AWS Well-Architected

フレームワーク

2018 年 6 月

注意

本書は、情報提供の目的のみのために提供されるものです。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 製品の使用について独自に評価する責任を負うものとします。これらの情報は、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」提供されるものです。本書のいかなる内容も、AWS、その関係者、サプライヤー、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 契約により規定されます。本書は、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 1
定義 2
アーキテクチャについて 3
一般的な設計の原則 5
Well-Architected フレームワークの 5 本柱 8
運用上の優秀性 8
セキュリティ 16
信頼性 26
パフォーマンス効率 33
コスト最適化 44
レビュープロセス 53
まとめ 57
寄稿者 58
参考資料 59
ドキュメント改訂履歴 62
付録: Well-Architected に関する質問、回答、ベストプラクティス 63
運用上の優秀性 63
セキュリティ 70
信頼性 77
パフォーマンス効率 83
コスト最適化 89

はじめに

AWS Well-Architected フレームワークは、AWS 上でシステムを構築する際の決定における利点と欠点を理解するのに役立ちます。フレームワークを利用して、信頼性が高く、セキュアで、効率的で、コスト効果の高いシステムをクラウド上に設計し運用するためのアーキテクチャのベストプラクティスを学ぶことができます。フレームワークは、設計したアーキテクチャをベストプラクティスに照らして評価し、改善すべき分野を特定する一貫した方法を提供します。アーキテクチャをレビューするプロセスは、監査の過程ではなく、アーキテクチャの決定に役立つ話し合いを行うことです。私たちは、優れたアーキテクチャのシステムは、ビジネスが成功する可能性を大幅に高めることができると確信しています。

AWS ソリューションアーキテクトは、幅広いビジネス領域とユースケースにわたって、ソリューションのアーキテクチャを検討してきた長年の経験があります。私たちは、何千ものお客様に対して AWS におけるアーキテクチャの設計およびレビューを支援してきました。この経験から、クラウド上でシステムを設計するためのベストプラクティスと主要な戦略を特定してきました。

AWS Well-Architected フレームワークは、特定のアーキテクチャがクラウドのベストプラクティスに沿っているかどうかを判断するための、一連の基本的な質問を文書化しています。このフレームワークでは、最新のクラウドベースのシステムに要求される品質に照らしてシステムを評価する一貫したアプローチと、品質を満たすために必要な改善方法を提供します。AWS は、常に進化し続けているので、お客様とかかわりから継続して学び、優れたアーキテクチャの定義を継続して改良していきます。

本書は、最高技術責任者 (CTO)、アーキテクト、開発者、運用チームのメンバーなどの技術担当者を対象としています。クラウドワークロードを設計して運用するときに使用する AWS のベストプラクティスと戦略について説明し、さらに詳しい実装情報と、ア

ーキテクチャパターンへのリンクを提供します。詳細については、AWS Well-Architected ホームページを参照してください。

定義

AWS のエキスパートは、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のアーキテクチャ設計が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。実際の環境にシステムをデプロイして確認することで、システムがどの程度うまく機能するかについて、またアーキテクチャのトレードオフによる影響について学んできました。

このようにして私たちが学んできたことをベースにして、AWS Well-Architected フレームワークは作成されました。これは、お客様や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と、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を提供しています。

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

AWS Well-Architected フレームワークの柱

柱の名前	説明
運用上の優秀性	ビジネス価値をもたらし、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。
セキュリティ	リスクの評価と軽減戦略によって、ビジネス価値を生み出しながら、システム、アセット、情報を保護する能力。

信頼性	インフラストラクチャまたはサービスの中断から復旧し、需要に合わせて動的にコンピューティングリソースを取得し、設定ミスや一時的なネットワーク問題のような障害を軽減するシステムの能力。
パフォーマンス効率	コンピューティングリソースを効率的に使ってシステム要件を満たし、需要の変化や技術の進歩に合わせてこの効率性を維持する能力。
コスト最適化	最も安価にシステムを実行して、ビジネス価値を実現する能力。

ソリューションのアーキテクチャを設計する際に、ビジネスの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ます。そして、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して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、パフォーマンスは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上の優秀性は、通常、他の柱とトレードオフになることはありません。

アーキテクチャについて

お客様のオンプレミス環境では、お客様内にテクノロジーアーキテクチャを管理する中心的なチームがあり、ベストプラクティスに従っていることを確実にするために、他の製品チームや機能チームのオーバーレイとして機能しているケースが多くみられます。テクノロジーアーキテクチャチームは、多くの場合、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、ネットワークアーキテクト、セキュリティアーキテクトなどの一連のロールで構成されます。これらのチームは、エンタープライズアーキテクチャ機能の一部として

TOGAF または Zachman Framework を使用することがよくあります。

AWS では、その機能を持つ一元化されたチームを持つことよりも、その機能を複数のチームに分散させる方が良いと考えています。例えば、意思決定機関を分散することを選択した場合は、チームが内部の標準を満たしていることを保証できないリスクが発生しますが、私たちは 2 つの方法でこれらのリスクを軽減しています。第 1 に、AWS には各チームにその機能を持たせることに焦点を当てたプラクティス[1] (慣行) 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にアクセスできるようにしています。第 2 に、基準を確実に満たすための自動チェックを実行するメカニズム[2]を導入しています。この分散型アプローチは Amazon のリーダーシッププリンシプルによって支持されており、お客様を起点に物事を考えるという文化を、[3]すべてのロールにわたって確立します。お客様に徹底的にこだわる (Customer-obsessed) チームがお客様のニーズに応じて製品を開発します。

アーキテクチャに関しては、すべてのチームが、ベストプラクティスに従いアーキテクチャを作成できる能力を持つことを期待されます。新しいチームがこの能力を持ち、既存のチームがバーを上げることができるように、設計のレビューや AWS ベストプラクティスの理解に役立つサポートを行うプリンシパルエンジニアの仮想的なコミュニティにアクセスできるようにしています。プリンシパルエンジニアのコミュニティは、ベストプラクティスが参照できアクセスできるようにします。これを実行する 1 つの方法として、実例に対してベストプラクティスを適用することにフォーカスしたランチタイムのトークイベントなどが挙げられます。これらの話を記録して、新しいチームメンバーのオンボーディング向け資料の一部として使用することもできます。

AWS のベストプラクティスは、インターネット規模の数千のシステムを運用してきた

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

経験から作られています。AWS ではベストプラクティスを定義するためにデータ内容を重視しますが、プリンシパルエンジニアといった特定分野のエキスパートの知識も活用しています。新しいベストプラクティスをプリンシパルエンジニアが認識すると、コミュニティとして機能して、チームがそれらに確実に従うようにします。これらのベストプラクティスでは、正式な内部レビュープロセスと、実施するメカニズムに準拠しています。Well-Architected は、AWS の社内レビュープロセスのお客様向けの実装で、現場にいるソリューションアーキテクト、および社内エンジニアリングチームなど主なエンジニアリング思考を体系化したものです。Well-Architected フレームワークは、これらの経験からの学習をお客様が活用できるスケーラブルなメカニズムです。

プリンシパルエンジニアのコミュニティによる、分散されたアーキテクチャのオーナーシップに基づくアプローチによって、お客様のニーズに基づいた Well-Architected なエンタープライズアーキテクチャが実現できると確信しています。テクノロジーリーダー (CTO または開発マネージャーなど) が Well-Architected レビューをすべてのシステムに対して行うことで、自社のテクノロジーのポートフォリオにおけるリスクの理解を深めることができます。このアプローチを使用することにより、チームにまたがるテーマを明らかにすることができ、メカニズムやトレーニング、またはプリンシパルエンジニアが特定分野の知見を複数のチームと共有するランチタイムトークなどにより、組織として解決することができます。

一般的な設計の原則

Well-Architected フレームワークは、クラウドにおける優れた設計を促進する一般的な設計の原則を特定します。

- **必要なキャパシティーを勘に頼らない**: 必要なインフラストラクチャ容量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、高価で無駄なリソースが発生したり、限られたキャパシティーによるパフォーマンス影

響への対応が必要になったりします。クラウドコンピューティングでは、このような問題は発生しません。必要なだけ使用することができ、必要に応じてキャパシティーを自動的にスケールアップおよびスケールダウンできます。

- **本番規模でシステムをテストする**: クラウドでは、本番規模のテスト環境をオンデマンドで作成することができ、テストを実行して、完了後すぐにリソースの使用を終了できます。テスト実行中に使用したテスト環境のリソース分の料金だけを支払うので、オンプレミスでのテスト費用のほんの一部で本番環境をシミュレートしたテストを行うことができます。

- **自動化によってアーキテクチャ上の実験を容易にする**: 自動化により、システムの構築と複製を低コストで行うことができ、手作業によるコストも回避することができます。自動化によって実行された変更を追跡し、その影響を監査し、必要に応じて以前のパラメータに戻すことができます。

- **発展的なアーキテクチャを受け入れる**: 発展的なアーキテクチャを受け入れます。従来の環境では、アーキテクチャ上の決定は、多くの場合、静的な 1 回限りのイベントとして実装され、システムのライフサイクル中に、数回の大きなメジャーバージョンの変更を伴うかもしれません。ビジネスおよびその状況は常に変化するので、これらの最初の決定によっては、変化するビジネス要件に対応するシステムの機能が阻害されてしまうかもしれません。クラウドでは、自動化とオンデマンドなテストを実施することが可能であり、設計変更のリスクを低減することができます。これにより、システムを継続的に進化させることができ、ビジネスはイノベーションの恩恵を通常のプラクティスとして享受することができます。

- **データ計測に基づいてアーキテクチャを決定する**: クラウドでは、アーキテクチャの選択がシステムの動作にどのように影響を与えるかを示すデータを収集することができます。これにより、システムを改善する方法を事実に基づいて決定することができます。クラウドインフラストラクチャはコード化できるので、収集したデータを使ってアーキテクチャの選択や改善を継続的に実施することができます。

- **本番で想定されるトラブルをあらかじめテストし、対策する**: 定期的にゲームデーを開催し、本番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、イベントに備えて組織的な経験を育成したりするために役立ちます。

Well-Architected フレームワークの 5 本柱

ソフトウェアシステムを構築するのは、建造物を建築するのとよく似ています。基礎が強固でなければ、構造上の問題が建造物の完全性と機能を台なしにしてしまいます。技術ソリューションのアーキテクチャを設計するときに、運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化の 5 本の柱を軽視すると、期待と要件を満たすシステムを構築することは困難かもしれません。これらの柱をアーキテクチャに組み込むことは、安定した効率性の高いシステムを構築する助けになります。これによって、機能要件など設計の他の側面に集中することができます。

運用上の優秀性

運用上の優秀性の柱には、ビジネス価値を実現するためのシステムを実行してモニタリングし、それらをサポートするプロセスと手順を継続的に改善する機能が含まれています。

運用上の優秀性の柱では、設計の原則、ベストプラクティス、質問の概要について説明します。実装の手引きとなるガイダンスについては、運用上の優秀性の柱に関するホワイトペーパーを参照してください。

設計の原則

クラウドにおける運用上の優秀性には、6 つの設計の原則があります。

- **運用をコード化する**: クラウドでは、アプリケーションコードの管理に使用するのと同じエンジニアリング規律を環境全体に適用することができます。システム全体 (ア

プリケーション、インフラストラクチャ) をコード化して定義し、コードを使ってそれらを更新することができます。運用手順をスクリプト化し、イベントによるトリガーにより自動的に実行させることができます。運用をコードで実行することにより、人為的なミスを抑制し、イベントに対する一貫した対応を可能にします。

- **ドキュメントに注釈を付ける**: オンプレミス環境では、ドキュメントは手動で作成され、ユーザーによって使用され、変化のスピードに同期することは困難です。クラウドでは、すべての構築後に注釈付きのドキュメントを自動的に作成することができます (または、手作業で作成したドキュメントに自動的に注釈を付けることができます)。注釈付きのドキュメントはユーザーやシステムが使用できます。注釈を、運用で使うコードへのインプットとして使用します。

- **頻繁に、小さく、可逆的な変更を行う**: コンポーネントが定期的に更新できるようにシステムを設計します。小さく、可逆的な変更を行い、更新が失敗した場合は、(可能であればユーザーに影響することなく) 変更を元に戻せるようにします。

- **運用手順を頻繁に見直す**: 運用手順を実施するたびに、手順の改善点を検討します。システムを進化させるのに合わせて、運用手順も適切に進化させることが必要です。定期的にゲームデーを開催し、すべての運用手順が有効で、チームがそれらに精通していることをレビューして確認します。

- **発生しうる障害を予想する**: 「事前」の演習により潜在的な障害の原因を特定し、その原因を排除するか、影響を軽減することができます。障害シナリオをテストし、障害の影響に対する理解が正しいかを確認します。対応手順をテストして、手順が有効であり、チームがそれらの実行を十分理解していることを確実にします。定期的にゲームデーを開催し、シミュレートされた障害イベントに対するシステムの動作とチームの対応をテストします。

- **運用上の失敗を改善に役立てる**: すべての運用上のイベントや障害から学んだ教訓、

運用の改善に役立てます。学びをチームや組織全体に共有します。

定義

クラウドにおける運用上の優秀性には、3 つの分野のベストプラクティスがあります。

1. **準備**

2. **運用**

3. **進化**

運用チームは、ビジネスやお客様のニーズを理解しておく必要があり、そうすることで効果的かつ効率的にビジネスの成果をサポートできるようになります。運用チームは運用上のイベントに対応する手順を作成および利用し、ビジネスニーズを効果的にサポートできるかどうかを検証します。運用チームは、求められるビジネス成果の達成を測定するためのメトリクスを収集します。ビジネスの状況、ビジネス上の優先事項、顧客のニーズなど、あらゆる事柄が常に変化し続けています。変化に対応するための継続的な進化をサポートし、実績から学んだ教訓を取り入れることのできる運用を設計することが重要です。

ベストプラクティス

準備

運用上の優秀性を実現するには、効果的な準備が必要です。ビジネスの成功は、目標を共有し、ビジネス、開発、および運用を横断的に理解することによって実現されます。共通の標準により、システムの設計と管理が簡素化され、運用上の成功をもたらします。アプリケーションやプラットフォーム、インフラストラクチャのコンポーネントお

よび顧客の体験と反応をモニタし、インサイトを取得するメカニズムを備えたシステムを設計します。

システムや変更が本番環境に移行する準備ができており、運用によりサポートされることを検証するメカニズムを構築します。運用の準備状況はチェックリストにより検証され、システムが定義された標準を満たしており、必要な手順がランブックとプレイブックに適切に記載されていることを確実にします。トレーニングを受けた要員が十分にいることを確認し、システムを効果的にサポートできるようにします。移行の前に、運用上のイベントと障害に対する対応をテストします。障害を意図的に発生させたり、ゲームデーを開催したりすることで、サポートする環境での対応を訓練します。

