AWS による優れた設計の
フレームワーク

2016 年 *11* 月

注意

本書は情報提供のみを目的としています。本書の発行時点における AWS の現行製品と慣行を表したものであり、それらは予告なく変更されることがあります。お客様は本書の情報、および AWS 製品またはサービスの利用について、独自の評価に基づき判断する責任を負います。いずれの AWS 製品またはサービスも、明示または黙示を問わずいかなる保証も伴うことなく、「現状のまま」提供されます。本書のいかなる内容も、AWS、その関係者、サプライヤ、またはライセンサーからの保証、表明、契約的責任、条件や確約を意味するものではありません。お客様に対する AWS の責任は AWS 契約によって規定されています。また、本文書は、AWS とお客様との間の契約に属するものではなく、また、当該契約が本文書によって修正されることもありません。.

目次

要約

本書では **AWS による優れた設計のフレームワーク**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、AWS による優れた設計の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。

はじめに

アマゾン ウェブ サービス (AWS) では、クラウドで信頼性、セキュリティ、効率、コスト効果が高いシステムを設計し運用するために、アーキテクチャのベストプラクティスについてお客様に学習していただくことが重要だと考えます。この一環として、AWS でシステムを構築する際の選択肢の長所と短所の理解に役立つように、AWS による優れた設計のフレームワークを開発しました。システムを上手に構築することで、ビジネスが成功する可能性が飛躍的に増加すると確信しています。

AWS ソリューションアーキテクトは、多様なビジネス業界およびユースケースにおけるソリューションの設計において長年の経験を持ち、何千という AWS のお客様のアーキテクチャの設計および検証のお手伝いをしてきました。この経験から、クラウドにおけるシステム設計のベストプラクティスおよび核となる戦略を見つけ出しました。

AWS による優れた設計のフレームワークには、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。

本書は技術責任者 (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。本書を読むことで、クラウドアーキテクチャを設計し運用する際の AWS のベストプラクティスおよび取るべき戦略が理解できます。本書では、実装の詳細またはアーキテクチャパターンは扱いません。しかし、この情報に関する適切なリソースへの参照が含まれています。

AWS による優れた設計のフレームワークの定義

AWS の専門家は、日々、お客様がクラウドの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。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。ライブ環境にシステムをデプロイするたびに、システムがどの程度うまく機能するか、トレードオフの影響はどうかが判ります。

このように学んできたことをベースに、AWS による優れた設計のフレームワークは作成されました。これは、アーキテクチャが AWS ベストプラクティスにどの程度合致しているかを評価できる質問集です。

AWS による優れた設計のフレームワークは、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、運用性という 5 本の柱を基本としています。それぞれは以下のように定義されます。

柱名	説明
セキュリティ	リスクの評価と軽減戦略を通して、ビジネス価値を提供しながら、情報、システム、資産を保護する能力です。
信頼性	インフラストラクチャまたはサービスの障害から復旧したり、必要に応じて動的にコンピューティングリソースを獲得したり、設定ミスや一時的なネットワークの問題などによる障害を軽減したりといったシステムの能力です。
パフォーマンス効率	システムの要求に合わせてコンピューティングリソースを効率的に使用し、需要の変化や技術の進歩に合わせてこの効率を維持する能力です。
コストの最適化	不要なコストや最適でないリソースを回避または排除する能力です。
運用性	ビジネスの価値をもたらし、サポートプロセスと手順の継続的な向上を実現するために、システムを実行およびモニタリングする能力です。

ソリューションを設計する際にビジネスコンテキストに基づいて柱の間でトレードオフを行うことになり、こうしたビジネス上の決定がエンジニアリングの優先付けにつながります。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入優先順位に影響を与える可能性があります。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。

一般的な設計の原則

AWS による優れた設計のフレームワークは、クラウド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。システムのデプロイに先立ってキャパシティを決定すると、高価なアイドルリソースが発生したり、キャパシティが制限されることによりパフォーマンスに影響したりします。クラウドコンピューティングにはこのような問題はありません。必要な分のみキャパシティーを使用し、自動的にスケールアップまたはスケールダウンできます。
- **本稼働スケールでシステムをテストする:** クラウドでは、本稼働スケールのテスト環境を作成してテストを完了したら、リソースを破棄することができます。支払いはテスト環境を実行しているときのみ発生するため、オンプレミスでテストを実施する場合と比べて僅かなコストで、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上の実験を容易にする:** 自動化によって、システムを低コストで作成およびレプリケートでき、手動作業の手間がかかりません。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ーに戻すことができます。

- **発展するアーキテクチャが可能に:** 従来の環境では、アーキテクチャ上の決定は、しばしば静的な 1 回限りのイベントとして実装され、その運用期間中にシステムのいくつかの主要なバージョンが実装されます。ビジネスとそのコンテキストは変化し続けているのに、当初の決定に縛られ、システムがビジネスの需要の変化に対応できない場合があります。クラウドでは、自動化とオンデマンドなテスト機能によって、設計変更によって生じる影響のリスクを低減できます。これによって、システムが何度でも進化でき、新しいイノベーションをビジネスに活用することを、標準的なプラクティスとして実施できるようになります。

- **データ駆動型アーキテクチャ:** クラウドでは、選択したアーキテクチャ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することができます。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。クラウドインフラストラクチャはコード化できるので、そのデータを使用してアーキテクチャを見直し、継続的に改善していくことができます。

- **ゲームデーを通じた向上:** 定期的にゲームデーを開催し、実稼働環境で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がどのように実行されるのかをテストします。これは、改善可能な箇所を把握したり、イベントに対応する体験を組織的に育成するのに役立ちます。

AWS による優れた設計のフレームワークの 5 本柱

ソフトウェアシステムの作成はビルの建築に似ています。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、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、運用性の 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。

このセクションでは、5 本柱のそれぞれについて、定義、ベストプラクティス、質問、考慮点、および関連する AWS の主要サービスを説明します。

セキュリティの柱

セキュリティの柱には、リスクの評価と軽減戦略を通して、ビジネス価値を提供しながら、情報、システム、アセットを保護する能力が含まれます。

設計の原則

クラウドでは、システムセキュリティを強化する役に立つ数多くの原則があります。

- **すべてのレイヤーでセキュリティを適用:** インフラストラクチャのエッジでセキュリティアプライアンス (例: ファイアウォール) を実行するだけではなく、ファイアウォールや他のセキュリティ制御をすべてのリソース (例: 各仮想サーバー、ロードバランサー、ネットワークサブネット) でも使用します。
- **トレーサビリティを有効化:** 環境に対する全てのアクションと変更を記録し監査します。
- **最小権限の原則の導入:** AWS リソースとのそれぞれのやり取りに適切な認可が与えられていることを確認し、リソース対する強力な論理的アクセスコントロールを実装します。
- **お客様システムの保護にフォーカス:** AWS 責任共有モデルにもとづき、安全なインフラストラクチャやサービスを AWS が提供します。お客様は、その上で稼働するお客様のアプリケーション、データ、オペレーティングシステムの保護に集中できます。
- **セキュリティのベストプラクティスを自動化する:** ソフトウェアベースのセキュリティメカニズムによって、安全に、より迅速かつコスト効率よくスケールすることができます。パッチが適用されセキュリティ強化された仮想サーバーのイメージを作成して保存し、そのイメージを自動的にお客様が起動する各新規サーバーに使用します。リビジョン管理を通して、テンプレートで定義・管理された信頼ゾーンアーキテクチャ全体を作成します。ルーチンおよび異常なセキュリティイベントへの対応を自動化します。

定義

クラウドでのセキュリティには、次の 5 つのベストプラクティスの領域があります。

1. アイデンティティ管理とアクセス管理
2. 発見的統制
3. インフラストラクチャの保護
4. データ保護
5. インシデントへの対応

システムを構築する前に、セキュリティに影響を及ぼす基礎的なプラクティスが適切に導入される必要があります。誰が何を実行できるのかをコントロールする必要があります。さらに、セキュリティインシデントを特定し、システムとサービスを保護して、データ保護によりデータの機密性と整合性を維持する必要があります。それには、セキュリティインシデントに対応する明確に定義された実績のあるプロセスが必要となります。これらのツールや技術は、経済的損失の防止や、法令上の義務の遵守などの目的をサポートするために重要です。

AWS 責任共有モデルを使用して、クラウドを使用する企業がセキュリティおよびコンプライアンスの目的を達成できます。AWS がクラウドサービスをサポートするインフラストラクチャを物理的に保護するため、AWS のお客様は、サービスを使用してお客様の目的を達成することに集中できます。また、AWS クラウドでは、セキュリティデータへのアクセス機能が強化され、セキュリティイベントへの応答方法が自動化できます。

ベストプラクティス

アイデンティティ管理とアクセス管理

アイデンティティ管理とアクセス管理は情報セキュリティプログラムの主要部分です。権限を持つユーザーおよび認証されたユーザーのみが、意図した方法でのみ、確実にリソースにアクセスできるようにします。たとえば、プリンシパル (ユーザー、グループ、サービス、アカウントでアクションを実行するロール) を定義し、これらのプリンシパルに合わせてポリシーを構築し、強力な認証情報管理を導入します。これらの権限管理の構成要素は、認証と許可の主要な概念です。

AWS では、権限管理は主に AWS Identity and Access Management (IAM) サービスでサポートされており、AWS のサービスやリソースへのユーザーのアクセスを制御することができます。アクセス権限をユーザー、グループ、ロール、リソースに割り当てるポリシーは細かく適用できます。複雑さのレベル、パスワード再利用の禁止、多要素認証 (MFA) の使用など、強力なパスワードポリシーを要求することもできます。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できます。AWS にアクセスするシステムを必要とするワークロードでは、IAM がインスタンスプロファイル、ID フェデレーション、一時的な認証情報を通じて安全なアクセスを実現します。

次の質問はセキュリティのための権限管理の考慮事項に関するものです (セキュリティに関するすべての質問、回答、およびベストプラクティスの一覧については、付録を参照してください)。

SEC 1. AWS ルートアカウントの認証情報へのアクセスと使用をどのように保護していますか。

SEC 2. AWS マネジメントコンソールと API へのユーザーによるアクセスを制御するために、システムユーザーのロールと責任をどのように定義していますか。

SEC 3. AWS リソースへの自動化されたアクセスをどのように制限していますか (アプリケーション、スクリプト、サードパーティーのツールやサービスなど)。

ルートアカウント認証情報を保護し続けることは非常に重要です。このため、AWS では MFA をルートアカウントに設定し、MFA 付きの認証情報を物理的に安全な場所に保管することをお勧めします。IAM サービスでは、その他の (ルートではない) ユーザーアクセス権限を作成して管理し、またリソースへのアクセスレベルを確立できます。

発見的統制

発見的統制は潜在的なセキュリティインシデントの識別に使用できます。発見的統制は管理フレームワークに不可欠の部分であり、品質プロセス、法律上の準拠義務、脅威の識別や対応に利用されます。発見的統制には、さまざまな種類があります。たとえば、資産およびその詳細属性を目録化して、より効果的に意思決定 (およびライフサイクル統制) を促進し、オペレーションの基礎を確立します。あるいは、情報システムに関連するコントロールを精査する内部監査を使用し、実態がポリシーおよび要件に一致しているか、定義した条件に基づいたアラート通知の自動化が正しく設定されているかを確認できます。これらの統制は重要なリアクティブファクターであり、企業が変則的なアクティビティの範囲を識別して理解するのに役立ちます。

AWS では、ログ、イベント、モニタリングの処理によって、監査や自動分析、アラームを実現し、発見的統制を実装することができます。AWS CloudTrail ログ、AWS API 呼び出し、および Amazon CloudWatch はアラームを伴うメトリクスのモニタリングを提供し、AWS Config は設定履歴を提供します。サービスレベルログも使用可能です。たとえば、Amazon Simple Storage Service (S3) を使用してアクセスリクエストのログを作成できます。最後に、Amazon Glacier はボールトロック機能を提供しているため、ミッションクリティカルなデータを、監査可能な長期保持期間をサポートするように設計されたコンプライアンス統制を保って保持できます。

次の質問は、セキュリティのための発見的統制の考慮事項に関するものです。

SEC 4. ログをどのように取得して分析していますか。

セキュリティやフォレンジックに関連する理由のため、また規制および法令の要件のため、優れた設計においてはログ管理が重要です。潜在的なセキュリティインシデントを特定できるように、ログを分析してそれに対応することが重要です。AWS には、お客様がデータ保持ライフサイクルを定義したり、データを保管、アーカイブ、場合によっては削除する場所を定義したりすることで、ログ管理をより簡単に実装できる機能があります。この機能により、予測可能で信頼できるデータ操作を、よりシンプルに高いコスト効果で行うことができます。

インフラストラクチャの保護

インフラストラクチャの保護には、多層に渡る防御や Multi-Factor Authentication などのコントロール方法が含まれ、ベストプラクティスや、業界あるいは法令上の義務を満たす必要があります。クラウドでもオンプレミスでも、オペレーションの実行を成功させるには、これらの手段を使用することが非常に重要です。

