AWS Well-Architected Framework

運用上の優秀性の柱

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

運用上の優秀性の柱: AWS Well-Architected Framework

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

運用上の優秀性の柱 – AWS Well-Architected フレームワーク

公開日: 2024 年 6 月 27 日 (改訂履歴)

このホワイトペーパーは AWS Well-Architected フレームワークの運用上の優秀性の柱に焦点を当てています。本書は、お客様が AWS のワークロードの設計、提供、メンテナンスにベストプラクティスを適用するうえで役立つガイダンスを提供します。

はじめに

それらの AWS Well-Architected Framework は、AWS でワークロードを構築する際に行う決定の利点とリスクを理解するのに役立ちます。このフレームワークを使用すれば、クラウド内で信頼性、安全性、効率性、コスト効率に優れ、持続可能なワークロードを設計および運用するための運用上およびアーキテクチャ上のベストプラクティスを学ぶことができます。運用とアーキテクチャを、ベストプラクティスに照らし合わせて一貫した方法で評価し、改善すべき領域を特定する方法を提供します。AWS は、運用を念頭に置いて設計された Well-Architected ワークロードがあれば、ビジネスを成功させる可能性は大幅に高まると確信しています。

このフレームワークは次の 6 つの柱に基づいています。

- オペレーショナルエクセレンス
- セキュリティ
- 信頼性
- パフォーマンス効率
- コスト最適化
- サステナビリティ

このホワイトペーパーは、運用上の優秀性の柱と、それを優れた設計のソリューションの基礎としてどのように適用するかに焦点を当てています。運用が、サポートする事業部門や開発チームとは別の、独立した機能として認識される環境では、運用上の優秀性を達成することは困難です。このホワイトペーパーに記載されたプラクティスを採用することで、状況の把握と、効果的かつ効率的な運用とイベント対応を可能にするアーキテクチャを構築して、ビジネス上の目標を継続的に改善し、サポートできます。

このホワイトペーパーの対象者は、最高技術責任者 (CTO)、アーキテクト、デベロッパー、オペレーションチームメンバーなどの技術担当者です。このホワイトペーパーを読むと、運用上の優秀性のためのクラウドアーキテクチャを設計する際に使用する AWS のベストプラクティスと戦略を理解できます。このホワイトペーパーでは、実装の詳細やアーキテクチャ上のパターンは扱いません。ただし、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。

運用上の優秀性

Amazon では、運用上の優秀性とは、優れたカスタマーエクスペリエンスを着実に提供しながら、ソフトウェアを正しく構築するために取り組むことであると定義しています。これには、チームの編成、ワークロードの設計、ワークロードの大規模な運用、経時的な進化のためのベストプラクティスが含まれます。運用上の優秀性により、チームはメンテナンスや問題解決のアクティビティに費やす時間を低減し、お客様に利益をもたらす新機能の構築に専念できるようになります。当社では適切な構築のために、システムの適正な実行、業務とチームのワークロードバランス、そして最重要事項として、優れたカスタマーエクスペリエンスを実現するためのベストプラクティスを重視しています。

運用上の優秀性の目的は、新機能とバグ修正を迅速かつ確実にお客様に提供することです。運用上の優秀性に投資している組織は、新しい機能を構築し、変更を加え、障害に対処しながら、着実に顧客満足を実現しています。その過程で、運用上の優秀性は、デベロッパーが高品質の結果を常に達成するために役立ち、継続的インテグレーションと継続的デリバリー (CI/CD) を促進します。

設計原則

以下は、運用上の優秀性を実現するための設計原則です。

- ビジネス成果に基づいてチームを編成する: チームがビジネス成果を達成できるかどうかは、リーダーシップのビジョン、効果的な運用、ビジネスに沿った運用モデルによって決まります。リーダーシップは、チームが最も効率的な方法で業務を行い、ビジネス成果を達成するようチームにインセンティブを与える適切なクラウド運用モデルを用いて、CloudOps の変革に全力で取り組む必要があります。適切な運用モデルでは、人材、プロセス、テクノロジーの能力を活用してスケールと生産性の最適化を実現し、俊敏性、即応性、適応性を通して差別化を図ります。組織の長期的なビジョンは、エンタープライズ全体にわたってステークホルダーおよびクラウドサービスや消費者に伝える目標に変換されます。目標と運用上の KPI はすべてのレベルで一致します。このプラクティスは、以下の設計原則の実装から得られる長期的な価値を維持します。
- オブザーバビリティを実装して実用的なインサイトを取得する: ワークロードの動作、パフォーマンス、信頼性、コスト、健全性などを包括的に理解します。主要業績評価指標 (KPI) を設定し、オブザーバビリティのテレメトリーを活用して、ビジネス成果の達成が脅かされている場合に情報に基づいた意思決定を行い、迅速に対処します。実用的なオブザーバビリティデータに基づいて、パフォーマンス、信頼性、コストを積極的に改善します。
- 可能な限り安全に自動化する: クラウドでは、アプリケーションコードに使用しているものと同じエンジニアリング原理を環境全体に適用します。ワークロード全体とその運用 (アプリケーション、インフラストラクチャ、設定、手順) をコードとして定義し、更新できます。その後、イベン

トに応じてワークロードの操作を開始することで、ワークロードの操作を自動化できます。クラウドでは、レート制御、エラーしきい値、承認などのガードレールを設定することで、自動化における安全性を実現できます。効果的な自動化により、イベントへの一貫した対応を実現し、人為的ミスを最小限に抑え、オペレーターの労力を軽減できます。

- 小規模かつ可逆的な変更を頻繁に行う: コンポーネントを定期的に更新できるように、スケーラブルで疎結合のワークロードを設計します。デプロイの自動化の手法と併せて、小さく段階的に変更していくことで、障害が発生した場合でも影響範囲を小さく抑え、迅速に復旧することができます。そのため、自信を持ってワークロードに有益な変化を加えられるようになり、一方で品質も維持し、市場の変化にも迅速に適応できます。
- 運用手順を定期的に改善する: ワークロードの進化に伴い、運用手順も適宜変更してください。運用手順を実施するときに、改善の機会を探します。定期的にレビューを実施し、すべての手順が効果的であり、チームに周知されていることを検証します。ギャップが見つかった場合は、手順を適宜更新してください。手順の更新について、すべてのステークホルダーとチームに伝えます。運用のゲーミフィケーションを行ってベストプラクティスを共有し、チームを教育します。
- 障害を想定する: 障害シナリオを作成し、ワークロードのリスクプロファイルとそれがビジネス成果に与える影響を理解することで、運用の成功を最大化します。こうしてシミュレートした障害に対する手順とチームの対応の有効性をテストします。テストで特定された未解決のリスクを管理するために、情報に基づいた意思決定を行います。
- 運用上のあらゆるイベントやメトリクスから学ぶ: 運用のあらゆるイベントや障害から学んだ教訓を通じて、改善を推進します。チーム間と組織全体で教訓を共有します。教訓は、運用がビジネス成果にどのように貢献するかについてのデータや工ピソードに焦点を当てたものである必要があります。
- マネージドサービスを使用する: 可能な限り AWS のマネージドサービスを利用して、運用上の負担を軽減します。それらのサービスの操作に関する運用手順を作成します。

定義

クラウドにおける「運用上の優秀性」には 4 つのベストプラクティス領域があります。

- 組織
- 準備
- 運用
- 進化

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

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

組織

ビジネスの成果に対応できるように、組織の優先順位、組織構造、および組織がチームメンバーをサポートする方法を把握する必要があります。

運用上の優秀性を実現するには、以下の点を理解する必要があります。

トピック

組織の優先順位

チームは、ビジネスの成功を実現する優先順位を設定するために、ワークロード全体、その役割、共有されるビジネス目標に関する理解を共有する必要があります。優先順位を明確に定義することで、努力を通じて得られるメリットが最大限に活かされます。組織のニーズの変化に応じて更新できるように、優先順位を定期的に確認します。

ベストプラクティス

OPS01-BP01 顧客のニーズを評価する

ビジネス、開発、運用チームを含む主要ステークホルダーと協力して、外部顧客のニーズに対する重点領域を決定します。これにより、目的のビジネス成果を達成するために必要なオペレーションサポートについて十分に理解していることを確かめることができます。

期待される成果:

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

一般的なアンチパターン:

- 営業時間外にカスタマーサポートを設けないこととしましたが、サポートリクエストの履歴データを確認していません。あなたには、これが顧客に影響を与えるかどうかはわかりません。
- 新しい機能を開発していますが、当該機能が望まれているかどうか、望まれている場合はどのような形式なのかを見出すために、顧客に関与してもらっておらず、また、提供の必要性および提供方法を検証するための実験も行っていません。

このベストプラクティスを活用するメリット: ニーズが満たされている顧客は、定着率が高くなる傾向があります。外部の顧客のニーズを評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ビジネスニーズの理解: ビジネスの成功は、ビジネス、開発、運用の各チームを含むステークホルダー全体で目標を共有し、理解を深めることで実現できます。

外部顧客のビジネス目標、ニーズ、優先順位の確認: ビジネス、開発、運用の各チームを含む主要ステークホルダーと連携し、外部顧客の目標、ニーズ、優先順位について話し合います。これにより、ビジネスおよび顧客成果を達成するために必要なオペレーションサポートについて十分に理解できます。

共通理解の確立: ワークロードのビジネス機能、ワークロードの運用における各チームのロール、およびこれらの要因が内部および外部顧客の共通のビジネス目標をどのようにサポートするかについて、共通の理解を確立します。

リソース

関連するベストプラクティス:

- OPS11-BP03 フィードバックループを実装する

OPS01-BP02 内部顧客のニーズを評価する

ビジネス、開発、運用チームを含む主要関係者と協力して、内部顧客のニーズに対する重点領域を決定します。これにより、ビジネス成果を達成するために必要なオペレーションサポートについて十分に理解できます。

期待される成果:

- 確立された優先順位に基づいて、改善の努力を最も影響があるところに集中させる (チームのスキルの開発、ワークロードのパフォーマンスの改善、コストの削減、ランブックの自動化、モニタリングの強化など)。
- ニーズの変化に応じて優先順位を更新する。

一般的なアンチパターン:

- ネットワーク管理を容易にするため、製品チームの IP アドレスの割り当てを変更することとしました。あなたは、これが製品チームに与える影響を知りません。
- あなたは、新しい開発ツールを実装しようとしていますが、当該ツールが必要とされているかどうか、または既存のプラクティスと互換性があるかどうかを知るために、社内クライアントを関与させていません。
- 新しいモニタリングシステムを実装しようとしていますが、検討されるべきモニタリングまたはレポートのニーズがあるかどうかを把握するために社内クライアントに問い合わせていません。

このベストプラクティスを活用するメリット: 内部顧客のニーズを評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- ビジネスニーズの理解: ビジネスの成功は、ビジネス、開発、運用の各チームを含むステークホルダー全体で目標を共有し、理解を深めることで実現できます。
- 内部顧客のビジネス目標、ニーズ、優先順位の確認: ビジネス、開発、運用の各チームを含む主要ステークホルダーと連携し、内部顧客の目標、ニーズ、優先順位について議論します。これにより、ビジネスおよび顧客成果を達成するために必要なオペレーションサポートについて十分に理解できます。

- 共通理解の確立: ワークロードのビジネス機能、ワークロードの運用における各チームの役割、およびこれらの要因が内部および外部顧客の共通のビジネス目標をどのようにサポートするかについて、共通の理解を確立します。

リソース

関連するベストプラクティス:

- OPS11-BP03 フィードバックループを実装する

OPS01-BP03 ガバナンス要件を評価する

ガバナンスとは、企業がビジネス目標を達成するために使用する、ポリシー、ルール、フレームワーク一式です。ガバナンス要件は、組織内から生まれます。選択する技術の種類に影響する場合も、ワークロードを運用する方法に関連する場合もあります。組織のガバナンス要件を、ワークロードに組み込みます。コンフォーマンスとは、ガバナンス要件が組み込まれていることを示す能力のことです。

期待される成果:

- ガバナンス要件が、アーキテクチャの設計およびワークロードのオペレーションに組み込まれています。
- ガバナンス要件に従っている証拠を提供できます。
- ガバナンス要件は定期的に見直され更新されています。

一般的なアンチパターン:

- 組織が、ルートアカウントを多要素認証とすることを義務としている。この要件を実装できなかったため、ルートアカウントが侵害された。
- ワークロードの設計中に、IT 部門が承認していないインスタンスタイプを選択した。ワークロードを起動できず、再設計を行わなければならなくなった。
- ディザスタリカバリ計画を備えることが必須となっている。計画を作成しなかったため、ワークロードの停止が長引いた。
- チームは新しいインスタンスの使用を希望していたが、ガバナンス要件が更新されていないため、許可されなかった。

このベストプラクティスを活用するメリット:

- ガバナンス要件に従うと、ワークロードを組織のより大きなポリシーに合わせることができます。
- ガバナンス要件は、業界の標準と組織のベストプラクティスを反映しています。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

関係者やガバナンス組織と協力して、ガバナンス要件を特定します。ガバナンス要件をワークロードに含めます。ガバナンス要件に従っている証拠を提供できるようにします。

お客様事例

AnyCompany Retail では、クラウドオペレーションチームが組織全体の関係者と協力して、ガバナンス要件を作成しました。例えば、Amazon EC2 インスタンスへの SSH アクセスを禁止しています。チームがシステムにアクセスする必要がある場合、AWS Systems Manager Session Manager を使用する必要があります。クラウドオペレーションチームは、新しいサービスを利用できるようになるたびに、ガバナンス要件を定期的に更新しています。

実装手順

1. 一元化されたチームがあればそれも含め、ワークロードの関係者を特定します。
2. 関係者と協力して、ガバナンス要件を特定します。
3. リストを作成したら、改善項目に優先順位を付け、ワークロードへの実装を開始します。
    a. AWS Config などのサービスを使用して、Governance-as-Code を作成し、ガバナンス要件に従っていることを検証します。
    b. AWS Organizations を使用する場合は、サービスコントロールポリシーを活用してガバナンス要件を実装できます。
4. 実装を検証するドキュメントを提供します。

実装計画に必要な工数レベル: 中。ガバナンス要件を満たさずに実装すると、ワークロードをやり直すことになる場合があります。

リソース

関連するベストプラクティス:

- OPS01-BP04 コンプライアンス要件を評価する - コンプライアンスはガバナンスに似ていますが、組織外に由来するものです。

関連するドキュメント:

- AWS Management and Governance Cloud Environment Guide (AWS の管理およびガバナンスに関するクラウド環境ガイド)
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment (マルチアカウント環境の AWS Organizations サービスコントロールポリシーのためのベストプラクティス)
- Governance in the AWS クラウド: The Right Balance Between Agility and Safety (AWS クラウドのガバナンス: 俊敏性と安全性のバランスを取る)
- ガバナンス、リスク、コンプライアンス (GRC) とは

関連動画:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks (AWS の管理とガバナンス: 設定、コンプライアンス、監査 - AWS Online Tech Talks)
- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1) (AWS re:Inforce 2019: クラウド時代のガバナンス (DEM12-R1))
- AWS re:Invent 2020: Achieve compliance as code using AWS Config(AWS re:Invent 2020: AWS Config を使用してコードとしてのコンプライアンスを実現する)
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)(AWS re:Invent 2020: AWS GovCloud (US) における俊敏なガバナンス)

関連する例:

- AWS Config のコンフォーマンスパックの例

関連サービス:

- AWS Config
- AWS Organizations - サービスコントロールポリシー

OPS01-BP04 コンプライアンス要件を評価する

規制、業界、および社内のコンプライアンス要件は、組織の優先順位を定義するための重要な推進要素です。コンプライアンスフレームワークによって、特定の技術や地理的場所を使用できない場合もあります。外部コンプライアンスフレームワークが特定されない場合は、デューデリジェンスを適用します。コンプライアンスを検証する監査またはレポートを作成します。

自社製品が特定のコンプライアンス基準を満たしていることを宣伝する場合、継続的なコンプライアンスを確保するための内部プロセスが必要です。コンプライアンス標準の例としては、PCI DSS、FedRAMP、HIPAA があります。適用されるコンプライアンス標準は、ソリューションが保存または送信するデータの種類、ソリューションがサポートするリージョンなど、さまざまな要因によって決まります。

期待される成果:

- 規制、業界、および社内のコンプライアンス要件がアーキテクチャの選択に組み込まれています。
- コンプライアンスを検証して監査レポートを作成できます。

一般的なアンチパターン:

- ワークロードの一部が、クレジットカード業界のデータセキュリティ基準 (PCI DSS) フレームワークの対象となっているが、ワークロードはクレジットカードデータを暗号化せずに保存している。
- ソフトウェア開発者とアーキテクトが、組織が遵守すべきコンプライアンスフレームワークに気付いていない。
- 年次の Systems and Organizations Control (SOC2) Type II 監査が近く行われるが、コントロールが配置されていることを検証できない。

このベストプラクティスを活用するメリット:

- ワークロードに適用されるコンプライアンス要件を評価し、理解することで、ビジネス価値を実現するためにどのような優先順位で注力すべきかを知ることができます。
- コンプライアンスフレームワークに合致する適切な場所や技術を選択します。
- 可監査性を持たせてワークロードを設計すると、コンプライアンスフレームワークを遵守していることを証明できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

このベストプラクティスを実装することで、コンプライアンス要件をアーキテクチャ設計プロセスに組み込みます。チームメンバーは必要なコンプライアンスフレームワークを認識します。フレームワークに沿ってコンプライアンスを検証します。

お客様事例

AnyCompany Retail は、顧客のクレジットカード情報を保存しています。カードストレージチームの開発者は、PCI-DSS フレームワークに準拠する必要があることを理解しています。クレジットカード情報が PCI-DSS フレームワークに沿って安全に保存およびアクセスされていることを検証する手順を踏んできています。毎年、セキュリティチームと協力して、コンプライアンスを検証しています。

実装手順

1. セキュリティチームやガバナンスチームと協力して、ワークロードが準拠しなければならない業界、規制、組織内部のコンプライアンスフレームワークを精査します。コンプライアンスフレームワークをワークロードに組み込みます。
   a. AWS Compute Optimizer や AWS Security Hub などのサービスを使用して、AWS のリソースの継続的なコンプライアンスを検証します。
2. チームメンバーがコンプライアンス要件に沿ってワークロードを運用および進化できるように、コンプライアンス要件を教育します。コンプライアンス要件は、アーキテクチャや技術を選択する際に含める必要があります。
3. コンプライアンスフレームワークによっては、監査またはコンプライアンスレポートを作成する必要があります。組織と協力して、このプロセスをできるだけ自動化します。
   a. AWS Audit Manager のようなサービスを使用して、コンプライアンスを検証し、監査レポートを作成します。
   b. AWS Artifact を使用して AWS のセキュリティとコンプライアンスに関するドキュメントをダウンロードできます。

実装計画に必要な工数レベル: 中。コンプライアンスフレームワークの実装は課題が多い場合があります。監査レポートやコンプライアンスドキュメントを作成するとさらに複雑になります。

リソース

関連するベストプラクティス:

- SEC01-BP03 管理目標を特定および検証する - セキュリティ管理目標は、コンプライアンス全体における重要な部分です。
- SEC01-BP06 パイプラインのセキュリティコントロールのテストと検証を自動化する - パイプラインの一部として、セキュリティ管理を検証します。新しい変更に関するコンプライアンスドキュメントを作成することもできます。
- SEC07-BP02 データ保護コントロールを定義する - 多くのコンプライアンスフレームワークは、データ処理およびストレージに関するポリシーベースです。
- SEC10-BP03 フォレンジック機能を備える - フォレンジック機能は、監査のコンプライアンスで使用できることがあります。

関連するドキュメント:

- AWS Compliance Center (AWS コンプライアンスセンター)
- AWS のコンプライアンスのリソース
- AWS Risk and Compliance Whitepaper (Amazon Web Services リスクとコンプライアンスホワイトペーパー)
- AWS 責任共有モデル
- コンプライアンスプログラムによる対象範囲内の AWS のサービス

関連動画:

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer(AWS re:Invent 2020: AWS Config を使用してコードとしてのコンプライアンスを実現する)
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing (AWS re:Invent 2021 - クラウドのコンプライアンス、保証、監査)
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202) (AWS Summit ATL 2022 - AWS におけるコンプライアンス、保証、監査の実装 (COP202))

関連する例:

- AWS での PCI DSS および AWS Foundational Security Best Practices

関連サービス:

