスクを管理し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。一部のリスクや選択肢は、一定期間許容される可能性があり、関連するリスクを軽減できる場合もあれば、リスクが残るのを容認できない場合もあります。その場合、リスクに対処するための措置を講じることになります。

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

アプリケーション、ワークロード、プラットフォーム、インフラストラクチャの各コンポーネントの所有者が特定されていること、および各プロセスと手順の定義を担当する所有者、およびそのパフォーマンスに責任を持つ所有者が特定されていることを確認します。各コンポーネント、プロセス、手順のビジネス価値、それらのリソースが配置されている理由やアクティビティが実行されている理由、所有権が存在する理由を理解することで、チームメンバーのアクションが明らかになります。チームメンバーの責任を明確に定義することで、当該メンバーが適切に行動し、責任と所有権を識別するメカニズムを持つことができます。イノベーションを制約しないように、追加、変更、例外をリクエストするメカニズムを備えます。チームがどのように連携して相互にサポートするのか、また、ビジネスの成果について、チーム間の合意を定義します。

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

学習を加速するための実験を奨励し、チームメンバーが関心と当事者意識を持ち続けるようにします。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。専ら学習のために設けられた

時間を提供することで、これをサポートし、奨励します。チームメンバーが成功し、ビジネスの成果をサポートするためにスケールできるように、ツールとチームメンバーの両方のリソースを持っていることを確認します。組織間の多様性を活用して、複数のユニークな視点を追求します。この視点を使用して、イノベーションを高め、想定に挑み、確証バイアスに傾くリスクを軽減します。チーム内のインクルージョン、多様性、アクセシビリティを向上させ、有益な視点を得ます。

組織に適用される外部の規制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 では、開発前のアプローチ、本番稼働前のワークロードの状態、本番稼働中のワークロードの状態などを確認するのに役立つ AWS Well-Architected Tool を提供しています。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 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顧客は、優先順位をさらに高めることができるセキュリティ、信頼性、パフォーマンス、コストの最適化に重点を置いた追加のチェックにアクセスできます。

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

AWS Organizations など、アカウント間で環境を集中管理できるツールまたはサービスを使用して、運用モデルの管理に役立てる必要があります。AWS Control Tower などのサービスでは、この管理機能が拡張され、アカウントのセットアップに関する設計図 (運用モデルのサポート) を定義し、AWS Organizations を使用して進行中のガバナンスを適用し、新しいアカウントのプロビジョニングを自動化できます。AWS マネージドサービス、AWS マネージドサービスパートナー、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: 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が成功するためのそれぞれの役割を理解し、自分たちのチームが成功するための他のチームの役割を理解し、目標を共有する必要があります。責任、所有権、意思決定方法、意思決定を行う権限を持つユーザーを理解することは、労力を集中的に投入し、チームの利点を最大化するのに役立ちます。
OPS 3: 組織の文化はビジネスの成果をどのようにサポートしますか?
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。

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

準備

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

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

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。これらは、本番環境移行時における有益な変更を促進し、デプロイされた問題を制限するとともに、お客様の環境において、デプロイメントアクティビティを通じて生じた問題、または検出された問題をすばやく特定し、修正できるようにします。

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

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

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

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 4: どのようにワークロードを設計して、その状態を理解できるようにするのですか?
ワークロードを設計する際には、すべてのコンポーネント (メトリクス、ログ、トレースなど) にわたって内部状態を理解するために必要な情報が送出されるようにします。そうすることによって、適時に有用な返答を提供できるようになります。
OPS 5: どのように欠陥を減らし、修正を容易にして、本番環境へのフローを改善するのですか?
リファクタリング、品質についてのすばやいフィードバック、バグ修正を可能にし、本番環境への変更のフローを改善するアプローチを採用します。これらにより、本番環境に採用される有益な変更を加速させ、デプロイされた問題を制限できます。またデプロイアクティビティを通じて挿入された問題をすばやく特定し、修復できます。
OPS 6: 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 7: 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。

運用アクティビティをコードとして実装することに投資することにより、運用担当者の生産性を最大に引き上げ、エラーの発生を最小限に抑え、自動応答を可能にします。「事前予測」のアプローチで、失敗を予測し、必要に応じて手順を作成します。一貫したタグ付け戦略に従って、リソースタグと AWS リソースグループを使用してリソースを識別できるようにします。組織、原価計算、アクセスコントロールのリソースにタグを付け、自動化された運用アクティビティの実行に的を絞ります。クラウドの伸縮性を活用したデプロイ方法を導入し、開発活動を促進し、システムの事前デプロイを促進して実装を高速化します。ワークロードを評価するために使用するチェックリストに変更を加える場合は、準拠しなくなった本番システムで行うことを計画します。

運用

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

運用上の優秀性を実現するには、運用上のイベントを効率的かつ効果的に管理する必要があります。計画的および予期しない運用イベントの両方に適用されます。十分に把

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

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

AWS では、ワークロードおよび AWS からネイティブに収集したメトリクスのダッシュボードビューを作成できます。CloudWatch またはサードパーティ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 8: 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 9: 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 10: 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。

収集するすべてのメトリクスは、ビジネスニーズと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、イベントを認識した場合のパフォーマンスを自動化します。

進化

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

リクエスト、問題の修正、コンプライアンス要件など) を定期的に評価し、優先順位を付けます。手順にフィードバックループを取り入れ、改善が必要な分野をすばやく特定し、実際に運用して教訓を学びます。

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

AWS では、ログデータを Amazon S3 にエクスポートしたり、ログを直接 Amazon S3 に送信して、長期保存したりできます。AWS Glue を使用すると、分析のために Amazon S3 でログデータを検出して準備し、関連するメタデータを AWS Glue データカタログに保存できます。Amazon Athena は 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 11: 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。

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、このセキュリティの柱では、データ、システム、資産を保護して、クラウドテクノロジーを活用し、セキュリティを向上させる機能について説明します。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。アイデンティティ管理を一元化し、長期にわたる静的な認証情報に依存しないようにすることを目的とします。
- 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。ログとメトリクスの収集をシステムに統合して、自動的に調査しアクションを実行します。
- 全レイヤーでセキュリティを適用する: 複数のセキュリティコントロールを使用して深層防御アプローチを適用します。ネットワークのエッジ、VPC、ロードバランシング、すべてのインスタンスとコンピューティングサービス、オペレーティングシステム、アプリケーション、コードなど、すべてのレイヤーに適用します。
- セキュリティのベストプラクティスを自動化する: 自動化されたソフトウェアベースのセキュリティメカニズムにより、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したり、排除したりするメカニズムとツールを使用します。これにより、機密性の高いデータを扱う際の誤処理、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に備える: 組織の要件に合わせたインシデント管理および調査のポリシーとプロセスを導入し、インシデントに備えます。インシデント対応シミュレーションを実行し、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、6 つのベストプラクティスの分野があります。

- セキュリティ
- Identity and Access Management
- 検出
- 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

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

AWS の責任共有モデルにより、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

セキュリティ

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

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

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。(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: ワークロードを安全に運用するには、どうすればよいですか?

ワークロードを安全に運用するには、セキュリティのすべての領域に包括的なベストプラクティスを適用する必要があります。組織レベルおよびワークロードレベルにおいて、運用上の優秀性で定義した要件とプロセスを抽出し、それらをすべての領域に適用します。AWS や業界のレコメンデーションおよび脅威インテリジェンスを最新に保つことで、脅威モデルと管理の目標を進化させることができます。セキュリティプロセス、テスト、検証を自動化することで、セキュリティオペレーションをスケールできます。

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

Identity and Access Management

アイデンティティとアクセスの管理は情報セキュリティプログラムの重要な要素です。これにより、お客様が意図した方法で承認、認証されたユーザーおよびコンポーネントのみがリソースにアクセスできるようになります。たとえば、プリンシパル (つまり、お客様のアカウントに対してアクションをとるアカウント、ユーザー、ロール、サービス) を定義し、これらのプリンシパルに合わせたポリシーを構築し、強力な認証情報管理を実装できます。これらの権限管理機能は認証と承認の中枢となっています。

AWS では、権限管理は主に AWS Identity and Access Management (IAM) サービスによってサポートされ、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。

SEC 2: ユーザー ID とマシン ID はどのように管理したらよいでしょうか?
安全に AWS ワークロードを運用するアプローチには、管理する必要があるアイデンティティが 2 種類あります。管理およびアクセス権を付与する必要があるアイデンティティのタイプを理解することで、適切な ID が適切な条件下で適切なリソースにアクセスできるようになります。ユーザー ID: 管理者、開発者、オペレーター、エンドユーザーは、AWS 環境とアプリケーションにアクセスするために ID を必要とします。これらの者は、あなたの組織のメンバー、または共同作業を行う外部ユーザーで、ウェブブラウザ、クライアントアプリケーション、またはインタラクティブなコマンドラインツールを介して AWS リソースを操作する人たちです。マシン ID: サービスアプリケーション、運用ツール、およびワークロードには、たとえばデータの読み取りを行うために、AWS のサービスにリクエストを送るための ID が必要です。このような ID には、Amazon EC2 インスタンスや AWS Lambda 関数など、AWS 環境で実行されているマシンが含まれます。また、アクセスを必要とする外部関係者のマシン ID を管理することもできます。さらに、AWS 環境にアクセスする必要があるマシンが AWS 外にある場合もあります。
SEC 3: 人とマシンのアクセス許可はどのように管理すればよいでしょうか?
アクセス許可を管理して、AWS とワークロードへのアクセスを必要とするユーザー ID やマシン ID へのアクセスを制御します。アクセス許可は、どの条件で誰が何にアクセスできるかを制御します。

すべてのユーザーやシステムが認証情報を共有してはいけません。ユーザーアクセス権は、パスワード要件や MFA の強制などのベストプラクティスを実践した上で、最小権限で与えられるべきです。AWS 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、認証情報と認証の管理、人的アクセスの制御、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参照してください。

検出

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。

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 5: ネットワークリソースをどのように保護しますか?