AWS では、AWS ネイティブの技術を使用するか、AWS Marketplace で入手できるパートナー製品およびサービスを使用して、ステートフルおよびステートレスなパケットインスペクションを実装できます。また、Amazon Virtual Private Cloud (VPC) を使用してプライベートで安全でスケーラブルな環境を作成 し、ゲートウェイ、ルーティングテーブル、パブリックサブネット、およびプライベートサブネットを含むトポロジを定義できます。

次の質問は、セキュリティのためのインフラストラクチャ保護の考慮事項に関するものです。

SEC 5. ネットワークおよびホストレベルの境界保護をどのように実施していますか。

SEC 6. AWS サービスレベルセキュリティ機能をどのように活用していますか。

SEC 7. Amazon EC2 インスタンス上のオペレーティングシステムの整合性はどのように保護していますか。

多層防御はどのタイプの環境にもお勧めです。インフラストラクチャの保護に関しては、多くの概念や対応方法が、オンプレミスでもクラウドでも有効です。境界保護の徹底、送受信のモニタリング、広範囲のログ記録、モニタリングおよびアラートはすべて、効果的な情報セキュリティ計画に不可欠です。

前述の「設計の原則」セクションで説明したとおり、AWS のお客様は EC2 インスタンスの設定をカスタマイズまたはハードニングして、この設定を Amazon Machine Image (AMI) に配置できます。その後、Auto Scaling によるトリガーでも手動起動でも、この AMI で起動するすべての新規仮想サーバー (インスタンス) では、ハードニングされた設定が適用されます。

データ保護

システムを構築する前に、セキュリティに影響を及ぼす基礎的なプラクティスが適切に導入される必要があります。たとえば、データ分類は、企業のデータを重要度に基づいてカテゴリー分けする方法です。また、暗号化は、認証されていないアクセスでは解読できないようにデータを変換することでデータを保護します。これらのツールや技術は、経済的損失の防止や、法令上の義務の遵守などの目的をサポートするために重要です。

AWS では、次のようなプラクティスでデータの保護が実行されます。

- AWS のお客様は、お客様のデータに対する完全な統制を保持します。
- AWS ではデータの暗号化や定期的なキーローテーションを含むキーの管理がより簡単になります。これらは AWS によってネイティブに自動化することも、お客様が保守することもできます。
- ファイルアクセスや変更などの重要な内容を含む詳細なログ記録を使用できます。

- AWS は、非常に高い弾力性を持つようにストレージシステムを設計しました。たとえば、Amazon Simple Storage Service (S3) はほぼ完全な耐久性を求めた設計になっています。(たとえば、Amazon S3 を使用して 10,000 のオブジェクトを保存する場合、平均で、10,000,000 年間に 1 つのオブジェクトを消失する可能性しかありません)。
- バージョニングは、より大容量データのライフサイクル管理処理の一部であり、意図しない上書きや削除、その他の障害に対してデータを保護することができます。
- AWS ではリージョン間のデータの移動は実行しません。1 つのリージョンに置かれたコンテンツは、お客様が明示的に機能を有効にするか、その機能を提供するサービスを利用しないかぎり、そのリージョンに残り続けます。

次の質問は、データセキュリティの考慮事項に関するものです。

SEC 8. データをどのように分類していますか。

SEC 9. 保管時のデータをどのように暗号化、保護していますか。

SEC 10. キーと認証情報をどのように管理していますか。

SEC 11. 伝送中のデータをどのように暗号化、保護していますか。

AWS では、保管時および伝送時のデータの暗号化用に複数の手段を用意しています。これらの機能を製品とサービスに組み込み、データの暗号化を容易にします。たとえば、サーバー側の暗号化 (SSE) を Amazon S3 に実装し、暗号化されたフォームでのデータ保存をより簡単にします。また、HTTPS 全体の暗号

化および復号化処理 (一般には SSL ターミネーション と呼ばれます) を Elastic Load Balancing で実行するように調整することもできます。

インシデントへの対応

非常に成熟した予防的統制と発見的統制が行われているとしても、企業は、セキュリティインシデントの潜在的な影響に対応し、それを軽減するためのプロセスを導入する必要があります。ワークロードのアーキテクチャは、お客様のチームがインシデント発生時に、システムを分離したり、影響の拡大を抑えたり、オペレーションを通常の状態に復旧させたりして、効果的に対処できるかどうかに大きな影響を与えます。セキュリティインシデントが発生する前にツールを導入してアクセスし、定期的にインシデント対応を実施することで、調査と復旧をタイムリーに行えるようにアーキテクチャを更新できるようになります。

AWS では、次のようなプラクティスで効果的なインシデント対応が実行されます。

- ファイルアクセスや変更などの重要な内容を含む詳細なログ記録を使用できます。
- イベントは自動的に処理され、AWS API を使用して自動的に Runbook を実行するスクリプトをトリガーすることができます。
- AWS CloudFormation を使用して、プレプロビジョニングツールと「クリーンルーム」を実現できます。これにより、安全で分離された環境でフォレンジックを実施できます。

次の質問は、インシデント対応の考慮事項に関するものです。

SEC 12. どのようにして適切なインシデント対応が取れることを確認できますか。

InfoSec チームに対して迅速にアクセスを許可し、インスタンスの隔離と、フォレンジックのためのデータと状態の取得を自動化する方法を確立します。

関連する AWS サービス

セキュリティに不可欠な AWS サービスは AWS Identity and Access Management (IAM) です。AWS のサービスやリソースへのユーザーのアクセスを安全に管理できます。以下のサービスと機能は、セキュリティの 4 つの領域をサポートします。

Identity and Access Management: IAM により、AWS サービスおよびリソースへのアクセスを安全にコントロールすることができます。多要素認証 (MFA) は、ユーザー名およびパスワードに加え、拡張された保護レイヤーを追加します。

発見的統制: AWS CloudTrail は AWS API コールを記録します。AWS Config は AWS リソースおよび設定の詳細なインベントリーを提供し、Amazon CloudWatch は AWS リソースのサービスをモニタリングします。

インフラストラクチャの保護: Amazon Virtual Private Cloud (VPC) では、AWS クラウド上にプライベートで隔離されたネットワークをプロビジョニングすることができ、仮想ネットワーク内に AWS リソースを起動することができます。

データ保護: Elastic Load Balancing、Amazon Elastic Block Store (EBS)、Amazon Simple Storage Service (S3)、および Amazon Relational Database Service (RDS) などのサービスには、伝送中や保管中のデータを保護する暗号化機能があります。AWS Key Management Service (KMS) では、データの暗号化に使用する暗号キーの作成や制御を簡単に行うことができます。

インシデント対応: IAM を使用して、インシデント対応チームに適切な許可を付与する必要があります。Amazon CloudFormation を使用して、調査を実行するための信頼できる環境を構築することができます。

リソース

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

ドキュメント & ブログ

- AWS セキュリティセンター
- AWS コンプライアンス
- AWS セキュリティブログ

ホワイトペーパー

- AWS セキュリティの概要
- AWS セキュリティのベストプラクティス
- AWS リスクおよびコンプライアンス

動画

- AWS クラウドのセキュリティ
- 責任共有モデルの概要

信頼性の柱

信頼性の柱には、インフラストラクチャまたはサービスの障害からの復旧、必要に応じた動的なコンピューティングリソースの獲得、設定ミスや一時的なネットワークの問題などによる障害軽減などのシステムの能力が含まれています。

設計の原則

クラウドでは、信頼性を高める役に立つ数多くの原則があります。

- **復旧手順のテスト:** オンプレミス環境では、テストはたいてい特定のシナリオによってシステムが機能することを証明するために行われます。復旧戦略を検証するためにテストが使用されることはあまりありません。クラウドでは、システム障害の発生過程をテストし、復旧手順を検証できます。自動化により、異なる障害をシミュレートしたり以前に障害が発生したシナリオを再作成したりできます。これらによって障害経路が洗い出され、実際に障害が発生する前にテストし修正することで、テストしていないコンポーネント障害のリスクを軽減できます。
- **障害から自動的に復旧する:** システムの主要なパフォーマンスインジケーター (KPI) をモニタリングすることで、しきい値を超過した場合に自動処理をトリガーできます。それにより自動的に障害を通知および追跡し、障害を回避または修正する復旧プロセスを自動化できます。より高度な自動化を使用して、障害が発生する前にそれを予測し修正することも可能です。

- **水平方向にスケールして集約システム能力を強化する:** 1 つの大きなリソースを複数の小さなリソースに置換して、1 つの障害がシステム全体に及ぼす影響を軽減します。複数のより小さなリソースに要求を分散させて、障害の共通点を共有しないようにします。
- **キャパシティーの推測が不要に:** オンプレミスシステムで発生する障害の一般的な原因はリソースの飽和です。システムに対する要求がそのシステムの能力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になります)。クラウドでは、需要とシステム使用率を監視し、需要を満たすために最適なレベルを維持するためのリソースの追加や削除を自動化することができます。
- **自動化における変更管理**: インフラストラクチャの変更を自動化により実行する必要があります。管理する必要のある変更は自動化の変更です。

定義

クラウドで高い信頼性を実現するには、次の 3 つのベストプラクティスの領域があります。

1. 基礎
2. 変更管理
3. 障害管理

信頼性を確保するには、システムによく計画された基礎があり適切な箇所をモニタリングし、また要望や要件に応じた変更を行う仕組みが必要です。障害を検出し自動的に自己修復するようにシステムを設計する必要があります。

ベストプラクティス

基礎

システムを構築する前に、信頼性に影響を及ぼす基礎的な要件が適切に導入される必要があります。たとえば、データセンターには十分なネットワーク帯域幅が必要です。これらの要件は時として放置されがちです (それぞれのプロジェクトスコープ外のため)。これらを放置すると、信頼性の高いシステムを構築する際に重大な影響を与える可能性があります。オンプレミス環境では、依存関係の影響によりこれらの要件を満たすには長い時間を必要とします。そのため計画の初期に組み込まれる必要があります。

AWS では、ほとんどの基礎要件は組み込み済みであるか、必要に応じて利用できます。クラウドは基本的に制限がないように設計されています。そのため、十分なネットワーキングおよびコンピューティングキャパシティの要件を満たすのは AWS の責任であり、お客様はオンデマンドでストレージデバイスのサイズなどのリソースサイズおよび割り当てを自由に変更できます。

次の質問は、信頼性のための基本的な考慮事項に関するものです。

REL 1. アカウントの AWS サービス制限はどのように管理していますか。

REL 2. AWS でのネットワークトポロジをどのように計画していますか。

AWS は、お客様がリソースを意図せずよけいにプロビジョニングしてしまう事故を防止するため、サービス制限 (チームが要求できる各リソースの数の上限) を設定しています。これらの制限をモニタリングしビジネスのニーズに合わせて変更するために、適切なガバナンスおよびプロセスを設ける必要があります。クラウドを取り入れるにあたって、既存のオンプレミスリソースとの統合 (ハイブリッドアプローチ) を計画しなければならない場合があります。ハイブリッドモデルにより、時間をかけて段階的にオールインのクラウド利用に移行することができます。そのため、AWS とオンプレミスのリソースがネットワークトポロジとして相互作用する方法を考えて設計することが重要です。

変更管理

変更がシステムに及ぼす影響を把握すれば事前に計画を立てることができます。モニタリングによって容量の問題や SLA 未達を引き起こす傾向をすばやく見つけることができます。旧来の環境では、変更制御のプロセスは手動であることが多く、変更を行う担当者や変更実施時期を効果的に制御するには、監査を受けながら注意深く調整しなければなりません。

AWS を使用することで、システムの挙動をモニタリングし KPI の変化への対応を自動化できます。たとえば、システムのユーザーが増えた場合にサーバーを追加するなどです。誰にシステムを変更する権限を持たせるかを制御し、これらの変更の履歴を監査できます。

次の質問は、変更管理に関連する信頼性のための考慮事項に関するものです。

REL 3. システムは需要の変化にどのように対応できますか。

REL 4. AWS リソースをどのようにモニタリングしていますか。

REL 5. 変更をどのように実行していますか。

需要の変更に対応してリソースを自動的に追加および削除するようにシステムを設計すると、信頼性が向上するだけでなく、ビジネスの成功が重荷ではなくなります。適切にモニタリングしていれば、KPI が予想した水準を逸脱した場合、チームは自動的にアラートを受信します。環境に対する変更を自動的に記録することで、それを監査し信頼性に影響を与えるアクションをすばやく識別できます。変更管理を統制することで、必要とされる信頼性を達成するためのルールを強化できます。

障害管理

合理的な複雑性を持つシステムでは、障害は発生するものです。このような障害を感知し、対応し、再発を防止する方法には、誰しも関心を持っています。

AWS では、モニタリングデータに対応するために自動化を活用できます。たとえば、特定のメトリックスがしきい値を超えたとき、自動化されたアクションをトリガーして問題を修正できます。また、本番環境の一部である、障害が発生したリソースを診断して修正するのではなく、新しいものに置換して、障害が発生したリソースを本番環境外で分析できます。クラウドでは低価格で一時的なバージョンとしてシステム全体を立ち上げることができるため、自動化されたテストを使用して復旧プロセス全体を検証できます。