- AWS Artifact

- AWS Audit Manager
- AWS Compute Optimizer
- AWS Security Hub

OPS01-BP04 脅威の状況を評価する

ビジネスに対する脅威 (競合、ビジネスリスクと負債、運用リスク、情報セキュリティの脅威など) を評価し、リスクのレジストリで現在の情報を維持します。注力する場所を決定する際に、リスクの影響を考慮します。

Well-Architected フレームワークは学習、測定、改善に重点を置いています。アーキテクチャを評価し、時間の経過とともにスケールする設計を実装するための一貫したアプローチを提供します。AWS が提供する AWS Well-Architected Tool は、開発前のアプローチ、本番稼働前のワークロードの状態、本番稼働中のワークロードの状態などを確認するのに役立ちます。最新の AWS アーキテクチャのベストプラクティスと比較して、ワークロードの全体的なステータスをモニタリングし、潜在的なリスクについてインサイトを得ることができます。

AWS のお客様は、AWS のベストプラクティスに照らしてアーキテクチャを評価するために、ミッションクリティカルなワークロードのガイド付き Well-Architected レビューを受けることもできます。エンタープライズサポートのお客様は、クラウドでの運用へのアプローチにおけるギャップの特定を支援するよう設計された運用レビューを受けることができます。

これらのレビューのチーム間での関与は、ワークロードとチームの役割の成功への貢献方法に関する共通理解を確立するのに役立ちます。レビューを通じて特定されるニーズは、優先順位を決定するのに役立ちます。

AWS Trusted Advisor は、最適化を推奨する中心的なチェックのセットへのアクセスを提供するツールであり、優先順位の決定に役立ちます。ビジネスおよびエンタープライズサポートのお客様は、優先順位の決定にさらに役立つ、セキュリティ、信頼性、パフォーマンス、コストの最適化に重点を置いた追加のチェックにアクセスできます。

期待される成果:

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

一般的なアンチパターン:

- 自社製品に古いバージョンのソフトウェアライブラリを使用しています。あなたは、ワークロードに意図しない影響を及ぼす可能性のある問題について、ライブラリのセキュリティ更新が必要なことを認識していません。
- 最近、競合他社は、あなたの製品に関する顧客からの苦情の多くに対処する製品のバージョンをリリースしました。あなたは、これらの既知の問題の対処について優先順位付けを行っていません。
- 規制当局は、法規制コンプライアンス要件を遵守していない企業の責任を追求してきました。未対応のコンプライアンス要件への対応に優先順位が付けてられていません。

このベストプラクティスを活用するメリット: 組織とワークロードに対する脅威を特定して理解することで、対処すべき脅威、その優先度、対処に必要なリソースを判断しやすくなります。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

- 脅威の状況の評価: ビジネスに対する脅威 (競合、ビジネスリスクと責任、運用リスク、情報セキュリティの脅威など) を評価し、重点領域を決定する際にその影響を織り込めるようにします。
  - AWS セキュリティ速報
  - AWS Trusted Advisor
- 脅威モデルの維持: 潜在的な脅威、計画および実施された緩和策、またその優先度を特定する脅威モデルを確立し、維持します。脅威がインシデントとして出現する確率、それらのインシデントから回復するためのコスト、発生が予想される損害、およびそれらのインシデントを防ぐためのコストを確認します。脅威モデルの内容の変更に伴って、優先順位を変更します。

リソース

関連するベストプラクティス:

- SEC01-BP07 脅威モデルを使用して脅威を特定し、緩和策の優先順位を付ける

関連するドキュメント:

- AWS クラウド コンプライアンス
- AWS セキュリティ速報

- AWS Trusted Advisor

関連動画:

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

OPS01-BP06 メリットとリスクを管理しながらトレードオフを評価する

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

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

AWS は、AWS とそのサービスについてチームを教育し、選択がどのようにワークロードに影響を与えるかについての理解を深める支援を行います。チームの教育には、AWS Support 提供のリソース (AWS ナレッジセンター、AWS ディスカッションフォーラム、AWS Support センター) や AWS ドキュメントを使用します。さらに質問がある場合は、AWS Support までお問い合わせください。

また AWS は、The Amazon Builders' Library で運用に関するベストプラクティスとパターンも共有しています。AWS ブログと公式の AWS ポッドキャストでは、その他さまざまな有益な情報を入手できます。

期待される成果: クラウドデリバリー組織内のあらゆるレベルでの重要な意思決定を促進する、意思決定ガバナンスフレームワークが明確に定義されています。このフレームワークには、リスク登録簿、意思決定の権限を持つ定義済みの役割、考えられる意思決定の各レベルに対する定義済みモデルなどの機能が含まれています。このフレームワークでは、対立の解決方法、提示すべきデータ、オプションの優先順位付けの方法が事前に定義されているため、決定が下されたらすぐに決定にコミットできます。意思決定のフレームワークには、すべての意思決定のメリットとリスクを確認して比較検討し、トレードオフを理解するための標準的アプローチが含まれています。これには、規制コンプライアンス要件の順守などの外部要因が含まれる場合があります。

一般的なアンチパターン:

- 投資家からは、Payment Card Industry Data Security Standards (PCI DSS) への準拠を実証することが求められています。投資家の要求に応えることと、現在の開発活動を継続することとのトレードオフについて検討しません。代わりに、準拠を実証することなく、開発作業を進めます。投資家は、プラットフォームのセキュリティと、投資の是非に懸念を抱いて、会社に対する支援を停止します。
- 開発者の 1 人がインターネットで見つけたライブラリを含めることにしました。不明なソースからこのライブラリを採用するリスクを評価しておらず、脆弱性や悪意のあるコードが含まれているかどうかはわかりません。
- 当初ビジネスが移行を正当化した理由は、アプリケーションワークロードの 60% のモダナイゼーションに基づくものでした。しかし、技術的な問題により、モダナイゼーションは 20% に留めるという決断が下されました。これにより、計画していた長期的メリットは減少し、インフラストラクチャチームがレガシーシステムを手動でサポートするためにオペレーターの労力が増え、この変更を予定していなかったインフラストラクチャチームでの新しいスキルセットの構築に大きく依存することになりました。

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

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

メリットとリスクの管理は、主要な意思決定の要件を決定する運営組織が定義すべきです。関連するリスクを理解した上で、決定が組織にもたらすメリットに基づいて意思決定を行い、優先順位を付けます。組織の意思決定には正確な情報が不可欠です。この情報は、信頼性の高い測定に基づき、費用対効果分析という一般的な業界慣行によって定義されたものである必要があります。このような決定を下すには、中央集権型と権限分散型のバランスを取ります。必ずトレードオフはあるため、それぞれの選択肢が定義された戦略と期待されるビジネス成果にもたらす影響を理解することが重要です。

実装手順

1. 包括的なクラウドガバナンスフレームワーク内でメリットの測定方法を定式化します。
   a. 中央集権型の意思決定と、権限分散型の意思決定のバランスを取ります。

   b. あらゆる意思決定で負担の大きい意思決定プロセスを実施することが遅延につながることを理解します。
   c. 意思決定プロセスに外部要因 (コンプライアンス要件など) を組み込みます。
2. さまざまなレベルの意思決定について、合意に基づいた意思決定フレームワークを確立します。このフレームワークには、利害の対立が関わる意思決定を解決すべき人物が含まれます。
   a. 取り消しが効かない可能性のある「ワンウェイドア」(一方通行の扉) の意思決定を一元化します。
   b. 下位レベルの組織リーダーが「ツーウェイドア」(双方向に行き来できる扉) の意思決定を行えるようにします。
3. メリットとリスクを理解し、管理します。決定のメリットと関連するリスクのバランスを取ってください。
   a. メリットの特定: ビジネスの目標、ニーズ、優先順位に基づいてメリットを特定します。例として、ビジネスケースへの影響、市場投入までの時間、セキュリティ、信頼性、パフォーマンス、コストなどがあります。
   b. リスクの特定: ビジネスの目標、ニーズ、優先順位に基づいてリスクを特定します。例として、市場投入までの時間、セキュリティ、信頼性、パフォーマンス、コストなどがあります。
   c. リスクに対するメリットの評価と十分な情報に基づく意思決定: ビジネス、開発、運用を含む主要関係者の目標、ニーズ、優先順位に基づいてメリットとリスクの影響を決定します。メリットの価値を、リスクが現実化する可能性とその影響のコストに照らして評価します。たとえば、信頼性よりも市場投入までのスピードを重視すると、競争上の優位性が得られます。ただし、信頼性の問題がある場合、稼働時間が短くなる場合があります。
4. コンプライアンス要件の順守を自動化する主な意思決定をプログラム的に実施します。
5. バリューストリーム分析や LEAN など既知の業界フレームワークと機能を活用して、現状のパフォーマンスやビジネスメトリクスのベースラインを定め、これらのメトリクスの改善に向けた進捗の反復を定義します。

実装計画に必要な工数レベル: 中〜高

リソース

関連するベストプラクティス:

- OPS01-BP04 脅威の状況を評価する

関連するドキュメント:

- Amazon の 1 日目の文化の要素 | 高品質で高速な決定を下す
- クラウドガバナンス
- Management & Governance Cloud Environment
- Governance in the Cloud and in the Digital Age: Parts One & Two

関連動画:

- Podcast | Jeff Bezos | On how to make decisions

関連する例:

- Make informed decisions using data (The DevOps Sagas)
- Using development value stream mapping to identify constraints to DevOps outcomes

運用モデル

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

チームのニーズは、業界、組織、チームの構成、およびワークロードの特性によって形成されます。1 つの運用モデルによって、すべてのチームとそのワークロードをサポートできると期待するのは合理的ではありません。

組織に存在する運用モデルの数は、開発チームの数とともに増加する傾向があります。運用モデルの組み合わせを使用することが必要になる場合もあります。

標準規格を採用しサービスを使用することで、運用を簡素化し、運用モデルのサポートの負担を軽減することができます。共有された標準に沿って開発作業を行うことの利点は、標準を採用し、新しい機能を採用するチームの数によって増大します。

チームの活動をサポートする標準の追加、変更、例外をリクエストするメカニズムを持つことが重要です。このオプションがなければ、標準はイノベーションの障壁になります。リクエストは、実行可能な場合は承認され、利点とリスクを評価した後、適切であると判断される必要があります。

明確に定義された一連の責任によって、競合や重複する作業の頻度を減らすことができます。ビジネス、開発、運用チームとの間に強い連携と関係があれば、ビジネス成果は達成しやすくなります。

運用モデル 2 x 2 の表現

これらの運用モデル 2 x 2 の表現は、環境内でのチーム間の関係を理解するのに役立つ図です。これらの図では、誰が何をするかということと、チーム間の関係に重点を置いていますが、これらの例に沿ったガバナンスと意思決定についても説明します。

チームはサポートするワークロードに応じて、複数のモデルの複数の部分を担当する場合があります。説明されている高レベルの分野よりも、さらに専門的な分野に分割したい場合があります。アクティビティを分離または集計したり、チームをオーバーレイしてより具体的な詳細を提供したりすると、これらのモデルで無限のバリエーションが生じる可能性があります。

チーム間で機能が重複する、または認識されていない機能があり、それらがさらなる利点をもたらしたり、効率化につながったりする可能性があることに気づくことがあります。また、組織内で満たされていないニーズを見つけ、それに取り組むことができる可能性もあります。

組織の変化を評価する際は、モデル間のトレードオフ、個々のチームがモデル内で存在する場所 (現在および変更後)、チームの関係と責任がどのように変化するか、およびメリットが組織への影響に見合っているかどうかを調べます。

次の 4 つの各運用モデルを使用して成功を実現することができます。一部のモデルは、特定のユースケースや、開発における特定のポイントに適しています。これらのモデルによっては、現在の環境で使用しているモデルよりも利点が大きい場合があります。

トピック

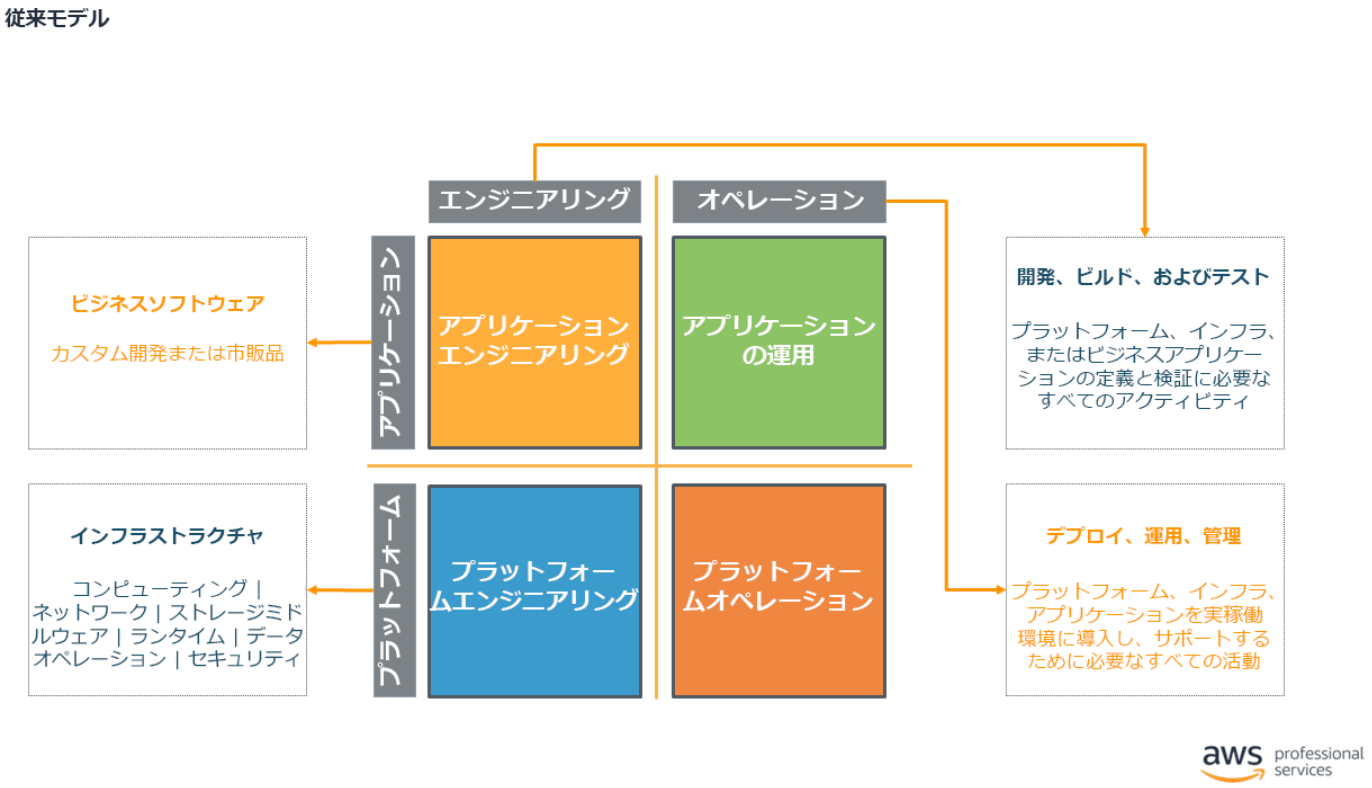
- 完全に分離された運用モデル
- 分離されたアプリケーションのエンジニアリングと運用 (AEO) および一元化されたガバナンスを備えたインフラストラクチャのエンジニアリングと運用 (IEO)
- 分離された AEO および一元化されたガバナンスとサービスプロバイダーを備えた IEO
- 分離された AEO および一元化されたガバナンスとサービスプロバイダーコンサルティングパートナーを備えた IEO
- 分離された AEO と一元化されていないガバナンスを備えた IEO

完全に分離された運用モデル

次の図では、縦軸にアプリケーションとインフラストラクチャが設定されています。アプリケーションとは、ビジネス成果を提供するワークロードを指し、カスタム開発または購入したソフトウェアで

あると考えます。インフラストラクチャとは、物理および仮想インフラストラクチャと、そのワークロードをサポートするその他のソフトウェアを指します。

横軸には、エンジニアリングと運用が設定されています。エンジニアリングとは、アプリケーションやインフラストラクチャの開発、構築、テストを指します。運用とは、アプリケーションとインフラストラクチャのデプロイ、更新、および継続的なサポートのことです。



多くの組織では、この「完全に分離された」モデルが存在します。各クアドラントのアクティビティは、個別のチームによって実行されます。 作業は、作業リクエスト、作業キュー、チケットなどのメカニズムを介して、または IT サービス管理 (ITSM) システムを使用してチーム間で渡されます。

チームへの、またはチーム間でのタスクを移行すると、複雑性が増し、ボトルネックや遅延が生じてしまいます。リクエストは、優先順位が高くなるまで遅延することがあります。遅れて特定された不具合は大幅な再処理が必要になる可能性があり、同じチームとその機能を再び通過する必要が生じます。エンジニアリングチームによるアクションを必要とするインシデントがある場合、引き渡しのアクティビティによって対応が遅れてしまいます。

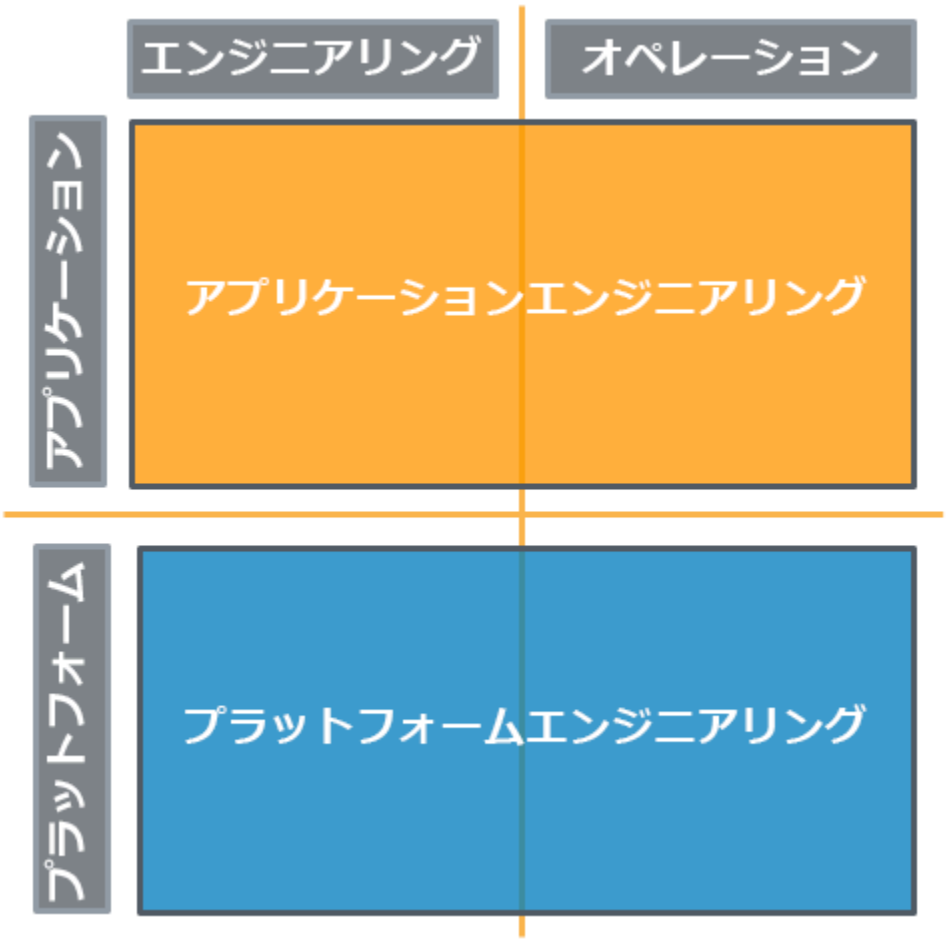
業務チーム、開発チーム、運用チームが、実行されているアクティビティや機能を中心に編成されている場合には、調整不良のリスクが高くなります。これでは、チームはビジネス成果の達成に集中するのではなく、特定の責任に集中することになってしまいます。チームは専門的、物理的、あるいは

論理的に細かく分けられて隔離されることがあり、コミュニケーションや共同作業の妨げになる場合があります。

分離されたアプリケーションのエンジニアリングと運用 (AEO) および一元化されたガバナンスを備えたインフラストラクチャのエンジニアリングと運用 (IEO)

この「分離された AEO と IEO」モデルでは、「自分で構築して実行する」という方法論に従います。

アプリケーションエンジニアとデベロッパーは、ワークロードのエンジニアリングと運用の両方を実行します。同様に、インフラストラクチャエンジニアは、アプリケーションチームをサポートするために使用するプラットフォームのエンジニアリングと運用の両方を実行します。