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

SEC 6: 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 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。
SEC 8: 保管時のデータをどのように保護していますか?
複数のコントロールを実装して保管中のデータを保護し、不正アクセスや不正処理のリスクを低減します。
SEC 9: 転送時のデータをどのように保護していますか?
複数のコントロールを実装して、転送中のデータを保護し、不正アクセスや損失のリスクを軽減します。

AWS には、保存中および伝送中のデータを暗号化する手段が複数あります。データの暗号化を容易にする機能を各サービスに搭載しています。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 10: インシデントの予測、対応、復旧はどのように行いますか?
組織に支障をきたすことを最小限に抑えるために、セキュリティインシデントのタイムリーで効果的な調査、対応、復旧に備えることが重要です。

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

リソース

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、期待されるタイミングで、意図した機能を正確かつ一貫して実行するワークロードの能力。これには、そ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機能が含まれます。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。

- 障害から自動的に復旧する: ワークロードで主要業績評価指標 (KPI) をモニタリングすることで、しきい値を超えた場合にオートメーションをトリガーできます。この場合の KPI は、サービスの運用の技術的側面ではなく、ビジネス価値に関する指標である必要があります。これにより自動的に障害を通知および追跡することができ、障害を回避または修正する復旧プロセスも自動化できます。より複雑な自動化を使用すると、障害が発生する前に修正を予期できます。
- 復旧手順をテストする: オンプレミス環境においては、多くの場合、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障害が発生するかをテストでき、復旧の手順も検証できます。自動化により、さまざまな障害のシミュレーションや過去の障害シナリオの再現を行うことができます。このアプローチでは、実際の障害シナリオが発生する前にテストおよび修正できる障害経路が公開されるため、リスクが軽減されます。
- 水平方向にスケールしてワークロード全体の可用性を高める: 1 つの大規模なリソースを複数の小規模なリソースに置き換えることで、単一の障害がワークロード全体に与える影響を軽減します。リクエストを複数の小規模なリソースに分散させることで、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することをやめる: オンプレミスのワークロードにおける障害の一般的な原因はリソースの飽和状態で、ワークロードに対する需要がそのワークロードのキャパシティーを超えたときに発生します (よくサービス妨害攻撃の目標となります)。クラウドでは、需要とワークロード使用率をモニタリングし、リソースの追加と削除を自動化することで、プロビジョニングが過剰にも過小にもならない、需要を満たせる最適なレベルを維持できます。制限はまだありますが、制御および管理できるクォータもあります (サービスクォータと制約の管理を参照)。
- オートメーションで変更を管理する: インフラストラクチャに対する変更は、オートメーションを使用して実行する必要があります。管理する必要がある変更には、自動化に対する変更が含まれており、それを追跡して確認することができます。

定義

クラウドでの信頼性には、4 つのベストプラクティスの分野があります。

- 基盤
- ワークロードアーキテクチャ
- 変更管理
- 障害の管理

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

基盤

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

AWS では、このよう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: サービスクォータと制約はどのように管理しますか?
クラウドベースのワークロードアーキテクチャには、サービスクォータ (サービスの制限とも呼ばれます) というものがあります。このようなクォータは、誤って必要以上のリソースをプロビジョニングするのを防ぎ、サービスを不正使用から保護することを目的として API 操作のリクエスト頻度を制限するために存在します。リソースにも制約があります。たとえば、光ファイバーケーブルのビットレートや、物理ディスクの記憶容量などです。
REL 2: ネットワークトポロジをどのように計画しますか?
多くの場合、ワークロードは複数の環境に存在します。このような環境には、複数のクラウド環境 (パブリックにアクセス可能なクラウド環境とプライベートの両方) と既存のデータセンターインフラストラクチャなどがあります。計画する際は、システム内およびシステム間の接続、パブリック IP アドレスの管理、プライベート IP アドレスの管理、ドメイン名解決といったネットワークに関する諸点も考慮する必要があります。

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

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、5 つの Well-Architected の柱すべてにかけてワークロードの動作に影響を与えます。高い信頼性を保つには、特定のパターンに従う必要があります。

AWS では、ワークロード開発者は使用する言語とテクノロジーを選択できます。AWS SDK は、AWS のサービス用に言語固有の API を提供することで、コーディングの複雑さを排除します。これらの SDK と言語の選択により、開発者はここにリスト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、The Amazon Builders' Library で Amazon がソフトウェアを構築および運用する方法について読み、学ぶこともできます。

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: 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、拡張性と信頼性の高いワークロードを構築します。サービス指向アーキテクチャ (SOA) は、サービスインターフェイスを介してソフトウェアコンポーネントを再利用できるようにする方法です。マイクロサービスアーキテクチャは、その一歩先を行き、コンポーネントをさらに小さくシンプルにしています。
REL 4: 障害を防ぐために、分散システムの操作をどのように設計しますか?
分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。これらのベストプラクティスは、故障を防ぎ、平均故障間隔 (MTBF) を改善します。
REL 5: 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?
分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。これらのベストプラクティスに従うことで、ワークロードはストレスや障害に耐え、より迅速に復旧し、そのような障害の影響を軽減できます。その結果、平均復旧時間 (MTTR) が向上します。

分散システムは、サーバーやサービスなどのコンポーネントを相互接続するために通信ネットワークを利用しています。このネットワークでデータの損失やレイテンシーがあっても、ワークロードは確実に動作する必要があります。分散システムのコンポーネントは、他のコンポーネントやワークロードに悪影響を与えない方法で動作する必要があります。

変更管理

ワークロードを信頼できる形で実行するには、ワークロードやその環境に対する変化を予測して対応することが不可欠です。変更には、需要の急増などのワークロードに負荷

がかかる変更や、機能のデプロイやセキュリティパッチの適用といった内部からの変更があります。

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

以下の質問は、信頼性に関するこれらの考慮事項に焦点を当てています。

REL 6: ワークロードリソースをモニタリングするにはどうすればよいですか?
ログとメトリクスは、ワークロードの状態についての洞察を得るための強力なツールです。ワークロードは、しきい値を超えたり重大なイベントが発生したりしたときに、ログとメトリクスがモニタリングされて通知が送信されるように構成できます。モニタリングにより、ワークロードは、低パフォーマンスのしきい値を超えたときや障害が発生したときにそれを認識できるため、それに応じて自動的に復旧できます。
REL 7: 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?
スケーラブルなワークロードには、リソースを自動で追加または削除する伸縮性があるので、リソースは常に、現行の需要に厳密に適合します。
REL 8: 変更はどのように実装するのですか?
変更制御は、新しい機能をデプロイしたり、アプリケーションと運用環境で既知のソフトウェアが実行されており、予測できる方法でパッチを適用または置換できることを確認したりするために必要です。変更が制御されていないと、変更の影響を予測したり、変更によって発生した問題に対処したりすることが困難になります。

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

障害の管理

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。信頼性は、発生時点で障害を認識し、可用性への影響を回避するための措置を講じることをワークロードに要求します。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 9: データはどのようにバックアップするのですか?
目標復旧時間 (RTO) と目標復旧時点 (RPO) の要件を満たすように、データ、アプリケーション、設定をバックアップします。
REL 10: ワークロードを保護するために障害分離をどのように使用しますか?
障害部分を分離した境界は、ワークロード内の障害の影響を限られた数のコンポーネントに限定します。境界外のコンポーネントは、障害の影響を受けません。障害部分を切り離した境界を複数使用すると、ワークロードへの影響を制限できます。
REL 11: コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?
高い可用性と低い平均復旧時間 (MTTR) の要件を持つワークロードは、高い弾力性があるように設計する必要があります。
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、期待する弾力性を実現することを確認する唯一の方法が、テストを行うことです。
REL 13: 災害対策 (DR) はどのように計画するのですか?
バックアップと冗長ワークロードコンポーネントを配置することは、DR 戦略の出発点です。RTO と RPO は、可用性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ロケーションと機能を考慮して、目標を達成するための戦略を実装します。

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

リソース

信頼性に関する 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 Documentation
- AWS Global Infrastructure
- AWS Auto Scaling: How Scaling Plans Work
- What Is AWS Backup?

ホワイトペーパー

- Reliability Pillar: AWS Well-Architected
- Implementing Microservices on AWS

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。 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: 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、チームがより簡単に高度なテクノロジーを実装できるようにします。IT チームに新しいテクノロジーのホス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。たとえば、NoSQL データベース、メディアトランスコーディング、および機械学習は、すべて特化された専門知識を必要とするテクノロジーです。クラウドでは、これらのテクノロジーがチームによる消費が可能なサービスとなり、チームはリソースのプロビジョニングと管理ではなく、製品の開発に集中できるようになります。
- わずか数分でグローバル展開する: 世界各地にある複数の 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は、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して維持する必要性を取り除きます。たとえば、サーバーレスストレージサービスは静的ウェブサイトとして機能させることができ (ウェブサ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。
- システムに対する精通の程度を考慮する: 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。たとえば、データベース、またはストレージのアプローチを選択するときは、データアクセスパターンを考慮します。

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

データ駆動型アプローチを採用して、高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 クラウドを十分に活かしていることを確認します。モニタリングは、期待されるパフォーマンスからの逸脱を確実に把握できるようにします。パフォーマンスを向上させるために、圧縮やキャッシングの使用、または整合性要件の緩和などのアーキテクチャ面でのトレードオフを実施します。

ベストプラクティス

選択

特定のワークロードに最適な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、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を使用し、異なる機能を有効化します。

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、ニーズにしっかりと適合するアプローチを見つけることが容易になります。また、オンプレミスのインフラストラクチャでは実現が難しいオプションも見つけることができます。たとえば、Amazon DynamoDB などのマネージドサービスは、どのような規模でもレイテンシーが 10 ミリ秒未満の完全マネージド型 NoSQL データベースサービスを提供します。

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: どのように最良パフォーマンスのアーキテクチャを選択するのですか?
多くの場合、ワークロード全体での最適なパフォーマンスのためには、複数のアプローチが必要になります。Well-Architected なシステムでは、パフォーマンスを向上させるために複数のソリューションと機能が使用されています。

データ主導のアプローチを用いて、アーキテクチャのパターンと実装を選択し、コスト効率性に優れたソリューションを実現します。AWS ソリューションアーキテクト、AWS リファレンスアーキテクチャ、および AWS パートナーネットワーク (APN) のパートナーは、業界知識に基づいたアーキテクチャを選択する支援を行うことができますが、アーキテクチャを最適化するには、ベンチマークまたは負荷テストを通じて得られ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)。アーキテクチャの実装には、アーキテクチャのパフォーマンスの最適化に固有の AWS のサービスが使用されます。以下のセクションでは、考慮すべき 4 つの主なリソースタイプ、つまりコンピューティング、ストレージ、データベース、およびネットワークについて説明します。

コンピューティング

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

AWS では、インスタンス、コンテナ、関数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であり、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。クラウドではリソースを自由に決定できることから、異なるサーバータイプで実験できます。AWS では、これらの仮想サーバーに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) およびグラフィック処理ユニット (GPU) などの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法であり、リソースが分離されているプロセスでアプリケーションとその依存関係を実行することが可能です。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。または、コンピューティング環境のインストール、設定、および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。

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

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

ストレージ

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

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

- 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の場所からデータへのアクセスを可能にする、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、パフォーマンスを備えたオブジェクトストレージサービスです。Amazon S3 は 99.999999999% (9 が 11 個) の耐久性を実現するように設計されており、世界中の企業向けに数百万ものアプリケーションのデータを保存しています。
- ブロックストレージは、仮想ホストごとに、高可用性かつ一貫性のある低レイテンシーのブロックストレージを提供します。これは、ダイレクトアタッチトストレージ (DAS) またはストレージエリアネットワーク (SAN) に類似しています。Amazon Elastic Block Store (Amazon EBS) は、EC2 インスタンスからアクセスできる永続的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 (EFS) などのファイルストレージソリューションは、大規模なコンテンツリポジトリ、開発環境、メディアストア、ユーザーのホームディレクトリなどのユースケースに最適です。Amazon FSx を使用する

と、一般的なファイルシステムを簡単かつコスト効率よく起動して実行できるため、広く使用されていて、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

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

AWS では、リレーショナル、key-value、ドキュメント、インメモリ、グラフ、時系列、および台帳データベースを含めた複数の専用データベースエンジンから選択できます。AWS のデータベースを使えば、サーバーのプロビジョニング、パッチ適用、セットアップ、設定、モニタリング、バックアップ、復旧などのデータベース管理タスクについて頭を悩ます必要はありません。AWS はクラスターを継続的に監視して、自己修復ストレージと自動スケーリングを使用してワークロードを稼働させ、より高価値なアプリケーション開発に集中できるようにします。

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。

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は、パフォーマンス効率に大きな影響を与えます。この領域では、多くの場合、データ駆動型のアプローチではなく、組織の標準に従って選択がなされます。ストレージと同様、ワークロードのアクセスパターンを検討

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

ネットワーク

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

AWS ではネットワーキングが仮想化されており、数多くの異なるタイプと設定で利用することができます。ネットワーキング手法とニーズの適合が容易になります。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。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、および AWS Global Accelerator などの、ネットワーク距離を縮め、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: どのようにネットワーキングソリューションを選択するのですか?