次の質問は、障害管理に関する信頼性のための考慮事項に関するものです。

REL 6. データをどのようにバックアップしていますか。

REL 7. システムはコンポーネントの障害にどのように対応しますか。

REL 8. 弾力性をどのようにテストしていますか。

REL 9. 災害対策についてどのように計画していますか。

論理エラーおよび物理エラーの両方から確実に復旧するように、定期的にデータをバックアップし、バックアップファイルをテストしてください。障害管理の要は、障害から復旧までの自動化されたシステムのテストを頻繁に実行することです (定期スケジュールに加えて、大規模なシステム変更後にも実行されるのが理想です)。システムの弾力性 (特に障害テストシナリオ内で) を評価して、目標復旧時間 (RTO) および目標復旧時点 (RPO) などの KPI を能動的にトラッキングしてください。KPI をトラッキングすることで、単一障害点を識別し修正する手がかりになります。目標は、システム復旧プロセスを徹底的にテストし、問題が解決できない局面においても、すべてのデータが復旧可能で顧客へのサービスを継続できる確信を持ち続けることです。復旧プロセスは通常の本番稼働プロセスと同様に習熟されていなければなりません。

関連する AWS サービス

信頼性を確実にする要となる AWS サービスは、実行中のメトリックスをモニタリングする Amazon CloudWatch です。信頼性の 3 つの領域をサポートするその他のサービスと機能は、次のとおりです。

基礎: AWS Identity and Access Management (IAM) により、お客様のユーザーの AWS サービスおよびリソースへのアクセスを安全にコントロールすることができます。Amazon VPC を使用すると、AWS クラウドの隔離されたプライベートなセクションをプロビジョニングし、仮想ネットワークで AWS リソースを起動することが可能になります。

変更管理: AWS CloudTrail は、アカウントの AWS API コールを記録し、アカウント保持者に監査用のログファイルを送信します。AWS Config は AWS リソースのおよび設定の詳細インベントリーを提供し、継続して設定の変更を記録します。

障害管理: AWS CloudFormation では AWS リソース作成用にテンプレートを提供して、整然とした予測可能な方法でプロビジョニングできます。

リソース

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

ビデオおよびアナリストレポート

- 障害の受け入れ: 障害の意図的な生成とサービスの信頼性
- クラウドでの可用性と信頼性のベンチマーク

ドキュメントおよびブログ

- サービスの制限
- サービス上限についてのブログ

ホワイトペーパー

- AWS を用いたバックアップアーカイブおよびリカバリのアプローチ
- 大規模環境での AWS インフラストラクチャの管理
- AWS 災害対策
- AWS Amazon VPC のネットワーク接続オプション

AWS サポート

- AWS プレミアムサポート
- Trusted Advisor

パフォーマンス効率の柱

パフォーマンス効率の柱は、コンピューティングリソースを要求に合わせて効率的に使用し、需要の変化や技術の進化に合わせてその効率を維持することです。

設計の原則

クラウドでは、パフォーマンス効率を達成するいくつかの原則があります。

- **先端技術の一般化:** 技術の知識や複雑性をクラウドベンダーの領域に押しやることで、実装が難しかった技術の利用がより簡単になります。IT チームに新技術のホストおよび実行方法を学習してもらうよりも、単純にサービスとして使用してもらうことができます。たとえば、NoSQL データベース、メディア変換、機械学習はどれも専門知識を必要とする技術ですが、この専門知識は技術コミュニティに広く行き渡っているわけではありません。クラウドでは、これらの技術はチームが使用できるサービスとなり、リソースのプロビジョニングや管理よりも製品開発に注力できます。

- **グローバル化を即座に達成:** わずか数クリックで、世界中の複数のリージョンにシステムを簡単にデプロイします。これにより、最小限のコストで、より低いレイテンシーとより良いエクスペリエンスを顧客に提供できます。
- **サーバーレスアーキテクチャの使用:** クラウドでは、サーバーレスアーキテクチャにより、従来のコンピューティング活動を実行するサーバーを稼働させ維持する必要がなくなります。たとえば、ストレージサービスは静的ウェブサイトとして動作でき、ウェブサーバーの必要がなくなります。イベントサービスならお客様のプログラムコードをホストできます。サーバーを管理するオペレーション上の負担がなくなるだけでなく、トランザクションコストが低くなります。なぜなら、これらのマネージドサービスは、クラウドのスケールで実行されるからです。
- **実験の頻度の増加:** 仮想化された自動化可能なリソースを使用して、異なった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行できます。
- **適合したテクノロジーアプローチ:** 実現させようとしていることに最も適した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アプローチを選択する際にデータアクセスパターンを検討します。

定義

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

1. 選択 (コンピューティング、ストレージ、データベース、ネットワーク)
2. 確認
3. モニタリング
4. 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。ハイレベルな設計からリソースタイプの選択と設定まで、アーキテクチャのあらゆる側面についてデータを収集します。定期的に選択内容を見直すことで、AWSプラットフォームを継続的に発展させるというメリットが得られます。モニタリングにより、期待するパフォーマンスからの逸脱を認識して、それに対するアクションをとることができます。最後に、圧縮またはキャッシュなどを使用するか、整合性要件を緩和することで、アーキテクチャのトレードオフを通じてパフォーマンスを向上させることができます。

ベストプラクティス

選択

特定のシステムに向けた適切なソリューションは、抱えているワークロードの種類によって異なり、多くの場合、複数のアプローチを組み合わせたものになります。優れた設計のシステムでは、複数の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を向上させます。

AWS では、リソースは仮想化され、さまざまなタイプおよび設定で利用できます。これにより、ニーズに忠実に一致するアプローチを見つけることがより簡単になるとともに、オンプレミスのインフラストラクチャでは簡単に実現できないストレージの選択肢も得られます。たとえば、Amazon DynamoDB などのマネージド型サービスは、どのような規模でも、数ミリ秒台のレイテンシーのフルマネージド型 NoSQL データベースを提供します。

次の質問例は、選択の考慮事項に関するものです。

PERF 1. 最良の実行アーキテクチャをどのように選択していますか。

アーキテクチャのパターンと実装を選択するときは、最適なソリューションにデータ駆動型のアプローチを使用します。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 パートナーは経験や実績に基づくアーキテクチャの選択に役立ちますが、アーキテクチャの最適化にはベンチマークまたはロードテストから取得したデータが必要です。

アーキテクチャは、さまざまなアーキテクチャアプローチ (イベント駆動型、ETL、パイプラインなど) を組み合わせる可能性が高くなります。アーキテクチャの実装では、アーキテクチャのパフォーマンスの最適化に特有の AWS 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 つの主要なリソースタイプ (コンピューティング、ストレージ、データベース、およびネットワーク) を見ていきます。

コンピューティング

特定のシステムに向けた適切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設計、使用パターン、構成設定によって異なります。アーキテクチャでは、パフォーマンスを向上させるために、さまざまなコンポーネントに対して異なるコンピューティングソリューションを使用し、異なる機能を有効にします。システムに対して適切でないコンピューティングソリューションや機能を選択すると、パフォーマンス効率が低下する可能性があります。

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

- **インスタンス**は仮想化されたサーバーであるため、ボタンクリックまたは API コールで機能を変更できます。クラウド上でのリソースの決定が変更できないものではなくなったため、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インスタンスは異なるファミリーおよびサイズで使用でき、ソリッドステートドライブ (SSD) やグラフィック処理ユニット (GPU) を含む幅広い機能を提供します。
- **コンテナ**は、アプリケーションとそのリソースを分離したプロセスの依存関係を実行できるオペレーティングシステム仮想化の方法です。
- **関数**は実行するコードから実行環境を抽象化します。たとえば、AWS Lambda ではインスタンスを実行せずにコードを実行できます。

次の質問例は、コンピューティングの考慮事項に関するものです。

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

コンピューティングの使用を設計するときは、需要の変化に応じてパフォーマンスを維持するのに十分なキャパシティを確保するために利用できる伸縮自在性のメカニズムを利用する必要があります。

ストレージ

特定のシステムの最適なストレージソリューションは、アクセス方法の種類 (ブロック、ファイル、またはオブジェクト)、アクセスのパターン (ランダムまたはシーケンシャル)、必要なスループット、アクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、および可用性と耐久性の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を向上させます。

AWS では、ストレージは仮想化され、さまざまなタイプで利用できます。これにより、データ保管方法をニーズに一層忠実に一致させることがより簡単になるとともに、オンプレミスのインフラストラクチャでは簡単に実現できないストレージの選択肢も得られます。たとえば、Amazon S3 は、イレブンナイン (99.999999999%) の堅牢性を実現するよう設計されています。また、マグネティックハードドライブ (HDD) の使用をソリッドステートドライブ (SSD) に変更することも、仮想ドライブを 1 つのインスタンスから別のインスタンスに数秒で簡単に移動することもできます。

次の質問例は、パフォーマンス効率を向上させるためのストレージの考慮事項に関するものです。

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。

ストレージソリューションを選択する際には、アクセスパターンと一致することを確認して、必要なパフォーマンスを達成することが不可欠です。

データベース

特定のシステムに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、拡張性、およびクエリ機能についての要件によって異なります。多くのシステムでは、パフォーマンスを向上させるために、さまざまなサブシステムに対して異なるデータベースソリューションを使用し、異なる機能を有効化しています。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可能性があります。

AWS では、Amazon Relational Database Service (RDS) が、完全マネージド型リレーショナルデータベースサービスを提供します。Amazon RDS を使用すれば、データベースのコンピューティングリソースやストレージリソースをスケールすることができ、多くの場合、ダウンタイムは発生しません。Amazon DynamoDB は、どのような規模でも、数ミリ秒台に抑えられたレイテンシーを提供する、フルマネージド型 NoSQL データベースです。Amazon Redshift は、パフォーマンスまたはキャパシティーのニーズが変わったときにノードの数やタイプを変更することができる、マネージド型でペタバイトスケールのデータウェアハウスです。

次の質問例は、パフォーマンス効率を向上させるためのデータベースの考慮事項に関するものです。

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。

ワークロードのデータベース手法 (RDBMS、NoSQL など) はパフォーマンス効率に大きな影響を与えますが、これはデータ駆動型のアプローチを通じてではなく、組織のデフォルトに従って選択されることがしばしばある領域です。ストレージと同様に、ワークロードのアクセスパターンを検討し、他の非データベースソリューションが問題をより効率的に解決できるかどうかを検討することも重要です (検索エンジンやデータウェアハウスの使用など)。

ネットワーク

特定のシステムに最適なネットワークソリューションは、レイテンシー、スループット要件などによって異なります。ユーザーやオンプレミスリソースなどの物理的な制約によりロケーションの選択が決まりますが、エッジの手法やリソースの配置を使用して制約を相殺することができます。

AWS では、ネットワーキングは仮想化され、さまざまなタイプおよび設定で利用できます。これにより、ネットワーキングの方法をニーズにさらに近づけることが容易になります。AWS はネットワークトラフィックを最適化するための製品機能 (非常に高度なネットワークインスタンスタイプ、Amazon EBS 最適化インスタンス、Amazon S3 Transfer Acceleration、Dynamic Amazon CloudFront など) を提供します。AWS はネットワーク距離またはジッターを低減するためのネットワーク機能 (Amazon Route53 レイテンシールーティング、Amazon VPC エンドポイント、AWS Direct Connect など) も提供します。

次の質問例は、パフォーマンス効率を向上させるためのストレージの考慮事項に関するものです。

PERF 5. ネットワークソリューションをどのように選択していますか。

ネットワークソリューションを選択するときは、ロケーションを考慮する必要があります。AWS を使用すると、距離を縮めるために使用する場所の近くにリソースを配置することができます。リージョン、プレースメントグループ、エッジロケーションを利用することで、パフォーマンスを大幅に向上させることができます。

確認

ソリューションの設計には、選択できる一定のセットのオプションがあります。しかし、時間がたつと、アーキテクチャのパフォーマンスを向上させる可能性のある新しいテクノロジとアプローチが利用可能になります。

AWS を使用することで、お客様のニーズに基づいた継続的なイノベーションを活用できます。新しいリージョン、エッジロケーション、サービス、機能を定期的にリリースします。これらのいずれも、お客様のアーキテクチャーのパフォーマンス効率を積極的に向上させる可能性があります。

次の質問例は、パフォーマンス効率を向上させるための確認に関するものです。

> **PERF 6. 新しいリソースタイプや機能の提供が開始される中で、どのようにして最適なリソースタイプを使い続けていることを担保していますか。**

アーキテクチャのパフォーマンスが制限されている場所を理解することで、その制約を緩和できるリリースを検討することができます。

モニタリング

アーキテクチャを実装したら、パフォーマンスを監視して、顧客が認識する前に問題を修正する必要があります。しきい値を超過した場合にアラームを送信

させるには、モニタリングメトリックを使用する必要があります。アラームは、パフォーマンスの悪いコンポーネントを回避するための自動化されたアクションをトリガーすることができます。

Amazon CloudWatch は AWS を使用して通知アラームを監視および送信し、自動化を使用して Amazon Kinesis、Amazon Simple Queue Service (SQS)、および AWS Lambda を介してアクションをトリガーすることでパフォーマンスの問題を回避することができます。

次の質問例は、パフォーマンス効率を向上させるためのモニタリングに関するものです。

PERF 7. 起動後のリソースが期待どおりの性能を出すように、リソースをどのようにモニタリングしていますか。

あまりにも多くの誤検出がないか、データを処理しきれなくなっていないかを確認することは、効果的なモニタリングソリューションを実現する上で重要です。自動トリガーは人為的なミスを防ぎ、問題を修正する時間を短縮することができます。本番環境でシミュレーションを行う「ゲームデー」を計画して、モニタリングソリューションをテストし、問題を正しく認識できるようにします。