AWS では、クラウド上での運用をコードで実行することができるため、安全に実験を行い、運用手順を開発し、障害の訓練を行うことができます。AWS CloudFormation を使用して、一貫したテンプレートベースの方法で、サンドボックス開発環境、テスト環境、本番環境を、運用の統制を向上させながら維持することができます。AWS では、さまざまなログ収集やモニタリングの機能を利用して、すべてのレイヤーにわたる可視性を維持することができます。リソースの使用、アプリケーションプログラミングインターフェイス (API)、ネットワークトラフィックに関するデータは、Amazon CloudWatch、AWS CloudTrail、VPC Flow Logs により収集されます。collectd プラグインや CloudWatch Logs エージェントを利用することで、OS に関する情報を CloudWatch に集約することができます。

このような運用上の優秀性に関する考慮事項に注意を促す質問を以下に掲載します(運用上の優秀性の質問、回答、ベストプラクティスの一覧については、付録を参照してください)。

運用上の優秀性 1: オペレーションの優先順位を決定する要因は明確ですか？

運用上の優秀性 2: 運用性を向上させるためのワークロード設計をしていますか？

運用上の優秀性 3: ワークロードが運用可能であることをどのように確認していますか？

システムの設計では、最小限のアーキテクチャ標準を実装するようにしてください。システムに対するメリット、運用の負荷、標準を実装するためのコストのバランスを考慮してください。サポートすべき標準を減らして、エラーが原因で許容できる水準より低くなってしまう可能性を低減してください。多くの場合、運用担当者のリソースは限られています

運用の作業のスクリプト化に時間をかけることで運用者の生産性を最大化し、エラー率を最小化し、自動的な対応ができるようにしてください。クラウドの伸縮性を活用できるデプロイ方法を採用し、すばやい実装のためにシステムの事前デプロイができるようにします。

運用

システムの運用がうまくできているかどうかは、ビジネスや顧客の成果の達成度合いによって計測されます。期待される成果を定義して、どのように成果が測定されるかを決定し、正しく運用できているかどうかを判断するために使われるシステムと運用に関するメトリクスを特定します。システムとシステムに対する運用の正常性 (例: デプロイとインシデント対応) の両方を含めて運用の健全性を考慮してください。発見された運用の改善あるいは運用の劣化を基にベースラインを確立し、メトリクスを収集して分析し、運用がうまくできているか、どのように推移してきたかについての理解を確認してください。収集されたメトリクスにより、顧客やビジネスニーズを満たせているかを確認し、改善できる部分を発見します。

運用イベントを効果的かつ効率的に管理することは、運用上の優秀性を実現するために重要です。これは計画されたイベントにも計画外のイベントにも当てはまります。事前にわかっているイベントに対しては、確立されたランブックを使用し、それ以外に解決が必要なイベントに対してはプレイブックを助けにします。ビジネスと顧客のニーズに応じて、イベント対応の優先度を決定します。イベントに対するアラートが正しく上げられ、関連するプロセスが実行され、関連するプロセスのオーナーが明確になるように

してください。イベントの解決に必要な担当者を事前に定義して、影響の度合い (イベントの発生期間やスケールやスコープ) に応じて、追加の担当者を巻き込むためのエスカレーションのトリガーも含めてください。イベント対応によりビジネスインパクトが発生する場合に適切な対応策を決定するため、適切な権限を持った人を特定し、必要に応じて関与させることができるようにしてください。

それぞれの対象者 (顧客、ビジネス担当者、開発担当者、運用担当者など) 向けにカスタマイズされたダッシュボードや通知を使って、システムの運用状況を共有して、イベントに対して適切な措置を行い、運用の期待値を正しく管理し、通常状態が再開される場合に知らせるようにします。

計画外のイベントの根本原因と、計画されたイベントで発生した想定外の影響を確認します。この情報は、将来イベントが発生した場合に影響を緩和するための手順を更新するのに使われます。必要に応じて、イベントにより影響を受けた関係者に根本原因を共有します。

AWS では、システムによって取得されたメトリクスを表示するダッシュボードを、AWS ネイティブの機能で作成することができます。CloudWatch やサードパーティのアプリケーションを活用して、ビジネスレベル、システムレベル、運用レベルなどさまざまなレベルの運用活動に関するビューを集約して表示することができます。AWS は、システムの問題を発見して根本原因の分析や改善を行うために、AWS X-Ray、CloudWatch、CloudTrail、VPC Flow Logs などのロギング機能を使うことで、システムに関して深く理解することができます。

このような運用上の優秀性に関する考慮事項に注意を促す質問を以下に掲載します。

運用上の優秀性 4: 運用状態をどのように確認していますか？

運用上の優秀性 5: 運用上のイベントをどのように管理していますか?

定期的な運用のみではなく、計画外のイベントに対する対応も自動化されるべきです。

デプロイやリリース管理、変更ロールバックなどのプロセスはマニュアルで実施されるべきではありません。リリースは、まれに行われる大規模変更により実施されるべきではありません。大規模な変更では、ロールバックはより困難になります。ロールバック計画が失敗したり、障害の影響を最小化できなかったりした場合、システムの運用が継続できなくなってしまいます。メトリクスをビジネスニーズと整合させることで、運用対応によるビジネス継続性の維持を効果的に行うことができます。1 回限りの一元化されていない手動で取得されたメトリクスは、計画外のイベントが発生した際にシステムの運用を大きく妨げる結果につながります。

進化

運用を進化させていくことは、運用上の優秀性を維持していくために重要です。継続的に少しずつ改善していくサイクルに専念します。システムと運用手順の両方に関して、定期的に改善できる部分を評価し優先順位付けをしてください (例えば、機能追加の要望、問題の改善、コンプライアンス要件など)。手順の中にフィードバックループを含め、改善できる部分をすばやく特定して、運用実績からの学習を取り込めるようにします。

メリットを共有するために、チームにまたがって運用から得た学びを共有します。学びから傾向を分析し、チームにまたがって運用メトリクスの振り返りと分析を行い、改善の機会と方法を特定します。改善をもたらす変更を実施し、その結果を評価して成功を判断します。

AWS 開発者用ツールを使用すると、ビルド、テスト、デプロイのアクティビティを継続的デリバリーとして実装できます。そのアクティビティでは、AWS とサードパーティから提供されるさまざまなソースコード、ビルド、テスト、デプロイ用のツールを使用できます。デプロイ作業の結果は、デプロイと開発の双方での改善できる部分を見つけるのに利用できるかもしれません。運用とデプロイ作業に関連した統合されたメトリクスデータを分析することで、ビジネスや顧客の成果に対しての運用やデプロイの影響

を分析することができます。このデータの振り返りと分析を複数のチームで行うことで、改善の機会と方法を発見するために役立てることができます。

このような運用上の優秀性に関する考慮事項に注意を促す質問を以下に掲載します。

運用上の優秀性 6: 運用をどのように進化させていますか？

運用をうまく進化させられるかどうかは、頻繁で小規模な変更や、改善を試し、開発し、テストすることができる安全な環境と時間を提供することや、障害から学ぶことを奨励する環境などによって支えられます。サンドボックスや開発環境、テスト環境、本番環境において高いレベルで統制された運用を行うことにより、開発を促進し、変更を本番環境にデプロイする際の成功をうまく予測できるようになります。

AWS の主要なサービス

運用上の優秀性に不可欠な AWS のサービスは、ベストプラクティスに基づいてテンプレートを作成するために使用できる **AWS CloudFormation** です。これにより、開発環境から本番環境まで、整った一貫性のある方法でリソースをプロビジョニングすることができます。以下のサービスと機能によって、運用上の優秀性の 3 つの分野がサポートされます。

- **準備**: AWS Config および AWS Config ルールは、システムの標準を作成し、環境を本番稼働させる前にそれがこれらの標準に準拠しているかどうかを判断するために使用できます。
- **運用**: Amazon CloudWatch を使用してシステムの運用状態をモニタリングできます。
- **進化**: Amazon Elasticsearch Service (Amazon ES) を使用してログデータを分析し、適切な対策につながるインサイトをすばやく安全に見つけることができます。

リソース

運用上の優秀性に関する AWS のベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメント

- DevOps と AWS

ホワイトペーパー

- 運用上の優秀性の柱

動画

- DevOps at Amazon

セキュリティ

セキュリティの柱には、リスクの評価と軽減戦略をとお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が含まれます。

セキュリティの柱では、設計の原則、ベストプラクティス、質問の概要について説明します。実装の手引きとなる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。

設計の原則

クラウドにおけるセキュリティには、7 つの設計の原則があります。

- **強固な認証基盤を整備する**: 最小権限の原則を実装し、適切な職務分掌を徹底して、AWS リソースに対するアクセス権限を適切に管理します。権限管理を一元化し、長

期的な認証情報への依存を低減、または解消します。

- **追跡可能性を実現する**: 環境に対するアクションおよび変更をリアルタイムでモニタリング、警告、監査します。ログとメトリクスをシステムと統合し、自動的に応答してアクションを実行します。

- **すべてのレイヤーにセキュリティを適用する**: 外側の単一レイヤーの保護のみに注目するのではなく、他のセキュリティ制御を使った深層防御アプローチを適用します。すべてのレイヤーにセキュリティを適用します (エッジネットワーク、VPC、サブネット、ロードバランサー、すべてのインスタンス、オペレーティングシステム、アプリケーションなど)。

- **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って、より迅速かつコスト効率の良い方法で、安全にスケールすることができるようになります。バージョン管理されたテンプレートを使用して、コードとして定義および管理された制御の実装を含む、安全なアーキテクチャを作成します。

- **転送中および保管中のデータを保護する**: 機密レベルでデータを分類し、必要に応じて暗号化、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使用します。

- **データに人を近づけない**: データに対する直接的なアクセスや手動処理の必要性を、低減または排除するためのメカニズムとツールを作成します。これにより、機密データの処理時における損失または改変のリスクやヒューマンエラーを低減することができます。

- **セキュリティイベントに対して準備する**: 組織の要件に合わせたインシデント管理のプロセスを導入し、インシデントに備えます。インシデント対応のシミュレーションを実施し、自動化されたツールを使用して、検出、調査、復旧の速度を向上させます。

定義

クラウドにおけるセキュリティには、5 つの分野のベストプラクティスあります。

1. **アイデンティティとアクセス管理**

2. **発見的統制**

3. **インフラストラクチャ保護**

4. **データ保護**

5. **インシデント対応**

システムを設計する前に、セキュリティに影響するプラクティスを導入する必要があります。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。また、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の保護し、データ保護によってデータの機密性と整合性の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には、明確に定義された実績のあるプロセスが不可欠です。このようなツールと技術は、財務上の損失の防止、規制遵守といった目的をサポートするために重要です。

AWS 責任共有モデルにより、企業はクラウドを導入してセキュリティとコンプライアンスの目標を達成できます。AWS は、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しているため、AWS のお客様はサービスを使用して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を自動化することができます。

ベストプラクティス

アイデンティティとアクセス管理

アイデンティティとアクセスの管理は情報セキュリティプログラムの主要部分です。権限を持つユーザーおよび認証されたユーザーのみが、管理者の意図した方法でリソースにアクセスできるようにします。例えば、プリンシパル (アカウント内でアクションを実行するユーザー、グループ、サービス、ロール) を定義し、これらのプリンシパルに合わせてポリシーを構築し、強力な認証情報管理を実装する必要があります。これらの権限管理の構成要素は、認証と認可の主要な部分です。

AWS の権限管理は、主に AWS Identity and Access Management (IAM) サービスによってサポートされており、これによって AWS のサービスとリソースに対するユーザーとプログラムのアクセスを制御できます。きめ細かなポリシーを適用して、ユーザー、グループ、ロール、リソースにアクセス許可を割り当てる必要があります。また、複雑さのレベル、再利用の禁止、Multi-Factor Authentication (MFA) の使用など、強力なパスワード設定をする機能もあります。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 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 によって、ロール、インスタンスプロファイル、ID フェデレーション、一時的な認証情報を使用して安全なアクセスを実現します。

このようなセキュリティに関する考慮事項に注意を促す質問を以下に掲載します(セキュリティの質問、回答、ベストプラクティスの一覧については、付録を参照してください)。

セキュリティ 1: ワークロードの認証情報をどのように管理していますか？

セキュリティ 2: AWSサービスへの人為的なアクセスをどのように制御していますか？

セキュリティ 3: AWSサービスへのプログラムによるアクセスをどのように制御していますか？

認証情報は、どのユーザーまたはシステム間でも共有しないようにする必要があります。ユーザーアクセスは、パスワード要件と MFA の使用などのベストプラクティスに従ったうえで、最小権限で付与させるべきです。AWS のサービスに対する API コール

といったプログラムによるアクセスは、一時的で権限が制限された認証情報 (AWS Security Token Service (STS) によって発行されたものなど) を使って行われるようにしてください。

発見的統制

発見的統制を使用して、潜在的なセキュリティの脅威またはインシデントを特定できます。検出制御は、ガバナンスフレームワークにおいて不可欠な要素であり、品質プロセス、法的またはコンプライアンス義務、脅威の識別と対応の取り組みをサポートするために使用できます。発見的統制にはさまざまな種類があります。例えば、アセットとその詳細な属性の一覧を作成することで、より効果的な意思決定やライフサイクル管理が促進され、運用の基準を確立することができます。また、情報システムに関連する制御を精査する内部監査を使用し、実態がポリシーと要件に一致していることや、定義した条件に基づいてアラート通知の自動化が正しく設定されていることを確認することもできます。このような統制は重要な反応因子で、組織が異常なアクティビティの範囲を特定して理解するのに役立ちます。

AWS では、監査、自動分析、アラームを許可しているログ、イベント、モニタリングを処理することによって、発見的統制を実現することができます。CloudTrail ログ、AWS API コール、CloudWatch は、アラームを使った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。Amazon GuardDuty は、悪意のある動作や不正な動作がないかどうかを継続的にモニタリングするマネージド型の脅威検知サービスで、AWS アカウントとシステムを保護するために役立ちます。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作成できます。

このようなセキュリティに関する考慮事項に注意を促す質問を以下に掲載します。

セキュリティ 4: ワークロードのセキュリティイベントをどのように検知していますか？

ログ管理は、セキュリティやフォレンジックから、規制上の要件または法的な要件までに関係する理由のため、優れた設計において重要です。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、データ保持のライフサイクルを定義することや、データの保管、アーカイブ、最終的な削除を行う場所を定義することで、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、よりシンプルでコスト効率の優れた方法で、予測可能で信頼できるデータ処理を行うことができます。