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

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

レビュー

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

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

以下の質問は、パフォーマンス効率に関する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、時間と共に、ワークロードのパフォーマンスを向上させることができる新しいテクノロジーとアプローチが利用できるようになります。

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

モニタリング

ワークロードを実装した後は、そのパフォーマンスをモニタリングして、問題が顧客に影響を及ぼす前に修正できるようにする必要があります。モニタリングメトリクスは、しきい値を超えたときにアラームを発行するために使用するようにしてください。

Amazon CloudWatch は、モニタリングおよび可観測性のサービスで、ワークロードをモニタリングし、システム全体のパフォーマンスの変化に対応し、リソースの使用率を最適化し、運用状態の統合ビューを取得するためのデータと実用的な洞察を提供します。CloudWatch は、AWS およびオンプレミスのサーバーで実行されるワークロードから、ログ、メトリクス、イベントの形式でモニタリングおよび運用データを収集します。AWS X-Ray は、開発者が本稼働の分散アプリケーションを分析およびデバッグするのに役立ちます。AWS X-Ray を使用すると、アプリケーションの動作に関する洞察を得て、根本原因を検出し、パフォーマンスのボトルネックを特定できます。これらのインサイトを使用することで、速やかに対応し、ワークロードのスムーズな動作を維持できます。

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

PERF 7: リソースが稼働していることを確実にするためのリソースのモニタリングはどのように行いますか?

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

誤判定がないことを確実にすることは、効果的なモニタリングソリューションの鍵です。自動化されたトリガーは、人為的なミスを防ぎ、問題の修正にかかる時間を短縮で

きます。解決までの時間を短縮できます。本番環境でシミュレーションを実施するゲームデーを計画して、アラームソリューションをテストし、それが問題を正しく認識することを確認します。

トレードオフ

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

AWS を使用すれば、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、プライマリデータベースの負荷を減らすため、情報ストア (データベースシステムなど) に読み取り専用のレプリカを動的に追加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 8: トレードオフをどのように使用するとパフォーマンスが向上するのですか?

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

ワークロードに変更を加えた後、メトリクスを収集して評価し、それらの変更の影響を判断します。システムおよびエンドユーザーに対する影響を測定して、トレードオフがワークロードにどのような影響を与えるかを理解します。負荷テストなどの体系的なアプローチを使用して、トレードオフによってパフォーマンスが向上するかどうかを調べます。

リソース

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 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。

- クラウド財務管理の実装: クラウドで財務面での成功を達成し、ビジネス価値の実現を加速するには、クラウド財務管理/コスト最適化に投資する必要があります。組織は、テクノロジーと使用状態の管理というこの新しい領域における能力を獲得するために、時間とリソースを投入する必要があります。コスト効率の高い組織になるには、セキュリティまたはオペレーションの能力と同様、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: 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。
- 差別化につながらない高負荷の作業に費用をかけるのをやめる: AWS は、サーバーのラック運用、積み上げ、電源供給などのデータセンターの運用の高負荷の作業を行います。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オペレーション上の負担も解消されます。この結果、お

客様は 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。

- 費用を分析および属性化する: 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の分野があります。

- 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
- 経時的な最適化

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

ベストプラクティス

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、より迅速にイノベーションを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。AWS Budgets は、コストと使用状況に関する事前通知を提供します。AWS ブログでは、お客様が新しいサービスリリースの最新の状況を常に知ることができるように、新しいサービスと機能に関する情報を提供しています。

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: クラウド財務管理はどのように実装しますか?
組織はクラウド財務管理の実装により、AWS によってコストと使用状況の最適化とスケーリングをすることで、ビジネス価値と財務上の成功を実現できます。

コスト最適化機能を構築するときは、メンバーを引き入れるとともに、CFM および CO のエキスパートでチームを補完することも検討してください。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、および改善を迅速に実施する方法を理解します。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ようとする際には、既存のプログラムおよびプロセスを改善または構築することを検討します。新しいプロセスおよびプログラムを構築するよりも、既存のものに追加する方がはるかに迅速です。これにより、結果を得るまでの時間が大幅に短縮されます。

経費支出と使用量の認識

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

多くのビジネスは、さまざまなチームが運用する複数のシステムによって構成されています。リソースのコストをそれぞれの組織や製品オーナーに帰属させることができると、リソースを効率的に使用し、無駄を削減できます。コストの帰属を明確にすること

で、実際に利益率の高い製品を把握し、予算の配分先についてより多くの情報に基づいた決定ができるようになります。

AWS では、AWS Organizations または AWS Control Tower を使用してアカウント構造を作成します。これにより、分離が可能になり、コストおよび使用量の配分に役立ちます。リソースにタグ付けして、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、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 2: 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 3: 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 4: 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。

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 で日々の詳細を使用します。より詳細な分析と検査を行うには、AWS Cost Explorer で時間単位の粒度を使用するか、コストと使用状況レポート (CUR) を使用して時間単位の粒度で Amazon Athena と Amazon QuickSight を使用します。

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

費用対効果の高いリソース

ワークロードにとって適切なインスタンスとリソースを使用することが、コスト削減の鍵になります。たとえば、レポート処理を小規模なサーバーで実行すると 5 時間かか

るものの、費用が 2 倍の大規模なサーバーでは 1 時間で実行できるとします。どちらのサーバーでも同じ結果が得られますが、長期的に見れば小規模なサーバーで発生するコストの方が大きくなります。

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

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により、最小コミットメントの定めなく、コンピューティングキャパシティーに対して 1 時間単位で料金を支払うことができます。Savings Plans とリザーブドインスタンスは、オンデマンド料金に対して最大 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 5: 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 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。
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 8: 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。

需要を管理し、リソースを供給する

クラウドに移行すると、お支払いは必要な分のみになります。必要な時にワークロードの需要に合わせたリソースを供給できるため、コストがかかる無駄なオーバープロビジョニングの必要性を排除できます。また、スロットル、バッファ、またはキューを使用して、需要を滑らかにし、少ないリソースで供給することで、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これらの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。需要を管理する際には、適切なサイズのキューまたはバッファがあること、および必要な時間内にワークロードの需要に応答していることを確認します。

経時的な最適化

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。

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、ワークロードを定期的に確認し、新しいサービスや機能を実装することが不可欠です。

以下の質問は、コスト最適化に関するこれらの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。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。

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

- AWS Documentation

ホワイトペーパー

- Cost Optimization Pillar

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。[1]

変更が困難な一方通行のドア (のような決定) 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

本ドキュメントは、次の人物および組織が寄稿しました。

- Rodney Lester: 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
- Eric Pullen: Well-Architected パフォーマンスリード、アマゾン ウェブ サービス
- Seth Eliot: Well-Architected 信頼性リード、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 Cloud Compliance

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

The Amazon Builders' Library

改訂履歴

表2 主な改訂

日付	説明
2020 年 7 月	ほとんどの質問と回答を見直して書き換えました。
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

組織

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重点領域を決定します。これにより、望ましいビジネス成果を達成するために必要なオペレーションサポートについて十分に理解できます。
- 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。
- ガバナンス要件を評価する: 特定のことに焦点を置くように求め、その焦点を強調する組織の定義したガイドラインや義務を把握します。組織のポリシー、標準、要件などの内部要因を評価します。ガバナンスへの変更を識別するメカニズムがあることを検証します。ガバナンス要件が特定されていない場合は、必ずこの決定にデューデリジェンスが適用されていることを確認してください。
- 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、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。
- トレードオフを評価する: 競合する利益または代替アプローチ間のトレードオフの影響を評価し、重点領域を決定するか、一連のアクションを選択する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、新しい機能の市場投入までの時間を短縮することは、コストの最適化よりも重視されることがあります。または、非リレーショナルデータ用にリレーショナルデータベースを選択すれば、データ型に合わせて最適化されたデータベースに移行してアプリケーションを更新するよりも、システムの移行が簡素化されます。
- メリットとリスクを管理する: メリットとリスクを管理し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、重要な新機能を顧客に公開できるように、未解決の問題を記録するワークロードをデプロイしておくと便利な場合があります。関連するリスクを軽減できる場合もあれば、リスクが残るのを容認できない場合もあります。その場合、リスクに対処するための措置を講じることになります。

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

ベストプラクティス:

- 特定された所有者がリソースについて存在している: 各アプリケーション、ワークロード、プラットフォーム、インフラストラクチャコンポーネントの所有権を持つユーザーが誰か、そのコンポーネントによって提供されるビジネス価値、およびその所有権が存在する理由を理解します。これらの個々のコンポーネントのビジネス価値、およびそれらがビジネスの成果をどのようにサポートするかを理解すると、それらに適用されるプロセスと手順がわかります。
- プロセスと手順が所有者を特定: 個々のプロセスと手順の定義を誰が所有しているか、特定のプロセスと手順が使用されている理由、およびその所有権が存在する理由を理解します。特定のプロセスと手順が使用される理由を理解することで、改善の機会を特定できます。
- 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ついて存在している: 定義されたワークロードに対して特定のアクティビティを実行する責任を持つ者と、その責任が存在する理由を理解します。運用アクティビティを実行することに責任を負う者を理解すると、誰がアクティビティを実行し、結果を検証し、アクティビティの所有者にフィードバックを提供するかを通知します。
- チームメンバーが自らの責任範囲を知る: 役割の責任と、ビジネスの成果にどのように貢献するかを理解することで、タスクの優先順位付けと役割が重要である理由を知ることができます。これにより、チームメンバーはニーズを認識し、適切に対応できます。
- 責任と所有権を特定するためのメカニズムが存在する: 個人またはチームが特定されない場合、対処される必要がある事項についての所有権または計画を割り当てる権限を持つ者へのエスカレーションパスが定義されています。
- 追加、変更、例外をリクエストするメカニズムが存在する: プロセス、手順、およびリソースの所有者にリクエストを送信できます。利点とリスクを評価した後、実行可能で適切であると判断されたリクエストを、十分な情報に基づいて承認します。
- チーム間の責任は事前定義済みまたは交渉済み: チーム間には、チームがどのように連携してサポートするかを説明する定義済みまたは交渉済みの合意があります (応答時間、サービスレベル目標、サービスレベルアグリーメントなど)。チームの仕事がビジネスの成果に及ぼす影響、および他のチームや組織の成果を理解することで、タスクの優先順位付けを知り、適切に対応できるようになります。

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:

- エグゼクティブスポンサー: シニアリーダーシップは、組織に対する期待を明確に設定し、成功を評価します。シニアリーダーシップは、ベストプラクティスの採用と組織の進化の協賛者、支持者、および推進者です。
- 結果にリスクがあるときにアクションを実行する権限が、チームメンバーに付与されている: ワークロード所有者は、結果にリスクがあるときにチームメンバーが対応できるようにするためのガイダンスと範囲を定義しています。エスカレーションメカニズムは、イベントが定義された範囲外にある場合に対応の方向性を取得するために使用されます。
- エスカレーションが推奨されている: チームメンバーにはメカニズムがあり、結果にリスクがあると思われる場合は、意思決定者や利害関係者に懸念をエスカレートすることが推奨されます。エスカレーションは、リスクを特定し、インシデントの発生を防ぐために、早く、頻繁に実行する必要があります。
- コミュニケーションがタイムリーで明確で実用的なものである: メカニズムが存在し、チームメンバーに既知のリスクや計画されたイベントをタイムリーに通知するために使用されます。アクションが必要かどうか、どのようなアクションが必要かを判断し、タイミングよくそのアクションを実行するために、必要なコンテキスト、詳細、および時間 (可能な場合) が提供されます。たとえば、パッチ適用を迅速に行えるようにソフトウェアの脆弱性を通知したり、サービス中断のリスクを回避するために変更のフリーズを実装できるように計画された販売プロモーションの通知を提供したりします。
- 実験が推奨されている: 実験は、学習を加速し、チームメンバーが関心と当事者意識を持ち続けることの一助となります。望ましくない結果は、成功につながらないパスを特定することに成功した実験です。望ましくない結果が得られた実験が成功してもチームメンバーが罰せられることはありません。イノベーションを起こし、アイデアを成果に変えるには、実験が必要です。
- チームメンバーがスキルセットを維持し、強化することが可能であり、推奨されている: チームは、ワークロードに対応するに際して、新しいテクノロジーを採用し、需要と責任の変化をサポートするために、スキルセットを強化する必要があります。新しいテクノロジーにおけるスキルの発達は、多くの場合、チームメンバーの満足度の源となり、イノベーションをサポートします。チームメンバーが強化している自らのスキルを検証および認識し、業界認証を追求および維持できるように支援します。組織の知識とスキルを持ち、熟練したチームメンバーを失った場合は、クロストレーニングによって知識の伝達を促進し、重大な影響のリスクを緩和します。学習のために専用の時間を割り当てます。
- チームに適正なリソースを提供する: チームメンバーの力量を維持し、ワークロードのニーズを満たすツールとリソースを提供します。チームメンバーに過剰な負荷がかかることは、人為的ミスに起因するインシデントのリスクを高めます。ツールやリソースへの投資 (頻繁に実行されるアクティビティのオートメーションなど) によって、チームの効果をスケールし、付加的なアクティビティをサポートできます。
- チーム内およびチーム間でさまざまな意見が奨励され、求められる: 組織間の多様性を活用して、複数のユニークな視点を追求します。この視点を使用して、イノベーションを高め、想定に挑み、確証バイアスに傾くリスクを軽減します。チーム内のインクルージョン、多様性、アクセシビリティを向上させ、有益な視点を得ます。

準備

OPS 4 どのようにワークロードを設計して、その状態を理解できるようにするのですか?

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

ベストプラクティス:

- 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 (クリックストリームのほか、開始、放棄、完了済みトランザクションなど) が送出されるよう、アプリケーションコードをインストルメント化します。この情報を使用して、アプリケーションの使用方法や使用パターンを理解したり、応答が必要とされるタイミングを特定したりできます。
- 依存関係のテレメトリーを実装する: 依存するリソースの状態 (到達可能性や応答時間など) に関する情報を出力するようにワークロードを設計および設定します。外部依存関係の例としては、外部データベース、DNS、ネットワーク接続などがあります。この情報を使用して、応答が必要とされるタイミングを特定します。
- トランザクショントレーサビリティを実装する: ワークロード全体のトランザクションフローに関する情報が送出されるよう、アプリケーションコードを実装し、ワークロードコンポーネントを設定します。この情報を使用して、応答が必要とされるタイミングを特定し、問題につながる要素の特定に役立てます。

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

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

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、テストにかかる労力を減らすためにテストを自動化します。
- 設定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。環境が本稼働環境に近づくにつれて増加するコントロールレベルを使用して、デプロイ時にワークロードが意図したとおりに運用するように確信を強化します。
- 小規模かつ可逆的な変更を頻繁に行う: 頻繁に、小さく、可逆的な変更を行うことで、変更の範囲と影響を減らします。これにより、トラブルシューティングが容易になり、修復がすばやくできるようになります。また変更を元に戻すこともできます。
- 統合とデプロイを完全に自動化する: 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 6 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 Canary 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正常な既知の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 7 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:

- 従業員の対応力を確保する: 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。
- 運用準備状況の継続的な確認を実現する: ワークロードの運用に関する準備状況を継続的に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要件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、イベントへの応答と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、イベントへの応答と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問題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。
- 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 8 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: 希望するビジネス上の業績 (注文率、顧客定着率、営業費用に対する利益など) と顧客に関する成果 (顧客満足度など) に基づいて、主要業績評価指標 (KPI) を特定します。KPI を評価して、ワークロードの成功を判別します。
- ワークロードのメトリクスを定義する: KPI の達成度を測定するワークロードメトリクスを定義します (たとえば、中止されたショッピングカート、注文数、コスト、価格、配分されたワークロード費用)。ワークロードの正常性 (インターフェイスの応答時間、エラー率、リクエスト数、完了したリクエスト、使用率など) を測定するワークロードのメトリクスを定義します。メトリクスを評価して、ワークロードが必要な成果に達しているかを判定し、ワークロードの正常性を把握します。
- ワークロードメトリクスを収集および分析する: メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。
- 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。改善、調査、および介入のしきい値を特定します。
- 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して異常な動作を識別し、必要に応じて適切に対応できるようにします。
- 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 9 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: 希望するビジネス (新機能など) と顧客の結果 (カスタマーサポートケースなど) に基づいて、主要業績指標 (KPI) を特定します。KPI を評価して、オペレーションの成功を判別します。
- 運用メトリクスを定義する: 運用メトリクスを定義して、KPI の達成度 (デプロイの成功、失敗したデプロイなど) を測定します。運用アクティビティの正常性を測定する運用メトリクスを定義します (たとえば、インシデントを検出する平均時間 (MTTD)、インシデントからの平均復旧時間 (MTTR) など)。メトリクスを評価して、運用アクティビティが必要な成果に達しているかを判定し、運用の正常性を把握します。
- 運用メトリクスを収集し、分析する: メトリクスのプロアクティブなレビューを定期的に行うと、傾向を把握し、適切な対応が必要な領域を特定できます。
- 運用メトリクスの基準値を設定する: 運用アクティビティのパフォーマンスを比較し、過不足を特定する基準となる期待値として、メトリクスに対する基準値を設定します。
- 運用に対して予想されるアクティビティのパターンを知る: 運用アクティビティのパターンを確立して異常なアクティビティを識別し、必要に応じて適切に対応できるようにします。
- 運用の結果にリスクがある場合に警告する: 運用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。
- 運用の異常が検出された場合に警告する: 運用の異常が検出された場合、必要に応じて適切な対応ができるよう、アラートを発生させます。
- 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 10 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。
- 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たときに、ユーザーと直接通信し (E メールや SMS など)、再び通常運用状態に復帰したときに再度通信し、ユーザーが適切な対応アクションを起こせるようにします。
- 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 11 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。
- インシデント後の分析を実行する: 顧客に影響を与えるイベントを確認し、寄与する要因と予防措置を特定します。この情報を使用して、再発を制限または回避するための緩和策を開発します。迅速で効果的な対応のための手順を開発します。対象者に合わせて調整された、寄与する要因と是正措置を必要に応じて伝えます。
- 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。
- ナレッジマネジメントを実行する: チームメンバーが探している情報をタイムリーに検出し、アクセスして、最新かつ完全であることを確認するメカニズムが存在しています。必要なコンテンツ、更新が必要なコンテンツ、および今後参照されることのないようにアーカイブする必要があるコンテンツを特定するためのメカニズムが存在しています。
- 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

セキュリティ

SEC 1 ワークロードを安全に運用するには、どうすればよいですか?

ワークロードを安全に運用するには、セキュリティのすべての領域に包括的なベストプラクティスを適用する必要があります。組織レベルおよびワークロードレベルにおいて、運用上の優秀性で定義した要件とプロセスを抽出し、それらをすべての領域に適用します。AWS や業界のレコメンデーションおよび脅威インテリジェンスを最新に保つことで、脅威モデルと管理の目標を進化させることができます。セキュリティプロセス、テスト、検証を自動化することで、セキュリティオペレーションをスケールできます。

ベストプラクティス:

- アカウントを使用してワークロードを分ける: 会社のレポート構造をミラーリングするのではなく、機能または共通のコントロールセットに基づいて、ワークロードを別々のアカウントに整理し、アカウントをグループ化します。セキュリティとインフラストラクチャを念頭に置いて、ワークロードが増大するにつれて組織が共通のガードレールを設定できるようにします。
- AWS アカウントのセキュリティを確保する: たとえば、MFA を有効にしてルートユーザーの使用を制限したり、アカウントの連絡先を設定したりすることで、アカウントへのアクセスを保護します。
- 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。管理目標と管理を継続的に検証することは、リスク軽減の有効性を測定するのに役立ちます。
- セキュリティ脅威に関する最新情報を入手する: 最新のセキュリティ脅威を常に把握して攻撃ベクトルを認識し、適切な管理を定義して実装できるようにします。
- セキュリティのレコメンデーションに関する更新情報を入手する: AWS と業界の両方のセキュリティのレコメンデーションを常に最新に保ち、ワークロードのセキュリティ体制を進化させます。
- パイプラインのセキュリティコントロールのテストと検証を自動化する: ビルド、パイプライン、プロセスの一環としてテストおよび検証されるセキュリティメカニズムの安全なベースラインとテンプレートを確立します。ツールとオートメーションを使用して、すべてのセキュリティコントロールを継続的にテストおよび検証します。たとえば、マシンイメージやインフラストラクチャなどの項目をコードテンプレートとしてスキャンして、セキュリティの脆弱性、不規則性、およびドリフトを各ステージで確立されたベースラインから確認します。
- 脅威モデルを使用してリスクを特定し、優先順位を付ける: 脅威モデルを使用して潜在的な脅威を特定し、その登録を最新の状態に維持します。脅威に優先順位を付け、セキュリティコントロールを調整して防止、検出、対応を行います。進化するセキュリティ環境の状況に応じてセキュリティコントロールを再確認および維持します。
- 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: AWS および APN パートナーは、ワークロードのセキュリティ体制を進化させることができる新しい機能とサービスを継続的にリリースしています。

Identity and Access Management

SEC 2 ユーザー ID とマシン ID はどのように管理したらよいでしょうか?