トレードオフ

ソリューションを構築するときに、トレードオフを考慮することで、最適なアプローチを選択できます。状況に応じて、整合性、耐久性、スペースと時間、およびレイテンシーをトレードして、より高いパフォーマンスを実現できます。

AWS を使用すれば、即座にグローバル化を達成し、世界各地の複数の場所でリソースをデプロイして、エンドユーザーに近づけることができます。また、読み取り専用レプリカをデータベースシステムなどの情報ストアに動的に追加し、プライマリデータベース上の負荷を減らすこともできます。AWS は、インメモリデータストアまたはキャッシュを提供する Amazon ElastiCache や、静的コンテンツのコピーをエンドユーザーの近くにキャッシュする Amazon CloudFront などのキャッシュソリューションも提供しています。

次の質問例は、パフォーマンス効率を向上させるための容量と時間のトレードオフに関するものです。

PERF 8. パフォーマンスを向上させるためにトレードオフをどのように使用しますか。

トレードオフにより、アーキテクチャの複雑さが増すことがあります。また、どのくらいの利点があるか確認するためロードテストが必要になることがあります。

関連する AWS サービス

パフォーマンス効率を高めるための主要な AWS サービスは、Amazon CloudWatch です。これはリソースやシステムをモニタリングし、パフォーマンス全体および運用健全性を可視化します。以下のサービスは、パフォーマンス効率の領域において重要です。

選択:

コンピューティング: Auto Scaling は、需要を満たし応答性を維持するために十分なインスタンスを確保するための鍵となります。

ストレージ: Amazon EBS は、ユースケースにあわせて最適化するための広範囲なストレージオプション (SSD やプロビジョニングされた 1 秒あたりの入力/出力オペレーション数 (PIOPS) など) を提供します。Amazon S3 はサーバーレスのコンテンツ配信機能を提供し、Amazon S3 Transfer Acceleration によって長距離間でファイルを高速、簡単、安全に転送できます。

データベース: Amazon RDS は、ユースケースにあわせて最適化できる広範囲なデータベース機能 (プロビジョンド IOPS やリードレプリカなど) を提供します。Amazon DynamoDB は、どのような規模でも、数ミリ秒台に抑えらえれたレイテンシーを提供します。

ネットワーク: Amazon Route 53 はレイテンシーに基づくルーティングを提供します。Amazon VPC エンドポイントと Direct Connect を使用すると、ネットワークの距離やジッターを低減できます。

確認: AWS ブログと AWS ウェブサイトの「最新情報」セクションは、新機能や新しいサービスについて学習するためのリソースです。

モニタリング: Amazon CloudWatch は、既存の監視ソリューションと統合でき、AWS Lambda でアクションをトリガーするために使用できるメトリックス、アラーム、および通知を提供します。

トレードオフ: Amazon ElastiCache、Amazon CloudFront、AWS Snowball によりパフォーマンスを改善できます。Amazon RDS のリードレプリカにより、読み込み負荷の高いワークロードを拡大することができます。

リソース

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

動画

- Performance Channel
- AWS におけるパフォーマンスベンチマーキング

ドキュメント

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

コストの最適化の柱

コストの最適化の柱には、ライフサイクル全体にわたるシステムの改善と継続的な改善のプロセスが含まれます。最初の概念実証の初期設計からプロダクションワークロードの実行中の操作まで、このホワイトペーパーのプラクティスを採用することで、ビジネス上の成果を達成しコストを最小限に抑えるコスト意識の高いシステムを構築して運用することができ、投資収益率を最大化できます。

設計の原則

クラウド上では、コストの最適化を達成できる数多くの原則に従うことができます。

- **消費モデルの採用:** 詳細な予測を使用するのではなく、消費するコンピューティングリソースに対してのみ支払いを行い、ビジネス要件に応じて使用量を増減します。たとえば、開発環境とテスト環境は、通常、

週中 1 日 8 時間しか使用されません。使用しないときはこれらのリソースを停止して最大 75% のコストを節約できます (40 時間対 168 時間)。

- **大幅なスケールメリット:** クラウドコンピューティングを使用すると、自社環境よりも低い変動コストを実現できます。AWS には高いスケールメリットがあるためです。数十万単位のユーザーの使用が AWS クラウドに集約されるため、従量課金制の料金も低くなります。
- **データセンターの運用への投資が不要に:** サーバーのラッキング、スタック、電源供給といった手間のかかる作業は AWS が行うため、お客様は IT インフラストラクチャではなく自社の顧客やビジネスプロジェクトに集中することができます。
- **投資の分析と要因の把握:** クラウドを使用すると、システムの使用量とコストを正確に特定することが容易になり、IT コストを個々のビジネスオーナーに明確に帰属させることができます。これは投資効果 (ROI) を測定する助けとなり、リソースを最適化して、コストを削減化するための機会をシステムオーナーに与えることになります。
- **マネージドサービスを使用して所有コストを減らす:** クラウド上では、マネージドサービスは E メールの送信といったタスクを行うサーバーの保守や、データベースの管理のような運用面の負荷を取り除きます。さらに、マネージドサービスはクラウドスケールで実行されるため、トランザクションまたはサービスあたりのコストが低くなります。

定義

クラウド上でのコストの最適化には、次の 4 つのベストプラクティスの領域があります。

1. コスト効果が高いリソース
2. 供給と需要の一致

3. 費用の把握
4. 継続した最適化

他の柱と同様に、考慮すべきトレードオフがあります。たとえば、最適化したいのは市場投入時間ですか、またはコストですか。場合によっては、先行コストの最適化に投資するのではなく、市場投入時間、新機能の提供、または単純に期日の順守といった、スピードの最適化が最善であることもあります。設計上の決定は、経験的データを使用せずにあわてて行われる場合があります。コスト最適化したデプロイのベンチマークに時間を費やすよりも、「念のため」に過度な補償を行いたくなるためです。これにより、過度にプロビジョニングしすぎ、最適化されていないデプロイがしばしば生まれます。以下のセクションでは、デプロイの初期コストの最適化および継続的なコストの最適化についての技術的および戦略的なガイダンスを示します。

ベストプラクティス

コスト効果が高いリソース

システムに対して適切なインスタンスとリソースを使用することが、コスト削減の鍵となります。たとえば、小規模なサーバーでレポートを作成する処理に 5 時間かかるところ、コストが 2 倍の大規模なサーバーでは、これを 1 時間で実行できるとします。どちらのジョブでも結果は同じですが、小規模なサーバーでは時間とともにより多くのコストが発生します。

優れた設計のシステムでは、最もコスト効率の高いリソースが使用され、それにより大きなコスト効果が得られます。また、コストを削減するために、マネージドサービスを使用することもできます。たとえば、E メール配信のために複数のメールサーバーを維持する代わりに、メッセージごとに課金するサービスを使用することができます。

AWS は柔軟でコスト効率が高いさまざまな料金オプションを提供し、ニーズに最も合った方法で Amazon EC2 インスタンスを利用できるようにします。オンデマンドインスタンスは、最低契約金なしに、時間単位で、コンピューティング性能に対して料金をお支払いいただくものです。リザーブドインスタンス (RI) では、キャパシティーを予約し、オンデマンド料金の最大 75 パーセントを節約できます。スポットインスタンスでは、使用されていない Amazon EC2 キャパシティーを、大幅な割引で価格を指定して利用することができます。スポットインスタンスは、HPC やビッグデータのように、サーバー群の中の個々のサーバーが、動的に追加/削除されるような状況での利用に適しています。

次の質問例は、コスト最適化のために、コスト効率の高いリソースを選択する方法に関するものです。

COST 1. ソリューションに AWS サービスを選択する際のコストを検討していますか。

COST 2. コスト目標を達成するためにリソースのサイズを決定しましたか。

COST 3. コスト目標を達成するために適切な料金モデルを選択していますか。

AWS Trusted Advisor などのツールを使用して定期的に AWS の使用量を確認することで、使用率をアクティブにモニタリングし、それに応じてデプロイを調整できます。

供給と需要の一致

供給と需要を最適に一致させることで、システムのコストは最少となりますが、プロビジョニングの時間と個別のリソースの障害を考慮して、十分な供給の余力を確保しておく必要もあります。需要は固定または可変の場合があるため、管理が大きなコストとならないようにメトリックスと自動化が必要になります。

AWS では、需要に合わせて自動的にリソースをプロビジョニングできます。Auto Scaling、需要、バッファ、時間ベースの手法により、必要に応じてリソースを追加または削除できます。需要の変動を予想できる場合、より費用を節約し、リソースをシステムのニーズに一致させることができます。

次の質問例は、コスト最適化のための供給と需要の一致に関するものです。

> **COST 4. キャパシティーが必要量を満たしているが大幅に超えていないことをどのように実現していますか。**

需要と供給とを一致させるように設計するときは、新しいリソースをプロビジョニングするのに使用されるパターンや時間のパターンについて積極的に考える必要があります。

費用の把握

クラウドで柔軟性と俊敏性が高まることにより、イノベーションと速いペースの開発およびデプロイが容易になります。ハードウェア仕様の確認、価格見積りの交渉、発注書の管理、配送のスケジュール、リソースのデプロイなど、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する手動のプロセスと時間

が排除されます。ただし、使いやすさと事実上無制限のオンデマンドキャパシティーでは、支出に関する新しい考え方が必要になる場合があります。

多くのビジネスは、さまざまなチームによって実行される複数のシステムで構成されています。それぞれのビジネスオーナーまたは製品オーナーにリソースのコストを割り当てるようにすると、リソースを効率的に利用するようになり、無駄を低減できます。正確なコストを特定することで、実際に利益率の高い製品を把握することができ、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。

次の質問例は、コストの最適化のための費用の把握に関するものです。

COST 5. アーキテクチャを設計するときに、データ転送料金について考慮しましたか。

COST 6. 使用状況と支出をどのようにモニタリングしていますか。

COST 7. 必要なくなったリソースは廃棄していますか。または、一時的に必要ないリソースを停止していますか。

COST 8. AWS の使用状況を管理するためにどのような管理と手順を実行していますか。

コスト配分タグを使用して、AWS コストをカテゴライズおよび追跡できます。使用している AWS リソース (たとえば Amazon EC2 インスタンスや Amazon S3 バケット) にタグを適用すると、使用量とコストをタグごとに集計したコスト配分レポートが生成されます。自社のカテゴリ (たとえばコストセンター、シ

ステム名、所有者) を表すタグを適用すると、複数のサービスにわたってコストを分類することができます。

タグ付けされたリソースとエンティティライフサイクル追跡 (従業員、プロジェクト) を結合することで、ビジネス上の価値を生み出しておらず、削除する必要のある孤立したリソースやプロジェクトを確認できます。予測される費用超過を通知する請求アラートを設定できます。また、AWS 簡易見積りツールによりデータ転送コストを計算することもできます。

継続した最適化

AWS が新しいサービスや機能を公開するときに、既存のアーキテクチャ上の決定を見直し、それらが引き続き最もコスト効果が高いことを確認するのがベストプラクティスです。要件が変わったら、必要なくなったリソースとサービス全体またはシステムを積極的に廃棄します。

AWS のマネージドサービスにより、しばしばソリューションを大幅に最適化できるため、利用可能になった新しいマネージドサービスについて確認することをお勧めします。たとえば、Amazon RDS データベースの実行は、Amazon EC2 で独自のデータベースを実行するよりも安価になる場合があります。

次の質問例は、コスト最適化のためのコストの再評価に関するものです。

COST 9. 新しいサービスの採用をどのように管理または検討していますか。

デプロイされたシステムを定期的に確認することで、新しい AWS サービスを利用してコストを下げることがしばしば可能になります。また、より新しいサービスが費用を節約するためにどのように役立つかを評価します。たとえば、Aurora 用の AWS RDS 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。

関連する AWS サービス

コストの最適化をサポートする主な AWS 機能はコスト配分タグで、システムのコストを理解するうえで役立ちます。以下のサービスと機能は、コスト最適化の 4 つの領域に関して重要です。

コスト効果が高いリソース: リザーブドインスタンスと前払いのキャパシティを使用してコストを削減できます。AWS Trusted Advisor を使用してお客様の AWS 環境を検査し、コスト削減の機会を見つけることができます。

供給と需要の一致: Auto Scaling では、費用の超過を抑えて、需要に合わせてリソースを追加または削除できます。

費用の把握: Amazon CloudWatch アラームと Amazon Simple Notification Service (SNS) では、予算額を超えた場合や、予算額を超える見通しの場合に警告を通知することができます。

継続した最適化: AWS ブログと AWS ウェブサイトの「最新情報」セクションは、新機能や新しいサービスについて学習するためのリソースです。AWS Trusted Advisor は AWS 環境を検査し、使用されていないリソースやアイドル状態のリソースを排除したり、リザーブドインスタンス容量をコミットしたりすることで、コスト削減の可能性を見つけます。