この例では、ガバナンスを一元管理として扱います。標準はアプリケーションチームに分散、提供、または共有されます。

などの、アカウント間で環境を集中管理できるツールまたはサービスを使用する必要があります。AWS Organizations。などのサービス AWS Control Tower では、この管理機能が拡張されており、アカウントのセットアップに関するブループリント (運用モデルのサポート) を定義し、AWS Organizations を使用して進行中のガバナンスを適用し、新しいアカウントのプロビジョニングを自動化することができます。

「自分で構築して実行する」ということは、アプリケーションチームが完全なスタック、ツールチェーン、およびプラットフォームの責任を負うという意味ではありません。

プラットフォームエンジニアリングチームは、標準化された一連のサービス (開発ツール、モニタリングツール、バックアップおよび復旧ツール、ネットワークなど) をアプリケーションチームに提供します。プラットフォームチームは、承認されたクラウドプロバイダーサービス、同じ特定の設定、またはその両方へのアクセスをアプリケーションチームに提供することもできます。

承認されたサービスと設定をデプロイするためのセルフサービス機能を提供するメカニズムである Service Catalogは、ガバナンスを実施しながらフルフィルメントリクエストに関連する遅延を制限するのに役立ちます。

プラットフォームチームはスタック全体の可視性を確保します。それにより、アプリケーションチームはアプリケーションコンポーネントに関する問題と、アプリケーションが消費するサービスやインフラストラクチャコンポーネントとを区別できます。プラットフォームチームは、これらのサービスの設定に関する支援や、アプリケーションチームの運用を改善する方法に関するガイダンスを提供することもできます。

前に説明したように、アプリケーションチームが、チームのアクティビティやアプリケーションのイノベーションをサポートする標準の追加、変更、例外をリクエストするメカニズムが存在することが不可欠です。

分離された AEO IEO モデルは、アプリケーションチームに強力なフィードバックループを提供します。ワークロードの日々の運用は、直接的なやり取り、またはサポートや機能のリクエストを介した間接的なやりとりを通じてお客様とのコンタクトを増やします。このような可視性の向上により、アプリケーションチームはより迅速に問題に対処できるようになります。より深いつながりと密接な関係により、お客様のニーズに対するインサイトが得られ、より迅速なイノベーションが可能になります。

これらはすべて、アプリケーションチームをサポートするプラットフォームチームにも当てはまります。

採用された標準は使用前に承認され、本番環境への投入に必要なレビューの量が減る場合があります。プラットフォームチームによって提供される、サポートおよびテスト済みの標準を使用すると、

これらのサービスに関する問題の頻度を減らすことができます。標準を採用することで、アプリケーションチームはワークロードの差別化に焦点を当てることができます。

分離された AEO および一元化されたガバナンスとサービスプロバイダーを備えた IEO

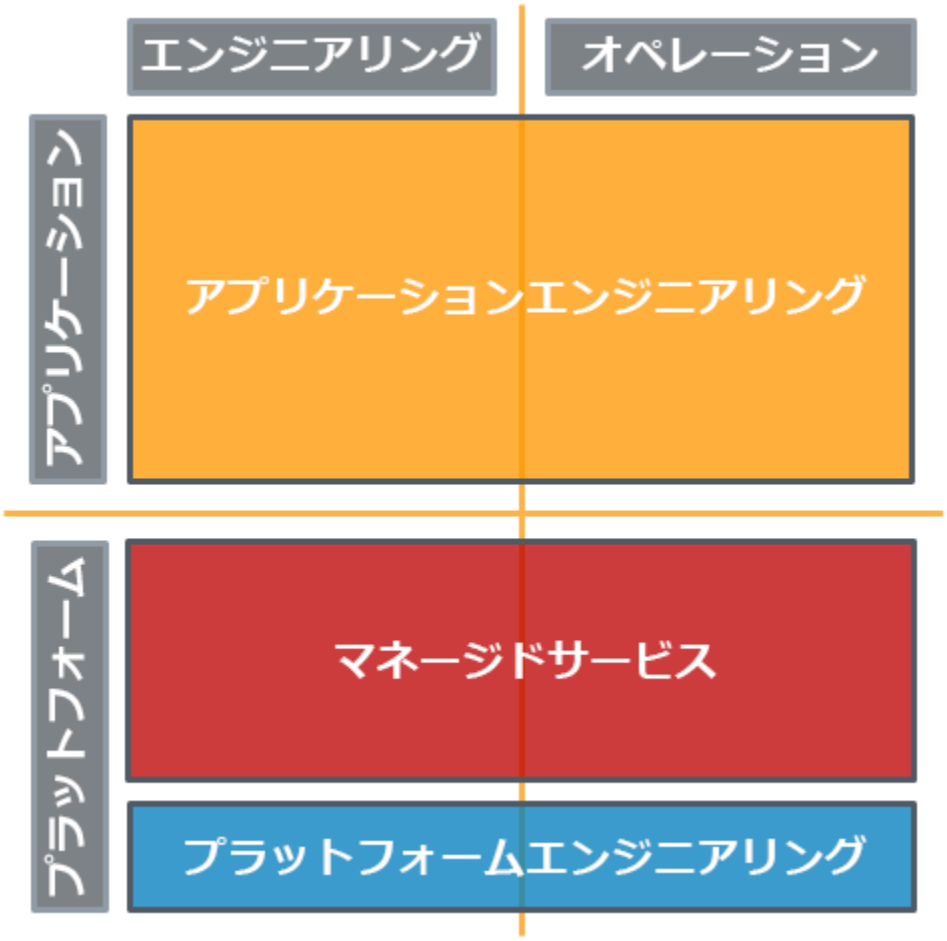
この「分離された AEO と IEO」モデルでは、「自分で構築して実行する」という方法論に従います。

アプリケーションエンジニアとデベロッパーは、ワークロードのエンジニアリングと運用の両方を実行します。

組織には、専用のプラットフォームエンジニアリングと運用チームをサポートするための既存のスキルやチームメンバーが存在しない場合があります。またそのための時間と労力の投資に積極的でないことも考えられます。

あるいは、ビジネスを差別化する機能の作成に重点を置いたプラットフォームチームが必要な場合もありますが、差別化につながらない日常業務は業者に任せたいという場合もあります。

などのマネージドサービスプロバイダー AWS Managed Services、AWS Managed Services パートナー、または AWS パートナーネットワークのマネージドサービスプロバイダーは、専門的な実装型のクラウド環境を提供し、セキュリティとコンプライアンスの要件、ビジネスの目標をサポートします。



このバリエーションでは、ガバナンスはプラットフォームチームによって一元管理され、アカウント作成と AWS Organizations および AWS Control Tower で管理されるポリシーがあります。

このモデルでは、サービスプロバイダーのメカニズムを操作するようにメカニズムを変更する必要があります。これは、サービスプロバイダーを含むチーム間のタスクの移行によって生じるボトルネックや遅延、または不具合の特定の遅れに関連する潜在的な再作業には対応していません。

プロバイダの標準、ベストプラクティス、プロセス、専門知識を活用できます。また、サービス提供の継続的な開発によるメリットも得られます。

マネージドサービスを運用モデルに追加すると、時間とリソースを節約でき、新しいスキルや能力を開発するのではなく、戦略的成果に集中して社内チームを維持できます。

分離された AEO および一元化されたガバナンスとサービスプロバイダーコンサルティングパートナーを備えた IEO

この「分離された AEO と IEO」モデルでは、「自分で構築して実行する」という方法論の確立を目指します。

アプリケーションチームのワークロードにエンジニアリングと運用のアクティビティを含め、より DevOps に近い文化の定着を目指します。

しかし、アプリケーションチームは、移行、クラウドの採用、ワークロードのモダナイゼーションの真っ只中で、クラウドやクラウド運用を十分にサポートするスキルをまだ持っていない場合があります。アプリケーションチームの能力の欠如や不慣れさが障害となることがあります。

この懸念に対処するために、質問と回答、議論、ソリューションの発見などを行うフォーラムを提供するクラウドセンターオブイネーブルメント (CCoE、Cloud Center of Enablement) チームを設置します。組織のニーズに応じて、CCoE は専門家の専属チームにすることも、組織全体から選ばれた人材で構成される仮想的なチームにすることもできます。CCoE を活用することで、チームのクラウドへのトランスフォーメーション、クラウドガバナンスの一元化、アカウントおよび組織管理標準の定義が可能になります。また、リファレンスアーキテクチャ、エンタープライズでのユースパターンの成功例を特定することもできます。

私たちは、CCoE を一般的に用いられているクラウドセンターオブエクセレンスの略ではなく、クラウドセンターオブイネーブルメントの略として用います。これは、サポートされるチームの成功とビジネス成果の達成をより重視しているためです。

アプリケーションチームが採用できるように、プラットフォームエンジニアリングチームはこれらの標準に基づいたコアとなる共有プラットフォーム機能を構築します。リファレンスアーキテクチャ、およびセルフサービスメカニズムを介してアプリケーションチームに提供されるパターンをコード化します。AWS Service Catalog などのサービスを使用することで、アプリケーションチームは承認済のリファレンスアーキテクチャ、パターン、サービス、構成、一元化されたガバナンスおよびセキュリティ標準による既定のコンプライアンスなどをデプロイできます。

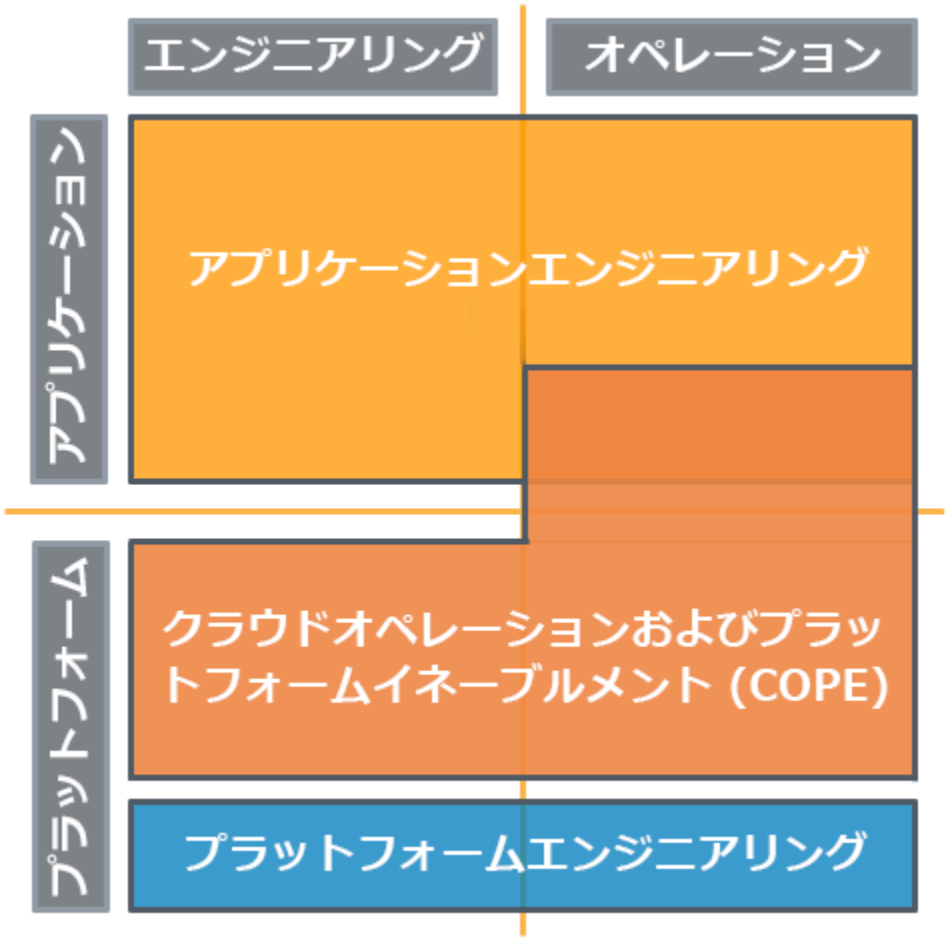
またプラットフォームエンジニアリングチームは、標準化された一連のサービス (開発ツール、モニタリングツール、バックアップおよび復旧ツール、ネットワークなど) をアプリケーションチームに提供します。

組織は標準化されたサービスを管理/サポートし、アプリケーションチームによるリファレンスアーキテクチャおよびパターンに基づくクラウドプレゼンスの確立を支援する「内部 MSP およびコンサルティングパートナー」を利用できます。これはクラウドオペレーションおよびプラットフォームイネーブルメント (COPE) チームと呼ばれ、アプリケーションチームによる基本的な運用の確立を支

援し、アプリケーションチームのシステムとリソースに対する役割と責任が継続的に増加するように促します。COPE チームは CCoE およびプラットフォームエンジニアリングチームと協力しながら継続的な改善を進め、アプリケーションチームの支持者として活動します。

アプリケーションチームは、環境、CICD パイプライン、変更管理、可観測性およびモニタリング、インシデント/イベント管理プロセスのセットアップにおいて COPE チームからの支援を得ます。COPE チームはアプリケーションチームによるこれらの運用アクティビティに参加しますが、時間の経過とともにアプリケーションチームに所有権を移すため、ゆるやかにフェードアウトします。

アプリケーションチームは、COPE チームのスキルとこれまでに組織で得た学びから恩恵を受けることができます。つまり、一元化されたガバナンスを通じたガードレールで守られることになります。アプリケーションチームは社内に存在する過去の成功の上に構築され、採用した組織標準の継続的な更新による恩恵を得ます。可観測性およびモニタリングを介してワークロードの運用に関する優れたインサイトを得ることができ、ワークロードに対して行った変更の影響をより深く理解できます。



COPE チームは運用アクティビティをサポートするために必要なアクセスを保持し、複数のアプリケーションチームにまたがったエンタープライズレベルの運用ビューと重大インシデントの管理サポートを提供します。また COPE チームは付加価値を生まない手間のかかる作業に分類されるアクティビティに対する責任を保持します。COPE チームは、スケール環境での標準のサポートソリューションを通じてこの責任を果たします。さらに、アプリケーションチームがアプリケーションの差別化に集中できるように、一般的なプログラミングや自動化された運用アクティビティの管理を引き続き行います。

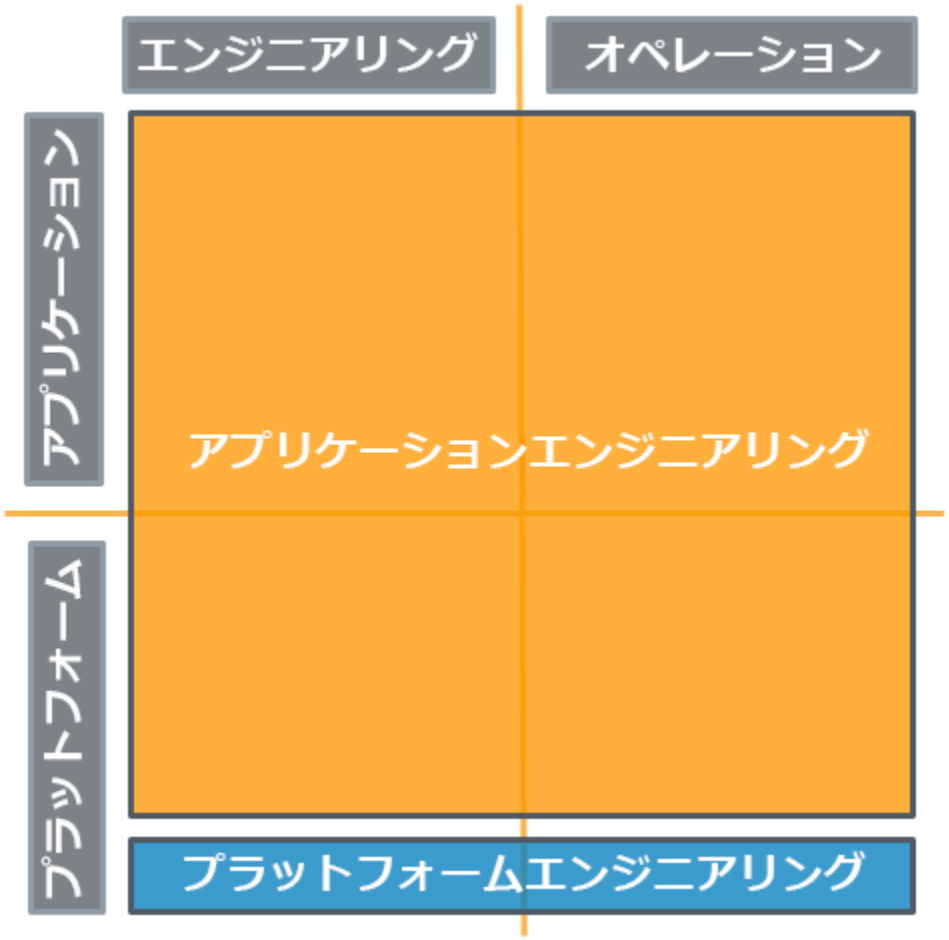
各チームがもたらす組織の標準、ベストプラクティス、プロセス、専門知識による恩恵を受けることができます。新しいチームがクラウドを採用したりモダナイゼーションを行ったりする際に、これらの成功パターンを繰り返せるようなメカニズムを確立します。このモデルでは、COPE チームによるアプリケーションチームの確立、知識とアーティファクトの継承を支援する能力を重視しています。アプリケーションチームによる広範囲の独立性の取得の失敗リスクに対処するために、アプリケーションチームの運用負荷を低減します。CCoE、COPE、アプリケーションチーム間の関係を確立し、進化と革新をさらに進めるためのフィードバックループを作り出します。

組織全体の標準を定義しながら CCoE と COPE チームを確立することで、クラウドの採用を促進し、モダナイゼーションプロセスをサポートできます。アプリケーションチームに対する COPE チームによる追加のサポートをコンサルタントやパートナーとして位置づけることで、アプリケーションチームは障壁を取り除き、より迅速にクラウド機能のメリットを採用できるようになります。

分離された AEO と一元化されていないガバナンスを備えた IEO

この「分離された AEO と IEO」モデルでは、「自分で構築して実行する」という方法論に従います。

アプリケーションエンジニアとデベロッパーは、ワークロードのエンジニアリングと運用の両方を実行します。同様に、インフラストラクチャエンジニアは、アプリケーションチームをサポートするために使用するプラットフォームのエンジニアリングと運用の両方を実行します。



この例では、一元管理されていないものとしてガバナンスを扱います。

標準はプラットフォームチームによってアプリケーションチームに分散、提供、または共有されますが、アプリケーションチームはワークロードに対応するために新しいプラットフォーム機能を自由に設計および運用できます。

このモデルでは、アプリケーションチームに対する制約が少なくなりますが、責任は大幅に増加します。追加のプラットフォーム機能をサポートするためには、追加のスキルや、チームメンバーになる可能性のある人が存在する必要があります。スキルセットが適切でなく、不具合が早期に認識されない場合、大幅な再作業のリスクが高まります。

アプリケーションチームに特に委任されていないポリシーを適用する必要があります。AWS Organizations などの、アカウント間で環境を一元管理できるツールまたはサービスを 使用します。AWS Control Tower などのサービスでは、 この管理機能が拡張されており、アカウントのセットアップに関するブループリント (運用モデルのサポート) を定義し、AWS Organizations を使用し

て進行中のガバナンスを適用し、新しいアカウントのプロビジョニングを自動化することができます。

アプリケーションチームが標準への追加や変更をリクエストするためのメカニズムを持つことは有益です。他のアプリケーションチームに利益をもたらす新しい標準にチームが貢献できる可能性があります。プラットフォームチームは、これらの追加機能の直接サポートを提供することが、ビジネス成果につながる効果的なサポートであると判断する可能性があります。

このモデルでは、高度なスキルやチームメンバーなどの要件によって、イノベーションに対する制約を制限します。チーム間のタスクの移行によって生成されるボトルネックや遅延の多くに対処しながら、チームとお客様との間の効果的な関係の発展を促進します。

関係性と所有権

運用モデルは、チーム間の関係を定義し、識別可能な所有権と責任をサポートします。

ベストプラクティス

- OPS02-BP01 リソースには特定の所有者が存在する
- OPS02-BP02 プロセスと手順には特定の所有者が存在する
- OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する
- OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する
- OPS02-BP05 追加、変更、除外をリクエストするメカニズムが存在する
- OPS02-BP06 チーム間の責任は事前定義済みまたは交渉済みである