安全に AWS ワークロードを運用するアプローチには、管理する必要があるアイデンティティが 2 種類あります。管理およびアクセス権を付与する必要があるアイデンティティのタイプを理解することで、適切な ID が適切な条件下で適切なリソースにアクセスできるようになります。ユーザー ID: 管理者、開発者、オペレーター、エンドユーザーは、AWS 環境とアプリケーションにアクセスするために ID を必要とします。これらの者は、あなたの組織のメンバー、または共同作業を行う外部ユーザーで、ウェブブラウザ、クライアントアプリケーション、またはインタラクティブなコマンドラインツールを介して AWS リソースを操作する人たちです。マシン ID: サービスアプリケーション、運用ツール、およびワークロードには、たとえばデータの読み取りを行うために、AWS のサービスにリクエストを送るための ID が必要です。このような ID には、Amazon EC2 インスタンスや AWS Lambda 関数など、AWS 環境で実行されているマシンが含まれます。また、アクセスを必要とする外部関係者のマシン ID を管理することもできます。さらに、AWS 環境にアクセスする必要があるマシンが AWS 外にある場合もあります。

ベストプラクティス:

- 強力なサインインメカニズムを使用する: パスワードの最小長を適用し、一般的なパスワードやパスワードの再使用を避けるようにユーザーを教育します。ソフトウェアまたはハードウェアのメカニズムを使用した Multi-Factor Authentication (MFA) を義務化することで、セキュリティを強化できます。
- 一時的な認証情報を使用する: 一時的な認証情報を動的に取得するために ID を要求します。人員 ID の場合、AWS Single Sign-On、または IAM ロールによるフェデレーションを使用して AWS アカウントにアクセスします。マシン ID の場合、長期的なアクセスキーではなく IAM ロールの使用を要求します。
- シークレットを安全に保存して使用する: サードパーティーアプリケーションへのパスワードなどのシークレットを必要とする人員 ID とマシンの ID については、専門的なサービスの最新の業界標準を使用して自動ローテーションを使用して保存します。
- 一元化された ID プロバイダーを利用する: 人員 ID の場合、一元化された場所で ID を管理できる ID プロバイダーを利用します。これにより、1 つの場所からアクセスを作成、管理、取り消すことができるため、アクセスの管理が容易になります。これにより、複数の認証情報の要件が軽減され、HR プロセスと統合する機会がもたらされます。
- 定期的に認証情報を監査およびローテーションする: 一時的な認証情報を利用できず、長期的な認証情報を必要とする場合、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。
- ユーザーグループと属性を活用する: 一般的なセキュリティ要件を持つユーザーを ID プロバイダーによって定義されたグループに配置し、アクセスコントロールに使用される可能性のあるユーザー属性 (部署や場所など) が正確で、最新の状態に保たれるようにするメカニズムを導入します。アクセスを制御するには、個々のユーザーではなくこれらのグループと属性を使用します。これにより、ユーザーのアクセスニーズが変化したときに多くの個別のポリシーを更新することなく、ユーザーのグループメンバーシップや属性を 1 回変更することで、アクセスを一元管理できます。

SEC 3 人とマシンのアクセス許可はどのように管理すればよいでしょうか?

アクセス許可を管理して、AWS とワークロードへのアクセスを必要とするユーザー ID やマシン ID へのアクセスを制御します。アクセス許可は、どの条件で誰が何にアクセスできるかを制御します。

ベストプラクティス:

- アクセス要件を定義する: ワークロードの各コンポーネントまたはリソースには、管理者、エンドユーザー、またはその他のコンポーネントからアクセスする必要があります。各コンポーネントにアクセスできるユーザーや内容を明確に定義し、適切な ID タイプと認証および承認の方法を選択します。
- 最小権限のアクセスを付与する: 特定の条件下で特定の AWS リソースに対する特定のアクションを行えるようにして、ID に必要なアクセスのみを付与します。グループと ID 属性を利用して、個々のユーザーのアクセス許可を定義するのではなく、規模に応じてアクセス許可を動的に設定します。たとえば、開発者のグループに、扱うプロジェクトのリソースのみを管理することを許可できます。これにより、開発者がグループから削除されると、アクセスポリシーに変更を加えることなく、そのグループがどこでアクセスコントロールに使用されたかを問わず、開発者のアクセスが取り消されます。
- 緊急アクセスのプロセスを確立する: 自動プロセスまたはパイプラインの問題が発生した場合に、ワークロードへの緊急アクセスを許可するプロセス。これにより、最小権限のアクセスを利用しながら、ユーザーは必要なときに適切なレベルのアクセスを取得できます。たとえば、管理者がリクエストを確認して承認するプロセスを確立します。
- アクセス許可を継続的に削減する: チームとワークロードが必要とするアクセスを決定したら、不要になったアクセス許可を削除し、最小権限のアクセス許可を達成するためのレビュープロセスを確立します。未使用の ID とアクセス許可を継続的にモニタリングし、削減します。
- 組織のアクセス許可ガードレールを定義する: 組織内のすべての ID へのアクセスを制限する共通コントロールを確立します。たとえば、特定の AWS リージョンへのアクセスを制限したり、中央セキュリティチームが使用する IAM ロールなどの一般的なリソースをオペレータが削除できないようにしたりできます。
- ライフサイクルに基づいてアクセスを管理する: アクセスコントロールをオペレーター、アプリケーションのライフサイクル、一元化されたフェデレーションプロバイダーと統合します。たとえば、ユーザーが組織を離れるとき、またはロールを変更するときに、ユーザーのアクセス権を削除するとします。
- パブリックおよびクロスアカウントアクセスの分析: パブリックおよびクロスアカウントアクセスに焦点を当てた結果を継続的にモニタリングします。パブリックアクセスとクロスアカウントアクセスを減らして、このタイプのアクセスを必要とするリソースのみへのアクセスに限定します。
- リソースを安全に共有する: アカウント間または AWS 組織内の共有リソースの消費を管理します。共有リソースをモニタリングし、共有リソースへのアクセスを確認します。

検出

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:

- サービスとアプリケーションのログ記録を設定する: アプリケーションログ、リソースログ、AWS のサービスログを含め、ワークロード全体でログ記録を設定します。例えば、組織内のすべてのアカウントで AWS CloudTrail、Amazon CloudWatch Logs、Amazon GuardDuty、AWS Security Hub が有効になっていることを確認します。
- ログ、結果、メトリクスを一元的に分析する: 異常や不正なアクティビティの兆候を検出するため、すべてのログ、メトリクス、テレメトリーを一元的に収集し、自動的に分析する必要があります。ダッシュボードを使用すると、リアルタイムの状態に関する洞察に簡単にアクセスできます。例えば、Amazon GuardDuty と Security Hub のログがアラートと分析のためにあるロケーションに一元的に送信されるようにします。
- イベントへの応答を自動化する: 自動化を使用してイベントを調査および修正することで、人為的な労力やエラーが軽減され、調査機能をスケールできます。定期的なレビューは、自動化ツールを調整するのに役立ちます。継続して繰り返し行います。例えば、最初の調査ステップを自動化して Amazon GuardDuty イベントへの応答を自動化し、同様の作業を繰り返して、人間の労力を徐々に排除します。
- 実用的なセキュリティイベントを実装する: チームに送信され、チームによるアクションが可能なアラートを作成します。チームがアクションを実行するための関連情報がアラートに含まれていることを確認します。例えば、Amazon GuardDuty と AWS Security Hub のアラートが、アクションを実行するためにチームに送信されたり、オートメーションフレームワークからのメッセージによってチームが通知される状態を保ちつつ、レスポンスオートメーションツールに送信されたりするようにします。

インフラストラクチャ保護

SEC 5 ネットワークリソースをどのように保護しますか?

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

ベストプラクティス:

- ネットワークレイヤーを作成する: 到達可能性要件をレイヤーに共有するコンポーネントをグループ化します。たとえば、インターネットアクセスを必要としない VPC 内のデータベースクラスターは、インターネットへのルート、またはインターネットからのルートがないサブネットに配置する必要があります。VPC を使用せずに稼働するサーバーレスワークロードでは、マイクロサービスを使用した同様の階層化とセグメント化でも同じ目標を達成できます。
- すべてのレイヤーでトラフィックをコントロールする: インバウンドトラフィックとアウトバウンドトラフィックの両方について、多層防御アプローチでコントロールを適用します。たとえば、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、VPC ベースのコントロールを使用してプライベート VPC で実行することを検討してください。
- ネットワーク保護を自動化する: 保護メカニズムを自動化し、脅威インテリジェンスと異常検出に基づく自己防御型ネットワークを提供します。たとえば、現在の脅威にプロアクティブに適応し、その影響を軽減できる侵入検知および防止ツールなどです。
- 検査および保護を実装する: 各レイヤーでトラフィックを検査し、フィルタリングします。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、アプリケーションネットワークレイヤーでの意図しないアクセスから保護します。Lambda 関数の場合、サードパーティーのツールは、アプリケーションレイヤーのファイアウォールをランタイム環境に追加できます。

SEC 6 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:

- 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。
- 攻撃領域を削減する: オペレーティングシステムを強化し、使用するコンポーネント、ライブラリ、外部から利用可能なサービスを最小限に抑えることで、攻撃対象領域を縮小します。
- マネージドサービスを活用する: Amazon RDS、AWS Lambda、Amazon ECS などのリソースを管理するサービスを実装し、共有責任モデルの一部としてのセキュリティメンテナンスタスクを減らします。
- コンピューティング保護を自動化する: 脆弱性管理、攻撃対象領域削減、リソース管理などのコンピューティング保護メカニズムを自動化します。
- ユーザーが遠距離でアクションを実行できるようにする: インタラクティブアクセスの機能を排除すると、人為的ミスのリスクが軽減され、設定や管理が手動で行われる可能性が低くなります。たとえば、変更管理ワークフローを使用して、Infrastructure as Code によって EC2 インスタンスをデプロイした後、直接アクセスや踏み台ホストを許可する代わりに、ツールを使用して EC2 インスタンスを管理します。
- ソフトウェアの整合性を検証する: ワークロードで使用されるソフトウェア、コード、ライブラリが信頼できるソースからのものであり、改ざんされていないことを検証するメカニズム (コード署名など) を実装します。

データ保護

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:

- ワークロード内のデータを特定する: これには、データのタイプと分類、関連するビジネスプロセス、データ所有者、該当する法的要件およびコンプライアンス要件、データの保存場所、および結果として実行が必要な統制が含まれます。これには、データが一般公開されることを意図しているかどうか、データが顧客個人識別情報 (PII) などの内部使用のみかどうか、データが知的財産である、法的な秘匿特権がある、機密性が高いと特記されているなど、より制限されたアクセス用であるかどうかを示す分類が含まれます。
- データ保護コントロールを定義する: 分類レベルに従ってデータを保護します。たとえば、関連するレコメンデーションを使用してパブリックとして分類されたデータを保護すると同時に、追加のコントロールで機密データを保護します。
- 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して、手動操作による人為的ミスのリスクを軽減します。
- データのライフサイクル管理を定義する: 定義されるライフサイクル戦略は、機密性レベル、また法的および組織の要件に基づいている必要があります。データを保持する期間、データ破壊、データアクセス管理、データ変換、データ共有などの側面を考慮する必要があります。

SEC 8 保管時のデータをどのように保護していますか?

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

ベストプラクティス:

- 安全なキー管理を実装する: 暗号化キーは、AWS KMS などのキー管理サービスを使用するなど、厳格なアクセスコントロールで安全に保存する必要があります。異なるキーを使用し、AWS IAM およびリソースポリシーと組み合わせて、データ分類レベルと分離要件に合わせて、キーに対するアクセスコントロールを検討してください。
- 保管中に暗号化を適用する: 最新の標準やレコメンデーションに基づき、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。
- 保管時のデータの保護を自動化する: 自動化ツールを使用して保管中のデータの保護を継続的に検証し、提供します。たとえば、すべてのストレージリソースが暗号化されていることを確認します。
- アクセスコントロールを適用する: 最低限の権限によるアクセスコントロールや、バックアップ、分離、バージョニングなどのメカニズムを適用することは、保管中のデータの保護に役立ちます。オペレーターがデータへのパブリックアクセスを許可しないようにします。
- 人をデータから遠ざけるメカニズムを使用する: 通常の運用状況で、すべてのユーザーが機密データおよびシステムに直接アクセスできないようにします。たとえば、クエリを実行するデータストアに直接アクセスする代わりにダッシュボードを提供します。CI/CD パイプラインを使用しない場合は、通常無効になっている特権アクセスメカニズムを適切に提供するために必要な制御とプロセスを決定します。

SEC 9 転送時のデータをどのように保護していますか?