インフラストラクチャ保護

インフラストラクチャの保護には、多層に渡る防御などの制御方法で、ベストプラクティスと組織または規制上の義務を満たすために必要があります。クラウドでもオンプレミスでも、実行中の運用を成功させるには、このような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。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、プライベートサブネットを含むトポロジを定義する必要があります。

このようなセキュリティに関する考慮事項に注意を促す質問を以下に掲載します。

セキュリティ 5: ネットワークをどのように保護していますか？

セキュリティ 6: AWS のセキュリティ機能と一般的なセキュリティ脅威に関する最新情報をどのように確認していますか？

セキュリティ 7: コンピューティングリソースをどのように保護していますか？

多層防御はどのタイプの環境にもお勧めします。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。境界保護の強化、出入口のモニタリング、広範囲のロギング、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不可欠です。

AWS のお客様は、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon EC2 Container Service (Amazon ECS) コンテナ、AWS Elastic Beanstalk インスタンスの設定をカスタマイズまたは強化して、その設定を変更不能なAmazon Machine Image (AMI) として保存することができます。その後、Auto Scaling によるトリガー、または手動での作成のいずれかにより、この AMI を使って作成されたすべての新しい仮想サーバー (インスタンス) で、強化された設定を使用できます。

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する基本的なプラクティスを設定しておく必要があります。例えば、データ分類は、企業のデータを機密性レベルに基づいて分類する方法です。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。このようなツールと技術は、財務上の損失の防止、規制遵守といった目的をサポートするために重要です。

AWS では、以下のようなプラクティスでデータの保護を行います。

- 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などのキー管理がより簡単になります。これらは AWS によって自動化することも、自分で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ギングを使用できます。
- AWS のストレージシステムは、非常に高い回復性を持つよう設計されています。例

えば、Amazon S3 スタンダード、S3 スタンダード – IA、S3 1 ゾーン – IA、Amazon Glacier はいずれも年間 99.999999999% の耐久性を実現するよう設計されています。この耐久性レベルは、年間に予想されるオブジェクトの喪失が平均 0.000000001% であることに相当します。

- バージョニングは大容量データのライフサイクル管理プロセスの一部であり、意図しない上書きや削除、および類似の障害からデータを保護できます。
- AWS がリージョン間でデータを移動させることはありません。あるリージョンに置かれたコンテンツは、お客様が明示的に移動の機能を有効にするか、そのような機能を提供するサービスを利用しない限り、そのリージョンから移動されることはありません。

このようなセキュリティに関する考慮事項に注意を促す質問を以下に掲載します。

セキュリティ 8: データをどのように分類していますか？

セキュリティ 9: データ保護メカニズムをどのように管理してしますか?

セキュリティ 10: 保存中のデータをどのように保護していますか？

セキュリティ 11: 転送中のデータをどのように保護していますか？

AWS では、保管中と転送中のデータの暗号化を複数の手段で実現できます。このような機能はサービスに組み込まれているため、データの暗号化は簡単です。例えば、Amazon S3 にはサーバー側暗号化 (SSE) が実装されているため、データを暗号化して保存できます。また、HTTPS の暗号化と復号のプロセス全体 (一般に SSL ターミネーションと呼ばれる) を、Elastic Load Balancing (ELB) に処理させることもできます。

インシデント対応

非常に成熟した予防的コントロールと発見的統制が実施されている場合でも、企業としてセキュリティインシデントに対応し、その潜在的な影響を軽減するプロセスを導入す

ることが必要です。ワークロードのアーキテクチャは、システムの分離や影響範囲拡大の抑制、および通常運用への復帰など、インシデント発生中にチームが効果的に機能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とアクセスを整備し、定期的にゲームデーによってインシデント対応を実践することが、調査と復旧を適時実施できるアーキテクチャ設計に役立ちます。

AWS では、以下のようなプラクティスによって効果的なインシデント対応を促進しています。

- 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ギングが使用できる。
- イベントは自動的に処理され、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る。
- 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ンできる。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施できる。

このようなセキュリティに関する考慮事項に注意を促す質問を以下に掲載します。

セキュリティ 12: インシデントに対してどのように備えていますか？

情報セキュリティチームのアクセスを迅速に許可し、インスタンスの隔離、および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。

AWS の主要なサービス

AWS Identity and Access Management (IAM) はセキュリティに不可欠なサービスで、AWS のサービスやリソースに対するユーザーのアクセスを安全にコントロールできます。以下のサービスと機能によって、セキュリティの 5 つの分野がサポートされます。

- **アイデンティティとアクセス管理**: IAM によって、AWS のサービスとリソースへの

アクセスを安全にコントロールできます。MFA はユーザ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できます。AWS Organizations によって、複数の AWS アカウントの管理を一本化し、ポリシーを強制的に適用できます。

- **発見的統制**: AWS CloudTrail によって AWS API コールが記録され、AWS Config によって AWS リソースと設定の詳細なインベントリが提供されます。Amazon GuardDuty はマネージド型の脅威検出サービスで、悪意のある動作や不正な動作がないかどうかが継続的にモニタリングします。Amazon CloudWatch は AWS リソースのモニタリングサービスで、CloudWatch Events をトリガーすることでセキュリティ対応を自動化できます。

- **インフラストラクチャの保護**: Amazon Virtual Private Cloud (Amazon VPC) によって、AWS リソースをお客様が定義した仮想ネットワークに起動できます。Amazon CloudFront はグローバルなコンテンツ配信ネットワークで、データ、動画、アプリケーション、API をエンドユーザーに安全に配信でき、DDoS 攻撃を緩和する AWS Shield と統合されています。AWS WAF は Amazon CloudFront または Application Load Balancer にデプロイするウェブアプリケーションファイアウォールで、一般的なウェブのエクスプロイトからウェブアプリケーションを守ります。

- **データ保護**: ELB、Amazon Elastic Block Store (Amazon EBS)、Amazon S3、Amazon Relational Database Service (Amazon RDS) などのサービスには、転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。Amazon Macie は機密データを自動的に検出、分類、保護するサービスです。また、AWS Key Management Service (AWS KMS) は、暗号化用のキーの作成と管理を容易します。

- **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と対応ツールに適切な認可を付与する必要があります。AWS CloudFormation を使用して、調査を実行するための信頼できる環境またはクリーンルームを構築できます。Amazon CloudWatch Events によって、AWS Lambda などの自動化された対応をトリガーするルールを作成できます。

リソース

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

ドキュメント

- AWS クラウドセキュリティ
- AWS コンプライアンス
- AWS セキュリティブログ

ホワイトペーパー

- セキュリティの柱
- AWS セキュリティの概要
- AWS セキュリティのベストプラクティス
- AWS リスクとコンプライアンス

動画

- AWS Security State of the Union
- 責任共有モデルの概要

信頼性

信頼性の柱には、インフラストラクチャやサービスの障害からの復旧、必要に応じたコンピューティングリソースの動的な取得、設定ミスや一時的なネットワーク問題などの障害の軽減といったシステムの機能が含まれます。

信頼性の柱では、設計の原則、ベストプラクティス、質問の概要について説明します。実装の手引きとなる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。

設計の原則

クラウドにおける信頼性には、5 つの設計の原則があります。

- **復旧手順をテストする**: オンプレミス環境において、ほとんどの場合、テストは特定のシナリオでシステムが機能することを証明するために実施されます。復旧戦略を検証するためにテストが行われることはあまりありません。クラウド上では、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、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。これによって障害が発生する経路を明らかにし、実際に障害が発生する前にテストし修正することで、テストされていないコンポーネントにより障害が発生するリスクを軽減できます。

- **障害から自動的に復旧する:** システム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化した処理をトリガーできます。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。自動化をより高機能にすることで、障害を発生前に予測して修正することも可能です。

- **水平方向にスケールして総合的なシステムの可用性を向上させる**: 1 つの大きなリソースを複数の小さなリソースに置き換えて、単一の障害がシステム全体に及ぼす影響を軽減します。複数の小さなリソースにリクエストを分散させて、共通の障害点を共有しないようにします。

- **キャパシティーを勘に頼らない**: オンプレミスシステムでの障害の一般的な原因はリソースの飽和で、システムに対する需要がそのシステム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)。クラウド上では、需要とシステム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、プロビジョンが過剰にも過小にもならない、適切に需要を満たせる最適レベルを維持できます。

- **自動化の変更を管理する**: インフラストラクチャの変更は自動化を利用して行うことが必要です。管理が必要となる変更は、自動化に対する変更です。

定義

クラウドにおける信頼性には、3 つの分野のベストプラクティスがあります。

1. **基盤**
2. **変更管理**
3. **障害管理**

信頼性を達成するため、システムの基盤を十分に計画してモニタリングを実施し、需要や要件の変更を処理するメカニズムを設けることが必要です。障害を検出し、自動的に修復できるよう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

基盤

システムのアーキテクチャを設計する前に、信頼性に影響を与える基盤についての要件が整っていることが必要です。例えば、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。こうした要件は時として無視されてしまいます (単独のプロジェクトのスコープからは外れるためです)。しかし、この点を無視すると、信頼性の高いシステムを実現する能力に大きく影響することがあります。オンプレミス環境の場合、このような

要件には依存関係のため長いリードタイムが必要になり、初期段階から計画に組み込んでおくことが必須になりました。

AWS の場合、このような基本的な要件は最初から組み込まれているか、必要に応じて対処することができます。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、十分なネットワークとコンピューティング性能の要件を満たすのは AWS の責任です。お客様はストレージデバイスのサイズといったリソースのサイズや割り当てをオンデマンドで自由に変更できます。

このような信頼性に関する考慮事項に注意を促す質問を以下に掲載します(信頼性の質問、回答、ベストプラクティスの一覧については、付録を参照してください)。

信頼性 1: AWS サービスの制限をどのように管理していますか？

信頼性 2: AWS上でのネットワーク構成をどのように設計していますか？

AWS では、お客様がリソースを意図せず過剰にプロビジョニングしてしまわないよう、サービスの制限 (AWS 利用者が要求できる各リソースの数の上限) が設定されています。このような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、適切なガバナンスとプロセスが必要になります。クラウド導入にあたって、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) の計画が必要な場合があります。ハイブリッドモデルでは、時間をかけて徐々にすべてをクラウドに移行することが可能になります。このため、AWS リソースとオンプレミスリソースとの連携方法をネットワークトポロジーとして設計しておくことが重要です。

変更管理

変更がシステムに及ぼす影響を把握していると、事前に計画を立てることができます。また、モニタリングによってキャパシティーの問題や SLA 違反につながりかねない傾向をすばやく見つけることができます。従来の環境では、変更制御のプロセスは手動で

あることが多く、誰がどの時点で変更を加えるかを効率的に管理するために、監査しながら慎重に調整することが必要でした。

AWS を使用することで、システムの動作をモニタリングして KPI への対応を自動化することができます。例えば、システムを使用するユーザーが増えたときに自動でサーバーを追加できます。誰がシステムの変更権限を持つかをコントロールし、その変更の履歴を監査することができます。

このような信頼性に関する考慮事項に注意を促す質問を以下に掲載します。

信頼性 3: システムに対する需要の変化にはどのように対応していますか？

信頼性 4: AWSリソースをどのようにモニタリングしていますか？

信頼性 5: 変更をどのように実施していますか？

需要の変更に対応してリソースの追加や削除が自動的に行われるようシステムを設計しておけば、信頼性は向上し、ビジネスの成功が負担に変わることも避けられます。適切なモニタリングが実施されていれば、KPI が予測された基準から逸脱したときに、アラートが自動的にチームに送られます。環境の変更に対する自動的なロギングによって、信頼性に影響を与えた可能性のあるアクションを監査によってすばやく特定できます。変更管理をコントロールすることで、必要な信頼性を達成するためのルールを適用できます。

障害管理

ある程度複雑なシステムでは、どうしても障害は発生するものです。それらの障害を検知して対応し、再発を防止することが重要です。

AWS では、自動化を利用してモニタリングデータに反応できます。例えば、特定のメトリクスがしきい値を超えたとき、自動化されたアクションをトリガーさせて、問題を修正するように設定できます。また、本番環境のリソースに障害が発生した場合には、

そのまま本番環境で診断して修復するのではなく、まず新しいリソースに置き換えて、障害が発生したリソースを本番環境外で別途分析することができます。クラウドではシステム全体を一時的なバージョンとして低価格で立ち上げられるため、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することもできます。

このような信頼性に関する考慮事項に注意を促す質問を以下に掲載します。

信頼性 6: データをどのようにバックアップしていますか？

信頼性 7: システムがコンポーネントのエラーに耐えるようにどのように設計していますか？

信頼性 8: システムの弾力性をどのようにテストしていますか？

信頼性 9: 災害時のリカバリプランはどうなっていますか？

論理エラーおよび物理エラーの両方から確実に復旧できるよう、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ます。障害管理の要になることは、システムに対して自動化されたテストを頻繁に実施して障害を発生させ、そこからどのように復旧するかを確認することです。これをスケジュールに従って定期的に実施し、このようなテストがシステムの大幅な変更後にもトリガーし実施されるようにします。目標復旧時間 (RTO) や目標復旧時点 (RPO) などの KPI を積極的に追跡して、システムの回復性を評価します (特に障害テストシナリオにおいて)。KPI の追跡は、単一障害点を特定して解消するために役立ちます。目標は、システムの復旧プロセスを徹底的にテストし、問題が長引く場合であってもすべてのデータを回復して、お客様へのサービス提供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、通常の本番プロセスと同様に熟知し、訓練しておく必要があります。

AWS の主要なサービス

信頼性の達成に不可欠な AWS のサービスが、ランタイムのメトリクスをモニタリングする **Amazon CloudWatch** です。以下のサービスと機能によって、信頼性の 3 つの分野がサポートされます。

- **基盤**: IAM によって、AWS のサービスとリソースへのアクセスを安全にコントロールできます。Amazon VPC を使用すると、AWS クラウドに隔離されたプライベートなセクションをプロビジョンし、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。AWS Trusted Advisor によって、サービスの制限を視覚的に把握できます。AWS Shield は分散サービス妨害攻撃 (DDoS) からの保護を提供するマネージドサービスで、AWS で実行されているウェブアプリケーションを保護します。