OPS02-BP01 リソースには特定の所有者が存在する

ワークロードのリソースには、変更管理、トラブルシューティング、その他機能を受け持つ、特定できる所有者が必要です。所有者は、ワークロード、アカウント、インフラストラクチャ、プラットフォーム、アプリケーションに割り当てられます。所有権は、一元登録などのツール、またはリソースに添付されたメタデータを使用して記録されます。コンポーネントのビジネス価値で、それらに適用されるプロセスと手順が決まります。

期待される成果:

- リソースに、メタデータまたは一元登録を使用して特定できる所有者がいます。
- チームメンバーが、リソースを誰が所有しているかを特定できます。

- アカウントの所有者は、可能な限り 1 人です。

一般的なアンチパターン:

- AWS アカウントのその他の連絡先が入力されていない。
- リソースに、それを所有するチームを特定するタグがない。
- E メールマッピングのない ITSM キューがある。
- インフラストラクチャの重要な部分で 2 つのチームの所有権が重複している。

このベストプラクティスを活用するメリット:

- リソースの変更管理は、所有権が割り当てられていて、わかりやすくなっています。
- トラブルシューティングが発生した場合に、適切な所有者を関与させることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

環境内のリソースユースケースにおける所有権の意味を定義します。所有権とは、リソースの変更を監督する人、トラブルシューティング中にリソースをサポートする人、または財務的な責任者を意味することもあります。名前、連絡先情報、組織、チームなどでリソースの所有者を指定し、記録します。

お客様事例

AnyCompany Retail は、所有権を、リソースの変更とサポートを担当するチームまたは個人と定義しています。AWS Organizations を活用して AWS アカウントを管理しています。予備のアカウント連絡先は、グループ受信箱を使用して設定されています。各 ITSM キューは、E メールエイリアスにマッピングされています。タグによって、誰が AWS リソースを所有しているかを特定できます。その他のプラットフォームやインフラストラクチャについては、所有権と連絡先情報を特定できる wiki ページがあります。

実装手順

1. 組織における所有権を定義することから始めます。所有権は、リソースのリスクに対して責任を持つ人、リソースの変更を担当する人、またはトラブルシューティング時にリソースをサポートする人などを意味します。また、リソースの財務的または管理的な所有権を意味することもあります。

2. AWS Organizations を使用してアカウントを管理します。アカウントのその他の連絡先を一元管理できます。
   a. 会社の E メールアドレスや電話番号を連絡先として使用することで、その E メールアドレスや電話番号の持ち主が組織から離れた場合でも、連絡先にアクセスすることができます。例えば、請求、オペレーション、セキュリティ用に別々の E メール配信リストを作成し、アクティブな AWS アカウントごとに請求、セキュリティ、オペレーションの連絡先として設定します。誰かが休暇を取っていたり、担当が変わったり、会社を辞めたりした場合でも、複数の人が AWS の通知を受け取り、対応できるようになります。
   b. アカウントが AWS Organizations で管理されていない場合、必要に応じてその他の連絡先を利用して AWS が適切な担当者に連絡を取ることができます。アカウントの代替連絡先は、個人ではなくグループを指定して設定してください。
3. タグを使用して AWS のリソースの所有者を特定します。所有者と連絡先情報の両方を、別々のタグで指定できます。
   a. AWS Config ルールを使用して、リソースに必須の所有権タグをつけるように強制できます。
   b. 組織においてタグ付け戦略を策定する方法に関する詳細なガイダンスについては、AWS リソースのタグ付けのベストプラクティス (ホワイトペーパー)を参照してください。
4. 生成 AI を活用する会話型アシスタント、Amazon Q Business を使用して、従業員の生産性を高め、質問に回答し、エンタープライズシステムの情報に基づいてタスクを完了します。
   a. Amazon Q Business を会社のデータソースに接続します。Amazon Q Business には、Amazon Simple Storage Service (Amazon S3)、Microsoft SharePoint、Salesforce、Atlassian Confluence など、40 を超えるサポート対象のデータソースへの事前構築済みのコネクタが用意されています。詳細については、「Amazon Q Business のコネクタ」を参照してください。
5. その他のリソース、プラットフォーム、インフラストラクチャについては、所有権を特定するドキュメントを作成します。このドキュメントはチームメンバーが誰でも利用できるようにします。

実装計画に必要な工数レベル: 低。アカウントの連絡先情報およびタグを利用して、AWS リソースの所有権を割り当てます。その他のリソースについては、wiki の表などシンプルなものを使用して所有権と連絡先情報を記録するか、ITSM ツールを使用して所有権をマッピングします。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する
- OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する

関連するドキュメント:

- AWS Account Management - Updating contact information
- AWS Organizations - Updating alternative contacts in your organization
- AWS リソースのタグ付けのベストプラクティス (ホワイトペーパー)
- Amazon Q と AWS IAM Identity Center を利用して、プライベートでセキュアなエンタープライズ生成 AI アプリケーションを開発する
- 生成 AI を使用して従業員の生産性向上を支援する Amazon Q Business の一般提供開始
- AWS クラウド Operations & Migrations ブログ - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security ブログ - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps ブログ - Integrating AWS CloudFormation Guard into CI/CD pipelines

関連ワークショップ:

- AWS Workshop - Tagging

関連する例:

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

関連サービス:

- AWS Config ルール - required-tags
- AWS Organizations

OPS02-BP02 プロセスと手順には特定の所有者が存在する

個々のプロセスと手順の定義を誰が所有しているか、特定のプロセスと手順が使用されている理由、およびその所有権が存在する理由を理解します。特定のプロセスと手順が使用される理由を理解することで、改善の機会を見極めることができます。

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- プロセスと手順に対し、その定義に責任を持つ所有者が指定されています。
  - ワークロードのサポートにおいて実施される運用アクティビティを特定します。これらのアクティビティを検出可能な場所に文書化します。
  - アクティビティの仕様に責任を有する個人またはチームを一意に特定します。当該個人またはチームは、適切なアクセス許可、アクセス、およびツールを持つ適切なスキルのあるチームメンバーが正常に実行できるようにする責任があります。そのアクティビティの実行に問題がある場合、アクティビティの改善に必要となる詳細なフィードバックを提供する責任はそのチームメンバーにあります。
  - AWS Systems Manager などのサービス、ドキュメント、AWS Lambda を通じて、アクティビティアーティファクトのメタデータの所有権をキャプチャします。タグまたはリソースグ

ループを使用してリソースの所有権をキャプチャし、所有権と連絡先情報を指定します。AWS Organizations を使用してタグ付けポリシーを作成し、所有権と連絡先情報をキャプチャします。

- 時間が経つにつれて、これらの手順はコードとして実行できるように進化し、人的介入の必要が減るはずです。
  - 例えば、AWS Lambda 関数、CloudFormation テンプレート、AWS Systems Manager オートメーションのドキュメントを検討します。
  - 適切なリポジトリでバージョン管理を行います。
  - 所有者と文書を簡単に識別できるように、適切なリソースタグを付けてください。

お客様事例

AnyCompany Retail では、所有権を 1 つまたは (共通のアーキテクチャプラクティスとテクノロジーを共有する) 複数のアプリケーションのプロセスを所有するチームまたは個人と定義しています。最初にプロセスと手順をステップバイステップガイドとしてドキュメント管理システムに文書化し、アプリケーションをホストする AWS アカウントと、アカウント内の特定のリソースグループのタグを使用して手順を検出可能にしています。同社は AWS Organizations を活用して AWS アカウントを管理しています。時間の経過に伴い、これらのプロセスはコードに変換され、リソースは Infrastructure as Code (CloudFormation または AWS Cloud Development Kit (AWS CDK) テンプレートなど) を使用して定義されます。運用プロセスは AWS Systems Manager または AWS Lambda 関数でオートメーションドキュメントとなります。これらの関数は、スケジュールされたタスクとして、AWS CloudWatch アラームや AWS EventBridge イベントなどのイベントへの応答として、または IT サービス管理 (ITSM) プラットフォーム内のリクエストによって起動できます。すべてのプロセスには、所有者を識別するタグが付いています。オートメーションとプロセスのドキュメントは、プロセスのコードリポジトリによって生成された Wiki ページ内で管理されます。

実装手順

1. 既存のプロセスと手順を文書化します。
   a. レビューを行い、最新の状態に保ちます。
   b. 各プロセスまたは手順の所有者を特定します。
   c. それらをバージョン管理下に置きます。
   d. 可能な場合は、アーキテクチャ設計を共有するワークロードや環境全体でプロセスと手順を共有します。
2. フィードバックと改善のためのメカニズムを確立します。

a. プロセスをレビューする頻度に関するポリシーを定義します。

b. レビュー担当者と承認者用のプロセスを定義します。

c. フィードバックを提供し、追跡するための問題やチケットキューを実装します。

d. プロセスと手順は、可能な限り、変更承認委員会 (CAB) による事前承認とリスク分類を受ける必要があります。

3. プロセスと手順は、それらを実行するユーザーがアクセスおよび検出できることを確認します。

a. タグを使用して、ワークロードのプロセスと手順にアクセスできる場所を示します。

b. 有意義なエラーやイベントのメッセージを活用して、問題に対処するための適切なプロセスや手順を示します。

c. Wiki とドキュメント管理を使用して、プロセスと手順を組織全体で一貫して検索できるようにします。

4. 必要に応じて自動化します。

a. サービスやテクノロジーが API を提供している場合は、オートメーションを開発する必要があります。

b. プロセスに関する指導を十分に行います。これらのプロセスを自動化するためのユーザーストーリーと要件を作成します。

c. プロセスと手順の使用状況を適切に評価し、問題があれば反復的な改善に役立てます。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS ホワイトペーパー - Introduction to DevOps on AWS
- AWS ホワイトペーパー | AWS リソースのタグ付けのベストプラクティス
- AWS ホワイトペーパー | Organizing Your AWS Environment Using Multiple Accounts

- AWS クラウド Operations & Migrations ブログ - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS クラウド Operations & Migrations ブログ - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations
- AWS Security ブログ - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps ブログ - Integrating AWS CloudFormation Guard into CI/CD pipelines

関連ワークショップ:

- AWS Well-Architected Operational Excellence Workshop
- AWS Workshop - Tagging

関連動画:

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

関連サービス:

- AWS Systems Manager Automation
- AWS Service Management Connector

OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する

定義されたワークロードに対して特定のアクティビティを実行する責任を持つ者と、その責任が存在する理由を理解します。運用アクティビティを実行することに責任を負うのが誰かを理解すると、アクティビティを実行する人物、結果を検証する人物、アクティビティの所有者にフィードバックを提供する人物を把握できます。

期待される成果:

組織において、定義されたワークロードで特定のアクティビティを実行し、そのワークロードによって生成されたイベントに対応する責任が明確に定義されています。組織は、プロセスの所有権と履行

を文書化し、この情報を検出可能にしています。組織で変更があった場合は責任を見直して更新し、チームは欠点や非効率を特定するアクティビティのパフォーマンスを追跡して測定します。フィードバックメカニズムを実装して、欠点や改善を追跡し、反復的な改善をサポートします。

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- アクティビティを実行する責任を持つのは誰か、アクションが必要なときに誰に通知すべきか、アクションを実行し、結果を検証してアクティビティの所有者にフィードバックを提供するのは誰かを理解できます。
- プロセスと手順が整備されていると、ワークロードの運用努力の効果が上がります。
- 新しいチームメンバーがより早く成果を出せるようになります。
- インシデントを軽減するための時間を短縮できます。
- さまざまなチームが同じプロセスと手順を使用して、一貫した方法でタスクを実行できます。
- 繰り返し使用可能なプロセスを持つことで、チームがプロセスをスケールすることができます。
- プロセスと手順が標準化されているため、チーム間でワークロードの責任を移転することによる影響を軽減できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

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

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

お客様事例

AnyCompany Retail では、所有権を 1 つまたは (共通のアーキテクチャプラクティスとテクノロジーを共有する) 複数のアプリケーションのプロセスを所有するチームまたは個人と定義しています。同社は、最初にプロセスと手順をステップバイステップガイドとしてドキュメント管理システムに文書化しています。アプリケーションをホストする AWS アカウントと、アカウント内の特定のリソースグループのタグを使用して手順を検出可能にし、AWS Organizations を使用して AWS アカウントを管理しています。時間の経過に伴って、AnyCompany Retail はこれらのプロセスをコードに変換し、Infrastructure as Code を使用して (CloudFormation または AWS Cloud Development Kit (AWS CDK) テンプレートなどのサービスを通じて) リソースを定義します。運用プロセスは AWS Systems Manager または AWS Lambda 関数でオートメーションドキュメントとなります。これらの関数は、スケジュールされたタスクとして、Amazon CloudWatch アラームや Amazon EventBridge イベントなどのイベントへの応答として、または IT サービス管理 (ITSM) プラットフォーム内のリクエストによって起動できます。すべてのプロセスには、所有者を識別するタグが付いています。チームは、プロセスのコードリポジトリによって生成された Wiki ページ内で、オートメーションとプロセスのドキュメントを管理しています。

実装手順

1. 既存のプロセスと手順を文書化します。
    a. レビューを行い、最新の状態であることを確認します。
    b. 各プロセスまたは手順に所有者がいることを確認します。
    c. 手順をバージョン管理下に置きます。
    d. 可能な場合は、アーキテクチャ設計を共有するワークロードや環境全体でプロセスと手順を共有します。
2. フィードバックと改善のためのメカニズムを確立します。

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

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS02-BP01 リソースには特定の所有者が存在する
- OPS02-BP02 プロセスと手順には特定の所有者が存在する
- OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する
- OPS02-BP05 責任と所有権を特定するためのメカニズムが存在する
- OPS11-BP04 ナレッジ管理を実施する

関連するドキュメント:

- AWS ホワイトペーパー | Introduction to DevOps on AWS
- AWS ホワイトペーパー | AWS リソースのタグ付けのベストプラクティス
- AWS ホワイトペーパー | Organizing Your AWS Environment Using Multiple Accounts

- AWS クラウド Operations & Migrations ブログ | Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Workshop - Tagging
- AWS Service Management Connector

関連動画:

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

関連する例:

- AWS Well-Architected Operational Excellence Workshop

OPS02-BP04 責任と所有権を管理するためのメカニズムが存在する

自分の役割の責任と、ビジネスの成果に自分がどのように貢献するかを理解することで、タスクの優先順位付けと役割が重要である理由を知ることができます。これにより、チームメンバーはニーズを認識し、適切に対応できます。チームメンバーが各自の役割を把握していると、所有権を確立し、改善の機会を特定できるとともに、影響を与える方法や適切な変更を行う方法を理解できます。

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

期待される成果: リソース、実行すべきアクション、プロセス、手順との関連性を含む責任が組織内のチームに対して明確に定義されています。これらの責任は、チームの責任と目標、および他のチームの責任と一致しています。エスカレーションパスを一貫性のある検出可能な方法で文書化し、決定内容を責任マトリクス、チーム定義、Wiki ページなどのドキュメントアーティファクトに反映させます。

一般的なアンチパターン:

- チームの責任が不明瞭、または明確に定義されていない。

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

このベストプラクティスを活用するメリット:

- 責任者または所有者が誰かを理解することで、適切なチームまたはチームメンバーに連絡して、リクエストをしたり、タスクを移行したりすることができる。
- 不作為やニーズへの未対応というリスクを低減するために、責任や所有権の割り当て権限を持つ個人を特定できる。
- 責任範囲が明確に定義されている場合、チームメンバーの自主性と所有権が高まる。
- 責任を理解することで、決定すべき事項、実行すべきアクション、および適切な所有者に引き渡すべきアクティビティが明確になる。
- 何がチームの責任範囲外かをしっかりと理解しているため、放棄された責任を特定しやすくなり、明確化のためにエスカレーションできるようになる。
- チームの混乱や緊張を防ぎ、チームがワークロードとリソースをより適切に管理できるようになる。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

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

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

お客様事例

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

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

実装手順

1. 既存のドキュメントから始めます。一般的なソースドキュメントには以下が含まれます。
    a. 責任または実行責任者 (responsible)、説明責任者 (accountable)、相談先 (consulted)、報告先 (informed) (RACI) のマトリクス
    b. チーム定義または Wiki ページ
    c. サービスの定義とサービス内容
    d. 役割または職務内容
2. 文書化された責任をレビューし、ディスカッションを設けます。
    a. チームとレビューを行って、文書化された責任と、チームが通常遂行している責任との間の不一致を特定します。
    b. 内部顧客が提供している可能性のあるサービスについて話し合い、チーム間での期待事項のギャップを特定します。
3. 相違点を分析して対処します。

4. 改善の機会を特定します。
    a. 頻繁にリクエストされ、リソースを大量に消費するリクエストを特定します。こうしたリクエストは一般的に有力な改善候補と考えられます。
    b. ベストプラクティス、パターン、規範ガイダンスを探し、このガイダンスを基に改善を簡素化し標準化します。
    c. 改善の機会を記録し、完了まで追跡します。
5. チームが責任の割り当てを管理および追跡する責任を負っていない場合、その責任を担うチームメンバーを指定します。
6. チームが責任の明確化を求めるためのプロセスを定義します。
    a. プロセスを見直し、明確で使いやすいことを確認します。
    b. 必ず誰かがエスカレーションに責任を持ち、完了まで追跡するようにします。
    c. 効果を測定するための運用メトリクスを設定します。
    d. フィードバックメカニズムを作成して、チームが改善の機会を強調できるようにします。
    e. 定期的なレビューのメカニズムを導入します。
7. 検出とアクセスが可能な場所にドキュメントを保管します。
    a. Wiki またはドキュメントポータルが一般的です。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS01-BP06 トレードオフを評価する
- OPS03-BP02 チームメンバーに、結果にリスクがあるときにアクションを実行する権限が付与されている
- OPS03-BP03 エスカレーションが推奨されている
- OPS03-BP07 チームに適正なリソースを提供する
- OPS09-BP01 メトリクスを使用して業務目標と KPI を測定する
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け
- OPS11-BP01 継続的改善のプロセスを用意する

関連するドキュメント:

- AWS ホワイトペーパー - Introduction to DevOps on AWS
- AWS ホワイトペーパー - AWS クラウド Adoption Framework: Operations Perspective
- AWS Well-Architected Framework Operational Excellence - Workload level Operating model topologies
- AWS 規範ガイダンス - Building your Cloud Operating Model
- AWS 規範ガイダンス - Create a RACI or RASCI matrix for a cloud operating model
- AWS クラウド Operations & Migrations ブログ - Delivering Business Value with Cloud Platform Teams
- AWS クラウド Operations & Migrations ブログ - Why a Cloud Operating Model?
- AWS DevOps ブログ - 組織のクラウドオペレーションをいかにモダナイズするか

関連動画:

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

OPS02-BP05 追加、変更、除外をリクエストするメカニズムが存在する

プロセス、手順、およびリソースの所有者にリクエストを送信できます。リクエストには、追加、変更、除外などがあります。このようなリクエストは変更管理プロセスを通ります。利点とリスクを評価した後、実行可能で適切であると判断されたリクエストを、十分な情報に基づいて承認します。

期待される成果:

- 割り当てられた所有権に基づいて、プロセス、手順、リソースの変更をリクエストできます。
- 変更は、メリットとリスクを検討して、熟考の上で行われます。

一般的なアンチパターン:

- アプリケーションをデプロイする方法を更新しなければならないが、オペレーションチームからデプロイプロセスの変更をリクエストする方法がない。
- ディザスタリカバリ計画を更新しなければならないが、変更のリクエスト先になる特定できる所有者がいない。

このベストプラクティスを活用するメリット:

- プロセス、手順、リソースを、要件の変更に合わせて進化させることができます。
- 所有者は十分な情報に基づいて変更時期を決定できます。
- 変更は熟考の上で行われます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

このベストプラクティスを実装するには、プロセス、手順、リソースに対する変更のリクエストが可能である必要があります。変更管理プロセスは簡単なものでかまいません。変更管理プロセスを文書化します。

お客様事例

AnyCompany Retail は責任割り当て (RACI) マトリックスを使用して、プロセス、手順、リソースの変更を所有しているのが誰かを特定しています。文書化された変更管理プロセスは、簡単で従いやすいものです。RACI マトリックスとこのプロセスを使用して、誰でも変更リクエストを送信できます。

実装手順

1. ワークロードのプロセス、手順、リソースと、それぞれの所有者を特定します。ナレッジマネジメントシステムにそれらを記録します。
   a. OPS02-BP01 リソースには特定の所有者が存在する、OPS02-BP02 プロセスと手順には特定の所有者が存在する、または OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する を実装していない場合は、先に実装します。
2. 組織の関係者と協力して、変更管理プロセスを作成します。このプロセスは、リソース、プロセス、手順の追加、変更、除外を対象とします。
   a. AWS Systems Manager Change Manager を、ワークロードリソースの変更管理プラットフォームとして使用できます。
3. ナレッジマネジメントシステムに、変更管理プロセスを記録します。