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

ベストプラクティス:

- 安全な鍵および証明書管理を実装する: AWS Certificate Manager (ACM) などの証明書管理サービスを使用するなど、厳格なアクセスコントロールを適用して、暗号化キーと証明書を安全に保管し、適切な時間間隔でローテーションします。
- 伝送中に暗号化を適用する: 組織的、法的、コンプライアンスの要件を満たすために、適切な基準とレコメンデーションに基づいて定義された暗号化要件を適用します。
- 意図しないデータアクセスの検出を自動化する: GuardDuty などのツールを使用して、データ分類レベルに基づいて定義された境界の外部にデータを移動する攻撃 (DNS プロトコルを使用して不明または信頼されないネットワークにデータをコピーするトロイの馬など) を自動検知します。
- 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。

インシデント対応

SEC 10 インシデントの予測、対応、復旧はどのように行いますか?

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

ベストプラクティス:

- 重要な人員と外部リソースを特定する: 組織がインシデントに対応するのに役立てるため、社内外の担当者、リソース、法的義務を特定します。
- インシデント管理計画を作成する: インシデントへの応答、インシデント時の伝達、インシデントからの復旧に役立つ計画を作成します。たとえば、ワークロードと組織にとって起こる可能性が最も高いシナリオで、インシデント応答計画を作成してみましょう。内部および外部に伝達およびエスカレーションする方法を含めます。
- フォレンジック機能を備える: 外部のスペシャリスト、ツール、オートメーションなど、適切なフォレンジック調査能力を特定し、準備します。
- 封じ込め機能を自動化する: インシデントの封じ込めおよび復旧を自動化し、対応時間を短縮するとともに組織的影響を軽減します。
- アクセスを事前プロビジョニングする: インシデント応答者が 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確認しておき、調査から復旧までの時間を短縮します。
- ツールを事前デプロイする: 復旧までの調査時間を短縮できるように、セキュリティ担当者は適切なツールを AWS に事前にデプロイしておきます。
- 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インシデント管理計画に組み込み、継続的に改善します。

信頼性

基盤

REL 1 サービスクォータと制約はどのように管理しますか?

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

ベストプラクティス:

- サービスクォータと制約を認識する: あなたは、ワークロードアーキテクチャに対するデフォルトのクォータとクォータ引き上げリクエストを認識しています。さらに、ディスクやネットワークなど、どのリソースの制約が潜在的に大きな影響を与えるかを知っています。
- アカウントおよびリージョンをまたいでサービスクォータを管理する: 複数の AWS アカウントまたは AWS リージョンをご利用の場合は、必ず本番ワークロードを実行するすべての環境で必要十分なクォータをリクエストしてください。
- アーキテクチャを通じて、固定サービスクォータと制約に対応する: サービスクォータと物理リソースには変更できないものもあることに注意し、これらが信頼性に影響を及ぼさないように設計します。
- クォータをモニタリングおよび管理する: 予想される使用量を評価し、クォータを必要に応じて引き上げて、使用量を予定通り増やせるようにします。
- 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。
- フェイルオーバーに対応するために、現在のクォータと最大使用量の間に十分なギャップがあることを確認する: リソース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、クォータ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クォータ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。

REL 2 ネットワークトポロジをどのように計画しますか?

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

ベストプラクティス:

- ワークロードのパブリックエンドポイントに可用性が高いネットワーク接続を使用する: これらのエンドポイントとそれらへのルーティングは、可用性が高い必要があります。これを実現するには、可用性の高い DNS、コンテンツ配信ネットワーク (CDN)、API ゲートウェイ、負荷分散やリバースプロキシを使用します。
- クラウド環境とオンプレミス環境のプライベートネットワーク間の冗長接続をプロビジョニングする: 個別にデプロイされたプライベートネットワーク間で複数の AWS Direct Connect (DX) 接続または VPN トンネルを使用します。高可用性のために複数の DX ロケーションを使用します。複数の AWS リージョンを使用している場合は、少なくとも 2 つのリージョンで冗長性を確保してください。VPN を終了する AWS Marketplace アプライアンスを評価したい場合があります。AWS Marketplace アプライアンスを利用する場合は、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。
- 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: 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、ワークロード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、EC2 インスタンス、コンテナーベースのアプリケーションが含まれます。
- 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: 2 つを超えるネットワークアドレス空間 (VPC およびオンプレミスネットワークなど) が VPC ピア接続、AWS Direct Connect、または VPN 経由で接続されている場合は、AWS Transit Gateway が提供するようなハブアンドスポークモデルを使用します。
- 接続されているすべてのプライベートアドレス空間において、重複しないプライベート IP アドレス範囲を指定します。: 各 VPC の IP アドレス範囲が、ピア接続時または VPN 経由での接続時に重複しないようにする必要があります。同様に、VPC とオンプレミス環境の間、または使用する他のクラウドプロバイダーとの IP アドレスの競合を回避する必要があります。また、必要に応じてプライベート IP アドレス範囲を割り当てる方法を用意する必要もあります。

ワークロードアーキテクチャ

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

ベストプラクティス:

- ワークロードをセグメント化する方法を選択する: モノリシックアーキテクチャは避ける必要があります。モノリシックアーキテクチャではなく、SOA とマイクロサービスのどちらかを選択する必要があります。どちらかの選択を行うときは、複雑さに比してどれだけメリットがあるかを考えてください。新製品のローンチ時に適しているものは、初期にスケーリングのことを考えて構築したワークロードが要するものとは異なります。小さなセグメントを使用する利点には、俊敏性、組織の柔軟性、スケーラビリティが高い点などがあります。複雑さにより、潜在的にレイテンシーが増加したり、デバッグが複雑化したり、運用上の負担が増加したりします

- 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: SOA は、ビジネスニーズに合わせて明確に定義された機能を備えたサービスを構築します。マイクロサービスはドメインモデルと境界付けられたコンテキストを使用してこれをさらに制限し、各サービスが 1 つのことだけを実行するようにします。特定の機能に焦点を当てることで、さまざまなサービスの信頼性要件を差別化し、より具体的に的を絞って投資することができます。簡潔なビジネス上の問題と各サービスに関連付けられた小さなチームにより、組織のスケーリングも容易になります。

- API ごとにサービス契約を提供する: サービス契約は、サービス統合に関するチーム間で文書化した合意で、機械で読み取ることができる API 定義、レート制限、パフォーマンスの期待値が含まれます。バージョニング戦略により、クライアントは既存の API を引き続き使用し、準備ができたらアプリケーションを新しい API に移行できます。契約に違反しない限り、デプロイはいつでも行うことができます。サービスプロバイダーチームは、選択したテクノロジースタックを使用して、API 契約の条件を満たすことができます。同様に、サービスコンシューマーは独自のテクノロジーを使用できます。

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

ベストプラクティス:

- 必要な分散システムの種類を特定する: ハードなリアルタイム分散システムでは、応答を同期的かつ迅速に行えるようにする必要がありますが、ソフトなリアルタイムシステムでは、応答に数分以上の余裕をもった時間枠があります。オフラインシステムは、バッチ処理または非同期処理を通じて応答を処理します。ハードなリアルタイム分散システムは、最も厳格な信頼性要件を持っています。

- 疎結合の依存関係を実装する: キューイングシステム、ストリーミングシステム、ワークフロー、ロードバランサーなどの依存関係は、緩やかに結合しています。疎結合は、コンポーネントの動作をそれに依存する他のコンポーネントから分離するのに役立ち、弾力性と俊敏性を高めます。

- すべての応答にべき等性を持たせる: べき等のサービスは、各リクエストが 1 回だけで完了することを約束します。そのため、同一のリクエストを複数回行っても、リクエストを 1 回行ったのと同じ効果しかありません。べき等サービスを使用すると、リクエストが誤って複数回処理されることを恐れる必要がなくなるため、クライアントが再試行を行いやすくなります。このために、クライアントはべき等性トークンを使用して API リクエストを発行できます。リクエストが繰り返される場合は常に同じトークンが使われます。べき等サービス API はトークンを使用して、リクエストが最初に完了したときに返された応答と同じ応答を返します。

- 動作を継続的に行う: 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。たとえば、何千台ものサーバーの状態をモニタリングするヘルスチェックシステムは、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、大きく急激な変更なしに常に作業を行っています。

REL 5 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?

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

ベストプラクティス:

- 該当するハードな依存関係をソフトな依存関係に変換するため、グレイスフルデグラデーションを実装する: コンポーネントの依存関係が異常な場合でも、コンポーネント自体は機能しますが、パフォーマンスが低下します。たとえば、依存関係の呼び出しが失敗した場合、事前に定義された静的レスポンスにフェイルオーバーします。

- スロットルリクエスト: これは、予想外の需要の増加に対応するための軽減パターンです。一部のリクエストは受け入れられますが、定義された制限を超えるリクエストは拒否され、スロットルされたことを示すメッセージが返されます。クライアントの期待は、リクエストが戻されて放棄されるか、遅い速度で再試行することです。

- 再試行呼び出しを制御および制限する: エクスポネンシャルバックオフを使用して、徐々に長い間隔で再試行します。これらの再試行間隔をランダム化するジッターを導入し、再試行の最大数を制限します。

- すぐに失敗し、キューを制限する: ワークロードがリクエストに正常に応答できない場合は、すぐに失敗するようにします。これにより、リクエストに関連付けられたリソースを解放でき、リソースが不足した場合にサービスを復旧できます。ワークロードは正常に応答できるが、リクエスト頻度が高すぎる場合は、代わりにキューを使用してリクエストをバッファします。ただし、長いキューは許可しないでください。クライアントがすでに処理を停止している古いリクエストを処理する原因となる可能性があるためです。

- クライアントタイムアウトを設定する: タイムアウトを適切に設定し、体系的に検証します。デフォルト値の設定は通常高すぎるため、デフォルト値のままにしないでください

- 可能な場合はサービスをステートレスにする: サービスは、状態を必要としないようにするか、あるいは異なるクライアントリクエスト間で、ディスクまたはメモリにローカルに保存されたデータに依存しないように、状態をオフロードするかのいずれかにする必要があります。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。Amazon ElastiCache や Amazon DynamoDB は、オフロードした状態に適した宛先です。

- 緊急レバーを実装する: これは、ワークロードの可用性に対する影響を軽減できる可能性がある迅速なプロセスです。根本原因がなくても操作できます。緊急レバーは、完全に決定的なアクティブ化と非アクティブ化の基準を指定することにより、リゾルバーの認知負荷をゼロに減らせるものが理想的です。緊急レバーの例には、すべてのロボットトラフィックをブロックしたり、静的応答を行ったりすることが含まれます。緊急レバーは多くの場合手動ですが、自動化することもできます。

変更管理

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

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

ベストプラクティス:

- 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。Personal Health Dashboard を使って AWS のサービスをモニタリングする
- メトリクスを定義および計算する (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
- 通知を送信する (リアルタイム処理とアラーム): 重要なイベントが発生すると、把握しておく必要のある組織が通知を受信します
- 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 (たとえば、障害が発生したコンポーネントを交換します)
- ストレージと分析: 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、より幅広いトレンドとワークロードの洞察を得ます
- 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する
- システムを通じたリクエストのエンドツーエンドのトレースをモニタリングする: AWS X-Ray またはサードパーティー製のツールを使用することで、開発者は分散システムの分析とデバッグをより簡単に行い、アプリケーションとその基盤となるサービスのパフォーマンスを把握できます

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

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

ベストプラクティス:

- リソースの取得またはスケーリング時に自動化を使用する: 障害のあるリソースを交換したり、ワークロードをスケーリングしたりする場合は、Amazon S3 や AWS Auto Scaling などのマネージド型の AWS のサービスを使用してプロセスを自動化します。サードパーティーのツールや 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。
- 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。
- 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。

REL 8 変更はどのように実装するのですか?