- **変更管理**: AWS CloudTrail は、アカウントの AWS API コールを記録し、監査用のログファイルを送信します。AWS Config は AWS のリソースと設定の詳細なインベントリを提供し、設定変更を継続的に記録します。Auto Scaling は、デプロイされたワークロードの需要管理を自動化するサービスです。CloudWatch には、カスタムメトリクスなどのメトリクスに基づいてアラートを送信する機能があります。また、CloudWatch にはロギング機能もあり、リソースからのログファイルを集約するためにも使用できます。

- **障害管理**: AWS CloudFormation では AWS リソースの作成用テンプレートが提供され、リソースを規則的で予測可能な方法でプロビジョニングできます。Amazon S3 は高い耐久性を持つサービスで、バックアップの保持に使用できます。Amazon Glacier では高い耐久性のアーカイブを実現できます。AWS KMS は信頼性の高いキー管理システムで、AWS の多数のサービスと統合できます。

リソース

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

ドキュメント

- サービスの制限
- サービスの制限のレポートについてのブログ

- Amazon Virtual Private Cloud
- AWS Shield
- Amazon CloudWatch
- Amazon S3
- AWS KMS

ホワイトペーパー

- 信頼性の柱
- 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を管理する
- AWS 災害対策
- AWS Amazon VPC のネットワーク接続オプション

動画

- AWS のサービスの制限の管理方法
- Embracing Failure: Fault-Injection and Service Reliability

製品

- AWS プレミアムサポート
- Trusted Advisor

パフォーマンス効率

パフォーマンス効率の柱には、コンピューティングリソースを効率的に使用してシステム要件を満たし、需要の変化や技術の進歩に合わせてこの効率を維持する能力が含まれます。

パフォーマンス効率の柱では、設計の原則、ベストプラクティス、質問の概要について説明します。実装の手引きとなる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。

設計の原則

クラウドにおけるパフォーマンス効率には、5 つの設計の原則があります。

- **最新テクノロジーの標準化**: 技術の知識や複雑な作業をクラウドベンダーに任せることで、実装が難しかった技術を簡単に取り入れられます。IT チームが新しい技術のホスト方法や実行方法を学習しなくても、単純にサービスとして使用することができます。例えば、NoSQL データベース、メディア変換、機械学習は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わた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して提供され、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実現**: 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できます。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環境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**:サーバーレスアーキテクチャの使用により、従来のコンピューティングリソースを維持して、運用する必要がなくなります。例えば、ストレージサービスは静的ウェブサイトとして動作することができ、ウェブサーバーの必要がなくなります。また、イベントサービスではお客様のプログラムコードをホストすることができます。サーバーを管理する運用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を削減できます。このようなマネージドサービスは、クラウ

ドスケールで運用されているためです。

- **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**: 実現し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。例えば、データベースやストレージのアプローチを選択する際にデータのアクセスパターンを考慮します。

定義

クラウドにおけるパフォーマンス効率には、4 つの分野のベストプラクティスがあります。

1. **選択**
2. **レビュー**
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、進化し続ける AWS クラウドを最大限に活用できます。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把握し、それに対するアクションを実行できます。最後に、圧縮またはキャッシュの使用や、整合性要件の緩和など、アーキテクチャのトレードオフによってパフォーマンスを向上できます。

ベストプラクティス

選択

特定のシステムに最適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、多くの場合は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、さまざまなタイプと設定を利用できます。これにより、ニーズに適したアプローチを見つけることがより簡単になります。また、オンプレミスのインフラストラクチャでは簡単に実現できないオプションも利用できます。例えば、Amazon DynamoDB などのマネージドサービスでは、あらゆる規模で数ミリ秒台のレイテンシーを実現するフルマネージド型の NoSQL データベースを提供します。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します (パフォーマンス効率の質問、回答、ベストプラクティスの一覧については、付録を参照してください)。

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

アーキテクチャのパターンと実装を選択するときは、データ駆動型のアプローチを使用して最適なソリューションを判断します。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、AWS パートナーネットワーク (APN) 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。

ほとんどの場合、アーキテクチャには、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) を組み合わせます。アーキテクチャの実装では、アーキテクチャパフォーマンスの最適化に特化した AWS のサービスを使用します。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、ネットワーク) について説明します。

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、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てパフォーマンスを向上します。アーキテクチャに対して適切でないコンピューティングソリューション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、ボタンのクリックまたは API コールで性能を変更できます。クラウド上で決定したリソースを変更できるようになったため、異なるサーバータイプを使って実験を行うことができます。AWS では、さまざまなファミリーやサイズの仮想サーバーインスタンスを用意しており、ソリッドステートドライブ (SSD) やグラフィック処理ユニット (GPU) などの幅広い機能を使用することができます。
- **コンテナ**はオペレーティングシステムの仮想化方式の 1 つで、アプリケーションとその資源をリソース上隔離されたプロセスとして実行することができます。
- **関数**は実行するコードから実行環境を抽象化します。例えば、AWS Lambda では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。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

パフォーマンス効率 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、利用可能な伸縮性のメカニズムを活用し、需要が変化してもパフォーマンスを維持できるように十分なキャパシティーを確保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、オブジェクト)、アクセスパターン (ランダムまたはシーケンシャル)、スループットの要件、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します。

AWS では、ストレージが仮想化されているため、さまざまなタイプを利用できます。これにより、ストレージ方法をニーズに厳密に合わせることが簡単になり、オンプレミスのインフラストラクチャでは簡単に実現できないストレージオプションを選択することもできるようになります。例えば、Amazon S3 は、イレブンナイン (99.999999999%) の耐久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 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。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

パフォーマンス効率 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、必要なパフォーマンスを達成するために、アクセスパターンとの一致を確認することが重要です。

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、スケーラビリティ、クエリ機能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および機能を選択すると、パフォーマンス効率が低下する可能性があります。

Amazon RDS は、フルマネージド型のリレーショナルデータベースサービスです。Amazon RDS を使用すると、データベースのコンピューティングリソースやストレージリソースをスケールすることができます。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、フルマネージド型の 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更された場合にノードの数やタイプを変更できる、ペタバイトスケールのマネージド型データウェアハウスです。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

パフォーマンス効率 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アプローチ (RDBMS、NoSQL) 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によってではなく、企業の方針に従って選択される場合が多い分野です。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することが重要です。また、データベースではないソリューション (検索エンジンやデータウェアハウスの使用など) 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できます。

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って場所の選択肢は限られますが、エッジ技術やリソースの配置によって補うことができます。

AWS では、ネットワークは仮想化され、さまざまなタイプや設定で利用できます。これにより、ネットワーク方法をニーズにより適合させることが容易になります。AWS では、ネットワークトラフィックを最適化するための機能 (拡張ネットワーク、Amazon EBS 最適化インスタンス 、Amazon S3 Transfer Acceleration 、Amazon CloudFront など) を提供しています。また、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 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) も提供しています。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

パフォーマンス効率 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、場所を考慮する必要があります。AWS では、使用する場所の近くにリソースを配置して距離を縮めることができます。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できます。

レビュー

ソリューションを設計するときには、選択できる一定のオプションセットがあります。それでも、時間がたつと、アーキテクチャのパフォーマンスをさらに向上させることができる新しい技術とアプローチが利用できるようになります。

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。AWS は新しいリージョン、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。これらにより、アーキテクチャのパフォーマンス効率を大幅に改善できる可能性があります。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

パフォーマンス効率 6: 新しいサービスや機能をどのように取り入れていますか？

アーキテクチャのパフォーマンスが何で制約されているかを把握しておくと、その制約を解決するリリースに気付くことができます。

モニタリング

アーキテクチャを実装したら、パフォーマンスをモニタリングして、エンドユーザー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ときにアラームを送信するには、モニタリングメトリクスを使用する必要があります。アラームによって自動化されたアクションをトリガーし、パフォーマンスの悪いコンポーネントに対応することができます。

Amazon CloudWatch では、モニタリングと通知アラーム送信の機能を利用できます。自動化を使用して、Amazon Kinesis、Amazon Simple Queue Service (Amazon SQS)、AWS Lambda によっ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に対処できます。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

パフォーマンス効率 7: 期待通りのパフォーマンスを発揮しているかをどのようにモニタリングしていますか？

誤検出が大量に発生していないか、または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。自動化されたトリガーにより、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、問題を正しく検知できることを確認します。

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、容量に対して、時間やレイテンシーとのトレードオフを検討し、より高いパフォーマンスを実現できます。

AWS を使用すると、すぐにグローバル化を行うことができ、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の負荷を減らすこともできます。AWS では、インメモリデータストアや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。Amazon DynamoDB Accelerator (DAX) は、Amazon DynamoDB の前に配置するリードスルー/ライトスルーの分散型キャッシュで、Amazon DynamoDB を同一の API をサポートしつつ、キャッシュ内のエンティティに対してミリ秒未満のレイテンシーを達成しています。

このようなパフォーマンス効率に関する考慮事項に注意を促す質問を以下に掲載します。

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

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、重要な利

点があることを確認するには、ロードテストを実施する必要があります。

AWS の主要なサービス

パフォーマンス効率を高めるために不可欠な AWS のサービスは、**Amazon CloudWatch** です。このサービスを使用すると、リソースやシステムをモニタリングし、全体的なパフォーマンスと運用状態を可視化することができます。以下のサービスと機能によって、パフォーマンス効率の 4 つの分野がサポートされます。

- **選択**
  - **コンピューティング**: Auto Scaling は、需要を満たし、応答性を維持するための十分なインスタンスを確保するうえで重要です。
  - **ストレージ**: Amazon EBS は、ユースケースに合わせて最適化できるように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 秒あたりの入出力オペレーション (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、ユースケースに合わせて最適化できるように広範囲なデータベース機能 (PIOPS やリードレプリカなど) を提供します。Amazon DynamoDB は、あらゆる規模で数ミリ秒台のレイテンシーを実現します。
  - **ネットワーク**: Amazon Route 53 では、レイテンシーに基づくルーティングを行います。Amazon VPC エンドポイントと AWS Direct Connect を使用すると、ネットワークの距離やジッターを削減できます。
- **レビュー**: AWS ブログと AWS ウェブサイトの「最新情報」セクションは、新しくリリースされた機能や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、既存のモニタリング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クス、アラーム、通知を提供します。
- **トレードオフ**: Amazon ElastiCache、Amazon CloudFront、AWS Snowball は、パフォーマンスを向上することができるサービスです。Amazon RDS のリードレプリカを使用すると、読み込み負荷の高いワークロードをスケールできます。

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。

ドキュメント

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

ホワイトペーパー

- パフォーマンス効率の柱

動画

- AWS re:Invent 2016: Scaling Up to Your First 10 Million Users (ARC201)
- Performance AWS re:Invent 2016: Deep Dive on Amazon EC2 Instances, Featuring Performance Optimization (CMP301)

コスト最適化

コスト最適化の柱には、最も安価にシステムを実行してビジネス価値を実現する能力が

含まれます。

コスト最適化の柱は、設計の原則、ベストプラクティス、質問の概要について説明します。実装の手引きとなる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。

設計の原則

クラウドにおけるコスト最適化には、5 つの設計の原則があります。

- **消費モデルを導入する**: 詳細な予測を立てるのではなく、必要な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。例えば、開発環境とテスト環境は、通常、1 週間のうちの稼働日に 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 (168 時間に対して 40 時間)。

- **全体的な効率を評価する**: システムがもたらすビジネス上の成果と、そうした成果の実現に関連するコストを評価します。この評価結果を使用して、ビジネス的な成果の拡大と、コスト削減による利益を理解することができます。

- **データセンターの運用費を排除する**: お客様はサーバーのラッキング、スタッキング、電源供給といった手間のかかる作業を AWS に任せ、IT インフラストラクチャを気にせず自社の顧客やビジネスプロジェクトに集中できます。

- **支出を分析し、帰属させる**: クラウドでは、システムの使用状況とコストを簡単かつ正確に把握できるため、個々のビジネスプロジェクトに属する IT コストを明確に特定できます。これは投資収益率 (ROI) を評価する上で役立ちます。システムの所有者はリソースを最適化し、コストを削減できます。

- **所有コストを削減するためにマネージドサービスとアプリケーションサービスを使用する**: クラウドでマネージドサービスやアプリケーションサービスを利用すると、E

メールの送信やデータベースの管理といったタスクのためにサーバーを維持せずに済むため、運用上の負担を取り除くことができます。さらに、マネージドサービスはクラウド規模で運用されているため、トランザクションやサービスあたりのコストが低くなります。

定義

クラウドにおけるコスト最適化には、4 つの分野のベストプラクティスがあります。

1. **コスト効率に優れたリソース**
2. **需要と供給の一致**
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。例えば、最適化する必要があるのは市場投入までのスピードですか、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、迅速な市場投入、新機能の提供、または単純に期日の遵守といった、スピードの最適化が最善であることがあります。設計上の決定は、経験的データを活用せずにあわてて行われる場合があります。コストが最適化されたシステムを構築するためのベンチマークに多く時間を費やすよりも、「万が一」を想定して多めにリソースに投資した方が良いと思える場合があるためです。この結果、過剰にプロビジョニングされたリソースや、最適化されていないリソースがデプロイされることになり、そうした状態を解消することはライフサイクル全体を通して不可能です。以下のセクションでは、デプロイにおける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技術的および戦略的なガイダンスを説明します。

ベストプラクティス

コスト効率に優れたリソース

システムに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。例えば、レポート生成プロセスについて考えてみましょう。小規模なサーバーで実行すると 5 時間かかるプロセスであっても、時間あたりの料金が 2 倍の大規模なサーバーでは 1 時間で済む場合があります。どちらのサーバーでも結果は同じですが、小規模なサーバーでは時間の経過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、最もコスト効率に優れたリソースが使用されており、優れたコスト効果が多く得られます。また、マネージドサービスを使用してコストを削減することもできます。例えば、メール配信のために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されるサービスを使用します。

AWS では、柔軟でコスト効率に優れたさまざまな料金オプションを使用し、ニーズに最も合った方法で Amazon EC2 インスタンスや他のサービスを利用できます。オンデマンドインスタンスでは、初期費用は不要で、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きます。リザーブドインスタンスでは、キャパシティーを予約し、オンデマンド料金から最大 75% 割引された価格で利用できます。スポットインスタンスでは、オンデマンド料金から最大 90% 割引された料金で、使用されていない Amazon EC2 のキャパシティーを利用することができます。スポットインスタンスは、ステートレスなウェブサーバー、バッチ処理、HPC、ビッグデータのように、サーバー群における個々のサーバーが動的に追加/削除されても問題ないシステムに適しています。