実装計画に必要な工数レベル: 中。変更管理プロセスの作成では、組織全体の複数の関係者と協調する必要があります。

リソース

関連するベストプラクティス:

- OPS02-BP01 リソースには特定の所有者が存在する - 変更管理プロセスを構築する前に、リソースに特定できる所有者がいる必要があります。
- OPS02-BP02 プロセスと手順には特定の所有者が存在する - 変更管理プロセスを構築する前に、プロセスに特定できる所有者がいる必要があります。
- OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する - 変更管理プロセスを構築する前に、オペレーションアクティビティに特定できる所有者がいる必要があります。

関連するドキュメント:

- AWS Prescriptive Guidance - Foundation palybook for AWS large migrations: Creating RACI matrices (AWS の規範的ガイダンス - AWS の大規模移行における基礎プレイブック: RACI マトリックスの作成)
- Change Management in the Cloud Whitepaper (クラウドにおける変更管理ホワイトペーパー)

関連サービス:

- AWS Systems Manager Change Manager

OPS02-BP06 チーム間の責任は事前定義済みまたは交渉済みである

チーム間には、チームがどのように連携し、互いにサポートするかを説明する、定義済みまたは交渉済みの契約があります (応答時間、サービスレベル目標、サービスレベルアグリーメントなど)。チーム間コミュニケーションチャネルが文書化されています。チームの仕事がビジネスの成果に及ぼす影響、および他のチームや組織の成果を理解することで、タスクの優先順位付けを知り、適切に対応できるようになります。

責任と所有権が未定義または不明な場合、必要な活動をタイムリーに処理できず、これらのニーズへの対応が重複し、競合する可能性のある作業が生じるリスクがあります。

期待される成果:

- チーム間の作業またはサポートに関する契約が、合意され文書化されています。
- 相互にサポートまたは協力するチームに、コミュニケーションチャネルおよび応答時間目標が定められています。

一般的なアンチパターン:

- 本稼働環境で問題が発生し、2 つの個別のチームが別個にトラブルシューティングを開始した。このサイロ化された作業のために停止時間が長くなった。
- オペレーションチームが開発チームの支援を必要としているが、応答時間の合意がない。リクエストが後回しにされる。

このベストプラクティスを活用するメリット:

- チームが相互にやり取りおよびサポートする方法を知っています。
- 応答性の目標が周知されています。
- コミュニケーションチャネルが明確に定義されています。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

このベストプラクティスを実装すると、チームが協力し合う方法についてあいまいさがなくなります。正式に合意を結ぶことで、チームの協力方法や互いにサポートする方法を体系化できます。チーム間コミュニケーションチャネルが文書化されます。

お客様事例

AnyCompany Retail の SRE チームは、開発チームとサービスレベルアグリーメントを結んでいます。開発チームがチケットシステムでリクエストを行う際に、15 分以内の応答を期待できます。サイトが停止した場合は、SRE チームが開発チームのサポートを受けながら調査を主導します。

実装手順

1. 組織全体の関係者と協力して、プロセスと手順に基づき、チーム間の契約を作成します。
    a. プロセスまたは手順が 2 チームで共有されている場合は、チームの協力方法に関するランブックを作成します。
    b. チーム間に依存関係がある場合は、リクエストについての応答 SLA を結びます。
2. 責任の所在をナレッジマネジメントシステムに記録します。

実装計画に必要な工数レベル: 中。チーム間に既存の契約がない場合、組織全体の関係者が合意に至るまでに工数がかかる場合があります。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する - チーム間で契約を結ぶ前に、プロセスの所有権を特定する必要があります。
- OPS02-BP03 パフォーマンスに責任を持つ所有者が運用アクティビティに存在する - チーム間で契約を結ぶ前に、オペレーションアクティビティの所有権を特定する必要があります。

関連するドキュメント:

- AWS Executive Insights - Empowering Innovation with the Two-Pizza Team (エグゼクティブのインサイト - 2 枚のピザチームでイノベーションを強化する)
- Introduction to DevOps on AWS - Two-Pizza Teams (AWS での DevOps 入門 - 2 枚のピザチーム)

組織カルチャー

チームメンバーにサポートを提供することで、チームメンバーがより効果的に行動し、 ビジネスの成果をサポートできるようにします。

ベストプラクティス

- OPS03-BP01 エグゼクティブスポンサーシップを提供する
- OPS03-BP02 チームメンバーに、結果にリスクがあるときにアクションを実行する権限が付与されている
- OPS03-BP03 エスカレーションが推奨されている
- OPS03-BP04 タイムリーで明確、かつ実用的なコミュニケーション
- OPS03-BP05 実験の推奨
- OPS03-BP06 チームメンバーがスキルセットを維持、強化することができ、それが推奨されている
- OPS03-BP07 チームに適正なリソースを提供する

OPS03-BP01 エグゼクティブスポンサーシップを提供する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット: エグゼクティブスポンサーがビジョン、方向性、目標を明確に伝え、共有していれば、チームメンバーはどのようなことが期待されているかを理解できます。リーダーが積極的に関与すると、個人とチームは定義された目標を達成するために同じ方向で集中的に尽力を開始します。この結果、組織は成功する能力を最大限に発揮できます。成功を評価すると、成功への障壁をより適切に特定して、エグゼクティブスポンサーの介入によって対処できるようになります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

- クラウドジャーニーのあらゆるフェーズ (移行、導入、または最適化) で成功を得るには、指名されたエグゼクティブスポンサーによるトップレベルのリーダーシップの積極的な関与が必要です。

エグゼクティブスポンサーは、定義された戦略に沿ってチームの考え方、スキルセット、作業方法を調整します。

- 理由を説明する: ビジョンと戦略の背後にある理由を明確に説明します。
- 期待値を設定する: 進捗状況や成功の測定方法など、組織の目標を定義して公開します。
- 目標の達成状況を追跡する: (タスクの完了のみでなく) 目標の段階的な達成状況を定期的に測定します。成果が危ぶまれる場合に適切な措置を講じることができるように、結果を共有します。
- 目標を達成するために必要なリソースを提供する: 従業員とチームを集めて協力体制を整え、定義された成果を実現する適切なソリューションを構築します。これにより、組織の摩擦が軽減または排除されます。
- チームを擁護する: チームとのエンゲージメントを維持し、進捗を理解して、影響を及ぼす外的要因があるかを把握します。チームの進行を妨げている障害を特定します。チームのために障害に対処し、不要な負担を取り除きます。チームが外部要因の影響を受けた場合、目標を再評価し、必要に応じてターゲットを調整します。
- ベストプラクティス導入の促進: 定量化可能な利点をもたらすベストプラクティスを認定し、その考案者と採用者を評価します。さらなる導入を推奨して、得られるメリットを拡大します。
- チームの進化を奨励する: 継続的改善の文化を創出して、進歩のみでなく失敗からも積極的に学びます。個人と組織の両者の成長と発展を奨励します。データやエピソードを利用して、ビジョンと戦略を進化させます。

お客様事例

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

実装手順

1. シングルスレッドリーダーシップを確立して、変革を主導し、推進する主要エグゼクティブスポンサーを割り当てます。
2. 変革のビジネス成果を明確に定義して、所有権と説明責任を割り当てます。主要エグゼクティブに重要な決定を主導して下す権限を付与します。
3. 変革戦略が非常に明確であり、エグゼクティブスポンサーから組織のあらゆるレベルに広く伝えられていることを確認します。
   a. IT とクラウドイニシアチブのビジネス目標を明確に定義します。
   b. IT およびクラウドトランスフォーメーションを推進するための主要なビジネスメトリクスを文書化します。

   c. 戦略の一端を担うすべてのチームおよび個人に着実にビジョンを伝えます。

4. 特定のリーダー、マネージャー、個別の貢献者にどのようなメッセージを伝える必要があるかを明記したコミュニケーション計画メトリクスを作成します。このようなメッセージを発信する人またはチームを指定します。

   a. コミュニケーション計画は、一貫性をもって確実に遂行します。

   b. 定期的に対面式のイベントを開催して、期待される内容を明確にして管理します。

   c. コミュニケーションの有効性に関するフィードバックを受け入れ、これに応じてコミュニケーションを調整し、計画を策定します。

   d. コミュニケーションイベントをスケジュールして、チームが抱える課題を積極的に把握し、必要に応じて方針を修正できるような一貫性あるフィードバックループを確立します。

5. リーダーシップの視点から各イニシアチブに積極的に関与して、影響を受けるすべてのチームが達成すべき成果を理解していることを確認します。

6. 各ステータスミーティングでは、エグゼクティブスポンサーは障害となる要因を探し、設定されたメトリクス、エピソード、またはチームからのフィードバックを調べ、目標に向けた進捗状況を測定する必要があります。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS03-BP04 タイムリーで明確、かつ実用的なコミュニケーション
- OP11-BP01 継続的改善のプロセスを用意する
- OPS11-BP07 オペレーションメトリクスのレビューを実行する

関連するドキュメント:

- Untangling Your Organisational Hairball: Highly Aligned
- The Living Transformation: Pragmatically approaching changes
- Becoming a Future-Ready Enterprise
- CCOE を構築するときに回避すべき 7 つの落し穴
- Navigating the Cloud: Key Performance Indicators for Success

関連動画:

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

関連する例:

- Prosci: 主要スポンサーの役割と重要性

OPS03-BP02 チームメンバーに、結果にリスクがあるときにアクションを実行する権限が付与されている

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

例えば、Amazon ではリーダーシッププリンシプルをガイドラインとして使用して、従業員が状況に応じて先に進み、問題を解決し、競合に対処して、アクションを起こすために期待される行動を促進しています。

期待される成果: リーダーシップは、組織の下位レベルであっても、個人やチームが重要な意思決定を下せるような新しい文化についての影響を与えています (ただし、意思決定が、監査可能な権限と安全メカニズムで定義されている場合に限る)。失敗を恐れないことが奨励され、チームは将来的に同様の状況に対処できるように、意思決定と対応を改善する方法を繰り返し学びます。その他のチームに利益をもたらすような改善につながったアクションがあれば、このようなアクションから学んだ知識を積極的に共有します。リーダーシップは、オペレーションの改善を測定し、個人や組織が同様のパターンを採用するようにインセンティブを提供します。

一般的なアンチパターン:

- リスクが特定された際に対処すべき内容についての明確なガイダンスやメカニズムが組織に存在しません。例えば、従業員がフィッシング攻撃を発見したときにセキュリティチームへの報告を怠った場合、組織の大部分が攻撃を受けてしまいます。これはデータ侵害の原因となります。
- サービスが利用できないことについて、顧客が不満を訴えています。サービスが利用できない主な原因は、デプロイの失敗です。デプロイツールは SRE チームが担当しており、デプロイの自動ロールバックは長期的なロードマップの対象となっています。最近のアプリケーションロールアウトで、エンジニアの 1 人がアプリケーションを以前のバージョンに自動的にロールバックするソ

リューションを考案しました。このソリューションは、SRE チームが採用するパターンとなる可能性があります。ただし、このような改善を追跡するプロセスがないため、その他のチームはこの方法を採用していません。組織は、顧客に影響を及ぼしてさらにマイナスのセンチメントを引き起こすデプロイの失敗に引き続き悩まされることになります。

- コンプライアンス維持のため、社内の情報セキュリティチームは、Amazon EC2 Linux インスタンスに接続するオペレーターに代わって、共有 SSH キーを定期的にローテーションするプロセスを長年管理しています。情報セキュリティチームがキーローテーションを完了するまでに数日かかり、その間の対象インスタンスへの接続はブロックされます。情報セキュリティにもその他のチームにも、同様の結果を得るために AWS のその他のオプションを利用することを提案する者はいません。

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

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

1. 失敗が起こり得ることが予想される文化を育みます。
2. 組織内のさまざまな業務領域について、明確な所有権と説明責任を定義します。
3. 所有権と説明責任を全員に伝え、分散型の意思決定を円滑に進めるうえで支援を提供する人物が誰であるかを各自が把握できるようにします。
4. 単一の方向と双方向の意思決定を定義し、より高いレベルのリーダーシップにエスカレーションする必要があるケースを各自が把握できるようにします。
5. 成果がリスクに直面した場合、すべての従業員がさまざまなレベルで対処する権限を付与されているという意識を組織全体で高めます。ガバナンス、アクセス許可レベル、ツール、機会に関するドキュメントをチームメンバーに提供して、効果的に対応するために必要なスキルを練習します。
6. さまざまな意思決定に対応するために必要なスキルを練習する機会をチームメンバーに提供します。意思決定レベルを定義したら、ゲームデーを開催して、各貢献者がプロセスを理解し、実際に実行できることを確認します。

   a. プロセスと手順のテストとトレーニングを実行できる安全な代替環境を用意します。

   b. 成果に既に定義されているレベルのリスクがある場合、チームメンバーにはアクションを起こす権限があるという意識を受け入れ、育みます。
   c. チームメンバーがサポートするワークロードとコンポーネントにアクセス許可とアクセス権を割り当てることで、アクションを実行するチームメンバーの権限を定義します。
7. チームが学んだこと (運用上の成功と失敗) を共有できるようにします。
8. チームが現状に問題意識を持てるようにして、改善点と組織に及ぼす影響を追跡して測定するメカニズムを提供します。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS01-BP06 メリットとリスクを管理しながらトレードオフを評価する
- OPS02-BP05 責任と所有権を特定するためのメカニズムが存在する

関連するドキュメント:

- AWS ブログ記事 | The agile enterprise
- AWS ブログ記事 | 成功を測定する：パラドックスと計画
- AWS ブログ記事 | Letting go : Enabling autonomy in teams
- 集中化か分散化か?

関連動画:

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

関連する例:

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

OPS03-BP03 エスカレーションが推奨されている

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

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

一般的なアンチパターン:

- エグゼクティブリーダーシップは、クラウドトランスフォーメーションプログラムのステータスミーティング中に、十分な質問をしておらず、問題や障害が発生している個所を発見することができません。ステータスとして良好な報告のみが提示されます。いかなる課題が提起されても、CEO はプログラムが失敗していると判断するため、良好な報告のみを発表したいと CIO が明言したためです。
- クラウド運用エンジニアが、新しいナレッジ管理システムがアプリケーションチームによって広く採用されていないことに気づきました。この企業では、この新しいナレッジ管理システムに数百万 USD を投資し、1 年かけて実装しました。しかし、チームは依然としてランブックをローカルで作成し、組織のクラウド共有で共有しているため、サポートされているワークロードに関連するナレッジを検索するのが困難となっています。このシステムを継続的に使用することで業務効率を向上できるため、クラウド運用エンジニアは、この件についてリーダーシップに報告しようとします。ナレッジ管理システムの実装を主導するディレクターにこの件について伝えたところ、この報告により投資が問題視されるという理由で、ディレクターはクラウド運用エンジニアを叱責します。
- コンピューティングリソースの強化を担当する情報セキュリティチームは、リソースを本番環境でリリースする前に、EC2 インスタンスが完全に保護されていることを確認するために必要となるスキャンをコンピューティングチームが実行する必要があるプロセスを導入することを決定しま

した。これにより、リソースのデプロイがこれまでより 1 週間遅延し、SLA に違反することになります。コンピューティングチームは、情報セキュリティ担当 VP の評判が悪くなることを懸念して、この問題についてクラウド担当の VP にエスカレーションすることを躊躇しています。

このようなベストプラクティスを確立することの利点:

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

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

実装手順

1. 組織のポリシー、基準、期待される内容を定義します。
    1. ポリシー、期待される内容、基準を幅広く採用し、理解されていることを確認します。
2. 基準が満たされない場合、早期かつ頻繁にエスカレーションを行うよう従業員を奨励し、トレーニングを行い、権限を付与します。
3. 早期かつ頻繁にエスカレーションすることがベストプラクティスであるという組織的な認識を確立します。エスカレーションした内容が事実ではないと判明する場合があること、およびエスカレーションしないことによってインシデントを阻止する機会を逃すよりもインシデントを阻止する機会が得られる方が好ましいということを受け入れます。
    a. エスカレーションの仕組みを構築します (アンドンシステムなど)。
    b. エスカレーションをいつどのように行うかを定義する手順を文書化します。
    c. アクションを実行または承認する人々を権限順に定義します。また、各ステークホルダーの連絡先の情報も追加します。
4. エスカレーションが行われた場合、リーダーシップが促進する一連のアクションによりリスクが軽減されたとチームメンバーが認めるまで、エスカレーションを続行する必要があります。
    a. エスカレーションには以下を含める必要があります。
        i. 状況およびリスクの性質の説明
        ii. 状況の重要度

iii. 影響が及ぶ人々や事項

iv. 影響の規模

v. 影響が発生した場合の緊急性

vi. 推奨される救済策と緩和計画

b. エスカレーションする従業員を保護します。対処を行わない意思決定者やステークホルダーを避けてにエスカレーションしたチームメンバーを報復行為から保護するポリシーを施行します。これが発生しているかどうかを特定し、適切に対応するメカニズムを備えます。

5. 組織が生み出すすべてのものに、継続的な改善のフィードバックループを取り入れる文化を奨励します。フィードバックループは責任者への軽微なエスカレーションとして機能し、エスカレーションが必要ない場合でも改善の機会を特定できます。継続的な改善の文化は、誰もがより積極的に行動することを押し進めます。
6. リーダーシップは、ポリシー、基準、メカニズム、報復されることのないオープンなエスカレーションと継続的なフィードバックループを奨励することを定期的に繰り返し強調すべきです。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- How do you foster a culture of continuous improvement and learning from Andon and escalation systems?
- The Andon Cord (IT Revolution)
- AWS DevOps ガイダンス | Establish clear escalation paths and encourage constructive disagreement

関連動画:

- Jeff Bezos on how to make decisions (& increase velocity)
- Toyota Product System: Stopping Production, a Button, and an Andon Electric Board
- Andon Cord in LEAN Manufacturing

関連する例:

- Working with escalation plans in Incident Manager

OPS03-BP04 タイムリーで明確、かつ実用的なコミュニケーション

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

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

一般的なアンチパターン:

- 組織には、すべてのワークロードを AWS に移行する 5 か年計画があります。クラウドのビジネスケースには、すべてのワークロードの 25% をモダナイズしてサーバーレステクノロジーを活用することが含まれています。CIO は、この戦略を直属部下に伝え、各リーダーが対面でのコミュニケーションなしにマネージャー、ディレクター、個別の貢献者にこのプレゼンテーションを伝達することを期待しています。CIO は現場に関与せず、この組織で新しい戦略が実行されることを期待しています。
- リーダーシップはフィードバックの仕組みを提供したり、利用したりすることはなく、期待のギャップが広がり、プロジェクトが行き詰まってしまいます。

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

このようなベストプラクティスを確立することの利点:

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

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

このベストプラクティスを実装するには、組織全体の関係者と協力して、コミュニケーション基準に関して合意を得る必要があります。この基準を、組織の誰もが確認できるようにします。大規模な IT 移行の場合、このようなベストプラクティスを考慮に入れない組織よりも、確立されたプランニングチームの方が、変更による従業員への影響をうまく管理できます。大規模な組織では、新しい戦略に対する個別の貢献者全員の強い賛同を得ることが重要となるため、変更管理がより困難になる可能性があります。このような移行プランニングチームを設置しない場合、効果的なコミュニケーションはリーダーシップが 100% 責任を負うことになります。移行プラニングチームを設立する際は、すべての組織のリーダーと協力し、あらゆるレベルにおける効果的なコミュニケーションを定義して、管理するチームメンバーを割り当てます。

お客様事例

AnyCompany Retail は、AWS エンタープライズサポートにサインアップして、クラウド運用は別のサードパーティープロバイダーに依存しています。同社は、業務活動の主要なコミュニケーション媒体としてチャットと ChatOps を利用しています。アラートやその他の情報は特定のチャネルに入力されます。誰かのアクションが必要な場合、期待される成果が明確に提示され、多くの場合、使用するランブックまたはプレイブックが指定されます。本稼働システムへの主要な変更については、変更カレンダーを使用してスケジュールしています。

実装手順

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

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

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

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

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

a. アラート向けのプレイブックとランブックの構築には、AWS Systems Manager ドキュメントを使用できます。

16.リスクや計画されたイベントの通知を明確かつ実用的な方法で提供し、適切な対応を可能にするのに十分な通知を提供するメカニズムが設けられています。計画されたイベントに先立ち、E メールリストまたはチャットチャネルを使用して、通知を送信します。

a. AWS Chatbot を使用すると、組織のメッセージングプラットフォーム内でアラートの送信やイベントの対応ができます。

17.計画されたイベントを知ることができる、アクセス可能な情報ソースを提供します。同じシステムから計画されたイベントを通知します。

a. AWS Systems Manager Change Calendar を使用すると、変更を実行できる変更ウィンドウを作成できます。これにより、チームメンバーは安全に変更を行うことができるタイミングを知ることができます。