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

ベストプラクティス:

- 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、具体的な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。標準的なアクティビティには、ワークロードのデプロイ、パッチの適用、DNS の変更などがあります。
- 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。
- デプロイの一部として回復力テストを統合する: (カオスエンジニアリングの一環としての) 回復力テストは、本番稼働前環境で自動デプロイパイプラインの一部として実行します。
- 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: これは、本番ワークロードで更新、セキュリティパッチ、または設定の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。
- 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は自動化されて、悪影響を排除します。

障害の管理

REL 9 データはどのようにバックアップするのですか?

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

ベストプラクティス:

- バックアップする必要があるすべてのデータを特定してバックアップするか、ソースからデータを再現する: Amazon S3 は、複数の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 のサービスには、バックアップを作成する機能が組み込まれています。サードパーティー製のバックアップソフトウェアも使用できます。また、データを他の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もあります
- バックアップを保護し、暗号化する: AWS IAM などの認証と許可を使用し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。
- データバックアップを自動的に実行する: 定期的なスケジュールに基づいて、またはデータセット内の変更に基づいて自動的にバックアップが作成されるように設定します。RDS インスタンス、EBS ボリューム、DynamoDB テーブル、および S3 オブジェクトはすべて、自動的にバックアップされるように設定できます。AWS Marketplace ソリューションまたはサードパーティーのソリューションも使用できます。
- 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。

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しますか?

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

ベストプラクティス:

- 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 リージョンにかけて分散します。これらのロケーションは、必要に応じて多様化できます。
- 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が 1 つ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実行できる場合は、定義された復旧目標内でワークロードを全面的に再構築する機能を実装する必要があります。
- バルクヘッドアーキテクチャを使用する: 船の隔壁のように、このパターンは、障害がリクエスト/ユーザーの小さなサブセットに確実にとどまるようにすることで、障害のあるリクエストの数が制限され、ほとんどがエラーなしで続行できるようにします。データの隔壁は通常パーティションまたはシャードと呼ばれ、サービスの隔壁はセルと呼ばれます。

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、高い弾力性があるように設計する必要があります。

ベストプラクティス:

- 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エラーを検知する: ワークロードの状態を継続的にモニタリングすることで、お客様と自動化システムがパフォーマンスの低下または完全な障害の発生を速やかに把握できるようにします。ビジネス価値に基づいて主要業績評価指標 (KPI) をモニタリングします。
- 障害のないロケーションにある正常なリソースにフェイルオーバーする: あるロケーションに障害が発生した場合でも、正常なロケーションのデータとリソースが引き続きリクエストに対応できるようにします。Elastic Load Balancing や AWS Auto Scaling などの AWS のサービスは、アベイラビリティーゾーン間で負荷を分散できるため、マルチゾーンのワークロードの方がこの点簡単です。マルチリージョンのワークロードの場合、状況はさらに複雑です。たとえば、クロスリージョンリードレプリカを使用すると、データを複数の AWS リージョンにデプロイできますが、プライマリロケーションに障害が発生した場合は、リードレプリカをマスターに昇格させ、そこへトラフィックを向ける必要があります。Amazon Route 53 と AWS Global Accelerator は、AWS リージョン間のトラフィックのルーティングにも役立ちます。
- 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。
- 静的安定性を使用してバイモーダル動作を防止する: バイモーダル動作とは、たとえばアベイラビリティーゾーンに障害が発生した場合に新しいインスタンスの起動に依存するなど、通常モードと障害モードでワークロードが異なる動作を示す場合をいいます。バイモーダル動作を防止するために、静的に安定し、1 つのモードでのみ動作するワークロードを構築する必要があります。この場合、1 つの AZ が削除された場合にワークロードの負荷を処理するのに十分な数のインスタンスを各アベイラビリティーゾーンにプロビジョニングしてから、Elastic Load Balancing または Amazon Route 53 ヘルスチェックを使用して、障害のあるインスタンスから負荷を分散します。
- イベントが可用性に影響する場合に通知を送信する: 重大なイベントが検出されると、イベントによって引き起こされた問題が自動的に解決された場合でも、通知が送信されます。

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、期待する弾力性を実現することを確認する唯一の方法が、テストを行うことです。

ベストプラクティス:

- プレイブックを使用して失敗を調査する: 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。
- インシデント後の分析を実行する: 顧客に影響を与えるイベントを確認し、寄与する要因と予防措置を特定します。この情報を使用して、再発を制限または回避するための緩和策を開発します。迅速で効果的な対応のための手順を開発します。対象者に合わせて調整された、寄与する要因と是正措置を必要に応じて伝えます。必要に応じて根本原因を他の人に伝える方法を確立します。
- 機能要件をテストする: 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。
- スケーリングおよびパフォーマンス要件をテストする: これには、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が含まれます。
- 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: 本番稼働前および運用環境に定期的に障害を挿入してテストを実行します。ワークロードが障害にどのように反応するかの仮説を立て、その仮説をテスト結果と比較して、一致しない場合はプロセスを繰り返します。本番稼働テストがユーザーに影響を与えないようにします。
- 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と一緒に、可能な限り本番環境に近い環境 (本番環境を含む) で障害対処手順を実行します。ゲームデーでは、本番環境のテス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。

REL 13 災害対策 (DR) はどのように計画するのですか?

バックアップと冗長ワークロードコンポーネントを配置することは、DR 戦略の出発点です。RTO と RPO は、可用性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ロケーションと機能を考慮して、目標を達成するための戦略を実装します。

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。
- DR サイトまたはリージョンでの設定ドリフトを管理する: インフラストラクチャ、データ、設定が DR サイトまたはリージョンで必要とされるとおりであることを確認します。たとえば、AMI とサービスクォータが最新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製のツールを使用して、システムの復旧を自動化し、トラフィックを DR サイトまたはリージョンにルーティングします。

パフォーマンス効率

選択

PERF 1 どのように最良パフォーマンスのアーキテクチャを選択するのですか?

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

ベストプラクティス:

- 利用可能なサービスやリソースを理解する: クラウドで利用できる幅広いサービスやリソースに関する情報を取得し、その内容を理解します。ワークロードに関連するサービスと設定のオプションを特定し、最適なパフォーマンスを実現する方法を理解してください。
- アーキテクチャにかかわる選択プロセスを決める: クラウドにおける社内の経験と知識、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。ワークロードで使用できると考えられるサービスの実験とベンチマークを促進するプロセスを定義するようにしてください。
- 意思決定においてコスト要件を考慮する: ワークロードには、多くの場合、運用のコスト要件があります。社内のコスト管理を使用し、予測されたリソースニーズに基づいて、リソースのタイプとサイズを選択してください。
- ポリシーやリファレンスアーキテクチャを使用する: 内部ポリシーと既存のリファレンスアーキテクチャを評価し、独自の分析を使用してワークロードのサービスと設定を選択することによって、パフォーマンスと効率性を最大化します。
- クラウドプロバイダー、または適切なパートナーからのガイダンスを利用する: 判断の指針とするために、ソリューションアーキテクト、プロフェッショナルサービス、または適切なパートナーなどのクラウド企業のリソースを利用します。これらのリソースは、アーキテクチャを確認して改善し、最適なパフォーマンスを実現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、クラウドでの実行状況を把握します。ベンチマークから収集されたデータを使用して、アーキテクチャ面での判断を導き出します。
- ワークロードの負荷テストを実施する: 異なるリソースタイプとサイズを使用して、最新のワークロードアーキテクチャをクラウドにデプロイします。デプロイメントをモニタリングして、ボトルネック、または過剰なキャパシティーを特定するパフォーマンスメトリクスを取得してください。このパフォーマンス情報を使用して、アーキテクチャとリソースを設計、またはそれらをより良く選択します。

PERF 2 コンピューティングソリューションはどのように選択するのですか?

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

ベストプラクティス:

- 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理解します。インスタンス、コンテナ、および関数の仕組みと、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認識します。
- 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には、インスタンスのファミリー、サイズ、機能 (GPU、I/O)、関数サイズ、コンテナインスタンス、シングルテナンシーとマルチテナンシーなどがあります。
- コンピューティング関連のメトリクスを収集する: コンピューティング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。このデータは、リソース要件についてより正確な判断を行うために使用できます。
- 適切なサイジングによって必要な設定を決定する: ワークロードのさまざまなパフォーマンス特性と、それらの特性とメモリ、ネットワーク、CPU 使用率との関連を分析します。このデータは、ワークロードのプロファイルに最適なリソースを選択するために使用します。たとえば、メモリ集約型のワークロード (データベースなど) には r ファミリーのインスタンスが最適ですが、バーストするワークロードでは、伸縮自在なコンテナシステムからより多くのメリットを得ることができます。
- 利用可能な伸縮性のあるリソースを使用する: クラウドは、需要の変化に対応するためのさまざまなメカニズムを通じて、リソースを動的に拡張または縮小する柔軟性を提供します。コンピューティング関連のメトリクスと組み合わせることによって、ワークロードは変化に自動的に対応し、最適な一連のリソースを利用して目標を達成できるようになります。
- メトリクスに基づいてコンピューティングニーズを再評価する: システムレベルのメトリクスを使用して、ワークロードの経時的な動作と要件を特定します。利用可能なリソースとこれらの要件を比較することによってワークロードのニーズを評価し、ワークロードのプロファイルに最も良く一致するようにコンピューティング環境を変更します。たとえば、時間がたつにつれて、システムが当初の想定よりもメモリ集約型であることがわかる場合があります。このため、別のインスタンスファミリー、またはインスタンスサイズに移行することでパフォーマンスと効率性の両方が向上する可能性があります。

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: オブジェクトストレージ、ブロックストレージ、ファイルストレージ、またはインスタンスストレージなど、ワークロードに最適なサービスを選択するために必要なさまざまな特性 (共有可能、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、およびデータの永続性など) を理解します。
- 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。プロビジョンド IOPS、SSD、磁気ストレージ、オブジェクトストレージ、アーカイブストレージ、またはエフェメラルストレージをどこでどのように使用するかを理解し、ワークロードのためのストレージ容量とパフォーマンスを最適化してください。
- アクセスパターンとメトリクスに基づいて意思決定を行う: ワークロードのアクセスパターンに基づいてストレージシステムを選択し、ワークロードがどのようにデータにアクセスするかを判断することによってそれらを設定します。ストレージの効率性を向上させるには、ブロックストレージではなくオブジェクトストレージを選択してください。選択したストレージオプションは、データのアクセスパターンに合わせて設定します。

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。ワークロードについて、トランザクションが必要かどうか、データとどのようにやり取りするか、およびパフォーマンス需要が何かを判断します。このデータを使用して、ワークロードに最高パフォーマンスを提供するデータベースアプローチを選択します (リレーショナルデータベース、NoSQL key-value 、ドキュメント、ワイドカラム、グラフ、時系列、またはインメモリストレージなど)。
- 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。データストレージのために、所定のサービスまたはシステムをいつどのように使用するのかを理解します。プロビジョンド IOPS、メモリとコンピューティングのリソース、およびキャッシングなど、データベースのパフォーマンスと効率性を最適化できる利用可能な設定オプションについて学びます。
- 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理解します。
- 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。たとえば、トランザクションを必要とするワークロード、または高スループットを提供するが、必要に応じて結果整合になるという key-value ストアには、リレーショナルデータベースを使用します。
- 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、キー配布、データウェアハウス設計、またはキャッシング戦略などの最適化が、システムのパフォーマンスまたは全体的な効率性にどのように影響するかを測定します。

PERF 5 どのようにネットワーキングソリューションを選択するのですか?