適切なサービスの選択によっても、使用量やコストを節減できます (例として、CloudFront によってデータ転送量を最少化する)。また、コストを完全に排除することもできます (例として、RDS で Amazon Aurora を使用し、高額なデータベースのライセンスコストを除外する)。

このようなコスト最適化に関する考慮事項に注意を促す質問を以下に掲載します(コスト

最適化の質問、回答、ベストプラクティスの一覧については、付録を参照してください)。

コスト最適化 1: AWSサービス選択の際にコストをどのように評価していますか？

コスト最適化 2: コスト目標を達成するためにインスタンスタイプとサイズをどのように選択していますか？

コスト最適化 3: コスト削減のために料金モデルをどのように選択していますか？

コスト最適化 4: データ転送料金についてどのように計画していますか？

コストも考慮してサービスを選択し、Cost Explorer や AWS Trusted Advisor などのツールを使用して定期的に AWS 使用料を確認することで、使用状況を頻繁にモニタリングし、それに応じてデプロイを調整します。

需要と供給の一致

需要と供給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、供給に十分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、大きな管理コストが発生しないようにメトリ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 や、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、より多くのコストを削減し、システムのニーズに合わせてリソースを確保できます。

このようなコスト最適化に関する考慮事項に注意を促す質問を以下に掲載します。

コスト最適化 5: リソースの供給と顧客の需要をどのように一致させていますか？

需要と供給を一致させる設計時には、使用パターンや新しいリソースのプロビジョニン

グにかかる時間についてよく考慮する必要があります。

費用の把握

クラウドの優れた柔軟性と俊敏性により、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。また、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プロセスと時間 (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、配送のスケジューリング、リソースのデプロイなど) 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ります。

ビジネスの多くは、さまざまなチームによって運用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属するコストを明確にすることで、リソースを効率的に使用し、無駄を削減できます。コストの帰属を明確にすることで、実際に利益率の高い製品を把握し、予算の配分先についてより多くの情報に基づいた決定ができるようになります。

AWS では Cost Explorer を利用し、支出を追跡して実際の内訳を把握できます。AWS Budgets を使用すると、使用量やコストが予測を超えた場合に通知を受け取ることができます。リソースのタグ付けにより、使用状況やコストに、ビジネスや組織上の情報を結び付け、組織という観点から最適化をさらに詳しく理解できます。

このようなコスト最適化に関する考慮事項に注意を促す質問を以下に掲載します。

コスト最適化 6: AWS 使用量とコストをどのようにモニタリングしていますか？

コスト最適化 7: AWS 使用量をどのように管理していますか？

コスト最適化 8: 不要なリソースをどのように削除していますか？

コスト配分タグを使用して AWS 使用料とコストを分類および追跡できます。AWS リ

ソース (EC2 インスタンスや S3 バケットなど) にタグを適用すると、使用状況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 (コストセンター、システム名、所有者など) を表すタグを適用し、複数のサービスにわたってコストを分類します。

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) と、リソースのタグ付けを組み合わせることで、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特定できます。請求アラートを設定し、コスト超過の予測に関する通知を受け取ることができます。また、AWS 簡易見積りツールで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 による新しいサービスや機能のリリース時は、アーキテクチャに関する既存の決定事項を見直し、アーキテクチャが引き続き最もコスト効率に優れたものであることを確認するのがベストプラクティスです。要件の変更後は、必要なくなったリソース、サービス全体、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、ソリューションを大幅に最適化できることがよくあるため、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。

例えば、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。

このようなコスト最適化に関する考慮事項に注意を促す質問を以下に掲載します。

コスト最適化 9: 新しい AWS サービスをどのように評価していますか？

定期的にデプロイについてレビューする際は、より新しいサービスがコストを削減する

うえでどのように役立つかを評価します。例えば、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る場合があります。

AWS の主要なサービス

コスト最適化に不可欠なツールが **Cost Explorer** であり、複数のシステムにわたって、またビジネス全体を通して使用状況を明確に把握することができます。以下のサービスと機能によって、コスト最適化の 4 つの分野がサポートされます。

- **コスト効率に優れたリソース**: Cost Explorer を使用してリザーブドインスタンス購入のレコメンデーションを取得し、AWS のリソースにかかったコストのパターンを時間とともに確認します。Amazon CloudWatch や Trusted Advisor を使用すると、リソースを適切なサイズに保つことができます。RDS で Amazon Aurora を使用すると、データベースのライセンスコストを排除できます。AWS Direct Connect や Amazon CloudFront によって、データ転送を最適化できる可能性があります。

- **需要と供給の一致**: Auto Scaling によって、需要に合わせてリソースを追加または削除することで費用の超過を抑えます。

- **費用の把握**: AWS Cost Explorer で、使用状況の詳細を確認および追跡します。AWS Budgets では、使用量やコストが予算を超えた場合や、超えると予測された場合に通知を受け取ることができます。

- **継続した最適化**: AWS ニュースブログと、AWS ウェブサイトの「最新情報」セクションでは、新機能や新しいサービスについて紹介しています。AWS Trusted Advisor では 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のキャパシティーを契約したりすることで、コストを削減できる部分を発見できます。

リソース

コスト最適化のベストプラクティスについて詳しくは、以下の資料を参照してください。

ドキュメント

- Cost Explorer によるコストの分析
- AWS クラウドエコノミクスセンター
- AWS の詳細な請求レポート

ホワイトペーパー

- コスト最適化の柱

動画

- Cost Optimization on AWS

ツール

- AWS 総所有コスト (TCO) 計算ツール
- AWS 簡易見積りツール

レビュープロセス

アーキテクチャのレビューは、一貫性のある方法と「誰も責めない」アプローチで詳細に行う必要があります。レビューは短時間 (数日ではなく数時間) で行います。これは話し合いであり監査ではありません。アーキテクチャをレビューする目的は、対応が必要な深刻な問題や、改善できる部分を特定することです。レビュー結果は、お客様の改善のためのアクションとなります。

「アーキテクチャについて」セクションで説明したように、各チームメンバーにアーキテクチャの品質に対する責任を負ってもらう必要があります。形式ばったレビューミーティングを開くのではなく、アーキテクチャの構築に携わったチームメンバーが Well-Architected フレームワークを使用して継続的にアーキテクチャをレビューすることをお勧めします。継続的なアプローチによって、チームメンバーはアーキテクチャの発展に合わせて回答を更新し、アーキテクチャを改良しながら機能を提供していくことができます。

AWS Well-Architected では、AWS が社内でシステムとサービスをレビューする方法を採用しています。このフレームワークは、設計方法を左右する一連の設計の原則と、根本原因分析 (RCA) でよく問題となる分野が軽視されないようにするための質問を土台に構築されています。社内システム、AWS のサービス、お客様に重大な問題があれば、AWS では必ず RCA を確認し、使用するレビュープロセスについて改善の余地を検討します。

レビューは、ワークロードのライフサイクル中に複数回実施する必要があります。まず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。[4]また本番運用前にもレビューを行います。本番運用の開始後、

[4] 多くの決定は、行き来が自由なドアに似ています。つまり、取り消しが可能です。こうした決定はすばやく行います。一方通行のドア (のような決定) で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。

ワークロードは新しい機能の追加や、テクノロジーの実装の変更によって発展し続けます。ワークロードのアーキテクチャは継続的に変化していきます。アーキテクチャが発展していくなかでその特徴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。アーキテクチャに大幅な変更を加えた場合は、Well-Architected のレビューを含む一連の改善プロセスを実施します。

一度限りや単独の評価としてレビューを実施する場合は、全ての適切な関係者がその対話に参加できるよう手配してください。何を実装しているのかチームが完全に理解したのは、レビュー時が初めてだったということがよくあります。別のチームのワークロードをレビューするには、そのチームのアーキテクチャについて立ち話程度の会話を何度かすることも有効です。これにより多くの疑問に対する答えを得ることができます。その後、ミーティングを数回行い、不明瞭な部分や認識したリスクについて明確にしたり、掘り下げたりすることができます

ミーティングを実施する際の推奨事項を以下に記載します。

- ホワイトボードのあるミーティングルーム
- 印刷した構成図や設計ノート
- 回答に別途調査が必要な質問のアクションリスト (「暗号化を有効化したかどうか」など)

レビュー終了後は、問題リストを作成し、ビジネスの状況に応じて優先順位を決定します。また、そうした問題がチームの日常業務に及ぼす影響も考慮します。リストの問題に早期に対処すれば、繰り返し発生する問題の解決ではなく、ビジネス価値の創出に時間を用いることができます。問題に対応しながら、レビューを更新してアーキテクチャの改良を確認できます。

レビューの価値は明らかであっても、新しいチームにはすんなり受け入れてもらえないかもしれません。チームにレビューの利点を伝えることで、以下の対処が必要な反対意

見に対処できます。

- 「忙しすぎて時間がありません!」(チームが大規模なローンチに向けて準備しているときによく耳にします)
  - 大規模なローンチの準備時には、ローンチをスムーズに進めたいと思われることでしょう。レビューによって、これまで見逃していた問題を把握できます。
  - 設計ライフサイクルの初期段階でレビューを行うことで、リスクを明らかにし、機能提供のロードマップに合わせてリスクの軽減プランを立てることをお勧めします。
- 「結果が出たところで対応する時間がありません!」(大きなスポーツイベントなど、スケジュールの変更がきかないイベントがターゲットである場合によく耳にします)
  - これらのイベントの日程を動かすことはできませんが、本当に、アーキテクチャのリスクを把握しないままイベント当日を迎えたいと思いますか。問題すべてに対処することはできなくても、問題が実際に発生した場合に備えて、対応方法を記載したプレイブックを用意することはできます。
- 「他チームにソリューション実装の秘密を知られたくありません!」
  - チームに、Well-Architected フレームワークのホワイトペーパーの付録に記載された質問を見てもらえば、いずれの質問も、取引や技術に関する秘密情報を公開するものではないことを理解してもらえます。

組織内で複数のチームとレビューを複数回実施するなかで、根本的な問題を特定できる場合があります。例えば、複数のチームが特定の柱またはトピックについて一連の問題を抱えている可能性があります。すべてのレビューを包括的に検証し、そうした根本的な問題の対応に役立つメカニズム、トレーニング、プリンシパルエンジニアの講義があるかどうか見極めます。レビュー終了後は、でき上がった改善リストを使ってビジネスの状況に応じて対処の優先順位を決定します。また、そうした問題がチームの日常業務

に及ぼす影響も考慮します。これらの問題に対応することで、問題の解決に追われるのではなく、ビジネス価値の創出に時間をかけることができます。問題に対応しながら、レビューを更新することで、アーキテクチャの改良を確認できます。

まとめ

AWS Well-Architected フレームワークの 5 つの柱では、クラウド上で信頼性、セキュリティ、効率、コスト効率に優れたシステムを設計および運用するため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やアーキテクチャ案のレビューに用いる一連の質問が掲載されています。また、各柱について AWS のベストプラクティスが複数記載されています。アーキテクチャでこのフレームワークを活用することにより、安定した効率的なシステムを構築し、機能面の要件により注力できるようになります。

寄稿者

このドキュメントを作成するにあたり、以下の個人および組織から寄稿していただきました。

- Philip Fitzsimons: Well-Architected シニアマネージャー、アマゾン ウェブ サービス
- Rodney Lester: Well-Architected の信頼性担当リーダー、アマゾン ウェブ サービス
- Nathan Besh: Well-Architected のコスト担当リーダー、アマゾン ウェブ サービス
- Brian Carlson: Well-Architected の運用担当リーダー、アマゾン ウェブ サービス
- Jon Steele: シニアテクニカルアカウントマネージャー、アマゾン ウェブ サービス
- 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
- Ben Potter: Well-Architected のセキュリティ担当リーダー、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
- 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

参考資料

AWS 災害対策

AWS KMS

AWS セキュリティのベストプラクティス

AWS re:Invent 2016: Scaling Up to Your First 10 Million Users (ARC201)

AWS Amazon VPC のネットワーク接続オプション

AWS クラウドエコノミクスセンター

AWS クラウドセキュリティ

AWS コンプライアンス

AWS の詳細な請求レポート

AWS Limit Monitor

AWS プレミアムサポート

AWS リスクとコンプライアンス

AWS セキュリティブログ

AWS セキュリティの概要

AWS Security State of the Union

AWS Shield

AWS 簡易見積りツール

AWS 総所有コスト (TCO) 計算ツール

AWS Well-Architected のホームページ

Amazon S3

Amazon CloudWatch

Amazon EBS ボリュームのパフォーマンス

Amazon S3 パフォーマンスの最適化

Amazon Virtual Private Cloud

Amazon leadership principles

Cost Explorer によるコストの分析

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

Cost Optimization on AWS

コスト最適化の柱

DevOps と AWS

DevOps at Amazon

Embracing Failure: Fault-Injection and Service Reliability

AWS のサービスの制限の管理方法

大規模環境での AWS インフラストラクチャを管理する

運用上の優秀性の柱

Performance AWS re:Invent 2016: Deep Dive on Amazon EC2 Instances, Featuring Performance Optimization (CMP301)

パフォーマンス効率の柱

信頼性の柱

セキュリティの柱

サービス
制限

サービスの制限のレポートについてのブログ

責任共有モデル
の概要 TOGAF

Trusted Advisor

Zachman フレームワーク

ドキュメント改訂履歴

大幅な改訂:

日付	説明
2018 年 6 月	質問文の簡素化、回答の標準化、読みやすさの向上のために更新しました。
2017 年 11 月	第 1 の柱として運用上の優秀性を移動し、他の柱の枠組みとなるよう改訂しました。AWS の進化を反映するように他の柱を更新しました。
2016 年 11 月	フレームワークを更新しました。運用上の優秀性の柱を追加し、他の柱を改訂、更新して重複箇所を削除しました。また、何千人ものお客様とのレビューから得られた知識を取り入れました。
2015 年 11 月	Amazon CloudWatch Logs の最新情報を付録に追記しました。
2015 年 10 月	初版

付録: Well-Architected に関する質問、回答、ベストプラクティス

運用上の優秀性

準備

運用上の優秀性 1: オペレーションの優先順位を決定する要因は明確ですか？

運用上の優先事項とは、運用作業において焦点を当てる分野です。運用上の優先事項を明確に定義し、その内容に同意することで、運用作業に関連した利点を最大化できます。

ベストプラクティス:

- **ビジネスニーズを評価する**: 運用上の優先事項の決定には、ビジネスチームと開発チームの両方に参加してもらいます。ビジネスで成果を達成するために必要な運用上のサポートを完全に理解する必要があります。
- **コンプライアンス要件を評価する**: 規制コンプライアンスの要件や業界標準などの外部要素を考慮し、潜在的な準拠事項を把握するようにします。こうした準拠事項によって、運用上の特定の優先事項について遵守が必要になったり、重要性が明確になったりする場合があります。