18.脆弱性の通知とパッチ情報をモニターして、ワークロードコンポーネントに関連する予期できない潜在的なリスクの脆弱性を理解します。チームメンバーが対応できるように通知を送信します。

a. AWS セキュリティ速報を購読して、AWS における脆弱性に関する通知を受信できます。

19.多様な意見や視点を求める: すべてのメンバーからの貢献を求めます。取り上げられることの少ないグループにコミュニケーションの機会を与えます。ミーティングでは、役割と責任の割り当てを定期的に変更します。

a. ロールと責任を拡張する: チームメンバーに、通常引き受けることがないであろうロールを引き受ける機会を提供します。チームメンバーは、このようなロールから、また、通常はやり取りしない新しいチームメンバーとのやり取りから、経験や視点を得ることができます。チームメンバーはまた、自身が得た経験と視点を、やり取りをする新しいロールやチームメンバーに提供することができます。視野が広がるにつれて、新たなビジネスチャンスや新たな改善機会を見極めます。チーム内のメンバーに、その他のメンバーが通常実行している日常的なタスクを交替で担当してもらい、このようなタスクを実行する需要と影響を理解してもらいます。

b. 安全かつ安心できる環境を提供する: 組織内のチームメンバーの精神的および物理的な安全を確保するポリシーとコントロールを施行します。チームメンバーは、報復を恐れずにやり取りできる必要があります。チームメンバーが安全で安心できると、エンゲージメントと生産性が向上する可能性が高くなります。組織の多様化が進むと、お客様を含め、サポート対象への理解が深まります。チームのメンバーが安心して自由に意見を出し、話を聞いてもらえることを確信すると、貴重なインサイトを共有する可能性が高まります (マーケティングの機会、アクセシビリティのニーズ、未開拓の市場セグメント、環境内の認識されていないリスクなど)。

c. チームメンバーが全面的に参加できるように奨励する: 従業員がすべての業務関連のアクティビティに全面的に参加するために必要なリソースを提供します。日々の課題に直面するチームメンバーは、このような課題を回避するうえでのスキルを身に付けています。このように独自に開発したスキルは、組織に大きな利点をもたらします。必要な調整を行いながらチームメンバーをサポートすることで、メンバーの貢献から得られる利点が拡大します。

リソース

関連するベストプラクティス:

- OPS03-BP01 エグゼクティブスポンサーシップを提供する
- OPS07-BP03 ランブックを使用して手順を実行する
- OPS07-BP04 プレイブックを使用して問題を調査する

関連するドキュメント:

- AWS ブログ記事 | 高パフォーマンスなアジャイル組織はアカウンタビリティとエンパワーメントがカギです
- AWS Executive Insights | 複雑さではなくイノベーションの拡大を学ぶ | シングルスレッドリーダー
- AWS セキュリティ速報
- Open CVE
- AWS Support App in Slack to Manage Support Cases
- Manage AWS resources in your Slack channels with AWS Chatbot

関連する例:

- Well-Architected Labs: Inventory and Patch Management (Level 100) (Well-Architected ラボ: インベントリおよびパッチ管理 (レベル 100))

関連サービス:

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

OPS03-BP05 実験の推奨

実験は、新しいアイデアを製品や機能に変える触媒となります。実験は、学習を加速し、チームメンバーが関心と当事者意識を持ち続けることの一助となります。イノベーションを促進するために、チームメンバーは頻繁に実験することが奨励されます。結果が思わしくないものであっても、何をすべきでないかを知ることに価値があります。実験が成功したものの望ましくない結果が得られた場合、チームメンバーが罰せられることはありません。

期待される成果:

- イノベーションを育むために、組織では実験が奨励されます。
- 実験は学びの機会として使用されます。

一般的なアンチパターン:

- A/B テストを実行したいが、実験を実行する仕組みがない。テストを行うことができないまま UI の変更をデプロイした。その結果、カスタマーエクスペリエンスが低下した。
- 会社にはステージ環境と本稼働環境しかない。新機能や新製品を実験するサンドボックス環境がないため、本稼働環境で実験を行わなければならない。

このベストプラクティスを活用するメリット:

- 実験はイノベーションを促進します。
- 実験を通して、ユーザーからのフィードバックにより迅速に対応できます。
- 組織に学習の文化が築かれます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

実験は安全な方法で実行する必要があります。複数の環境を活用して、本稼働リソースを危険に晒すことなく、実験を行います。A/B テストや機能フラグを使用して、実験をテストします。チームメンバーがサンドボックス環境で実験を行えるようにします。

お客様事例

AnyCompany Retail では実験が奨励されています。チームメンバーは週の仕事の 20% を実験や新技術の学習に使用できます。サンドボックス環境があり、イノベーションを行うことができます。新機能には A/B テストが使用され、実際のユーザーのフィードバックを使用して機能を検証します。

実装手順

1. 組織全体でリーダーたちと協力して実験をサポートしてもらいます。チームメンバーは安全な方法で実験を行うことが奨励されます。
2. チームメンバーに、安全に実験できる環境を提供します。チームメンバーが本稼働環境に似た環境にアクセスできるようにする必要があります。
   a. 個別の AWS アカウントを使用して実験用のサンドボックス環境を作成できます。アカウントのプロビジョニングには、AWS Control Tower を使用できます。
3. 機能フラグや A/B テストを使用して安全に実験を行い、ユーザーからのフィードバックを収集します。
   a. AWS AppConfig 機能フラグを使用して、機能フラグを作成できます。
   b. 限定されたデプロイに対する A/B テストの実行には、Amazon CloudWatch Evidently を使用できます。
   c. AWS Lambda のバージョンを使用して、関数の新しいバージョンをデプロイし、ベータテストを実行できます。

実装計画に必要な工数レベル: 高。安全な方法で実験できる環境をチームメンバーに提供し実験を行うには、多額の投資が必要です。また、機能フラグを使用したり A/B テストをサポートしたりするために、アプリケーションコードの変更が必要になる場合があります。

リソース

関連するベストプラクティス:

- OPS11-BP02 インシデント後の分析を実行する - 実験に加えて、インシデントから学習することは、イノベーションの重要な推進要素です。

- OPS11-BP03 フィードバックループを実装する - フィードバックループは、実験の重要な部分です。

関連するドキュメント:

- An Inside Look at the Amazon Culture: Experimentation, Failure, and Customer Obsession (Amazon 文化の裏側: 実験、失敗、顧客第一主義)
- Best practices for creating and managing sandbox accounts in AWS(AWS でのサンドボックスアカウントの作成と管理におけるベストプラクティス)
- Create a Culture of Experimentation Enabled by the Cloud (クラウドで可能になる実験文化の構築)
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros (SulAmérica Seguros におけるクラウドでの実験とイノベーションの実現)
- Experiment More, Fail Less (実験が多いほど失敗は少なくなる)
- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU (複数のアカウントを使用した AWS 環境の組織化 - サンドボックス OU)
- Using AWS AppConfig Feature Flags (AWS AppConfig の機能フラグを使用する)

関連動画:

- AWS On Air ft.Amazon CloudWatch Evidently | AWS Events (AWS On Air: Amazon CloudWatch Evidently 特集 | AWS イベント)
- AWS On Air San Fran Summit 2022 ft.AWS AppConfig Feature Flags integration with Jira (AWS On Air San Fran Summit 2022: Jira を使用した AWS AppConfig 機能フラグの統合)
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R) (AWS re:Invent 2022 - デプロイはリリースではない: 機能フラグで起動をコントロールする (BOA305-R))
- Programmatically Create an AWS アカウント with AWS Control Tower(AWS Control Tower を使用して AWS アカウントをプログラムで作成する)
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations(AWS Organizations のベストプラクティスを使用するマルチアカウント AWS 環境の設定)

関連する例:

- AWS Innovation Sandbox

- End-to-end Personalization 101 for E-Commerce (E コマース向けのエンドツーエンドのパーソナライゼーション 101)

関連サービス:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 チームメンバーがスキルセットを維持、強化することができ、それが推奨されている

チームは、ワークロードに対応するに際して、新しいテクノロジーを採用し、需要と責任の変化をサポートするために、スキルセットを強化する必要があります。新しいテクノロジーにおけるスキルの発達は、多くの場合、チームメンバーの満足度の源となり、イノベーションをサポートします。チームメンバーが磨いているスキルを検証し、認識するような業界認証を取得し、維持できるように支援します。組織の知識とスキルを持ち、熟練したチームメンバーを失った場合は、クロストレーニングによって知識の伝達を促進し、重大な影響のリスクを緩和します。学習のために専用の時間を割り当てます。

AWS では、お客様のチームの教育のために、ガイダンス、例、詳細なウォークスルーを利用できるAWS の使用開始リソースセンター、AWS ブログ、AWS オンラインテックトーク、AWS イベントとウェビナー、AWS Well-Architected ラボを提供しています。

AWS Support、(AWSre: Post 、AWS Support Center)、AWS ドキュメントなどのリソースは、技術的な障害を取り除き、業務を改善するのに役立ちます。不明な点があれば、AWS Supportセンター経由で AWS Support までお問い合わせください。

また、AWS では The Amazon Builders' Library で AWS の運用を通じて得られたベストプラクティスとパターンを AWS ブログ、公式 AWS ポッドキャストで共有しています。

AWS トレーニングと認定では、自分のペースで学習できるデジタルコースによる無料トレーニングや、役割または業界別の学習プランが提供されています。また、インストラクターが実施するトレーニングに登録して、チームの AWS スキルの開発をさらにサポートすることもできます。

期待される成果: 組織は常にスキルギャップを評価し、体系的な予算と投資によってそのギャップを解消します。チームでは、業界をリードする認定資格の取得などのスキルアップのアクティビティを行うようにメンバーを奨励しています。チームは、ランチアンドラーン、イマージョンデー、ハッ

カソン、ゲームデーなど、専用の相互共有ナレッジプログラムを活用します。組織のナレッジシステムを常に最新の状態に維持し、新入社員のオンボーディングトレーニングなど、クロストレーニングチームメンバーとの関連性を維持します。

一般的なアンチパターン:

- 体系的なトレーニングプログラムと予算がなく、チームはテクノロジーの進化に遅れずに対応しようとする際に不安を感じるため、離職率が増大します。
- AWS への移行の取り組みの中で、組織のチーム間でスキルギャップやクラウド知識のレベルの違いがあることが判明します。スキルアップに向けた取り組みを行わなければ、チームはレガシーで非効率的なクラウド環境の管理に追われ、オペレーターの労力が増大することになります。このような燃え尽き症候群は従業員の不満を増大します。

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

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

1. 体系的なクラウド支援プログラムを利用する: AWS Skills Guild は、クラウドスキルに関する自信を高め、継続的な学習文化の刺激となるコンサルタティブトレーニングを提供しています。
2. 教育のためのリソースを提供する: 構造的に設けられた専用の時間、トレーニング資料へのアクセス、ラボリソース、カンファレンスや専門家組織への参加のサポートにより、教育者と同僚の両方から学習する機会を得ることができます。上級チームのメンバーを下級チームのメンバーのメンターとして交流できるようにしたり、上級チームのメンバーの業務を下級チームのメンバーがシャドーイングして、手法やスキルを学ぶ機会を提供したりします。より広い視点を持つために、仕事に直接関係しないコンテンツについて学習することを奨励します。
3. エキスパートの技術リソースの利用を奨励する: AWS re:Post などのリソースを活用して、厳選されたナレッジや活気にあふれたコミュニティにアクセスします。

4. 最新のナレッジリポジトリを構築して管理する: wiki やランブックなどのナレッジ共有プラットフォームを利用します。AWS re:Post Private を使用して、独自の再利用できるエキスパートのナレッジソースを作成し、コラボレーションの合理化、生産性向上、従業員のオンボーディングの加速を実現します。
5. チームの教育とチーム間のエンゲージメント: チームメンバーの継続的な教育のニーズに応じた計画を策定します。チームメンバーが他のチームに (一時的または永続的に) 参加し、組織全体に役立つスキルやベストプラクティスを共有する機会を提供します。
6. 業界認証の取得と維持をサポートする: チームメンバーが学んだことを検証し、その成果を認める業界認証を取得して維持するのをサポートします。

実装計画に必要な工数レベル: 高。

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS ホワイトペーパー | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS トレーニング と認定
- AWS Support
- AWS re:Post
- AWS のご利用開始のためのリソースセンター
- AWS ブログ
- AWS クラウド コンプライアンス
- AWS ドキュメント
- 公式 AWS ポッドキャスト
- AWS オンラインテックトーク
- AWS イベントとオンラインセミナー

- AWS Well-Architected ラボ
- Amazon Builders' Library

関連動画:

- AWS re:Invent 2023 | Reskilling at the speed of cloud: Turning employees into entrepreneurs
- WS re:Invent 2023 | Building a culture of curiosity through gamification

OPS03-BP07 チームに適正なリソースを提供する

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

期待される成果:

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

一般的なアンチパターン:

- 複数年にわたるクラウド移行計画が間近に迫っているのにもかかわらず、スタッフの AWS スキルは向上していません。このため、ワークロードのサポートがリスクにさらされ、従業員の士気が低下しています。
- IT 組織全体がアジャイルな働き方にシフトしています。ビジネス部門は、製品ポートフォリオに優先順位を付け、どの機能を最初に開発する必要があるかについてのメトリクスを設定していま

す。アジャイルプロセスでは、チームが作業計画にストーリーポイントを割り当てる必要はありません。この結果、以降の労力に必要となるキャパシティレベルを把握することも、そのタスクに適切なスキルが割り当てられているかを判断することができません。

- AWS パートナーにワークロードを移行してもらっていますが、パートナーが移行プロジェクトを完了した後のチームのサポートの移行計画が策定されていません。チームはワークロードを効率的かつ効果的にサポートするのに苦労しています。

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

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

1. スタッフの生産性などの関連する運用指標 (ワークロードをサポートするためのコストやインシデント時に費やしたオペレーター時間など) を使用して、チームの成功の基準を定義します。
2. リソースキャパシティプランニングと検査のメカニズムを定義し、適切なバランスの適格なキャパシティが必要な際に利用でき、長期的に調整可能かを検証します。
3. チームに影響を及ぼす業務上の課題 (責任範囲の増大、テクノロジーの変化、人員の喪失、対応する顧客の増加など) を把握するためのメカニズムを作成します (チームに毎月アンケートを送信するなど)。
4. このようなメカニズムを使用してチームとのエンゲージメントを維持し、従業員の生産性に関する課題の一因となる可能性のある傾向を検出します。チームが外部要因の影響を受けた場合、目標を再評価し、必要に応じてターゲットを調整します。チームの進行を妨げている障害を特定します。

5. 現在提供されているリソースが十分であるか、追加のリソースが必要かどうかを定期的に確認して、サポートチームの適切な調整を行います。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS03-BP06 チームメンバーがスキルセットを維持、強化することができ、それが推奨されている
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け
- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する
- OPS10-BP07 イベントへの対応を自動化する

関連するドキュメント:

- AWS クラウド 導入フレームワーク: People Perspective
- Becoming a Future-Ready Enterprise
- Prioritize your Employees' Skills to Drive Business Growth
- 高いパフォーマンスを発揮する組織 - Amazon ピザ 2 枚チーム
- How Cloud-Mature Enterprises Succeed

準備

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

運用上の優秀性に備えるには、以下を実行する必要があります。

トピック

- オブザーバビリティを実装する
- 運用のための設計
- デプロイのリスクを緩和する
- 運用準備状況と変更管理

オブザーバビリティを実装する

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

オブザーバビリティは単なるモニタリングにとどまらず、外部からの情報に基づいてシステム内部の仕組みを包括的に明らかにします。メトリクス、ログ、トレースを柱とするオブザーバビリティは、システムの動作とダイナミクスに関する深いインサイトを提供します。効果的なオブザーバビリティによって、チームはパターン、異常、傾向を見極め、潜在的な問題に積極的に対処し、最適なシステムの状態を維持することができます。

主要業績評価指標 (KPI) を特定することは、モニタリングアクティビティと事業目標の連携を確保するうえで非常に重要です。このような連携により、チームは真に重要なメトリクスを使用してデータ主導の意思決定を行い、システムパフォーマンスとビジネス成果の両方を最適化できます。

さらに、オブザーバビリティにより、企業は事後的ではなく積極的に対処できるようになります。チームはシステム内の因果関係を理解し、問題に対処するのみでなく、問題を予測して防止することができます。ワークロードが進化するにつれて、オブザーバビリティ戦略を再検討して改善し、戦略の関連性と効果を維持することが重要です。

ベストプラクティス

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

- OPS04-BP03 ユーザーエクスペリエンステレメトリーを実装する
- OPS04-BP04 依存関係のテレメトリーを実装する
- OPS04-BP05 分散トレースを実装する

OPS04-BP01 主要業績評価指標を特定する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロード KPI を効果的に定義する方法:

1. ビジネス成果から始めます。 メトリクスの詳細に取り掛かる前に、望ましいビジネス成果を理解しておきます。売上の増加、ユーザーエンゲージメントの向上、または応答時間の短縮などがあります。
2. 技術的なメトリクスをビジネス目標と関連付けます。 すべての技術メトリクスがビジネスの成果に直接影響するわけではありません。直接影響するようなメトリクスを特定します。ただし、多くの場合、ビジネス KPI を使用して問題を特定する方が簡単です。
3. Amazon CloudWatch の使用: CloudWatch を採用して、KPI となるメトリクスを定義してモニタリングします。
4. KPI を定期的に見直して更新します。 ワークロードとビジネスが進化するにつれ、KPI を適切に調整します。
5. 関係者の参画: KPI の定義とレビューには、技術チームと業務チームの両方の関与が必要です。

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS オブザーバビリティのベストプラクティス
- CloudWatch ユーザーガイド
- AWS オブザーバビリティ Skill Builder コース

関連動画:

- オブザーバビリティ戦略の策定

関連する例:

- One Observability ワークショップ

OPS04-BP02 アプリケーションテレメトリーを実装する

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

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

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- 情報に基づいた意思決定: テレメトリーとビジネス KPI から得られるインサイトを活用して、データ主導の意思決定を行うことができます。
- 運用効率の向上: データ主導型のリソース利用は、高いコスト効率をもたらします。
- ワークロードの安定性の向上: 問題をより迅速に検出して解決し、稼働時間を改善します。
- CI/CD プロセスの効率化: テレメトリーデータから得られるインサイトにより、プロセスの改善と信頼性の高いコードの配信が容易になります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

実装手順

1. 収集するデータの特定: ワークロードの状態、パフォーマンス、動作に関する重要なインサイトを提供する主要なメトリクス、ログ、トレースを確認します。
2. CloudWatch エージェントのデプロイ: CloudWatch エージェントを使用すると、ワークロードとその基盤となるインフラストラクチャからシステムとアプリケーションのメトリクスとログを取得できます。CloudWatch エージェントを使用すると、OpenTelemetry や X-Ray トレースを収集して、X-Ray に送信することもできます。
3. ログとメトリクスの異常検出の実装: CloudWatch Logs 異常検出と CloudWatch メトリクス異常検出を使用して、アプリケーションの操作の異常なアクティビティを自動的に特定します。これらのツールは、機械学習アルゴリズムを使用して異常を検出して警告するのでモニタリング機能が強化され、潜在的な混乱やセキュリティ脅威への対応時間が短縮できます。これらの機能を設定すると、アプリケーションヘルスとセキュリティをプロアクティブに管理できます。
4. 機密ログデータの保護: Amazon CloudWatch Logs データ保護を使用して、ログ内の機密情報をマスキングします。この機能は、アクセス前に機密データを自動的に検出してマスキングするので