リソース

コストの最適化に関連する AWS ベストプラクティスの詳細については、以下の参考資料を参照してください。

動画

- AWS でのコスト最適化

ドキュメント

- AWS クラウドエコノミクスセンター

ツール

- AWS 総所有コスト (TCO) 計算ツール
- AWS 詳細な請求レポート
- AWS 簡易見積りツール
- AWS コストエクスプローラー

運用性の柱

運用の柱には、本番環境のワークロード管理に利用される運用手法と手順が含まれます。これには、どのように計画された変更を実行するのかや予期しない運用イベントへの応答が含まれます。変更の実行や応答は自動化されている必要があります。高い運用性のためのすべてのプロセスと手順は文書化して、テストし、定期的に確認する必要があります。

設計の原則

クラウド上では、高い運用性を実現するいくつかの原則があります。

- **コードによるオペレーションの実行:** 共通の反復プロセスまたは手順がある場合は、自動化を使用します。たとえば、構成管理、変更、およびイベントへの応答を自動化することを検討してください。
- **オペレーションプロセスをビジネス目標に合わせる:** ビジネス目標を達成する上で高い運用性を示すメトリックスを収集します。ゴールはメトリックのノイズ信号を低減することであり、運用上のモニタリングと応答は、ビジネスクリティカルなニーズをサポートすることを目標としています。不要なメトリックスを収集することで、モニタリングと応答が複雑になるため、予期しない運用イベントへの効果的な対応が妨げられます。
- **定期的に、小規模で、増分変更を加える:** ワークロードは、コンポーネントを定期的に更新できるように設計する必要があります。変更は、大規模バッチではなく、小さな増分で行う必要があり、オペレーションに影響を与えずにロールバックできる必要があります。メンテナンスや依

存するサービスコンポーネントの交換のための停止時間なしに、これらの変更を実装できるように、オペレーション手順を整えます。

- **予期しないイベントの応答のためのテスト:** ワークロードは、コンポーネントの障害や他の予期しないイベントについてテストする必要があります。運用イベントに応答する手順をテストして理解することが重要であり、運用イベントが発生したときにその手順に従うようにしてください。ゲームデーを設定して、シミュレートされた運用イベントや失敗したインジェクションに対する応答をテストできます。

- **運用イベントと障害から学ぶ:** すべてのタイプの運用イベントと障害がキャプチャされ、確認され、その後改善のために使用されるように、プロセスを定める必要があります。定期的な機能別のオペレーション確認によって、プロセスの改善がもたらされ、高い運用性が実現されるはずです。

- **オペレーション手順を最新に保つ:** プロセスと手順ガイドは、環境とオペレーションの進展に適応されている必要があります。これには、通常のオペレーションランブック (標準的なオペレーション手順) のほか、プレイブック (予期しない運用イベントまたはプロダクションの障害に対する応答プラン) の更新も含まれます。間違いの繰り返しを防ぐために、オペレーションのガイダンスと学習をチーム間で共有する必要があります。この情報についての wiki または社内のサポート技術情報の使用を検討してください。評価すべき情報には、オペレーションメトリックス、予期しない異常、失敗したデプロイ、システム障害、および障害に対する効果的でない、または不適切な応答が含まれます。環境とオペレーションが進化するにつれて、システムとアーキテクチャの文書化も自動化を使用し、キャプチャして更新する必要があります。

定義

クラウド上で高い運用性を実現するには、次の 3 つのベストプラクティスの領域があります。

1. 準備
2. 運用
3. 応答

高い運用性を実現するには、準備が不可欠です。ワークロードを設計する際のベストプラクティスに従うことで、多くの運用上の問題を回避することができます。また、実働環境ではなく設計フェーズで実装する方がコストがかかりません。オペレーションプロセスと手順は、徹底的に計画して、テストし、確認する必要があります。ワークロードは、自動化された管理可能な方法で進化して、変更される必要があります。変更は、継続的なオペレーションに影響を与えることなく、小規模に、頻繁に、増分で行う必要があります。オペレーションチームは、運用イベントや障害に対応する準備ができていなければなりません。

ベストプラクティス

準備

高い運用性を実現するには、効果的な準備が必要です。運用チェックリストにより、本番運用のためのワークロードの準備ができていることを確認でき、十分な準備なしに意図せず実稼働に入ってしまうことを防止できます。ワークロードには、オペレーションチームが通常の日常的なタスクを実行できるように (ランブック)、また予期しない運用イベントに対応できるように (プレイブック) 参照するためのオペレーションガイダンスが必要です。プレイブック

には、応答プラン、エスカレーションパス、ステークホルダー通知が含まれている必要があります。運営上の変化を促すビジネスサイクルイベント (マーケティングイベント、フラッシュセールなど) の定期的な確認も実行する必要があります。ギャップや課題を特定し、潜在的なリスクを軽減できるように、すべてのランブックとプレイブックをテストする必要があります。失敗を追跡し、失敗から学ぶ仕組みが整備されているべきです。環境、アーキテクチャ、およびそれらのリソースの設定パラメータは、追跡およびトラブルシューティングのためにコンポーネントを容易に識別できる仕方で文書化する必要があります。設定の変更も追跡可能で自動化されている必要があります。

AWS には、運用準備をサポートするために使用できる複数の方法、サービス、機能があり、また通常の日常運用と予期しない運用イベントに備える能力があります。運用の準備には、依然として手作業の色々な仕事や見落としを確実に防ぐためにピアレビューが含まれるかもしれません。AWS CloudFormation などの AWS サービスを使用して、実稼働環境にデプロイするときに必要なすべてのリソースが含まれていること、および環境設定がテストされたベストプラクティスに基づいていることを確認でき、ヒューマンエラーを減らせます。Auto Scaling またはその他の自動スケーリングメカニズムを実装することで、ビジネス関連のイベントが運用上のニーズに影響する場合に、ワークロードが自動的に応答するようになります。AWS Config ルール機能を備えた AWS Config のようなサービスは、AWS ワークロードや環境での変更を自動的に追跡して対応するためのメカニズムを作成します。運用中や対応中の必要なときにワークロードのすべてのリソースを簡単に識別できるように、タグ付けなどの機能を使用することも重要です。

次の質問は、高い運用性に関する準備のための考慮事項に関するものです。

OPS 1. クラウドオペレーションのどのベストプラクティスを使用しているか

OPS 2. ワークロードの設定管理はどのように行っているか

手順が変わった場合、ドキュメントが古いものにならないようにしてください。また、それがすべてを網羅していることを確認してください。アプリケーション設計、環境設定、リソース設定、対応プラン、および軽減プランがなければ、ドキュメントは完全とは言えません。ドキュメントが定期的に更新されてテストされなければ、予期しない運用イベントが発生した時に役に立ちません。運用前にワークロードを確認しないと、検出されない問題が発生した場合に運用に影響が出ます。運用イベントが発生した時にリソースが文書化されていない場合、正しいリソースを特定し対応方法を決定することはより困難になります。

運用

運用は標準化され、日常的に管理可能なものである必要があります。自動化、小規模で頻繁な変更、定期的な品質保証テスト、変更を追跡、監査、ロールバック、および確認するための定義されたメカニズムに重点を置くべきです。変更は、大規模なものや頻度が低いものであってはならず、スケジュールされたダウンタイムや手動での実行も不要であるべきです。ワークロードの主要な運用指標に基づく幅広いログとメトリクスを収集して確認し、継続的な運用を確実にする必要があります。

AWS では、継続的な統合/継続的デプロイメント (CI / CD) パイプライン (ソースコードリポジトリ、ビルドシステム、デプロイメントおよびテスト自動化など) を設定できます。手動であれ自動であれ、リリース管理プロセスはテストされ、

小さい差分変更と追跡されたバージョンに基づいていなければなりません。運用上の問題を引き起こすことなく、運用上の問題を引き起こす変更を元に戻すことができます。変更の品質保証には、Blue/Green、Canary、A/B テストなどのリスク低減戦略が含まれている必要があります。運用チェックリストは、本番環境のワークロードが準備できていることを評価するために使用する必要があります。一元管理されたモニタリングおよびアラート用にログを集計します。アラートが通知やエスカレーションなどの自動応答をトリガーするようにします。また、障害だけでなく、異常も監視するよう設計します。

次の質問は、高い運用性のためにワークロードをどのように運用するかに関するものです。

OPS 3. 変更の影響を最小限に抑えながら、ワークロードをどのように進化させるか

OPS 4. ワークロードが想定通りに動作していることを確認するために、どのように監視するか

ルーチンオペレーションや計画外のイベントへの応答を自動化する必要があります。デプロイメント、リリース管理、変更、およびロールバックのための手動プロセスは避ける必要があります。リリースは頻度の低い大規模バッチであってはなりません。大規模な変更ではロールバックがより困難になり、ロールバックプランを作成できないか、障害の影響を緩和できないため、オペレーションの持続性が損なわれます。モニタリングをビジネスニーズに合わせることで、ビジネスの継続性を維持する上で効果的な応答にすることができます。手動による応答で場当たり的かつ一元化されていない監視は、予期しないイベントでのオペレーションにより大きな影響を与えます。

応答

予期せぬ運用イベントへの応答は自動化されるべきです。これはアラートのためだけでなく、緩和、修復、ロールバック、復旧にも役立ちます。アラートはタイムリーである必要があり、また応答が運用イベントの影響を緩和するのに十分でない場合、エスカレーションを呼び出す必要があります。失敗したデプロイメントを自動的にロールバックするには、品質保証のメカニズムを整えておく必要があります。応答は、ステークホルダー、エスカレーションプロセスおよび手順を含む事前定義されたプレイブックに従う必要があります。エスカレーションパスを定義し、機能的および階層的エスカレーション機能の両方を含める必要があります。階層的エスカレーションは自動化され、エスカレートされた優先度はステークホルダーへの通知をもたらすべきです。

AWS には、予期せぬ運用イベントや自動化された応答に応じて、適切なアラートと通知の両方を確実に行うためのメカニズムがいくつかあります。ワークロードを一元的に監視し、主要な運用指標に関連するすべての利用可能なログとメトリクスに基づいて、効果的なアラートと通知を作成するためのツールも必要です。これには、サービスやコンポーネントの障害だけでなく、範囲外の異常に関するアラートや通知も含まれます。応答は、依存する AWS 環境とサービス、およびワークロードのアプリケーションの状態に対して有効にする必要があります。根本原因の分析 (RCA) は、運用イベントの後に実行し、アーキテクチャと対応プランの両方を改善するために使用します。

次の質問は、高い運用性のためのイベントへの対応に関するものです。

OPS 5. 計画外の運用イベントにどのように対応するか

OPS 6. 計画外の運用イベントに対応する場合、エスカレーションはどのように管理されるか

対応プランが場当たり的に行われ、定義されていない場合、結果は予測不可能であり、しばしばイベントに悪影響を与えます。期限切れのプレイブックに基づいた対応、または問題が発生しプレイブックにアクセスできない場合も、予測できない結果が生じます。もしプロセスがイベントを確認するのにふさわしくなく、そのため将来の運用イベントがより防止が困難になる場合は、同じようなインパクトがあります。

関連する AWS サービス

高い運用性を実現するために使用できる主なサービスが 2 つあります。AWS CloudFormation を使用すれば、ベストプラクティスに基づいてテンプレートを作成し、リソースを整然とした予測可能な方法でプロビジョニングできます。Amazon CloudWatch は、メトリクスの監視、ログの収集、アラートの生成、および応答のトリガーに使用できます。高い運用性の 3 つのエリアをサポートするその他のサービスと機能は、次のとおりです。

準備: AWS Config は AWS リソースおよび設定の詳細インベントリーを提供し、継続して設定の変更を記録します。AWS Service Catalog は、ベストプラクティスに合わせて標準化されたサービス提供を作成するのに役立ちます。ワークロードを、Auto Scaling や Amazon SQS などのサービスを利用した自動化を行うように設計すれば、予期しない運用イベントが発生した場合でも継続的な運用を確実にできます。

運用: AWS ワークロードへのコードの変更を管理し自動化するために、AWS CodeCommit、AWS CodeDeploy、および AWS CodePipeline を使用できます。オペレーションの変更を自動化するためには、AWS SDK またはサードパーティライブラリを使用します。AWS 環境に対して加えられた変更を監査し、追跡するには AWS CloudTrail を使用できます。

応答: Amazon CloudWatch サービスの機能を利用して、効果的かつ自動化された応答を実現できます。Amazon CloudWatch アラームを使用してアラートと通知のしきい値を設定でき、Amazon CloudWatch イベントは通知と自動化された応答をトリガーできます。

リソース

高い運用性に関連するベストプラクティスの詳細については、以下のリソースを参照してください。

動画

- AWS re:Invent 2015 – Amazon での DevOps
- AWS re:Invent 2015 - インフラストラクチャのスケーリング操作
- AWS Summit 2016 - DevOps、AWS における継続的統合とデプロイメント

ドキュメントおよびブログ

- AWS と DevOps
- 継続的な統合とは
- 継続的な配信とは
- AWS DevOps ブログ

ホワイトペーパー

- AWS クラウド導入フレームワーク - 運用の視点
- AWS の DevOps 入門
- AWS 運用チェックリスト

AWS サポート

- AWS クラウドオペレーションレビュー
- AWS プレミアムサポート
- AWS Trusted Advisor

まとめ

AWS による優れた設計のフレームワークは、クラウド上の信頼性、セキュリティ、効率、コスト効果が高いシステムを設計および運用するための 5 つの柱についてアーキテクチャのベストプラクティスを提供します。このフレームワークには、既存のアーキテクチャまたは提案されたアーキテクチャを確認するための質問と、それぞれの柱についての AWS ベストプラクティスが用意されています。アーキテクチャでフレームワークを使用すると、安定した効率的なシステムを構築することができ、より機能的な要件に注目することができます。