リスクを評価する: 運用上の優先事項は、競合する利益間のトレードオフによって成り立っていることがよくあります。例えば、新しい機能の市場投入時間を短縮するためには、コスト最適化を犠牲にすることが必要な場合があります。潜在的な利益に対するリスクを考慮し、十分な情報に基づいて運用上の優先事項を決定します。

運用上の優秀性 2: 運用性を向上させるためのワークロード設計をしていますか？

ワークロードの存続期間の大部分は、通常、運用されている状態です。ワークロードを長期間維持できるようにするために、システム設計の一環として運用ニーズを考慮してください。

ベストプラクティス:

- **設計標準を共有する**: 複数のチーム間で既存のベストプラクティス、ガイダンス、ガバナンスの要件を共有し、システム設計へ共有の設計標準を取り入れることで、複雑性を軽減して開発作業の利点を最大化します。設計標準に対する変更、追加、例外をリクエストする手順を作成し、継続的な改善とイノベーションをサポートします。

- **クラウド運用に向けて設計する**: ワークロード設計でクラウド環境の特徴 (伸縮性、オンデマンドのスケーラビリティ、従量制料金、自動化など) を活用し、迅速な反復作業による改善や低リスクの実験といった運用機能を実現します。

- **ワークロードの動作を深く理解する**: システム設計に測定ツール (ログ、メトリクス、カウンター) を組み込むことで、システムで何が起こっているかを把握し、個々のコンポーネント単位でシステムのパフォーマンスを測定できるようにします。

- **顧客の行動を深く理解する**: システム設計に測定ツール (ログ、メトリクス、カウンター) を組み込むことで、顧客によるシステムの使用状況と、顧客へのサービス品質を把握できるようにします。

- **障害を軽減し、簡単に修正を行い、フローを改善するためのプラクティスを実装する**: フローを改善し、品質、リファクタリング、バグ修正に対する迅速なフィードバックが可能なアプローチを導入し、デプロイ作業で発生した問題をすばやく特定および修正できるようにします。

- **デプロイのリスクを軽減する**: 頻度の高い変更の実施 (小規模かつ取り消し可能なもの)、自動デプロイ、テスト、カナリアデプロイまたはワンボックスデプロイ、ブルーグリーンなどのアプローチを導入し、変更の実装中に発生した問題の影響を軽減して迅速な復旧を可能にします。

運用上の優秀性 3: ワークロードが運用可能であることをどのように確認していますか？

ワークロード、プロセス、手順、担当者について運用への対応準備を評価し、ワークロードに関連した運用上のリスクを把握します。

ベストプラクティス:

- **継続的改善の文化を育てる**: 継続的な改善の文化を育て、担当者が改善の機会をとらえて適切な対応ができるようにします。変更は定期的に発生すること、障害は必ず発生すること、改善とイノベーションは実験によって達成できることを強調して継続的改善の文化を育てます。求められる成果が未達成の場合も批判しないようにすることで、安心して実験に取り組める環境を提供します。

- **ビジネスに対する価値の理解を共有する**: ビジネスに対するワークロードの重要性をすべてのチームで共有し、運用上の問題に対応するために必要なリソースをすべてのチームが活用できる手順を確立します。

- **担当者の能力を確保する**: 運用上のニーズに対応するため、トレーニングを受けた担当者が適切な人数配置されていることを検証するメカニズムを確立します。効果的な対応を継続的に行うため、必要に応じて担当者をトレーニングし、能力を向上させます。

- **ガバナンスとガイダンスを文書化し、共有する**: コンプライアンスについて、入手可能で、理解しやすく、評価可能な標準を共有し、担当者がコンプライアンスを達成できるようにガイドおよび教育します。標準に対する変更、追加、例外をリクエストする手順を作成し、継続的な改善とイノベーションをサポートします。

- **チェックリストを使用する**: チェックリストを使用し、ワークロードを運用する準備ができていることを一貫性のある方法で評価します。チェックリストには、少なくともチームとワークロードの運用への対応準備に関する項目と、セキュリティの考慮事項を記載する必要があります。チェックリストの項目を遵守する必要がある場合や、そうした項目の確認後、リスクを考慮して要件がすべて満たされていないワークロードを運用する判断を下せる場合があります。チェックリストの項目はあるワークロードやアーキテクチャに固有のものとすることも、実装依存とすることもできます。必要に応じてチェックリストをスクリプト化および自動化し、一貫性のある方法で迅速に実行し、人為的ミスが発生しないようにします。

- **ランブックを使用する**: 特定の結果を達成するための手順を文書化し、既知のイベントに一貫性のある方法で迅速に対応できるようランブックを作成します。効果的な手順には、適切なスキルを持ったスタッフが求められる成果を達成するうえで最低限の情報を記載します。必要に応じてランブックをスクリプト化および自動化し、一貫性のある方法で迅速に対応し、人為的ミスが発生しないようにします。

- **プレイブックを使用する**: 根本的な問題を特定するための手順を文書化し、障害シナリオに一貫性のある方法で迅速に対応できるようプレイブックを作成します。効果的なプロセスでは、適切なスキルを持ったスタッフが障害の潜在的な原因を特定し、障害を切り分けし、根本的な原因と修正方法を特定できます。必要に応じてプレイブックをスクリプト化および自動化し、一貫性のある方法で迅速に対応し、人為的ミスが発生しないようにします。

- **復旧演習を行う**: 潜在的な障害のシナリオを特定し、可能な場合は障害の原因を取り除きます。また、障害への対処方法を作成およびテストし、迅速かつ効果的に対応して発生時の影響を軽減できるようにします。

運用

運用上の優秀性 4: 運用状態をどのように確認していますか？

ワークロードとプロセスを評価するためのメトリクスを定義し、ビジネス成果を達成するうえでの運用の有効性を理解します。適切なアクションを実行できるように、メトリクスを取得および分析してプロセスとイベントを可視化します。

ベストプラクティス:

- **ビジネスと顧客について求められる成果を定義する**: ワークロードに関連して成功とはどのようなことを指すかをビジネスと顧客の観点から定義して文書化し、運用の成功を評価する際の基準として使用します。

- **成功のメトリクスを特定する**: ワークロードの動作をビジネスと顧客の要求事項に照らして評価するためのメトリクスを定義し、ワークロードではそうした事項が達成されているかどう

かを確認します。

- **ワークロードのメトリクスを特定する**: 成功のメトリクスに照らしてワークロードとそのコンポーネントのステータスを評価するためのメトリクスを定義し、ワークロードではそれらのメトリクスが達成されているかどうかを確認します。
- **運用のメトリクスを特定する**: 運用アクティビティ (ランブックおよびプレイブック) の実行を評価するためのメトリクスを定義し、ビジネスニーズに応える運用上の結果を達成しているかどうかを確認します。
- **基準値を定める**: メトリクスの基準値を定め、比較基準として求められる値を設定します。
- **メトリクスを収集および分析する**: メトリクスのレビューを定期的かつ積極的に実行し、トレンドを特定して適切な対応が必要な分野を判断します。
- **インサイトを検証する**: 分析結果をレビューし、職能上の枠を超えたチームとビジネスオーナーと共に対応します。これによって共通の理解を確立し、影響を詳しく特定し、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ます。
- **運用のビジネスレベルビュー**: 運用のビジネスレベルビューを作成してニーズに応えられているかどうかを判断し、ビジネスの目標を達成するために改善が必要な分野を特定します。

運用上の優秀性 5: 運用上のイベントをどのように管理していますか?

運用イベントに対応するための手順を作成および検証し、ワークロードが中断される可能性を最小化します。

ベストプラクティス:

- **ビジネスへの影響に基づいて運用イベントの優先順位を決定する**: ビジネスへの影響に基づいて運用イベントの優先順位を決定し、複数のイベントへの介入が必要な場合に、ビジネスにとって最重要なイベントに最初に対応できるようにします。例えば、死亡や怪我、経済的損失、評価または信頼の低下などの影響が考えられます。
- **イベント、インシデント、問題を管理するためのプロセス**: 観察されたイベント、介入が必要なイベント (インシデント)、介入が必要で、繰り返し発生するか現時点では解決できないイ

ベント (問題) に対応するためのプロセスを確立します。そのプロセスを使用してイベントに適切なタイミングかつ方法で対応し、これらのイベントによるビジネスと顧客への影響を軽減します。

- **アラートごとのプロセス**: アラートを設定したあらゆるイベントに対して、適切に定義した対応方法 (ランブックまたはプレイブック) を確立し、責任者を明確にします。これによって運用イベントに効果的かつ迅速に対応し、重要なイベントが重要度の低い通知によって無視されてしまうことを防げます。

- **意思決定者を特定する**: 組織を代表して運用上のアクションを判断する権利を持つ意思決定者を特定します。運用上のアクティビティがビジネスの成果に影響を与える可能性がある場合は、必要に応じて意思決定者にエスカレーションし、十分な情報に基づいた判断ができるようにします。意思決定者は必要に応じてランブックとプレイブックを事前に承認し、イベントに迅速に対応します。

- **エスカレーションパスを定義する**: ランブックおよびプレイブックでエスカレーションパス (エスカレーションがトリガーされる理由、エスカレーション手順、各アクションに対して明確に特定された責任者など) を定義し、運用イベントに効果的かつ迅速に対応できるようにします。エスカレーションにはサードパーティが関係する場合もあります。

- **プッシュ通知**: ユーザーが利用するサービスに影響が出た場合、またサービスが通常の運用状態に戻った場合にユーザーに直接メッセージを送信し (E メールまたは SMS など)、ユーザーがそれに応じて適切なアクションを取れるようにします。

- **ダッシュボードを通じてステータスを通知する**: 利用対象者 (社内テクニカルチーム、リーダーシップ、顧客など) に合わせたダッシュボードを提供し、
ビジネスの現在の運用ステータスを通知して関連するメトリクスを提示します。例えば、Amazon CloudWatch ダッシュボード、AWS Personal Health Dashboard、サービスヘルスダッシュボードなどを利用できます。

- **根本原因分析のためのプロセス**: イベントの根本原因を特定および文書化するためのプロセスを確立し、イベントの再発を制限または防止するための軽減対策と、迅速かつ効果的に対応するための手順を作成します。必要に応じて、利用対象者に合わせて根本原因を通知しま

す。

進化

運用上の優秀性 6: 運用をどのように進化させていますか？

継続的かつ段階的な改善のために時間とリソースを割り当て、運用の有効性と効率性を向上させます。

ベストプラクティス:

- **継続的な改善プロセス**: 継続的かつ段階的に運用プロセスを改善していくために時間とリソースを割り当てます。改善できる分野を定期的に評価し、そうした分野への対応を優先し、最大の成果が見込める分野にリソースを集中して割り当てます。
- **改善の要因を定義する**: 改善の要因を定義して改善できる分野を評価し、そうした分野への対応を優先します。必要な機能、能力、改善、許容できない問題、バグ、脆弱性、コンプライアンスへの準拠の維持に必要な更新 (ポリシー、サードパーティによるサポートを活用) を考慮に入れて改善できる分野を評価します。
- **フィードバックループを取り入れる**: 手順にフィードバックループを取り入れ、問題と改善が必要な分野を特定します。
- **得られた知識を文書化し共有する**: 運用アクティビティの実行から得られた知識を文書化および共有し、社内で、またチームの枠を超えて活用します。得られた知識のトレンドを分析して調査すべき問題と分野を確認し、改善の機会を特定します。
- **運用メトリクスをレビューする**: 異なるビジネス分野からさまざまなチームを募って過去の運用メトリクスをレビューし、改善できる分野や、一連のアクション候補を特定し、得られた知識を共有します。

セキュリティ

アイデンティティとアクセスの管理

セキュリティ 1: ワークロードの認証情報をどのように管理していますか？

認証情報とは、ワークロードを管理するためのアクセス権を直接または間接的に付与するパスワード、トークン、キーを指します。適切なメカニズムで認証情報を保護し、誤用または悪意のある使用のリスクを軽減します。

ベストプラクティス:

- **Multi-Factor Authentication (MFA) を導入する**: ソフトウェアトークンやハードウェアトークンを使用した Multi-Factor Authentication (MFA) を導入し、アクセス制御を強化します。
- **パスワードの要件を導入する**: パスワードの最小長と複雑さの要件を導入し、総当たり攻撃やその他のパスワード攻撃から保護します。
- **認証情報を定期的に更新する**: 認証情報を定期的に更新し、古い認証情報が以前のシステムや担当者によって使用されるリスクを軽減します。
- **認証情報を定期的に監査する**: 認証情報を監査し、適切な対策 (MFA など) が実施され、定期的に更新され、適切なアクセスレベルが設定されていることを確認します。
- **一元化されたIDプロバイダーを使用する**: IDプロバイダーやディレクトリサービスを使用してユーザーの認証を一元的に行い、複数の認証情報の必要性や管理の複雑さを低減します。

セキュリティ 2: AWS サービスへの人為的なアクセスをどのように制御していますか？

定義、制限、分離が適切に行われたアクセス権を使用して人間によるサービスへのアクセスを制御し、不正アクセスのリスクを軽減します。

ベストプラクティス:

- **認証情報を共有しない**: 認証情報はユーザー間で共有せず、ユーザーの分離とトレーサビリティを実現します。
- **ユーザーのライフサイクルによって管理する**: アクセスは、従業員のライフサイクルポリシーを使用して管理し、有効なユーザーにのみアクセス権を付与します。
- **最小権限を付与する**: ユーザーには、業務を遂行するために必要な最小限の権限のみを付与し、不正アクセスのリスクを軽減します。
- **アクセス要件を明確に定義する**: ユーザーの職務や役割に対してアクセス要件を明確に定義し、不要な権限を付与してしまうリスクを軽減します。
- **アクセス権はロールやフェデレーションを使用して付与する**: ロールを使用すると、安全なクロスアカウントアクセスとフェデレーティッドユーザーを実現できます。

セキュリティ 3: AWS サービスへのプログラムによるアクセスをどのように制御していますか?