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

ベストプラクティス:

- 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。
- 使用可能なネットワーク機能を評価する: パフォーマンスの向上に役立つ可能性のあるクラウドのネットワーク機能を評価します。これらの機能の影響を、テスト、メトリクス、および分析を使って測定してください。たとえば、レイテンシー、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。
- ハイブリッドワークロード用に適切なサイズの専用接続または VPN を選択する: オンプレミス通信の要件がある場合は、ワークロードパフォーマンスのために十分な帯域幅があることを確認します。帯域幅の要件によっては、単一の専用接続、または単一の VPN では十分でない場合があり、複数の接続間でトラフィックのロードバランシングを有効化する必要があります。
- ロードバランシングと暗号化のオフロードを活用する: トラフィックを複数のリソースやサービスに分散させ、ワークロードがクラウドの伸縮性を活用できるようにします。また、パフォーマンスを向上させ、トラフィックを効率的に管理およびルーティングするために、ロードバランシングを使用して暗号化終了をオフロードすることもできます。
- 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。
- ネットワーク要件に基づいてワークロードのロケーションを選択する: 利用可能なクラウドロケーションオプションを使用して、ネットワークレイテンシーを低減、またはスループットを向上させます。ネットワークレイテンシーの低減、またはスループットの向上には、AWS リージョン、アベイラビリティーゾーン、プレイスメントグループ、および Outposts、Local Regions、Waveledge などのエッジロケーションを活用します。
- 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。これらの変更の影響を測定して、その影響測定値を将来の意思決定に使用します。

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、時間と共に、ワークロードのパフォーマンスを向上させることができる新しいテクノロジーとアプローチが利用できるようになります。

ベストプラクティス:

- リソースとサービスに関する最新情報を常に入手する: 新しいサービス、設計パターン、製品が利用可能になったら、パフォーマンスの向上方法を検討します。アドホック評価、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。
- ワークロードのパフォーマンス向上プロセスを定める: 新しいサービス、設計パターン、リソースの種類、設定が利用できるようになった時点で、これらを評価するプロセスを明確に定めます。たとえば、新しいインスタンス製品で既存のパフォーマンステストを実行して、ワークロードを向上させる可能性を判断します。
- ワークロードのパフォーマンス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、新しいサービスまたはリソースが利用可能になるときに、それらの導入を積極的に推進します。

モニタリング

PERF 7 リソースが稼働していることを確実にするためのリソースのモニタリングはどのように行いますか?

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

ベストプラクティス:

- パフォーマンスに関連するメトリクスを記録する: モニタリングとオブザーバビリティサービスを使用して、パフォーマンス関連のメトリクスを記録します。たとえば、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、またはその他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。これらのビューは、ワークロードのどの部分が期待通りに機能していないかに関するインサイトを提供します。
- ワークロードのパフォーマンスを測定するための主要業績評価指標 (KPI) を確立する: ワークロード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。たとえば、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、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主要業績評価指標 (KPI) の測定値が予想された境界線の外側にある場合に自動的にアラームを生成します。
- メトリクスを定期的に見直す: 定期的なメンテナンスとして、またはイベントやインシデントに応じて、収集対象のメトリクスを見直します。これらのレビューを使用して、どのメトリクスが問題対応の鍵となったか、およびどのメトリクスを追加すると、それらが追跡される場合に問題の特定、対応、または防止に役立つと思われるかを特定します。
- モニタリングしてプロアクティブに警告する: 主要業績評価指標 (KPI) をモニタリングおよびアラート発行システムと組み合わせて使用し、パフォーマンス関連の問題に積極的に対処します。アラームを使用して、可能な場合に問題を修正する自動化されたアクションをトリガーします。自動化された応答が不可能な場合は、応答できるシステムにアラームをエスカレートします。たとえば、期待される主要業績評価指標 (KPI) 値を予測し、それらが特定のしきい値を超えた場合にアラームを発行できるシステム、または KPI が期待される値の範囲外である場合に、デプロイメントを自動的に停止、またはロールバックできるツールなどが考えられます。

トレードオフ

PERF 8 トレードオフをどのように使用するとパフォーマンスが向上するのですか?

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

ベストプラクティス:

- 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。たとえば、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顧客の近辺に移動させることでメリットを得ることができます。
- 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、より優れたパフォーマンスを達成するために何を引き替えにすることができるかを特定します。たとえば、キャッシュサービスの使用はデータベースシステムにかけられた負荷を低減できますが、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が必要となる、また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、どの選択肢が顧客とワークロードの効率性に影響するかを特定します。たとえば、key-value データストアの使用がシステムパフォーマンスを向上させる場合、その結果整合性の特質がお客様に与える影響を評価することが大切です。
- 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行うと同時に、収集されたメトリクスとデータを評価します。この情報を使用して、そのパフォーマンス向上がワークロード、ワークロードのコンポーネント、および顧客に与えた影響を判断します。この測定は、トレードオフに由来するパフォーマンス向上を理解する、および副次的な悪影響が生じたかどうかを判断するために役立ちます。
- 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、複数の戦略を活用してパフォーマンスを高めてください。たとえば、過剰なネットワークコールやデータベースコールを防止するためにデータキャッシングなどの戦略を使用する、データベースエンジンの読み込みレートを向上させるためにリードレプリカを使用する、データボリュームを削減するためにデータのシャーディングまたは圧縮を可能な限り使用する、ブロッキングを回避するために利用可能になった結果をバッファし、ストリーミングするなどの戦略を使用します。

コスト最適化

クラウド財務管理を実践する

COST 1 クラウド財務管理はどのように実装しますか?

組織はクラウド財務管理の実装により、AWS によってコストと使用状況の最適化とスケーリングをすることで、ビジネス価値と財務上の成功を実現できます。

ベストプラクティス:

- コスト最適化担当を設定する: 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を作成します。このチームには、組織全体の財務、テクノロジー、およびビジネスの役割を持つ人材が必要です。
- 財務とテクノロジーの連携を確立する: クラウドジャーニーのすべての段階で、コストと使用状況に関するディスカッションに財務チームとテクノロジーチームを参加させます。チームは、定期的に集まり、組織の最終目的や目標、コストと使用状況の現状、財務や会計のプラクティスなどのトピックについて話し合います。
- クラウドの予算および予測を確立する: 既存の組織の予算作成および予測プロセスを調整し、非常に変動しやすいクラウドのコストと使用状況の性質に対応できるようにします。プロセスは、トレンドベースまたはビジネスドライバーベースのアルゴリズム、またはそれらの組み合わせを使用して、動的なものとする必要があります。
- 組織のプロセスにコスト意識を採り入れる: 使用量に影響する新規または既存のプロセスにコスト意識を採り入れ、既存のプロセスを活用してコスト意識を高めます。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。
- コスト最適化に関して報告および通知する: 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。定例会議を開催して、このワークロードのコスト効率を分析し、社内にコスト意識を浸透させます。
- 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。通知を受け取ったときにコストとカテゴリを見るだけにとどまらないでください。これにより、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。
- 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。

経費支出と使用量の認識

COST 2 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。
- 目標およびターゲットを策定する: ワークロードのコストおよび使用量の両方について、目標を策定します。目標はコストと使用状況について組織に方向性を提供し、ターゲットはワークロードについての測定可能な結果を提供します。
- 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: ポリシーに沿ったグループおよびロールを実装し、各グループのインスタンスおよびリソースを作成、変更、削除できるユーザーを管理します。たとえば、開発、テスト、本番グループを実装します。これは、AWS のサービスやサードパーティーのソリューションに適用されます。
- 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 3 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:

- 詳細情報ソースを設定する: AWS のコストと使用状況レポートおよび Cost Explorer の時間単位の詳細を設定し、コストと使用状況の詳細情報を提供します。ワークロードが、もたらされるすべてのビジネス成果のログエントリを持つように設定します。
- 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、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。
- 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。
- コストと使用状況に組織情報を追加する: 組織、ワークロード属性、コスト配分カテゴリに基づいてタグ付けスキーマを定義します。すべてのリソースにタグを付けます。Cost Categories を使用して、組織の属性に従ってコストと使用状況をグループ化します。
- ワークロードメトリクスに基づいてコストを配分する: メトリクスや業績に基づいてワークロードのコストを配分し、ワークロードのコスト効率を測定します。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 4 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。
- リソースを削除する: 定期監査や使用状況の変化などのイベントを契機として、リソースを削除します。通常、削除は定期的に行われ、手動または自動で行われます。
- 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 5 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。
- 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ソフトウェアライセンスコストを排除し、ワークロードに対して、コスト面で大きなメリットをもたらすことができます。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。
- 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、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。
- 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をそのライフタイム全体にわたって維持できます。

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:

- 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。さまざまな予測負荷におけるワークロードのベンチマークアクティビティを実行し、コストを比較します。モデリングを行う際には、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。
- データに基づいてリソースのタイプやサイズを選択する: ワークロードに関するデータ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選択します。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。この選択は通常、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自動的にリソースタイプとサイズを選択する: 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ニングします。これは、自動スケーリングなどのフィードバックループまたはワークロードのカスタムコードで行うことができます。

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: ワークロードの各コンポーネントを分析します。コンポーネントとリソースが長期間実行されるか (コミットメント割引)、動的および短期実行 (スポットまたはオンデマンド) とするかを決定します。AWS Cost Explorer のレコメンデーション機能を使用して、ワークロードに関する分析を実行します。
- 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。
- 費用対効果の高い条件を提供するサードパーティーの契約を選択する: コスト効率に優れた契約と条件により、これらのサービスのコストが、提供されるメリットに見合ったものとなります。組織に追加のメリットを提供するときに、それに合わせてスケーリングする契約と料金を選択します。
- この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ー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 (リソースタイプに応じて、期間の 25% から 75%)。
- マスターアカウントレベルで料金モデル分析を実行する: Cost Explorer の Savings Plans およびリザーブドインスタンスのリコメンデーションを使用して、コミットメント割引のマスターアカウントレベルにおける定期的な分析を実行します。

COST 8 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 設定などの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。

需要を管理し、リソースを供給する

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。
- 需要を管理するためのバッファまたはスロットルを実装する: バッファリングとスロットリングは、ワークロードの需要を修正し、ピークを滑らかにします。クライアントが再試行を実行するときにスロットリングを実行します。バッファリングは、リクエストを保存し、後日まで処理を延期するために実装します。スロットルとバッファが、クライアントが要求された時間内にレスポンスを受け取るように設計されていることを確認します。
- リソースを動的に供給する: リソースを計画的なやり方でプロビジョニングします。これは、自動スケーリングなどの需要ベース、または需要が予測可能でリソースが時間に基づいて提供される時間ベースで行います。これらの手法を使用すると、過剰プロビジョニングやプロビジョニング不足を最小限に抑えることができます。

経時的な最適化

COST 10 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:

- 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。
- 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。