寄稿者

本書の執筆に当たり、次の人物および組織が寄稿しました。

- Philip Fitzsimons、Well-Architected シニアマネージャー、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アーキテクト、アマゾン ウェブ サービス
- John Steele、シニアテクニカルアカウントマネージャー、アマゾン ウェブ サービス
- Callum Hughes、ソリューションアーキテクト、アマゾン ウェブ サービス

ドキュメント履歴

2015 年 11 月 20 日: 最新の Amazon CloudWatch ログ情報で付録を更新しました。

2016 年 11 月 20 日: フレームワークを更新しました。卓越したオペレーションの柱を含め、他の柱を改訂、更新して重複を削減しました。また、数千のお客様とともにレビューを行った結果から学んだこと組み込みました。

付録: Well-Architected の質問、回答、ベストプラクティス

この付録には、ベストプラクティスを含む Well-Architected に関する質問と回答の完全な一覧が、柱ごとに整理されて含まれています。

セキュリティの柱

アイデンティティ管理とアクセス管理

SEC 1. AWS ルートアカウントの認証情報へのアクセスと使用をどのように保護していますか。

AWS ルートアカウントの認証情報は、他のオペレーティングシステムのルートまたはローカル管理者と似ており、非常に注意して使用する必要があります。最新のベストプラクティスでは、AWS Identity and Access Management (IAM) ユーザーを作成し、それらのユーザーを管理者グループに関連付けて、IAM ユーザーを使ってアカウントを管理します。AWS のルートアカウントは、API キーを保持せず、強力なパスワードがあり、ハードウェアの多要素認証 (MFA) デバイスと関連付けられている必要があります。これにより、ルートアイデンティティの使用を AWS マネジメントコンソール経由のみに強制し、アプリケーションプログラミングインターフェイス (API) コールにルートアカウントの使用を許可しません。注意：一部のリセラーやリージョンでは、AWS ルートアカウントのクレデンシャルの配布やサポートは行われません。

ベストプラクティス:

- **ルートの MFA および最低限の使用** AWS ルートアカウントの認証情報は、必要最小限の作業にのみ使用されている。
- **ルートを使用しない**

SEC 2. AWS マネジメントコンソールと API へのユーザーによるアクセスを制御するために、システムユーザーのロールと責任をどのように定義していますか。

お客様にとって最新のベストプラクティスは、ユーザーグループを作成してシステムユーザーの定義されたロールと責任を分離することです。ユーザーグループは複数の異なる技術で定義できます。Identity and Access Management (IAM) グループ、クロスアカウントアクセス用の IAM ロール、Security Assertion Markup Language (SAML) 統合によるウェブ認証 (例: Active Directory でのロール定義)、または SAML や AWS Security Token Service (STS) により統合されるサードパーティーソリューションの使用 (Okta、Ping Identity、別のカスタムテクニック) などです。共有アカウントは使用しないことを強くお勧めします。

ベストプラクティス:

- **従業員のライフサイクルが管理されている** 従業員のライフサイクルポリシーが定義、実施されている。
- **最小限の権限** ユーザー、グループ、およびロールが明確に定義され、ビジネス要件を達成するために必要な最小の権限のみが付与されている。

SEC 3. AWS リソースへの自動化されたアクセスをどのように制限していますか (アプリケーション、スクリプト、サードパーティーのツールやサービスなど)。

ユーザーグループは人について作成されるため、システムによるアクセスも同様の方法で定義する必要があります。Amazon EC2 インスタンスの場合、これらのグループは EC2 用の IAM ロールと呼ばれます。最新のベストプラクティスは、EC2 用の IAM ロールと AWS SDK または CLI を使用することです。これらには、EC2 認証情報のための IAM ロールを取得する機能が組み込まれています。従来、ユーザーの認証情報は EC2 インスタンス内に設定されていましたが、スクリプトやソースコードへの認証情報のハードコーディングはお勧めできません。

ベストプラクティス:

- **自動化されたアクセスに使用される静的認証情報** これらが安全に保管されている。
- **自動化されたアクセス用の動的認証** インスタンスプロファイルまたは Amazon STS を使用して管理する。

発見的統制

SEC 4. ログをどのように取得して分析していますか。

ログの取得は、パフォーマンスからセキュリティインシデントまですべてを調査するために重要です。最新のベストプラクティスは、ビジネスニーズに基づいて、ログを定期的にソースから直接ログ処理システム (CloudWatch Logs、Splunk、Papertrail など) に移動するか、後で処理するために Amazon S3 バケットに保存することです。ログの一般的なソースには、AWS API およびユーザー関連ログ (AWS CloudTrail など)、AWS サービス固有のログ (Amazon S3、Amazon CloudFront など)、オペレーティングシステム生成ログ、およびサードパーティー製アプリケーション固有のログがあります。Amazon CloudWatch ログを使用して、Amazon EC2 インスタンス、AWS CloudTrail、およびその他のリソースをモニタリング、保存、アクセスすることができます。

ベストプラクティス:

- **アクティビティが Amazon CloudWatch ログ、イベント、VPC フローログ、ELB ログ、S3 バケットログなどを適切にモニタリングしている。**
- **AWS Cloud Trail が有効化されている**
- **オペレーティングシステムまたはアプリケーションのログをモニタリングしている**

インフラストラクチャの保護

SEC 5. ネットワークおよびホストレベルの境界保護をどのように実施していますか。

オンプレミスデータセンターで、DMZ 手法ではファイアウォールを使用して異なるシステムを信頼済みゾーンと信頼されていないゾーンに分離します。AWS で、ステートフルなファイアウォールとステートレスのファイアウォールの両方が使用されます。ステートフルなファイアウォールはセキュリティグループと呼ばれ、ステートレスなファイアウォールは、Amazon Virtual Private Cloud (VPC) のサブネットを保護するネットワークアクセスコントロールリスト (ACL) と呼ばれます。最新のベストプラクティスは、VPC でシステムを実行し、セキュリティグループでロールベースのセキュリティ (ウェブ層、アプリケーション層など) を定義して、ネットワーク ACL で場所ベースのセキュリティ (アベイラビリティーゾーンごとに 1 つのサブネットでの Elastic Load Balancing 層、アベイラビリティーゾーンごとに別のサブネットでのウェブ層など) を定義することです。

ベストプラクティス:

- **VPC でネットワークトラフィックが制御されている** たとえば、ファイアウォール、セキュリティグループ、NACLS、踏み台ホストなどを使用する。
- **境界でネットワークトラフィックが制御されている** たとえば、AWS WAF、ホストベースのファイアウォール、セキュリティグループ、NACLS などを使用する。

SEC 6. AWS サービスレベルセキュリティ機能をどのように活用していますか。

AWS のサービスで追加のセキュリティ機能が提供されている場合があります (Amazon S3 バケットポリシー、Amazon SQS、Amazon DynamoDB、KMS キーポリシーなど)。

ベストプラクティス:

- **追加機能の適切な使用**

SEC 7. Amazon EC2 インスタンス上のオペレーティングシステムの整合性はどのように保護していますか。

別の従来の制御として、オペレーティングシステムの整合性の保護があります。これは、従来のホストベースの手法 (OSSEC、Tripwire、Trend Micro Deep Security など) を使用して Amazon EC2 で簡単に行うことができます。

ベストプラクティス:

- **ファイルの整合性** EC2 インスタンス用にファイル整合性のコントロールが使用されている。
- **EC2 侵入検出** EC2 インスタンス用にホストベースの侵入検出コントロールが使用されている。
- **AWS Marketplace またはパートナーソリューション**
  AWS Marketplace または APN パートナーからのソリューション。
- **設定管理ツール** カスタム Amazon Machine Image (AMI) またはデフォルトでセキュリティ保護される設定管理ツール (Puppet または Chef など) を使用している。

データ保護

SEC 8. データをどのように分類していますか。

データを分類することで、レベルや重要度に基づいて構造化されたデータをカテゴリ分けできます。これには、使用できるデータ型、データの場所、アクセスレベル、データの保護 (たとえば暗号化かアクセス制御か) も含まれます。

ベストプラクティス:

- **データ分類スキーマの使用**
- **すべてのデータを機密として扱う**

SEC 9. 保管時のデータをどのように暗号化、保護していますか。

伝統的なセキュリティ統制では、保管時のデータを暗号化します。AWS は、クライアント側 (SDK サポート、オペレーティングシステムサポート、Windows Bitlocker、dm-crypt、Trend Micro SafeNet など) およびサーバー側 (Amazon S3 など) の両方を使用してこれをサポートしています。サーバー側の暗号化 (SSE) や Amazon Elastic Block Store 暗号化ボリュームなどを使用することもできます。

ベストプラクティス:

- **不要** 保管時のデータ暗号化は不要
- **保存時の暗号化**

SEC 10. キーと認証情報をどのように管理していますか。

キーは、秘密情報であるため保護される必要があり、適切なローテーションポリシーを定義して管理する必要があります。ベストプラクティスは、これらの秘密情報を管理スクリプトやアプリケーションにハードコーディングしないことです。しかし、これはしばしば行われています。

ベストプラクティス:

- **AWS CloudHSM** AWS CloudHSM を使用する。
- **AWS サービスのコントロール** 保管時のデータを、AWS サービス固有の統制 (Amazon S3 SSE、Amazon EBS 暗号化ボリューム、Amazon Relational Database Service (RDS) Transparent Data Encryption (TDE) など) を使用して暗号化できる。
- **クライアント側を使用** 保管時のデータを、クライアント側の手法を使用して暗号化する。
- **AWS Marketplace またはパートナーソリューション**
  AWS Marketplace または APN パートナーからのソリューション。
  (SafeNet、TrendMicro など)。

SEC 11. 伝送中のデータをどのように暗号化、保護していますか。

暗号化を使用することによって、伝送中のデータを保護するのがベストプラクティスです。AWS は、サービス API に対する暗号化エンドポイントの使用をサポートします。さらに、お客様は Amazon EC2 インスタンス内でさまざまな技術を使用できます。

ベストプラクティス:

- **必須でない** 伝送中のデータは暗号化を要求されない。
- **暗号化された通信** 通信には TLS またはそれと同等のものが適切に使用されている。

インシデントへの対応

SEC 12. インシデント対応が適切であることをどのように確認しますか。

セキュリティインシデントが発生する前にツールを導入してアクセスし、定期的にインシデント対応を実施することで、調査と復旧をタイムリーに行えるようにアーキテクチャを更新できるようになります。

ベストプラクティス:

- **事前にプロビジョニングされたアクセス** 情報セキュリティに正しいアクセス権限がある、またはすばやくアクセスできる手段がある。インシデントに対して適切に対応できるように、事前にプロビジョニングされている必要があります。
- **事前にデプロイされたツール** インシデントに対して適切に対応できるように、情報セキュリティで AWS に正しいツールが事前にデプロイされている
- **本稼働以外でのゲームデー** 本稼働環境以外でインシデント対応シミュレーションが定期的に実行され、そこで学んだ教訓がアーキテクチャや運用に生かされている。
- **本稼働でのゲームデー** 本稼働環境でインシデント対応シミュレーションが定期的に実行され、そこで学んだ教訓がアーキテクチャや運用に生かされている。

信頼性の柱

基礎

REL 1. アカウントの AWS サービス制限はどのように管理していますか。

AWS アカウントは、新しいユーザーが誤って必要以上のリソースをプロビジョニングしないように、デフォルトのサービス制限でプロビジョニングされます。AWS のお客様は AWS サービスに関するニーズを評価し、使用している各リージョンについて制限の適切な変更をリクエストする必要があります。

ベストプラクティス:

- **制限のモニタリングと管理** AWS の使用量の可能性について評価し、リージョンの制限を適切に引き上げ、使用量の計画的な成長を可能にしている。
- **自動化されたモニタリングのセットアップ** SDK などのツールを実装して、しきい値に近づいたときに警告を発する。
- **サービスの固定の上限に注意** 変更できないサービスの上限に注意し、それらを考慮した設計を行っている。
- **フェイルオーバーに対応するために、サービスの上限とお客様の最大使用量の間に十分な余裕を持たせてある**
- **サービスの上限が、関連するすべてのアカウントとリージョンを含めて考慮されている**

REL 2. AWS でのネットワークトポロジをどのように計画していますか。

アプリケーションは、EC2 Classic、VPC、デフォルト VPC の中のひとつ、または複数の環境で実行することができます。システム接続性、Elastic IP/パブリック IP アドレスの管理、VPC/プライベート アドレスの管理、名前解決などのネットワークの考慮点は、クラウドのリソースを利用する上で基礎となるものです。よく計画され、文書化されたデプロイは、重複や競合のリスクを減らすために必要不可欠です。

ベストプラクティス:

- **データセンター内の接続性の考慮は不要**
- **AWS とオンプレミス環境 (ある場合) 間の可用性の高い接続** 複数の DX 接続、複数の VPN トンネル、AWS Marketplace アプライアンスの適切な利用。
- **負荷の高いユーザーのための可用性の高い接続** 可用性の高い負荷分散またはプロキシ、DNS ベースのソリューション、AWS Marketplace アプライアンス等の利用。
- **重複のないプライベート IP アドレス範囲** 仮想プライベートクラウドの IP アドレス範囲とサブネットが互いに重複せず、他のクラウド環境やオンプレミス環境とも重複しない。
- **IP サブネットの割り当て** 個別の Amazon VPC の IP アドレス範囲は、将来の拡張やアベイラビリティーゾーン間でのサブネットに対する IP アドレス割り当てを考慮し、アプリケーションの要件を満たすために十分な大きさがある。

変更管理

REL 3. システムは需要の変化にどのように対応できますか。

スケーラブルなシステムは、いつの時点でも最新の需要に合わせて、リソースを自動的に追加、削除する伸縮性を備えています。

ベストプラクティス:

- **自動スケーリング** Amazon S3、Amazon CloudFront、Auto Scaling、Amazon DynamoDB、AWS Elastic Beanstalk など、自動スケーラブルサービスを使用する。
- **ロードがテスト済み** ロードテスト手法を採用し、規模の拡大や縮小がアプリケーションの要件に合うかどうか測定する。

REL 4. AWS リソースをどのようにモニタリングしていますか。

ログとメトリクスは、アプリケーションの健全性についての洞察を得るための強力なツールです。ログとメトリックスをモニタリングし、しきい値を超えるか重要なイベントが発生したときに通知を送信するようシステムを設定できます。低パフォーマンスのしきい値を超えるか、障害が発生した場合、システムが自動的に自己修復するか、それに応じてスケールするよう構築されていることが理想的です。

ベストプラクティス:

- **モニタリング** Amazon CloudWatch またはサードパーティー製ツールでアプリケーションをモニタリングする。
- **通知** 重要なイベントが発生した場合に通知を受け取るように計画します。
- **自動応答** 自動化を使用して、障害が検出されたときに、失敗したコンポーネントを置き換えるなどのアクションを実行します。

REL 5. 変更をどのように実行していますか。

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

ベストプラクティス:

- **自動化** デプロイおよびパッチ適用を自動化する。

障害管理

REL 6. データをどのようにバックアップしていますか。

データ、アプリケーション、および運用環境 (アプリケーションで設定されたオペレーティングシステムと定義される) をバックアップし、平均修復時間 (MTTR) および目標復旧時点 (RPO) の要件を満たします。

ベストプラクティス:

- **自動バックアップ** AWS の機能、AWS Marketplace ソリューション、またはサードパーティー製ソフトウェアを使用してバックアップを自動化します。
- **定期的な復旧テスト** 復旧テストを通じて、バックアップの実装が RTO および RPO を満たすことを確認する。

REL 7. システムはコンポーネントの障害にどのように対応しますか。

アプリケーションが、高可用性と短い平均修復時間 (MTTR) を満たす、暗黙または明示の要件がありますか。そのような要件がある場合は、弾力性を持つようアプリケーションを設計し、機能停止に対処できるようにアプリケーションを配置します。より高いレベルの可用性を達成するため、アプリケーションは、複数の物理的なロケーションに分散して配置される必要があります。弾力性を持つように個別の Layer (ウェブサーバー、データベースなど) を構築します。これにはイベントの重要な中断と障害のモニタリング、自己修復、および通知が含まれます。

ベストプラクティス:

- **マルチ AZ/リージョン** 複数のアベイラビリティーゾーン/リージョンにまたがってアプリケーション負荷を分散する (例: DNS、ELB、Application Load Balancer、API Gateway)。
- **疎結合依存関係** たとえば、キューシステム、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を使用する。
- **グレースフルデグレデーション** コンポーネントの依存関係に異常がある場合、コンポーネントそのものは異常とはレポートされない。機能が低下した状態でリクエストの処理を続行できます。
- **自動修復** 自動的に障害を検出し、復旧対応を実行する。システムの状態を継続してモニタリングし、重要なイベントについて通知を受け取るように計画します。

REL 8. 弾力性をどのようにテストしていますか。

弾力性をテストすると、本稼働でのみ表面化する潜伏バグを見つけることがあります。定期的にゲームデーで手順を実践することで、組織で手順をスムーズに実行できるようになります。

ベストプラクティス:

- **プレイブック** 障害シナリオのプレイブックが用意されている。
- **障害の生成** 定期的に障害をテストして (Chaos Monkey を使用するなど)、障害経路の詳細を把握している。
- **ゲームデーのスケジュール**
- **根本原因の分析 (RCA)** 重要なイベントに基づいてシステム障害のレビューを行い、アーキテクチャを評価する。

REL 9. 災害対策についてどのように計画していますか。

データ復旧 (DR) は、バックアップからデータを復元する際に不可欠です。このデータに対して、RTO および RPO の目標と一致するように目標、リソース、場所、および機能が定義され、実行される必要があります。

ベストプラクティス:

- **目標の定義** RTO および RPO を定義します。
- **災害対策** DR 戦略を確立します。

- **構成情報の乖離** Amazon Machine Images (AMI) およびシステム設定状態が、DR サイト/リージョンで最新であることを確認している。
- **DR のテストと確認** DR サイトへのフェイルオーバーを定期的にテストして、RTO と RPO が満たされていることを確認している。
- **自動復旧の実装** AWS またはサードパーティー製ツール (またはその両方) を使用してシステム復旧を自動化します。

パフォーマンスの柱

選択

PERF 1. 最良の実行アーキテクチャをどのように選択していますか。

特定のシステムに向けた適切なソリューションは、ワークロードの種類によって異なり、複数の手法を組み合わせたものになります。優れた設計のシステムでは、複数の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を向上させます。

ベストプラクティス:

- **ベンチマーク** AWS で既知のワークロードのロードテストを行い、それを使用して最適な選択を行っている。
- **ロードテスト** さまざまなリソースタイプとサイズを使用して AWS でシステムの最新バージョンをデプロイし、モニタリングを使用してパフォーマンスメトリクスをキャプチャしてから、パフォーマンス/コストの計算に基づいて選択を行っている。

PERF 2. コンピューティングソリューションをどのように選択しましたか。

特定のシステムに向けた適切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設計、使用パターン、構成設定によって異なります。アーキテクチャでは、パフォーマンスを向上させるために、さまざまなコンポーネントに対して異なるコンピューティングソリューションを使用し、異なる機能を有効にします。システムに対して適切でないコンピューティングソリューションや機能を選択すると、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:

- **オプションの検討** 最良のパフォーマンスを達成するために、インスタンス、コンテナ、関数の使用方法について異なるオプションを検討している。
- **インスタンス設定のオプション** インスタンスを使用している場合、ファミリー、インスタンスサイズ、機能 (GPU、I/O、バースト可能) などの設定オプションを検討している。
- **コンテナ設定のオプション** コンテナを使用している場合、コンテナのメモリー、CPU、テナンシー設定などの設定オプションを検討している。
- **関数設定のオプション** 関数を使用している場合、メモリ、ランタイム、ステートなどの設定オプションを検討している。
- **伸縮性** 弾力性を使用して (Auto Scaling、Amazon EC2 Container Service (ECS)、AWS Lambda)、需要の変化に対応している。

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。

特定のシステムの最適なストレージソリューションは、アクセス方法の種類 (ブロック、ファイル、またはオブジェクト)、アクセスのパターン (ランダムまたはシーケンシャル)、必要なスループット、アクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、および可用性と耐久性の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を向上させます。

ベストプラクティス:

- **特性の検討** 使用する必要があるサービス (Amazon S3、Amazon EBS、Amazon Elastic File System (EFS)、EC2 インスタンスストア) を選択するために必要な別の特性 (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、データの永続性など) を検討している。
- **設定オプションの検討** PIOPS、SSD、マグネティック、および Amazon S3 Transfer Acceleration などの設定オプションを検討している。
- **アクセスパターンの検討** アクセスパターン (ストライピング、キー分散、パーティショニングなど) に基づいてストレージシステムの使用方法を最適化している。

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。

特定のシステムに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、拡張性、およびクエリ機能についての要件によって異なります。多くのシステムでは、パフォーマンスを向上させるために、さまざまなサブシステムに対して異なるデータベースソリューションを使用し、異なる機能を有効化しています。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可能性があります。

ベストプラクティス:

- **特性の検討** 最もパフォーマンスのよいデータベースの使用アプローチ (リレーショナル、No-SQL、ウェアハウス、インメモリ) を選択できるように、別の特性 (可用性、整合性、分断耐性、レイテンシー、耐久性、拡張性、クエリ機能など) を検討している。
- **設定オプションの検討** ストレージの最適化、データベースレベルの設定、メモリ、キャッシュなどの設定オプションを検討している。
- **アクセスパターンの検討** アクセスパターン (インデックス、キー分散、パーティション、水平スケーリングなど) に基づいてデータベースシステムの使用方法を最適化している。
- **他のアプローチの検討** 検索インデックス、データウェアハウス、ビッグデータなどのクエリ可能データの提供で他のアプローチを検討した。

PERF 5. ネットワーキングソリューションをどのように設定していますか。

特定のシステムに最適なネットワークソリューションは、レイテンシー、スループット要件などによって異なります。ユーザーやオンプレミスリソースなどの物理的な制約は、エッジの手法やリソースの配置を使用してオフセットすることができる場所のオプションを実行します。

ベストプラクティス:

- **場所の検討**ネットワークレイテンシーを低減するために場所のオプション (リージョン、アベイラビリティーゾーン、プレイスメントグループ、エッジなど) を検討する。

- **製品機能の検討** ネットワークトラフィックを最適化するために製品の機能 (EC2 インスタンスネットワーク性能、超高ネットワークインスタンスタイプ、Amazon EBS 最適化インスタンス、Amazon S3 Transfer Acceleration、Dynamic Amazon CloudFront など) を検討する。
- **ネットワーキング機能の検討** ネットワーク距離またはジッターを低減するために、ネットワーキング機能 (Amazon Route 53 レイテンシールーティング、Amazon VPC エンドポイント、AWS Direct Connect など) を検討する。
- **適切な NACLS** ネットワークスループットを維持するために、最低限のセットの NACLS を使用する。
- **暗号化のオフロードの検討** 暗号化停止 (TLS) をオフロードするために、負荷分散の使用を検討する。
- **プロトコルの検討** ネットワークパフォーマンスを最適化するために必要なプロトコルについて検討する。

確認

PERF 6. 新しいリソースタイプや機能の提供が開始される中で、どのようにして最適なリソースタイプを使い続けていることを担保していますか。

ソリューションの設計には、選択できる一定のセットのオプションがあります。しかし、時間がたつと、アーキテクチャのパフォーマンスを向上させる可能性のある新しいテクノロジとアプローチが利用可能になります。

ベストプラクティス:

- **レビュー** 新しいリソースタイプおよびサイズをレビューするプロセスがある。パフォーマンステストを再実行して、パフォーマンス効果に向上が見られるかどうかを評価します。

モニタリング

PERF 7. 起動後のリソースが期待どおりの性能を出すように、リソースをどのようにモニタリングしていますか。

内部または外部 (またはその両方) の要因により、時間とともにシステムパフォーマンスが低下する場合があります。システムのパフォーマンスをモニタリングすることで、パフォーマンスの低下を検知し、内部または外部の要因 (オペレーティングシステムまたはアプリケーションのロードなど) を修正することができます。

ベストプラクティス:

- **モニタリング** Amazon CloudWatch、サードパーティー製、またはカスタムのモニタリングツールを使用してパフォーマンスをモニタリングする。
- **アラームベースの通知** メトリクスが安全な境界を超えた場合に、モニタリングシステムから自動的なアラートを受信する。
- **トリガーベースのアクション** 自動化されたアクションで問題を修正またはエスカレーションするアラームを設定する。

トレードオフ

PERF 8. パフォーマンスを向上させるためにトレードオフをどのように使用しますか。

ソリューションを構築するときに、積極的にトレードオフを考慮することで、最適なアプローチを選択できます。整合性、耐久性、容量を、時間およびレイテンシーと引き換えにすることで、より高いパフォーマンスを達成できることがよくあります。

ベストプラクティス:

- **サービスの検討** Amazon ElastiCache、Amazon CloudFront、AWS Snowball など、パフォーマンスを向上させるサービスを使用する。
- **パターンの検討** キャッシング、リードレプリカ、シャーディング、圧縮、バッファリングなど、パフォーマンスを向上させるパターンを使用する。

コストの最適化の柱

コスト効果が高いリソース

COST 1. ソリューションに AWS サービスを選択する際のコストを検討していますか。

Amazon EC2、Amazon EBS、Amazon S3 などは AWS のサービスの基本的な「ビルディングブロック」です。Amazon RDS や Amazon DynamoDB などのマネージドサービスは、「ハイレベル」の AWS サービスです。適切な構成要素およびマネージドサービスを選択することで、コストに対してアーキテクチャを最適化できます。たとえば、これらのマネージドサービスを使用すると、管理オーバーヘッドまたは運用オーバーヘッドの大部分を減らすか排除することができ、アプリケーションやビジネス関連のアクティビティに労力を向けることができます。

ベストプラクティス:

- **コスト削減サービスの選択** サービスを分析して、コスト削減に使用できるサービスを確認する。
- **ライセンス費用の最適化**
- **サーバーレスおよびコンテナベースの手法を使用した最適化** AWS Lambda、Amazon S3 ウェブサイト、Amazon DynamoDB、および Amazon ECS を使用したコストの削減。
- **適切なストレージソリューションを使用した最適化** 使用パターンに基づいたもっともコスト効果の高いストレージソリューションを使用する

(Amazon EBS コールドストレージ、Amazon S3 Standard-Infrequent Access、Amazon Glacier など)。

- **適切なデータベースを使用した最適化** 該当する場合は、Amazon Relational Database Service (RDS) (Postgres、MySQL、SQL Server、Oracle Server) または Amazon DynamoDB (またはその他のキーと値のストア、NoSQL 代替策) を使用する。
- **その他のアプリケーションレベルのサービスを使用した最適化** 該当する場合は、Amazon Simple Queue Service (SQS)、Amazon Simple Notification Service (SNS)、Amazon Simple Email Service (SES) を使用する。

COST 2. コスト目標を達成するためにリソースのサイズを決定しましたか。

タスクに対して適切な AWS リソースサイズを選択していることを確認します。AWS では、ベンチマーク評価の使用によって、選択したタイプがそのワークロードに対して最適化されていることを確認します。

ベストプラクティス:

- **メトリクス駆動型のリソースサイズ設定** パフォーマンスメトリクスを活用して正しいサイズ/タイプを選択し、コストを最適化する。Amazon EC2、Amazon DynamoDB、Amazon EBS (プロビジョンド IOPS)、Amazon RDS、Amazon EMR、ネットワーキングなどのようなサービスのスループット、サイジング、ストレージについて適切にプロビジョニングを行います。

COST 3. コスト目標を達成するために適切な料金モデルを選択していますか。

ワークロードで費用を最小化するために最適な料金モデルを使用します。デプロイを最適化すると、すべてオンデマンドインスタンス、オンデマンドとリザーブドインスタンスの組み合わせとするか、可能な場合はスポットインスタンスを含めることもできます。

ベストプラクティス:

- **キャパシティーの予約および取引のコミット** 定期的に使用料を分析し、それに応じてリザーブドインスタンスを購入する (Amazon EC2、Amazon DynamoDB、Amazon S3、Amazon CloudFront など)。
- **スポット** 選択ワークロード (バッチ、EMR など) 向けのスポットインスタンス (スポットブロック、フリートなど) を使用する。
- **リージョンコストの検討** リージョンの選択でコストを考慮する。

供給と需要の一致

COST 4. キャパシティーが必要量を満たしているが大幅に超えていないことをどのように実現していますか。

料金とパフォーマンスの点でバランスが取れたアーキテクチャを構築するには、支払ったすべてのものが使用されるようにし、著しく使用率が低いインスタンスを避けます。いずれかの方向に偏った使用率のメトリックスは、運用コスト (過度の使用率によるパフォーマンスの低下) または無駄な AWS 支出 (オーバープロビジョニングによる) によるビジネスへの悪影響につながります。

ベストプラクティス:

- **デマンドベースの手法** Auto Scaling を使用して変化する需要に対応する。
- **バッファベースの手法** 作業をバッファして (Amazon Kinesis or Amazon Simple Queue Service (SQS) を使用するなど)、処理に十分なキャパシティーができるまで作業を保留する。
- **時間ベースの手法** 時間ベースの手法の例としては、Follow The Sun、週末に開発およびテストインスタンスをオフにする、四半期または年間スケジュールに従う (ブラックフライデーなど) など。

費用の把握

COST 5. アーキテクチャを設計するときに、データ転送料金について考慮しましたか。

データ転送料金をモニタリングし、それらのコストの一部を軽減できるようなアーキテクチャの決定を行います。たとえば、コンテンツプロバイダーがエンドユーザーに対して Amazon S3 バケットから直接コンテンツを提供している場合、コンテンツを Amazon CloudFront コンテンツ配信ネットワーク (CDN) にプッシュすれば、大幅なコスト削減が可能になる可能性があります。小規模でも効果的なアーキテクチャ変更によって、運用コストを大幅に削減できる場合があることを覚えておいてください。

ベストプラクティス:

- **最適化** データ転送を最適化するよう設計する (アプリケーション設計、WAN アクセラレーション、マルチ AZ、リージョン選択など)。
- **CDN** 該当する場合、CDN を使用する。

- **AWS Direct Connect** 状況を分析し、該当する場合は AWS Direct Connect を使用する。

COST 6. 使用状況と支出をどのようにモニタリングしていますか。

コストをモニタリング、管理、適切に割り当てるためのポリシーと手順を確立します。AWS 提供のツールを活用して、だれが何をどのぐらいのコストで使用しているのかを明らかにします。これにより、ビジネスニーズとチームのオペレーションについてより深く理解することができます。

ベストプラクティス:

- **すべてのリソースにタグ付け** タグ付け可能なリソースすべてにタグ付けして、請求の変化をインフラストラクチャと使用量の変更に関連付けられるようにする。
- **請求およびコスト管理レポートツールの活用** 詳細な請求レポートまたはコストエクスプローラーをロードし、解釈できるための標準プロセスがある。該当する場合は Amazon CloudWatch またはサードパーティープロバイダー (例:Cloudability、CloudCheckr、CloudHealth) を使用して定期的に使用量と費用をモニタリングします。
- **通知** 費用が明確に定義された限度を超える場合に、チームの主要メンバーに通知する。
- **財務駆動型のチャージバック/ショーバック方法** これを使用して、インスタンスとリソースをコストセンターに割り当てる (タグ付けの使用など)。

COST 7. 必要なくなったリソースは廃棄していますか。または、一時的に必要ないリソースを停止していますか。

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、適切な場合は必要なプロセス変更または機能強化を識別できるようにします。AWS サポートと協力して、お客様のワークロードに合わせてプロジェクトを最適化する方法を提案します。たとえば、どのような場合に Auto Scaling、AWS OpsWorks、AWS Data Pipeline、その他異なる Amazon EC2 プロビジョニング手法を使用するべきかについてや、Trusted Advisor のコスト最適化の提案のレビューなどです。

ベストプラクティス:

- **自動化** 重要でないリソースや不要なリソース、または使用率が低いリソースを確認して廃棄する際に、リソースの削除を適切に処理するようシステムを設計する。
- **定義されたプロセス** 既に使われていないリソースを確認して廃棄するプロセスがある。

COST 8. AWS の使用状況を管理するためにどのような管理と手順を実行していますか。

目標の達成のために適切なコストとなるようなポリシーとメカニズムを確立します。タグ付けと IAM コントロールを通じてチェックとバランスの手法を使用すると、浪費することなくイノベーションが可能になります。

ベストプラクティス:

- **グループとロールの確立** (例: 開発/テスト/本稼働) ガバナンスメカニズムを使用して、各々のグループのインスタンスとリソースをスピンアップできるのが誰かを管理する。(これは AWS サービスまたはサードパーティーのソリューションに適用されます)。
- **プロジェクトライフサイクルの追跡** プロジェクト、チーム、および環境のライフサイクルを追跡、測定、監査して、不要なリソースの使用と支払いを避ける。

継続した最適化

COST 9. 新しいサービスの採用をどのように管理または検討していますか。

AWS が新しいサービスや機能を公開するときに、既存のアーキテクチャ上の決定を見直し、それらが引き続き最もコスト効果が高いことを確認するのがベストプラクティスです。

ベストプラクティス:

- **コスト最適化関数の確立**
- **レビュー** 新しいサービス、リソースタイプおよびサイズをレビューするプロセスがある。パフォーマンステストを再実行して、コスト削減が見られるかどうかを評価します。

運用性の柱

準備

OPS 1. クラウドオペレーションのどのベストプラクティスを使用しているか

高い運用性を実現するには、効果的な準備が必要です。オペレーションチェックリストを使用すると、ワークロードが本稼働オペレーションの準備ができていることを確認できます。チェックリストを使用すると、十分な準備ができないまま意図せず本稼働に昇格させてしまうことがなくなります。

ベストプラクティス:

- **オペレーションチェックリスト** ワークロードの運用準備ができているかどうかを評価するために使用するオペレーションチェックリストを作成する。
- **事前計画** ビジネスに実体的な影響を与える機会とリスク (評価、財務など) に備えてイベント (マーケティングキャンペーン、フラッシュセールなど) の事前計画を立てる。
- **セキュリティチェックリスト** ワークロードを安全に運用する準備ができているかの評価に使用できるセキュリティチェックリストを作成する (ゼロデイ、DDoS、キー漏えいなど)。

OPS 2. ワークロードの設定管理はどのように行っているか

環境、アーキテクチャ、およびそれらのリソースの設定パラメータは、追跡およびトラブルシューティングのためにコンポーネントを容易に識別できる仕方で文書化する必要があります。設定の変更も追跡可能で自動化されている必要があります。

ベストプラクティス:

- **リソース追跡** ワークロード内のリソースとその機能を識別する方法を計画する (メタデータ、タグ付けの使用など)。
- **ドキュメント作成** アーキテクチャのドキュメントを作成する (コードとしてのインフラストラクチャ、CMDB、ダイアグラム、リリースノートなど)。
- **運用知識の習得** 継続的に運用知識を習得している (wiki、ナレッジベース、チケットなど)。
- **イミュータブルインフラストラクチャ** イミュータブルインフラストラクチャを確立し、パッチではなく再デプロイできるようにする。
- **変更手順の自動化** 変更手順を自動化する。
- **設定管理データベース (CMDB)** CMDB のすべての変更を追跡する。

運用

OPS 3. 変更の影響を最小限に抑えながら、ワークロードをどのように進化させるか

自動化、小規模の頻繁な変更、定期的な品質管理テスト、および変更の追跡、監査、ロールバック、レビューについて定義されたメカニズムに注力する必要があります。

ベストプラクティス:

- **デプロイメントパイプライン** CI/CD パイプラインを適切に配置する (ソースコードリポジトリ、ビルドシステム、デプロイメントとテストの自動化など)。
- **リリース管理プロセス** リリース管理プロセスを確立する (マニュアルまたは自動化など)。
- **小規模の増分変更** システムコンポーネントの小規模なバージョンアップをリリースできるようにする。
- **復元可能な変更** オペレーションに問題が出る場合は変更を元に戻せるように準備しておく (ロールバックや機能切り替えなど)。
- **リスク軽減戦略** Blue/Green、Canary、A/B テストなどのリスク軽減戦略を使用する。

OPS 4. ワークロードが想定通りに動作していることを確認するために、どのように監視するか

内部または外部 (またはその両方) の要因により、時間とともにシステムが機能低下する場合があります。システムの動作をモニタリングすることで、この低下の要因を確認し、修正することができます。

ベストプラクティス:

- **モニタリング** Amazon CloudWatch、サードパーティー製、またはカスタムのモニタリングツールを使用してパフォーマンスをモニタリングする。

- **ログの集約** 複数のソースからログを集約する (アプリケーションログ、AWS サービス固有のログ、VPC フローログ、CloudTrail など)。
- **アラームベースの通知** メトリクスが安全な境界を超えた場合に、モニタリングシステムから自動的なアラートを受信する。
- **トリガーベースのアクション** アラームによって自動化されたアクションが起動し、問題を修正またはエスカレーションする。

対応

OPS 5. 計画外の運用イベントにどのように対応するか

予期しない運用イベントへの応答を自動化しておきます。これにはアラートだけでなく、緩和、修復、ロールバック、復旧も含みます。

ベストプラクティス:

- **プレイブック** 従うべきプレイブック (オンコールプロセス、ワークフローチェーン、エスカレーションプロセスなど) があり、定期的に更新されている。
- **RCA プロセス** 問題を解決、記録、修正でき、今後の再発を防ぐ RCA プロセスがある。
- **自動化された対応** 自動化された対応を通じて、計画にない運用イベントに手順正しく対応する (Auto Scaling、Support API など)。

OPS 6. 計画外の運用イベントに対応する場合、エスカレーションはどのように管理されますか。

計画外の運用イベントへの対応は、ステークホルダー、エスカレーションプロセスおよび手順を含む事前定義されたプレイブックに従う必要があります。エスカレーションパスを定義し、機能的および階層的エスカレーション機能の両方を含めます。階層的エスカレーションは自動化され、エスカレートされた優先度でステークホルダーに通知されるべきです。

ベストプラクティス:

- **適切なドキュメントとプロビジョン** 必要なステークホルダーとシステムを適切に配置し、エスカレーションが発生した場合にアラートを受信できるようにします。
- **キューベースのアプローチを使用した機能的なエスカレーション** 優先度、深刻度、取り込みメカニズムに基づいた適切に機能するチームのキュー間でエスカレーションします。
- **階層的エスカレーション** デマンド型または時間ベースの手法を使用します。インシデントの深刻度、スケール、または解決/リカバリーまでの時間が増加すると、優先度が上がります。
- **外部エスカレーションパス** 外部サポート、AWS サポート、AWS パートナー、およびサードパーティーのサポート参加をエスカレーションパスに含める。
- **階層的優先度のエスカレーションが自動化されている** デマンドまたは時間のしきい値を超えると、優先度が自動的にエスカレーションします。