適切に制限された認証情報とロールを短期間付与することによりプログラムによるサービスへのアクセス、またはサービスへの自動アクセスを制御し、不正アクセスのリスクを軽減します。

ベストプラクティス:

- **認証情報を共有しない**: プログラムによるアクセスに使用される認証情報は、システム間で共有しません。
- **動的な認証**: 認証情報はサービスやシステムから動的に取得し、頻繁に更新します。
- **最小権限を付与する**: プログラムによるアクセスの要件を明確に定義し、最小権限のみをシステムに付与することで、不正アクセスのリスクを軽減します。
- **アクセス要件を明確に定義する**: アクセス要件を明確に定義し、不要な権限を付与してしまうリスクを軽減します。

発見的統制

セキュリティ 4: ワークロードのセキュリティイベントをどのように検知していますか？

ログとメトリクスを取得して分析し、セキュリティの脅威とイベントを可視化して適切なアクションを実行できるようにします。

ベストプラクティス:

- **利用可能なロギングを有効化する**: すべてのサービスと機能に対してロギングを有効にすることで、イベントの可視性が向上します。
- **AWS CloudTrail を分析する**: 疑わしい動作を検出するため CloudTrail トレイルが自動的に分析されるよう設定する必要があります。
- **ログを一元的に分析する**: すべてのログが一元的に収集され、自動的に分析されるよう設定し、疑わしい動作を検出する必要があります。
- **重要なメトリクスとイベントをモニタリングし、アラートを送信する**: セキュリティに関する主なメトリクスとイベントがモニタリングされ、自動的にアラートが送信されるよう設定します。
- **AWS Marketplace や APN パートナーのソリューションを活用する**: AWS Marketplace や APN パートナーのソリューションを活用します。

インフラストラクチャ保護

セキュリティ 5: ネットワークをどのように保護していますか？

パブリックとプライベートのネットワークとサービスでは、多層防御を使用してネットワークベースの脅威からワークロードを保護する必要があります。

ベストプラクティス:

- **Virtual Private Cloud (VPC) でトラフィックを制御する**: VPC を使用してワークロードのトラフィックを分離し、制御します。
- **トラフィックを境界で制御する**: トラフィックをワークロードの境界やネットワークエッジで制御することで、トラフィックを制御し、保護を強化できる最初の機会を利用します。
- **利用可能な機能を使用してトラフィックを制御する**: セキュリティグループ、ネットワーク ACL、サブネットなどの利用可能な機能を使用してトラフィックを制御することで、保護を強化します。
- **AWS Marketplace や APN パートナーのソリューションを活用する**: AWS Marketplace や APN パートナーのソリューションを活用します。

セキュリティ 6: AWS のセキュリティ機能と一般的なセキュリティ脅威に関する最新情報をどのように確認していますか？

最新の情報を入手し、AWS と業界のベストプラクティス (サービスや機能など) を実装することで、ワークロードのセキュリティを向上させます。最新のセキュリティ脅威を理解することで脅威モデルを構築し、保護対策を特定および実装します。

ベストプラクティス:

- **新しいセキュリティのサービスと機能を評価する**: セキュリティのサービスと機能がリリースされたら、そうしたサービスと機能を調べ、現在のセキュリティ体制を強化するために適切な保護を特定します。
- **セキュリティのサービスと機能を使用する**: セキュリティのサービスと機能を採用することで、ワークロードを保護するための対策を導入します。

セキュリティ 7: コンピューティングリソースをどのように保護していますか？

管理可能なコンポーネントを使用してコンピューティングリソースを設定して整合性を保護お

よびモニタリングし、適切なアクションを実行できるようにします。

ベストプラクティス:

- **デフォルトの設定を強化する**: コンピューティングのセキュリティを向上させるために、要件に合わせてコンピューティングリソースを設定および強化します。
- **ファイルの整合性を確認する**: ファイルの整合性を確認し、許可されていない変更を把握して適切なアクションを実行します。
- **侵入検知を利用する**: ホストベースのエージェントなど、侵入検知対策によって可視性を高めます。
- **AWS Marketplace や APN パートナーのソリューションを活用する**: AWS Marketplace や APN パートナーのソリューションを活用します。
- **設定管理ツールを利用する**: 設定管理サービスや設定管理ツールを使用し、デフォルトで自動的に安全な設定が適用されるようにします。
- **パッチ適用と脆弱性スキャンを実行する**: 定期的にパッチを適用し、ツールを使用して脆弱性をスキャンし、新しい脅威からの保護を実現します。

データ保護

セキュリティ 8: データをどのように分類していますか？

分類を使用して重要度に基づいてデータを分類し、適切な保護対策を判断します。

ベストプラクティス:

- **データ分類スキーマを使用する**: データ分類スキーマを使用してデータを分類します。
- **データ分類を適用する**: データは、スキーマの分類に基づいて保護します。

セキュリティ 9: データ保護メカニズムをどのように管理してしますか?

データ保護メカニズムには、転送中および保管中のデータを保護するサービスやキーがあります。このようなサービスとキーを保護し、システムとデータへの不正アクセスのリスクを軽減します。

ベストプラクティス:

- **安全なキー管理サービスを使用する**: AWS KMS、CloudHSM などのキー管理サービスを使用します。
- **サービスレベルの制御機能を使用する**: KMS 管理キーを使用した Amazon S3 の暗号化など、組み込みのサービスレベルの制御機能を使用します。
- **クライアントサイドのキー管理を使用する**: クライアント側の手法を使用してキーを管理します。
- **AWS Marketplace や APN パートナーのソリューションを利用する**: AWS Marketplace や APN パートナーのソリューションを使用します。

セキュリティ 10: 保存中のデータをどのように保護していますか？

保管中のデータを保護し、不正アクセスやデータ損失のリスクを軽減します。

ベストプラクティス:

- **保管中のデータを暗号化する**: 保管中のデータを暗号化します。

セキュリティ 11: 転送中のデータをどのように保護していますか？

転送中のデータを保護し、不正アクセスやデータ漏えいのリスクを軽減します。

ベストプラクティス:

- **転送中のデータを暗号化する**: 必要に応じて TLS または同等の技術を通信に使用します。

インシデント対応

セキュリティ 12: インシデントに対してどのように備えていますか？

セキュリティインシデントを調査して対応し、ワークロードの中断を最小限に抑えることができるよう準備します。

ベストプラクティス:

- **アクセス権を事前に設定する**: 情報セキュリティチームに適切なアクセスを事前に付与しておくか、すぐにアクセスを取得できるようにしておきます。このアクセスは、インシデントに対して適切に対応できるよう事前にプロビジョニングしておきます。
- **ツールを事前にデプロイする**: 情報セキュリティチームがインシデントに対して適切に対応できるよう、AWS に事前にデプロイされた適切なツールを用意しておきます。
- **ゲームデーを実施する**: インシデント対応シミュレーションを定期的に実行し、得られた知識をアーキテクチャや運用に活用します。

信頼性

基盤

信頼性 1: AWS サービスの制限をどのように管理していますか？

AWS アカウントのプロビジョニング時には、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の制限が設定されます。また、API を呼び出すことができる回数も制限されており、AWS インフラストラクチャが保護されます。AWS のサービスに関するニーズを評価し、各リージョンの制限に対して適切な変更をリクエストします。

ベストプラクティス:

- **能動的にモニタリングし、制限を管理する**: Amazon CloudWatch やサードパーティの製品を使用して AWS の使用量を予測し、必要に応じてリージョンの制限を緩和し、予定されている使用量の増加に対応します。
- **制限のモニタリングと管理について自動化を実装する**: AWS SDK を使用し、しきい値に近づいたときに通知するツールを実装します。エンタープライズサポートを契約している場合、制限の緩和リクエストが自動的に作成されるよう設定することもできます。
- **サービスに固定の制限に注意する**: 変更できないサービスの制限に注意し、それらを考慮した設計を行います。
- **フェイルオーバー時に備え、現在のサービスの制限と最大使用量の間に十分な余裕を確保する**: フェイルオーバーは、設備で故障が生じた際に発生します。これは、お客様のアーキテクチャの分離ゾーン、アベイラビリティーゾーン (AZ)、または AWS リージョンであることもあります。分離ゾーンや AZ 内でフェイルオーバーが実行されると、障害が発生したリソースの削除前に自動化によってリソースのリクエストが行われ、予定していた制限を上回ってしまうことがあります。リソースが完全に廃棄される前に分離ゾーンの障害に対応するため、リソースのリクエストが可能なように設定しておきます。

- **関連するすべてのアカウントとリージョンでサービスの制限を管理する**: 複数の AWS アカウントや AWS リージョンを使用している場合、本番ワークロードを実行しているすべての環境で必ず同じ制限をリクエストします。

信頼性 2: AWS 上でのネットワーク構成をどのように設計していますか？

アプリケーションは、EC2-Classic、デフォルト VPC、お客様が作成した VPCのうち、1 つ以上の環境で実行できます。ネットワークの考慮事項 (システム接続性、Elastic IP アドレスとパブリック IP アドレスの管理、VPC とプライベートアドレスの管理、名前解決など) は、クラウドのリソースを使用するうえで基礎となるものです。入念に計画され、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:

- **データセンターとの接続が不要である**: 既存のオンプレミスネットワークへの接続が必要ない場合は、簡単に計画を立てることができます。
- **AWS とオンプレミス環境の間に可用性の高い接続を実装する**: 複数の Direct Connect 回線と複数の VPN トンネルを使用します。複数の Direct Connect ロケーションを使用して高可用性を実現します。複数の AWS リージョンを使用している場合、少なくとも 2 つのリージョンに複数の Direct Connect ロケーションが必要になります。AWS Marketplace を確認し、VPN を終端するアプライアンスを検討するようお勧めします。AWS Marketplace のアプライアンスを使用する場合は、高可用性を実現するために複数のアベイラビリティーゾーンで冗長インスタンスをデプロイします。
- **ワークロードのエンドユーザー向けに可用性の高いネットワーク接続を実装する**: 可用性の高い DNS、負荷分散、リバースプロキシをアプリケーションのパブリック向けエンドポイントとして使用します。AWS Marketplace を確認し、負荷分散やプロキシを行うアプライアンスを検討するようお勧めします。
- **複数の VPC で重複しないプライベート IP アドレスの範囲を使用する**: VPN を経由してピア接続または接続する場合は、各 VPC の IP 範囲が競合しないようにする必要があります。オンプレミス環境や他のクラウドプロバイダーとのプライベート接続にも同じことが当てはま

ります。

- **IP サブネットの割り当ては拡張性と可用性を考慮して行う**: Amazon VPC の各 IP アドレス範囲は、将来の拡張や複数のアベイラビリティーゾーンのサブネットに対する IP アドレス割り当てを考慮して十分な広さを確保し、アプリケーションの要件を満たすことができるようにします。また、将来の拡張時に使用できる IP アドレスも確保しておきます。

変更管理

信頼性 3: システムに対する需要の変化にはどのように対応していますか？

スケーラブルなシステムは、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:

- **ワークロードを自動的にスケールする**: Amazon S3、Amazon CloudFront、Auto Scaling、Amazon DynamoDB、Amazon Aurora、Elastic Load Balancing、AWS Lambda などの自動的にスケールされるサービスを使用するか、サードパーティのツールや AWS SDK を使用して作成した自動化を使用します。
- **ワークロードの負荷テストを実施する**: ロードテスト手法を採用し、規模の拡大や縮小がワークロードの要件を満たすかどうかを評価します。

信頼性 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、ワークロードの健全性についてインサイトを得るための強力なツールです。ログとメトリクスをモニタリングし、しきい値を超えた場合や、重大なイベントが発生した場合に通知を送信するようシステムを設定します。低パフォーマンスのしきい値を下回るか、障害が発生した場合、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システムを構築することが理想的です。

ベストプラクティス:

- **すべての階層でワークロードをモニタリングする**: Amazon CloudWatch やサードパーティのツールを使用してワークロードの階層をモニタリングします。Personal Health Dashboard を使用して AWS のサービスをモニタリングします。
- **通知をモニタリングに基づいて送信する**: 重大なイベントを把握しておく必要のある組織は、そうしたイベントの発生時に通知を受け取ることができます。
- **イベントに自動的に応答する**: 自動化を使用し、イベントの検出時に、障害が発生したコンポーネントの置き換えといったアクションを実行します。
- **定期的にレビューを実施する**: 重大なイベントや変更に応じてシステムのモニタリングを定期的にレビューし、アーキテクチャと実装を評価します。

信頼性 5: 変更をどのように実施していますか？

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 リソースとワークロードの変更制御は、ワークロードと運用環境で既知のソフトウェアが実行されており、予測可能な方法でソフトウェアのパッチ適用や置換が行われるために必要です。

ベストプラクティス:

- **自動化を使用して変更をデプロイする**: デプロイとパッチ適用を自動化します。

障害管理

信頼性 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、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) 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:

- **手動でデータをバックアップする**: 重要なデータは、Amazon S3、Amazon EBS スナップシ

ョット、サードパーティのソフトウェアを使用してバックアップし、RPO を満たします。

- **自動化されたプロセスを使用してデータをバックアップする**: AWS の機能 (例、Amazon RDS と Amazon EBS のスナップショット、Amazon S3 のバージョン)、AWS Marketplace のソリューション、サードパーティのソリューションを使用してバックアップを自動化します。
- **データを定期的に復旧し、バックアップの整合性とプロセスを確認する**: 復旧テストを実施し、バックアッププロセスの実装が目標復旧時間と目標復旧時点を満たしていることを確認します。
- **バックアップをセキュリティで保護し、暗号化する**: 『AWS セキュリティのベストプラクティス』ホワイトペーパーを参照してください。

信頼性 7: システムがコンポーネントのエラーに耐えるようにどのように設計していますか？

高可用性と平均修復時間 (MTTR) の速さに関する要件がワークロードにある場合 (暗黙的か明示的かを問わず)、回復力を高めるようワークロードを設計し、機能停止に耐えうるようワークロードを分散します。

ベストプラクティス:

- **ワークロードのすべてのレイヤーでモニタリングを実行し、エラーを検知する**: システムの健全性を継続的にモニタリングし、パフォーマンスの低下と完全な障害を報告します。
- **複数のアベイラビリティーゾーン (必要に応じて複数の AWS リージョン)にデプロイする**: ワークロードの負荷を複数のアベイラビリティーゾーンと AWS リージョンに分散します (例、DNS、ELB、Application Load Balancer、API Gateway)。
- **依存関係を疎結合にする**: キューイングシステム、ストリーミングシステム、ワークフロー、ロードバランサーなどの依存関係は疎結合にします。
- **グレースフルデグラデーションを実装する**: コンポーネントの依存関係に異常がある場合、コンポーネントそのものが異常と報告されることはありません。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。

- **自動修復をすべてのレイヤーに実装する**: 障害の検出時に、自動機能を使用して修復アクションを実行します。
- **可用性に影響するイベントの発生時に通知を受け取る**: 重大なイベントの検出時に、そのイベントが自動的に修復された場合でも通知が送信されます。

信頼性 8: システムの弾力性をどのようにテストしていますか？

ワークロードの弾力性をテストすると、本番環境にデプロイするまでわからない隠れたバグを見つけることができます。このテストを定期的に行います。

ベストプラクティス:

- **プレイブックを使用する**: 想定外の障害シナリオに備えてプレイブックを作成します。
- **障害注入テストを行う**: 意図的な障害を定期的にテストし、障害経路の範囲を確認します。
- **ゲームデーをスケジュールに組み込む**: ゲームデーを使用し、障害発生時の手順を定期的に演習します。
- **根本原因分析 (RCA) を実行する**: 重大なイベントに基づいてシステムの障害を確認し、アーキテクチャを評価して根本原因を特定します。必要に応じてこの原因を他の人に連絡する手段を用意しておきます。

信頼性 9: 災害時のリカバリプランはどうなっていますか？

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

ベストプラクティス:

- **復旧目標を定義する**: 目標復旧時間 (RTO) と目標復旧時点 (RPO) を定義します。

- **復旧戦略を定義する**: 目標を満たすことができるように災害対策 (DR) 戦略を定義します。
- **構成のずれを管理する**: AMI とシステム構成の状態および AWS のサービスに対する制限が DR サイトやリージョンで最新の状態であることを確認します。
- **災害対策の実装をテストし、検証する**: DR サイト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。
- **復旧を自動化する**: AWS またはサードパーティのツールを使用してシステム復旧を自動化します。

パフォーマンス効率

選択

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

通常ワークロードに対して最適なパフォーマンスを得るためには、複数のアプローチが必要です。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:

- **ベンチマーク結果を元に検討する**: AWS で既知のワークロードのロードテストを行い、テスト結果を参考に最適な選択を行います。
- **負荷テスト結果を元に検討する**: さまざまなリソースタイプとサイズで AWS に最新バージョンのシステムをデプロイし、モニタリングによってパフォーマンスメトリクスをキャプチャしてから、パフォーマンスとコストの計算に基づいて選択を行います。

パフォーマンス効率 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？

あるシステムに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、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てパフォーマンスを向上します。アーキテクチャに対して適切でないコンピューティングソリューション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:

- **コンピューティングソリューションの選択肢を検討する**: 最良のパフォーマンスを得るために、インスタンス、コンテナ、関数の使用方法についてさまざまな選択を検討します。
- **インスタンスの設定オプションを検討する**: インスタンスを使用する場合、ファミリー、インスタンスサイズ、機能 (GPU、I/O、バースト機能) などの設定オプションを検討します。
- **コンテナの設定オプションを検討する**: コンテナを使用する場合、コンテナのメモリ、CPU、テナンシー設定などの設定オプションについて検討します。
- **関数の設定オプションを検討する**: 関数を使用する場合、メモリ、ランタイム、ステートなどの設定オプションについて検討します。
- **伸縮性を活用する**: 伸縮性 (例、AWS Auto Scaling、Amazon Elastic Container Service、AWS Lambda) を活用し、需要の変化に対応します。

パフォーマンス効率 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、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します。

ベストプラクティス:

- **ストレージ特性を検討する**: 使用する必要のあるサービス (Amazon S3、Amazon EBS、Amazon Elastic File System (Amazon EFS)、Amazon EC2 インスタンスストアなど) を

選択するうえで必要なさまざまな特性 (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、データの永続性など) について検討します。

- **設定オプションを検討する**: PIOPS、SSD、マグネティック、Amazon S3 Transfer Acceleration などの設定オプションを検討します。
- **アクセスパターンを検討する**: 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ます。

パフォーマンス効率 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が使用され、異なる機能が提供されています。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および機能を選択すると、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:

- **データベース特性を検討する**: パフォーマンスが最も優れたデータベースのアプローチ (リレーショナル、NoSQL、データウェアハウス、インメモリなど) を選択できるように、さまざまな特性 (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) について検討します。
- **設定オプションを検討する**: 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について検討します。
- **アクセスパターンを検討する**: 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ます。
- **他のアプローチを検討する**: クエリ可能なデータを提供するための他のアプローチを検討します (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)。

パフォーマンス効率 5: ネットワークソリューションをどのように選択していますか？

システムに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、そうした制約はエッジの活用やリソースの配置によってカバーすることができます。

ベストプラクティス:

- **ロケーションを検討する**: ネットワークのレイテンシーを低減するためにロケーションのオプション (AWS リージョン、アベイラビリティーゾーン、プレースメントグループ、エッジロケーションなど) を検討します。
- **サービス機能を検討する**: ネットワークトラフィックを最適化するためにサービス機能 (EC2 インスタンスのネットワーク機能、拡張ネットワーク、Amazon EBS 最適化インスタンス、Amazon S3 Transfer Acceleration、Amazon CloudFront による動的コンテンツ配信など) を検討します。
- **ネットワーク機能を検討する**: ネットワーク距離の短縮や、ジッターの低減を実現するために、ネットワーク機能 (Amazon Route 53 でのレイテンシーベースのルーティング、Amazon VPC エンドポイント、AWS Direct Connect など) を検討します。

レビュー

パフォーマンス効率 6: 新しいサービスや機能をどのように取り入れていますか？

ソリューションを設計するときには、選択できる一定のオプションセットがあります。しかし、時間がたつと、アーキテクチャのパフォーマンス向上につながる新しいテクノロジー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:

- **評価プロセスを使用する**: 新しいリソースタイプおよびサイズを評価するプロセスを作成しま

す。パフォーマンステストを再実行し、パフォーマンス効率に向上が見られるかどうかを評価します。

モニタリング

パフォーマンス効率 7: 期待通りのパフォーマンスを発揮しているかをどのようにモニタリングしていますか？

システムのパフォーマンスは、時間がたつにつれて低下してしまうことがあります。システムのパフォーマンスをモニタリングすることで、パフォーマンスの低下を検知し、内部要因や外部要因 (オペレーティングシステムやアプリケーションのロードなど) を修正します。

ベストプラクティス:

- **モニタリング**: Amazon CloudWatch、サードパーティ製ツール、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します。
- **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できます。
- **トリガーベースのアクション**: 問題を修正やエスカレーションするための自動化されたアクションをトリガーするようアラームを設定します。

トレードオフ

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

ソリューションの構築時は、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します。多くの場合、整合性、耐久性、時間および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:

- **サービスを使用する**: 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します。
- **パターンを使用する**: キャッシュ、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します。

コスト最適化

コスト効率に優れたリソース

コスト最適化 1: AWS サービス選択の際にコストをどのように評価していますか？

Amazon EC2、Amazon EBS、Amazon S3 は、基盤となるAWS のサービスです。Amazon RDS、Amazon DynamoDB などのマネージドサービスは、上位のレベル、つまりアプリケーションレベルの AWS のサービスです。基盤となるサービスとマネージドサービスを適切に選択することで、コストに対してアーキテクチャを最適化できます。例えば、マネージドサービスを使用すると、管理や運用のオーバーヘッドの大部分を削減または排除し、アプリケーションやビジネス関連のアクティビティに注力できます。

ベストプラクティス:

- **サービスの選択によってコストを削減する**: 実行および選択したサービスを分析し、コストを最小限に抑えられるよう取り組みます。
- **ライセンスコストを最適化する**: 例えば、EC2 ワークロードに Linux、Oracle データベースワークロードに Aurora、データ分析に Redshift と、サービスでオープンソースのソフトウェアを使用してコストを削減します。
- **サーバーレスとコンテナベースのアプローチを使用して最適化を行う**: AWS Lambda、Amazon S3 (静的ウェブサイト用)、Amazon DynamoDB、Amazon ECS を使用して全体的なビジネスコストを削減します。
- **適切なストレージソリューションを使用して最適化を行う**: 使用パターンに基づいて最もコスト効率に優れたストレージソリューション (Amazon EBS コールドストレージ、Amazon S3 標準 – 低頻度アクセス、Amazon Glacier など) を使用します。
- **適切なデータベースを使用して最適化を行う**: 必要に応じて、Amazon RDS (Aurora、PostgreSQL、MySQL、SQL Server、Oracle Database) や、Amazon DynamoDB (または

その他のキー値ストア、NoSQL の代替製品) を使用します。

- **その他のアプリケーションレベルサービスを使用して最適化を行う**: 必要に応じて、Amazon SQS、Amazon SNS、Amazon Simple Email Service (Amazon SES) を使用します。

コスト最適化 2: コスト目標を達成するためにインスタンスタイプとサイズをどのように選択していますか？

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するようお勧めします。

ベストプラクティス:

- **メトリクスに基づいてリソースのサイジングをする**: パフォーマンスメトリクスを活用して適切なサイズとタイプを選択し、コストに合わせて最適化します。Amazon EC2 、Amazon DynamoDB 、Amazon EBS (PIOPS)、Amazon RDS、Amazon EMR、ネットワークといった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。

コスト最適化 3: コスト削減のために料金モデルをどのように選択していますか？

ワークロードの費用を最小限に抑えるため最適な料金モデルを使用します。オンデマンドインスタンスのみや、オンデマンドインスタンスとリザーブドインスタンスを組み合わせて使用すること、または必要に応じてスポットインスタンスも使用することにより、デプロイを最適化できます。

ベストプラクティス:

- **リザーブドインスタンスとリザーブドキャパシティー**: Amazon EC2、Amazon DynamoDB、Amazon RDS、Amazon CloudFront などの使用状況を定期的に分析し、それに応じてリザーブドインスタンスを購入します。
- **スポットインスタンス**: EC2 Auto Scaling、AWS Batch、Amazon EMR などの一部のワー

クロードにスポットインスタンス (例、スポットブロックやスポットフリート) を使用します。

- **リージョンごとのコスト差異を検討する**: AWS リージョンを選択するときにコストを考慮します。

コスト最適化 4: データ転送料金についてどのように計画していますか？

データ転送料金をモニタリングし、そうしたコストの削減につながるアーキテクチャに関する決定を行います。例えば、お客様がコンテンツプロバイダーとして、エンドユーザーに対して 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) にプッシュすると、大幅にコストを削減でき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上の変更によって、運用コストを劇的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:

- **最適化を行う**: データ転送に合わせてアプリケーション設計、WAN アクセラレーション、マルチ AZ、AWS リージョンの選択を最適化します。
- **コンテンツ配信ネットワーク (CDN) を使用する**: 必要に応じて CDN を使用します。
- **AWS Direct Connect を使用する**: 状況を分析し、必要に応じて AWS Direct Connect を使用します。

需要と供給の一致

コスト最適化 5: リソースの供給と顧客の需要をどのように一致させていますか？

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、料金を支払っているリソースを無駄なく使用し、著しく使用率が低いインスタンスがないようにします。使用率のメトリクスがいずれかの方向に偏っている場合、運用コスト (著しく高い使用率によるパ

フォーマンスの低下) や、無駄な AWS の支出 (過剰なプロビジョニングが原因) によってビジネスに悪影響が及びます。

ベストプラクティス:

- **需要ベースのアプローチ**: Autoscaling を使用して変動する需要に対応します。
- **バッファベースのアプローチ**: 作業をバッファに入れ (例、Amazon Kinesis や Amazon SQS を使用)、作業を処理できるだけの十分なキャパシティーを確保してから作業を開始します。
- **時間ベースのアプローチ**: 時間ベースのアプローチの例としては、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、四半期や年間のスケジュールに従う (ブラックフライデーなど) などがあります。

費用の把握

コスト最適化 6: AWS 使用量とコスト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 のツールを活用し、誰が、何を、どのぐらいのコストで使用しているのかを把握します。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できます。

ベストプラクティス:

- **すべてのリソースをタグ付けする**: タグ付け可能なリソースすべてにタグ付けし、AWS の請求の変化をインフラストラクチャと使用状況の変化に関連付けられるようにします。
- **請求とコスト管理ツールを使用する**: 詳しい請求レポートを読み込んで解釈する標準プロセスを作成するか、Cost Explorer を使用します。必要に応じて Amazon CloudWatch またはサードパーティプロバイダー (例、Cloudability、CloudCheckr、CloudHealth) を使用して定期的に使用状況と支出をモニタリングします。
- **予算超過の通知**: 費用が明確に定義された限度を超えた場合、チームの主要メンバーに通知し

ます。

- **ビジネス成果によるコスト配分**: ビジネスの成果と収益に基づいてワークロードコストをタグ付けなどの方法で後から配分する手法です。

- **財務主導のチャージバック手法またはショーバック手法:** タグ付けなどの方法でリソースを各コストセンターに割り当てる手法です。

コスト最適化 7: AWS 使用量をどのように管理していますか？

コストを適正に保ちながら目標を達成できるよう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:

- **グループとロールを定義して設定する**: ガバナンスのメカニズムを使用し、開発グループ、テストグループ、本番グループなど、各グループでインスタンスとリソースを作成できるユーザーを制御します。このメカニズムは AWS のサービスとサードパーティのソリューションに適用します。

- **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、測定、監査し、不要なリソースを使用したり、そうしたリソースに料金を支払ったりすることがないようにします。

コスト最適化 8: 不要なリソースをどのように削除していますか？

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、必要に応じて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。ワークロードに合わせてプロジェクトを最適化するために、AWS サポートの提案を活用します。例えば、AWS Auto Scaling、AWS OpsWorks、AWS Data Pipeline、Amazon EC2 のさまざまなプロビジョニング手法を使用すべき場合や、AWS Trusted Advisor でコスト最適化の推奨事項をレビューすべき場合について提案を受けることができます。

ベストプラクティス:

- **削除を自動化する**: リソースの削除を適切に処理するようシステムを設計してから、重要でないリソースや不要なリソース (使用率が低い) を確認して廃棄します。
- **削除プロセスを定義する**: 既に使われていないリソースを確認して削除するプロセスを作成します。

継続した最適化

コスト最適化 9: 新しい AWS サービスをどのように評価していますか？

AWS による新しいサービスや機能のリリース時には、アーキテクチャに関する既存の決定事項を見直し、アーキテクチャが引き続き最もコスト効率に優れたものである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:

- **定期的にレビューする**: AWS ソリューションアーキテクト、コンサルタント、アカウントチームと定期的にミーティングを行い、全体的なコストを下げることができる新しいサービスや機能について検討します。
- **コスト最適化を担当する職務を設ける**: ビジネス全体のコストと使用状況を定期的にレビューするチームを作ります。
- **ワークロードをレビューおよび分析する**: 新しいサービス、リソースタイプ、サイズをレビューするプロセスを作成します。パフォーマンステストを再実行し、コスト削減の有無を評価します。