プライバシーとコンプライアンスの維持に役立ちます。データマスキングを実装して、個人を特定できる情報 (PII) などの機密情報を安全に処理し保護します。

5. ビジネスの KPI の定義とモニタリング: ビジネスの成果に沿ってカスタムメトリクスを確立します。
6. AWS X-Ray を使用したアプリケーションのインストルメント化: CloudWatch エージェントのデプロイに加えて、トレースデータを送出するようにアプリケーションをインストルメント化することが重要です。このプロセスにより、ワークロードの動作とパフォーマンスをさらに詳細に把握できます。
7. アプリケーション全体でデータ収集を標準化: アプリケーション全体でデータ収集方法を標準化します。均一性を確立することで、データの関連付けと分析が容易になり、アプリケーションの動作を包括的に把握しやすくなります。
8. クロスアカウントオブザーバビリティの実装: Amazon CloudWatch クロスアカウントオブザーバビリティを使用して、複数の AWS アカウントにわたるモニタリング効率を強化します。この機能を使用すると、さまざまなアカウントのメトリクス、ログ、アラームを 1 つのビューに統合できます。これにより、組織の AWS 環境全体で特定された問題の管理が容易になり、対応時間を短縮できます。
9. データの分析と対処: データの収集と正規化が完了したら、Amazon CloudWatch を使用してメトリクスとログを分析し、AWS X-Ray を使用してトレース分析を行います。このような分析を行うことで、ワークロードの状態、パフォーマンス、動作に関する重要なインサイトを入手し、意思決定プロセスの指針とすることができます。

実装計画に必要な工数レベル: 高。

リソース

関連するベストプラクティス:

- OPS04-BP01 ワークロード KPI を定義する
- OPS04-BP03 ユーザーアクティビティテレメトリーを実装する
- OPS04-BP04 依存関係のテレメトリーを実装する
- OPS04-BP05 トランザクショントレーサビリティを実装する

関連するドキュメント:

- AWS オブザーバビリティのベストプラクティス

- CloudWatch ユーザーガイド
- AWS X-Ray デベロッパーガイド
- 運用の可視性を高めるために分散システムを装備する
- AWS Observability: Skill Builder コース
- Amazon CloudWatch の最新情報
- AWS X-Ray の最新情報

関連動画:

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

関連する例:

- One Observability Workshop
- AWS ソリューションライブラリ: Amazon CloudWatch を使用したアプリケーションモニタリング

OPS04-BP03 ユーザーエクスペリエンステレメトリーを実装する

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

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

一般的なアンチパターン:

- リアルユーザーモニタリング (RUM) をしないアプリケーション:
  - 問題検出の遅延: RUM を行わない場合、ユーザーからの苦情があるまでパフォーマンスのボトルネックや問題に気付かない可能性があります。このような事後対応型のアプローチは、お客様の不満につながる可能性があります。

    - ユーザーエクスペリエンスに関するインサイトの欠如: RUM を採用しない場合、実際のユーザーがアプリケーションをどのように操作したかを示す重要なデータが得られず、ユーザーエクスペリエンスの最適化の面で限界があります。
- 合成トランザクションを行わないアプリケーション:
    - 細部を見逃すケース: 合成トランザクションは、一般的なユーザーには頻繁に使用されてない可能性があるにしろ、特定の業務部門にとっては重要であるパスや機能のテストに役立ちます。合成トランザクションを行わないと、このようなパスが誤動作しても検出されない場合があります。
    - アプリケーション非使用時の問題の確認: 定期的に合成テストを実行して、実際のユーザーがアプリケーションを積極的に操作していない時間のシミュレーションを行うことで、システムが常に適正に機能することを確認できます。

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ユーザーアクティビティテレメトリーのために RUM と合成トランザクションを活用するうえで、AWS は次のとおりのサービスを提供しています Amazon CloudWatch RUM と Amazon CloudWatch Synthetics。メトリクス、ログ、トレースをユーザーアクティビティデータと組み合わせることで、アプリケーションの動作のステータスとユーザーエクスペリエンスの両方を包括的に把握できます。

実装手順

1. Amazon CloudWatch RUM のデプロイ: アプリケーションを CloudWatch RUM と統合して、実際のユーザーデータを収集、分析、提示します。
   a. CloudWatch RUM JavaScript ライブラリを使用して、 RUM をアプリケーションと統合します。
   b. ダッシュボードを設定して、実際のユーザーデータを可視化してモニタリングします。
2. CloudWatch Synthetics の設定: ユーザーのアプリケーション操作をシミュレートする Canary、つまりスクリプト化したルーチンを作成します。
   a. 重要なアプリケーションのワークフローとパスを定義します。
   b. CloudWatch スクリプトを使用して Canary を設計し、 パスへのユーザーの操作をシミュレートします。
   c. Canary を指定した間隔で実行するようにスケジュールを設定してモニタリングを実行し、着実にパフォーマンスチェックを実行します。
3. データの分析と対処: RUM と合成トランザクションからのデータを活用してインサイトを取得し、異常が検出された場合は是正措置を講じます。CloudWatch ダッシュボードとアラームを使用して常に情報を入手します。

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Amazon CloudWatch RUM ガイド
- Amazon CloudWatch Synthetics ガイド

関連動画:

- Amazon CloudWatch RUM を使用したエンドユーザーインサイトを通してアプリケーションを最適化する
- AWS on Air ft.Real-User Monitoring for Amazon CloudWatch

関連する例:

- One Observability ワークショップ
- Amazon CloudWatch RUM ウェブクライアントの Git リポジトリ
- Amazon CloudWatch Synthetics を使用してページのロード時間を測定する

OPS04-BP04 依存関係のテレメトリーを実装する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- ワークロードの信頼性の向上: 外部依存を常に利用可能にして最適なパフォーマンスを発揮できるようにすることで実現できます。

- 問題の検出と解決の迅速化: ワークロードに影響が及ぶ前に、依存関係に関連する問題を事前に特定して対処できます。
- 包括的なビュー: ワークロードのヘルスに影響を及ぼす内部コンポーネントと外部コンポーネントの両方を全体的に把握できます。
- ワークロードのスケーラビリティ強化: 外部依存のスケーラビリティの限界とパフォーマンス特性を把握することにより実現できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

実装手順

依存関係のテレメトリーを効果的に実装する方法:

1. 外部依存関係の特定: ステークホルダーと協力して、ワークロードが依存している外部依存関係を特定します。外部依存関係には、外部データベース、サードパーティ API、その他の環境へのネットワーク接続ルート、DNS サービスなどのサービスが含まれます。効果的な依存関係のテレメトリーを得る第一歩は、依存関係を包括的に把握することです。
2. モニタリング戦略の策定: 外部依存を明確に把握した後、それに応じたモニタリング戦略を構築します。これには、各依存関係の重要度、予想される動作、関連するサービスレベルアグリーメントまたは目標 (SLA または SLT) を把握することなどがあります。ステータスの変化やパフォーマンスの逸脱を通知する積極的なアラートを設定します。
3. ネットワークモニタリングの使用: インターネットモニターおよびネットワークモニターを使用して、グローバルインターネットとネットワークの状態に関する包括的なインサイトを取得します。これらのツールは、外部の依存関係に影響を与える機能停止、中断、またはパフォーマンスの低下を把握し、それに対応するために役立ちます。
4. AWS Health Dashboard を使用した情報の取得: AWS でサービスに影響を及ぼす可能性のあるイベントが発生した場合にアラートと修正ガイダンスを提供します。

   a. Amazon EventBridge ルールを使用して AWS Health イベントをモニタリングしたり、プログラムで AWS Health API と統合して AWS Health イベントを受信したときのアクションを自動化したりできます。これらのアクションには、計画されたすべてのライフサイクルイベントメッセージをチャットインターフェイスに送信するなどの一般的なアクションや、IT サービス管理ツールでのワークフローの開始などの特定のアクションがあります。

   b. AWS Organizations を使用する場合は、アカウント全体で AWS Health イベントを集約します。

5. AWS X-Ray を使用したアプリケーションのインストルメント化: AWS X-Ray を使用すると、アプリケーションとアプリケーションの基盤となる依存関係のパフォーマンスに関するインサイトが得られます。リクエストを開始から終了までトレースすることにより、アプリケーションが依存している外部サービスや外部コンポーネントのボトルネックや障害を特定できます。
6. Amazon DevOps Guru の使用: この機械学習ベースのサービスは、運用上の問題を特定し、重大な問題が発生する可能性のあるタイミングを予測して、実行すべき具体的な対応措置を推奨します。依存関係のインサイトを得て、その関係性が運用上の問題の原因ではないことを突き止めるために、非常に有益です。
7. 継続的なモニタリング: 外部依存に関するメトリクスとログを継続的にモニタリングします。予期しない動作やパフォーマンスの低下についてのアラートを設定します。
8. 変更後の検証: 外部依存のいずれかが更新されたり変更されたりした場合は、そのパフォーマンスを検証して、アプリケーションの要件との整合性を確認します。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS04-BP01 ワークロード KPI を定義する
- OPS04-BP02 アプリケーションテレメトリーを実装する
- OPS04-BP03 ユーザーアクティビティテレメトリーを実装する
- OPS04-BP05 トランザクショントレーサビリティを実装する
- OP08-BP04 実践的なアラートを作成する

関連するドキュメント:

- Amazon Personal AWS Health Dashboard ユーザーガイド

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

関連動画:

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

関連する例:

- Gaining operational insights with AIOps using Amazon DevOps Guru
- AWS Health Aware
- タグベースフィルタリングを利用した大規模な AWS Health モニタリングおよびアラートの管理

OPS04-BP05 分散トレースを実装する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

計測が必要となるすべてのワークロードの要素を特定することから始めます。すべてのコンポーネントを把握したら、AWS X-Ray や OpenTelemetry などのツールを活用してトレースデータを収集し、X-Ray や Amazon CloudWatch ServiceLens Map などのツールを使用して分析を行います。デベロッパーとのレビューを定期的に実施し、Amazon DevOps Guru、X-Ray Analytics、X-Ray Insights などのツールをサポートとして使用した議論により、より詳細な検出を行います。トレースデータからアラートを設定して、ワークロードのモニタリング計画で定義されている結果に対してリスクが検出された場合に通知します。

実装手順

分散トレースを効果的に実装する方法:

1. AWS X-Ray の採用: X-Ray をアプリケーションに組み込むと、アプリケーションの動作に関するインサイトを取得したり、パフォーマンスを把握して、ボトルネックを特定したりできます。X-Ray Insights を自動トレース分析に活用します。
2. サービスの計測: AWS Lambda 関数から EC2 インスタンスまで、すべてのサービスがトレースデータを送信することを確認します。計測するサービスが多いほど、エンドツーエンドのビューが明確になります。
3. CloudWatch リアルユーザーモニタリング と 合成モニタリングの統合: リアルユーザーモニタリング (RUM) と X-Ray を使用した合成モニタリングを統合します。これにより、実際のユーザーエクスペリエンスをキャプチャしてユーザーの操作をシミュレートし、潜在的な問題を特定できます。
4. CloudWatch エージェントの使用: エージェントは X-Ray または OpenTelemetry のいずれからもトレースを送信できるため、インサイトがより詳細に拡張されます。
5. Amazon DevOps Guru の使用: DevOps Guru は X-Ray、CloudWatch、AWS Config、AWS CloudTrail からのデータを使用して、実践的なレコメンデーションを提供します。

6. トレースの分析: トレースデータを定期的に確認して、アプリケーションのパフォーマンスに影響を及ぼす可能性のあるパターン、異常、またはボトルネックを特定します。
7. アラートの設定: 異常なパターンや長時間のレイテンシーに対して CloudWatch でアラームを設定すると、 積極的に問題に対処できます。
8. 継続的な改善: サービスが追加または変更されたら、関連するすべてのデータポイントが取得できるように、トレース戦略を再検討します。

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS X-Ray デベロッパーガイド
- Amazon CloudWatch エージェントユーザーガイド
- Amazon DevOps Guru ユーザーガイド

関連動画:

- AWS X-Ray Insights を使用する
- AWS on Air ft.オブザーバビリティ: Amazon CloudWatch と AWS X-Ray

関連する例:

- AWS X-Ray を使用したアプリケーションのインストルメント化

運用のための設計

本番環境への変更プロセスを改善し、リファクタリング、品質についての迅速なフィードバック、バグ修正に役立つアプローチを採用します。これらにより、本番環境に採用される有益な変更を加速させ、デプロイされた問題を制限できます。またデプロイアクティビティを通じて導入された問題を迅速に特定し、修復できます。

AWS では、ワークロード全体 (アプリケーション、インフラストラクチャ、ポリシー、ガバナンス、運用) をコードとして表示できます。すべてコードで定義し、更新できます。つまり、スタックのすべての要素にアプリケーションコードに使用するのと同じエンジニアリング規律を適用できます。

ベストプラクティス

- OPS05-BP01 バージョン管理を使用する
- OPS05-BP02 変更をテストし、検証する
- OPS05-BP03 構成管理システムを使用する
- OPS05-BP04 構築およびデプロイ管理システムを使用する
- OPS05-BP05 パッチ管理を実行する
- OPS05-BP06 設計標準を共有する
- OPS05-BP07 コード品質の向上のためにプラクティスを実装する
- OPS05-BP08 複数の環境を使用する
- OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う
- OPS05-BP10 統合とデプロイを完全自動化する

OPS05-BP01 バージョン管理を使用する

変更とリリースの追跡を有効にするにはバージョン管理を使用します。

AWS の多くのサービスは、バージョン管理機能を備えています。 AWS CodeCommit などのリビジョンまたはソース管理システムを使用して、 インフラストラクチャのバージョン管理された AWS CloudFormation テンプレートなどのコードやその他のアーティファクトを管理します。

期待される成果: コードに関してチームが協力し合います。コードをマージすると、コードの一貫性が維持され、変更点が失われることはありません。エラーは、適正なバージョン管理によって簡単に元に戻すことができます。

一般的なアンチパターン:

- コードを開発し、ワークステーションに保存したのに、そのワークステーションで回復不可能なストレージ障害が発生し、コードが失われる。
- 既存のコードを変更で上書きした後、アプリケーションを再起動すると、操作できなくなる。変更を元に戻すことができない。
- レポートファイルへの書き込みがロックされていて、別のユーザーが編集する必要があるとき、編集をしようとするユーザーは、ほかのユーザーに作業を停止するように求める。
- 研究チームは、今後の業務を形作る詳細な分析に取り組んでいます。誰かが誤って最終レポートを買い物リストで上書きして保存してしまう。変更を元に戻すことができず、レポートを再作成する必要がある。

このベストプラクティスを活用するメリット: バージョン管理機能を使用すると、既知の良好な状態や以前のバージョンに簡単に戻すことができ、アセットが失われるリスクを低減できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

バージョン管理されたレポジトリでアセットを維持します。そうすることで、変更の追跡、新しいバージョンのデプロイ、既存バージョンへの変更の検出、以前のバージョンの回復 (障害が発生する場合に、その前の良好な状態に戻すなど) をサポートします。構成管理システムのバージョン管理機能を手順に統合します。

リソース

関連するベストプラクティス:

- OPS05-BP04 構築およびデプロイ管理システムを使用する

関連するドキュメント:

- AWS CodeCommit とは

関連動画:

- AWS CodeCommit の紹介

OPS05-BP02 変更をテストし、検証する

デプロイされた変更はすべてテストし、本稼働でのエラーを回避する必要があります。このベストプラクティスは、バージョンコントロールからアーティファクトビルドへの変更をテストすることに重点を置いています。テストには、アプリケーションコードの変更に加えて、インフラストラクチャ、設定、セキュリティコントロール、運用手順も含める必要があります。テストは、単体テストからソフトウェアコンポーネント分析 (SCA) まで、さまざまな形態があります。ソフトウェアの統合および配信プロセスでテストをさらに早めると、アーティファクト品質の確実性が増します。

組織はすべてのソフトウェアアーティファクトにおいてテスト基準を作成する必要があります。テストを自動化すると、手間を軽減し、手動テストによるエラーを回避できます。手動テストが必要な場合もあります。デベロッパーは自動テストの結果を確認して、ソフトウェアの品質を向上させるフィードバックループを構築する必要があります。

期待される成果: ソフトウェアの変更は、配信前にすべてテストされます。デベロッパーはテスト結果と検証にアクセスできます。組織には、すべてのソフトウェア変更に適用されるテスト基準があります。

一般的なアンチパターン:

- ソフトウェアの新しい変更を、テストせずにデプロイする。本稼働で実行に失敗し、その結果サービスが停止する。
- 新しいセキュリティグループが、本番前環境でのテストをせずに AWS CloudFormation にデプロイされる。そのセキュリティグループによって、ユーザーがアプリにアクセスできなくなる。
- メソッドが変更されても単体テストを行わない。本稼働へのデプロイ時にソフトウェアが失敗する。

このベストプラクティスを活用するメリット: ソフトウェアのデプロイでの変更失敗率が軽減されます。ソフトウェアの品質が向上します。デベロッパーのコードの実行可能性に関する意識が向上します。確信を持ってセキュリティポリシーをロールアウトし、組織のコンプライアンスをサポートできます。自動スケーリングポリシーの更新などインフラストラクチャの変更を事前にテストし、トラフィックのニーズを満たすことができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

継続的統合の実践の一部として、アプリケーションコードからインフラストラクチャまで、すべての変更に対してテストを行います。テスト結果は、デベロッパーが迅速にフィードバックを得られるように公開します。組織で、すべてのソフトウェア変更に適用されるテスト基準を施行します。

Amazon Q Developer の生成 AI の機能を活用して、開発者の生産性とコード品質を向上させます。Amazon Q Developer は、コード提案の生成 (大規模言語モデルに基づく)、単体テストの作成 (境界条件を含む)、およびセキュリティ脆弱性の検出と修復によってコードセキュリティを強化します。

お客様事例

AnyCompany Retail は、継続的な統合パイプラインの一部として、すべてのソフトウェアアーティファクトに対して複数種類のテストを実行しています。テスト駆動開発を実践しているため、すべてのソフトウェアに単体テストがあります。アーティファクトがビルドされると、エンドツーエンドのテストが実行されます。1 ラウンド目のテストが完了すると、静的アプリケーションセキュリティスキャンを実行し、既知の脆弱性を探します。デベロッパーは、各テストに合格するたびにメッセージを受け取ります。すべてのテストが完了すると、ソフトウェアアーティファクトはアーティファクトリポジトリに保存されます。

実装手順

1. 組織の関係者と協力して、ソフトウェアアーティファクトのテスト基準を作成します。すべてのアーティファクトが合格しなければならない基準のテストとは何でしょうか。 テスト範囲に含める必要があるコンプライアンスやガバナンスの要件はありますか。 コード品質テストを実施する必要がありますか。 テストが完了した際に通知が必要なのは誰ですか?
    1. AWS デプロイパイプラインリファレンスアーキテクチャには、統合パイプラインの一部としてソフトウェアアーティファクトに対して実行できる、テストの種類の信頼できるリストが含まれています。
2. ソフトウェアテスト基準に基づいて必要なテストを行い、アプリケーションを計測します。テストの各セットは 10 分以内に完了する必要があります。テストは統合パイプラインの一部として実行する必要があります。
    a. Amazon Q Developer を使用します。これは、単体テストケース (境界条件を含む) の作成、コードとコメントを使用した関数の生成、一般的なアルゴリズムの実装に役立つ生成 AI ツールです。
    b. Amazon CodeGuru Reviewer を使用して、アプリケーションコードをテストして欠陥を検出します。

c. AWS CodeBuild を使用して、ソフトウェアアーティファクトに対しテストを実施できます。

d. AWS CodePipeline は、ソフトウェアテストをパイプラインに組み込むことができます。

リソース

関連するベストプラクティス:

- OPS05-BP01 バージョン管理を使用する
- OPS05-BP06 設計標準を共有する
- OPS05-BP07 コード品質の向上のためにプラクティスを実装する
- OPS05-BP10 統合とデプロイを完全自動化する

関連するドキュメント:

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer センター
- Amazon CodeWhisperer でアプリケーションをより速く構築する 10 の方法
- Amazon CodeWhisperer でコードカバレッジの先を見る
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Automated AWS CloudFormation Testing Pipeline with TaskCat and CodePipeline
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools
- Getting started with testing serverless applications
- My CI/CD pipeline is my release captain
- AWS での継続的インテグレーションと継続的デリバリーの実践 (ホワイトペーパー)

関連動画:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development
- Testing Your Infrastructure as Code with AWS CDK

関連リソース:

- Amazon CodeWhisperer の生成 AI を使用したアプリケーションの構築
- Amazon CodeWhisperer Workshop
- AWS デプロイパイプラインリファレンスアーキテクチャ - アプリケーション
- AWS Kubernetes DevSecOps パイプライン
- Policy as Code ワークショップ – テスト駆動開発
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Use Serverspec for test-driven development of infrastructure code

関連サービス:

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

OPS05-BP03 構成管理システムを使用する

設定を変更し、変更を追跡記録するには、構成管理システムを使用します。これらのシステムは、手動プロセスによって発生するエラーと、変更を導入する労力を減らします。

静的な構成管理では、ライフタイムを通じて一貫性を維持することが期待されるリソースの初期化時に値を設定します。このケースの例として、インスタンス上のアプリケーションサーバーまたはウェブサーバー用を設定する場合や、 AWS Management Console 内または AWS CLI を介して AWS サービスの設定を定義する場合が挙げられます。

動的な構成管理では、ライフタイムを通じて変化する、または変化することが予測されるリソースの初期化時に値を設定します。例えば、構成変更を介してコードの機能を有効にするように機能トグルを設定したり、インシデント発生時にログの詳細レベルを変更してより多くのデータを取得し、インシデント終了時に詳細レベルを元に戻して不要なログや負荷を減らしたりすることができます。

AWS では、 AWS Config を使用して アカウント間とリージョン全体にわたる AWS リソースの設定を 継続的にモニタリングできます。これは、設定履歴を追跡し、設定変更がほかのリソースに与える影響を理解して、 AWS Config ルール と AWS Config コンフォーマンスパックを使用した期待される、または望まれる設定との比較監査を行えます。

Amazon EC2 インスタンス、AWS Lambda、コンテナ、モバイルアプリケーション、または IoT デバイスで実行されているアプリケーションで動的設定が使用されている場合、 AWS AppConfig を使用して環境全体にわたり、設定、検証、デプロイ、モニタリングできます。

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

期待される成果: 継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインの一部として設定、検証、デプロイを行います。モニタリングして、設定が正しいことを確認します。これにより、エンドユーザーや顧客への影響を最小限に抑えることができます。

一般的なアンチパターン:

- あなたがフリート全体でウェブサーバー設定を手動で更新したところ、更新エラーのために多数のサーバーが応答しなくなりました。
- あなたは、何時間もかけて、アプリケーションサーバーフリートを手動で更新します。変更中の設定の不整合が、予期しない動作を引き起こします。
- 誰かがセキュリティグループを更新したため、ウェブサーバーにアクセスできなくなりました。変更内容を把握しなければ、問題の調査にかなりの時間を費やすことになり、復旧までより長くの時間を要することになります。
- 検証をせずに CI/CD を使用して本番稼働前の設定を本番環境にプッシュします。ユーザーと顧客に正確でないデータやサービスを提供してしまいます。

このベストプラクティスを活用するメリット: 構成管理システムを採用することで、変更やその追跡の労力のレベルと、手動の手順に起因するエラーの頻度を軽減できます。構成管理システムを使用すると、ガバナンス、コンプライアンス、規制要件に関して保証が得られます。

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

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

実装手順

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

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS Control Tower
- AWS Landing Zone Accelerator

- AWS Config
- AWS Config とは?
- AWS AppConfig
- AWS CloudFormation とは
- AWS デベロッパーツール

関連動画:

- AWS re:Invent 2022 - AWS ワークロードのプロアクティブなガバナンスとコンプライアンス
- AWS re:Invent 2020: AWS Config を使用してコードとしてのコンプライアンスを実現する
- AWS AppConfig を使用してアプリケーションの設定を管理してデプロイする

OPS05-BP04 構築およびデプロイ管理システムを使用する

構築およびデプロイ管理システムを使用します。これらのシステムは、手動プロセスによって発生するエラーと、変更を導入する労力を減らします。

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

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

一般的なアンチパターン:

- 開発システムでコードをコンパイルした後、あなたは、実行可能ファイルを本稼働システムにコピーし、起動に失敗する。ローカルログファイルは、依存関係がないために失敗したことを示す。
- 開発環境でアプリケーションの新機能の構築を正常に完了し、品質保証 (QA) にコードを提供しても、静的アセットが欠如していたために、QA に合格しない。
- 金曜日に、多くの労力をかけて、開発環境でアプリケーションを手動で構築でき、これには、新しくコード化された機能も含まれるけれど、月曜日に、アプリケーションを正常に構築することを可能にするステップを繰り返すことができない。
- そこで、新しいリリース用に作成したテストを実行する。その後、あなたは、翌週いっぱいをかけて、テスト環境をセットアップし、すべての既存の統合テストを実行してから、パフォーマンステ

ストを実行する。新しいコードには許容できないパフォーマンスへの影響があり、再開発してから再テストする必要がある。

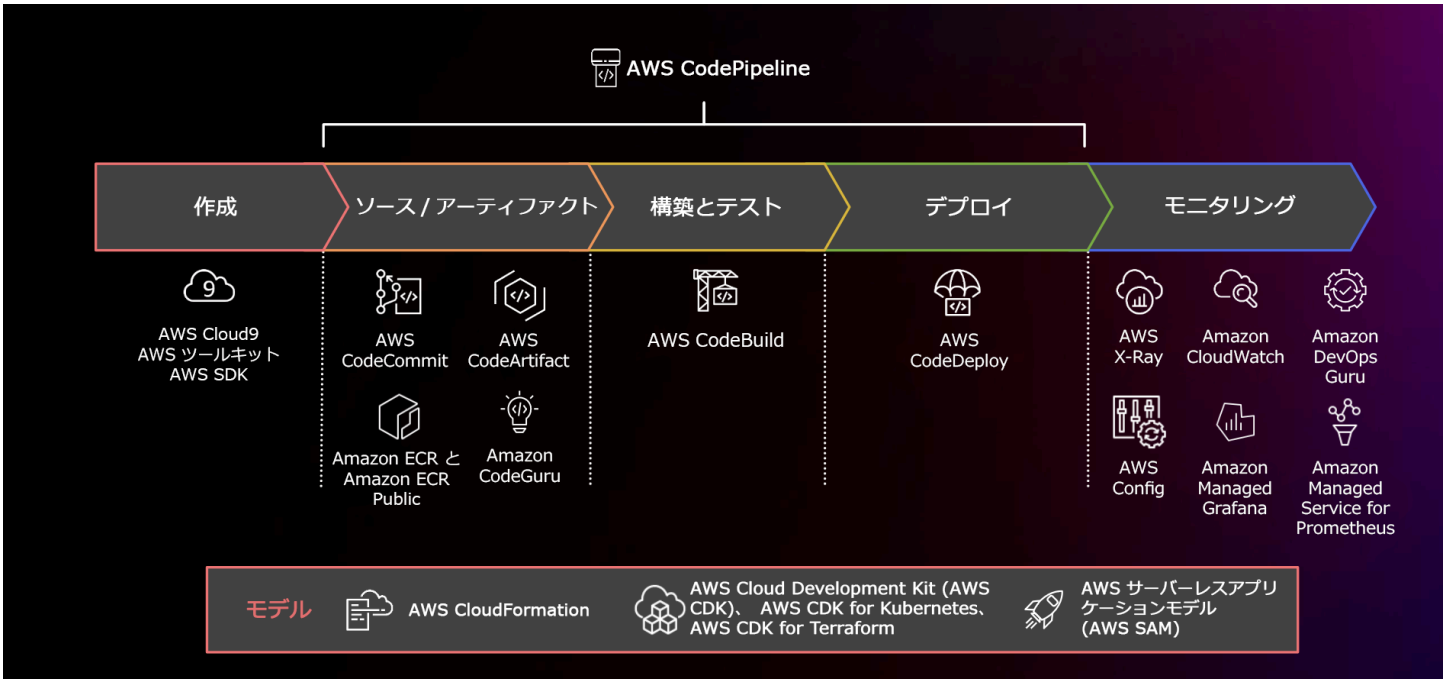
このベストプラクティスを活用するメリット: ビルドとデプロイのアクティビティを管理するメカニズムを提供することで、反復的なタスクを実行するための労力の程度を減らし、チームメンバーは高価値のクリエイティブなタスクに専念し、手動の手順によるエラーの発生を抑制できます。

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

構築およびデプロイ管理システムを使用すると、変更の追跡と実装、手動プロセスが原因で発生するエラーの削減、安全なデプロイに必要な労力の軽減につながります。コードのチェックインから、ビルド、テスト、デプロイ、検証を通じて統合とデプロイのパイプラインを完全自動化します。これにより、リードタイム短縮、コスト低減、変更頻度の増加、労力の軽減、コラボレーションの強化につながります。

実装手順



AWS CodePipeline と関連サービスを使用した CI/CD を説明する図

1. AWS CodeCommit を使用して、アセット (ドキュメント、ソースコード、バイナリファイルなど) のバージョン管理、保存、管理を行います。

2. CodeBuild を使用して、ソースコードのコンパイル、単体テストの実行、デプロイ準備が整ったアーティファクトの作成を行います。
3. CodeDeploy を Amazon EC2 インスタンス、オンプレミスインスタンス、サーバレス AWS Lambda 関数、または Amazon ECS へのアプリケーションのデプロイを自動化するデプロイメントサービスとして使用します。
4. 環境をモニタリングします。

リソース

関連するベストプラクティス:

- OPS06-BP04 テストとロールバックを自動化する

関連するドキュメント:

- AWS デベロッパーツール
- AWS CodeCommit とは
- AWS CodeBuild とは
- AWS CodeBuild
- AWS CodeDeploy とは

関連動画:

- AWSre\: Invent 2022 - AWS の DevOps 向け AWS Well-Architected ベストプラクティス

OPS05-BP05 パッチ管理を実行する

パッチ管理を実行し、問題を解決して、ガバナンスに準拠するようにします。パッチ管理の自動化により、手動プロセスによって発生するエラーを低減し、スケールして、パッチに関連する労力を減らすことができます。

パッチと脆弱性の管理は、利点とリスク管理のアクティビティの一環です。不変のインフラストラクチャを使用し、検証済みの正常な状態でワークロードをデプロイすることが推奨されます。これが現実的でない場合のオプションには、パッチの適用があります。

Amazon EC2 Image Builder は、マシンイメージ更新向けのパイプラインを提供します。パッチ管理の一環として、 AMI イメージパイプラインを使用する Amazon マシンイメージ (AMI) または Docker イメージパイプラインを備えた コンテナイメージを を検討します。一方、AWS Lambda は、脆弱性を排除するための カスタムランタイムと追加ライブラリのパターンを 提供します。

Linux または Windows Server イメージ用の AMI イメージパイプラインを使用する の更新管理については、 Amazon EC2 Image Builderを使用します。既存のパイプラインで Amazon Elastic Container Registry (Amazon ECR) を使用して、Amazon ECS と Amazon EKS イメージを管理できます。Lambda には、 バージョン管理機能が提供されていますを使用します。

パッチの本番環境のシステムへの適用は、まず安全な環境でテストした後とする必要があります。パッチは運用上またはビジネス上の成果に対応している場合にのみ適用してください。AWS では、AWS Systems Manager Patch Manager を使用して、 管理対象システムへのパッチ適用プロセスを自動化し、 Systems Manager Maintenance Windows を使用してアクティビティをを使用します。

期待される成果: AMI とコンテナイメージにパッチが適用されて最新の状態であり、起動の準備が整っています。デプロイされたすべてのイメージのステータスを追跡し、パッチのコンプライアンスを把握できます。現在のステータスを報告でき、コンプライアンスのニーズを満たすプロセスが施行されています。

一般的なアンチパターン:

- あなたには、すべての新しいセキュリティパッチを 2 時間以内に適用するために権限が付与されました。その結果、アプリケーションにパッチとの互換性がないため、複数の機能停止が発生しました。
- パッチが適用されていないライブラリは、不明な関係者がライブラリ内の脆弱性を使用してワークロードにアクセスするため、意図しない結果をもたらします。
- あなたは、デベロッパーに通知することなく、自動的にデベロッパー環境にパッチを適用します。あなたには、デベロッパーから、環境が想定どおりに動作しなくなったという苦情が複数寄せられます。
- 永続的なインスタンスの商用オフザシェルフのセルフソフトウェアにパッチを適用していない。ソフトウェアに問題があり、ベンダーに連絡すると、ベンダーから、バージョンがサポートされておらず、サポートを受けるためには、特定のレベルにパッチを適用する必要があることが伝えられます。
- 使用した暗号化ソフトウェアの最近リリースされたパッチにより、パフォーマンスが大幅に向上しますが、パッチが適用されていないシステムには、パッチを適用しない結果として、パフォーマンスの問題が残存している。

- 緊急に修正が必要なゼロデイ脆弱性についての通知を受けて、すべての環境に手動でパッチを適用する必要がある。

このベストプラクティスを活用するメリット: パッチ適用の基準や環境全体にわたる配布方法など、パッチ管理プロセスを確立することで、パッチレベルのスケールとレポート作成が実現します。これにより、セキュリティパッチの適用が保証され、実施されている既知の修正のステータスを明確に把握できます。これにより、必要な機能の導入、問題の迅速な解決、継続的なガバナンスへの遵守が実現します。パッチ管理システムと自動化を実装して、パッチをデプロイする労力を軽減し、手動プロセスに起因するエラーの発生を抑制します。

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

問題の修正、希望する機能や能力の取得、ガバナンスポリシーやベンダーのサポート要件への準拠継続を行うためにはシステムをパッチします。変更不可能なシステムでは、必要な成果を達成するために適切なパッチを使用してデプロイします。パッチ管理メカニズムを自動化することで、パッチ適用の経過時間、手動プロセスが原因で発生するエラー、パッチに関する労力を低減できます。

実装手順

Amazon EC2 Image Builder の場合:

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

6. レシピのハイジーンを定期的に管理します。

Systems Manager Patch Manager の場合:

1. パッチベースラインを作成します。
2. パス設定操作方法を選択します。
3. コンプライアンスレポートとスキャンを有効にします。

リソース

関連するベストプラクティス:

- OPS06-BP04 テストとロールバックを自動化する

関連するドキュメント:

- Amazon EC2 Image Builder とは
- Amazon EC2 Image Builder を使用してイメージパイプラインを作成する
- コンテナイメージパイプラインを作成する
- AWS Systems Manager Patch Manager
- Patch Manager の操作
- パッチコンプライアンスレポートの使用
- AWS デベロッパーツール

関連動画:

- AWS のサーバーレスアプリケーション用の CI/CD
- Ops を考慮に入れて設計する

関連する例:

- Well-Architected ラボ - インベントリおよびパッチ管理
- AWS Systems Manager Patch Manager チュートリアル

OPS05-BP06 設計標準を共有する

チーム全体でベストプラクティスを共有し、デプロイ作業における利点の認識を高め、それを最大限にします。標準を文書化し、アーキテクチャの進化に応じて最新の内容となるよう維持します。組織内で共有された標準が適用されている場合、標準の追加、変更、例外を申請するメカニズムを持つことは重要です。このオプションがなければ、標準はイノベーションの障壁になります。

期待される成果: 設計標準が組織のチーム間で共有されています。設計標準が文書化され、ベストプラクティスの進化に応じて内容が更新されます。

一般的なアンチパターン:

- 2 つの開発チームがそれぞれ独自のユーザー認証サービスを作成しました。ユーザーは、アクセスするシステムの各部分について、個別の一連の認証情報を維持する必要があります。
- 両チームは独自のインフラストラクチャを管理しています。新しいコンプライアンス要件により、インフラストラクチャの変更が必要になり、両チームは別々の方法で新たな要件を実装します。

このベストプラクティスを活用するメリット: 共有される標準を利用すると、ベストプラクティスの採用、開発作業の利点の最大化につながります。設計標準を文書化して更新することにより、組織はベストプラクティス、セキュリティ、コンプライアンス要件を最新の内容に維持できます。

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

既存のベストプラクティス、設計標準、チェックリスト、業務手順、ガイダンス、ガバナンス要件をチーム間で共有します。改善とイノベーションを支援するために、設計標準の変更、追加、例外を申請する手順を設けます。公開されたコンテンツについてチームに周知させます。新しいベストプラクティスが台頭すると、それに応じて設計標準を最新の内容に維持するメカニズムを導入します。

お客様事例

AnyCompany Retail には、ソフトウェアアーキテクチャのパターンを作成する機能横断的なアーキテクチャチームがあります。このチームでは、コンプライアンスとガバナンスを組み込んだアーキテクチャを構築しています。この共有標準を採用するチームは、コンプライアンスとガバナンスが組み込み済みであるという利点が得られ、この設計標準を基盤に迅速に構築できます。アーキテクチャチームは四半期ごとのミーティングでアーキテクチャのパターンを検討し、必要に応じて更新します。

実装手順

1. 設計標準の開発と更新の責任を担う部門横断的なチームを特定します。このチームは、組織全体にわたる関係者と協力して、設計標準、業務手順、チェックリスト、ガイダンス、ガバナンス要件を開発し、設計標準を文書化して、組織内で共有します。
   a. AWS Service Catalog を使用すると、IaC (Infrastructure as Code) を使用して設計標準を提示するポートフォリオを作成でき、ポートフォリオをアカウント間で共有できます。
2. 新しいベストプラクティスが特定されると、それに応じて設計標準を最新の内容に維持するメカニズムを導入します。
3. 設計標準が一元的に施行されている場合は、変更、更新、例外を申請するプロセスを設けます。

実装計画に必要な工数レベル: 中程度設計標準を作成して共有するプロセスを開発するには、組織全体のステークホルダーとの調整と協力が必要です。

リソース

関連するベストプラクティス:

- OPS01-BP03 ガバナンス要件を評価する - ガバナンス要件は設計標準に影響を及ぼします。
- OPS01-BP04 コンプライアンス要件を評価する - コンプライアンスは設計標準作成の際に重要な情報を提供します。
- OPS07-BP02 運用準備状況の継続的な確認を実現する - 運用準備状況チェックリストは、ワークロード設計時に設計標準を実装するメカニズムです。
- OPS11-BP01 継続的改善のプロセスを用意する - 設計標準の更新は継続的改善の一環です。
- OPS11-BP04 ナレッジ管理を実施する - ナレッジ管理プラクティスの一環として、設計標準を文書化して共有します。

関連するドキュメント:

- AWS Service Catalog を使用して AWS Backup を自動化する
- AWS Service Catalog Account Factory の機能を拡張
- Expedia Group が AWS Service Catalog を使用して Database as a Service (DBaaS) サービスを構築した方法
- クラウドアーキテクチャパターンの使用に関する可視性を維持する
- AWS Organizations を設定して AWS Service Catalog のポートフォリオの共有を簡素化する

関連動画:

- AWS Service Catalog – 開始方法
- AWS re:Invent 2020: エキスパートに学ぶ AWS Service Catalog ポートフォリオの管理

関連する例:

- AWS Service Catalog リファレンスアーキテクチャ
- AWS Service Catalog ワークショップ

関連サービス:

- AWS Service Catalog

OPS05-BP07 コード品質の向上のためにプラクティスを実装する

コード品質の向上のためにプラクティスを実装し、欠陥を最小限に抑えます。例としては、テスト駆動型デプロイ、コードレビュー、標準の導入、ペアプログラミングなどがあります。このようなプラクティスを継続的インテグレーションと継続的デリバリープロセスに組み込みます。

期待される成果: 組織はコードレビューやペアプログラミングなどのベストプラクティスを使用し、コード品質が向上します。デベロッパーとオペレーターは、ソフトウェア開発ライフサイクルの一環として、コード品質のベストプラクティスを採用しています。

一般的なアンチパターン:

- コードレビューを行わずに、アプリケーションの主幹にコードをコミットしています。変更が自動的に本番環境にデプロイされ、アプリケーションの停止が発生します。
- 新しいアプリケーションの開発が、ユニットテスト、エンドツーエンドテスト、または統合テストなしで行われています。デプロイする前にアプリケーションをテストする方法がありません。
- エラーの対応には、本番環境でチームが手動の変更を加えています。テストやコードレビューを行わずに変更を加えており、継続的インテグレーションと継続的デリバリープロセスを介して変更がキャプチャされたりログに記録されたりしていません。

このベストプラクティスを活用するメリット: コードの品質を向上させるためのプラクティスを採用することで、本稼働環境に発生する問題を最小限に抑えることができます。コード品質は、ペアプロ

グラミング、コードレビュー、AI 生産性ツールの実装などのベストプラクティスの活用を促進します。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

プラクティスを実装して、コード品質を向上し、デプロイする前にエラーを最低限に抑えます。テスト駆動開発、コードレビュー、ペアプログラミングなどのプラクティスを採用して、開発の質を向上します。

Amazon Q Developer の生成 AI の機能を活用して、開発者の生産性とコード品質を向上させます。Amazon Q Developer は、コード提案の生成 (大規模言語モデルに基づく)、単体テストの作成 (境界条件を含む)、およびセキュリティ脆弱性の検出と修復によってコードセキュリティを強化します。

お客様事例

AnyCompany Retail では、コード品質の向上のためにいくつかのプラクティスを採用しており、アプリケーションのコーディング基準として、テスト駆動開発を採用しています。新しい機能には、スプリント中にデベロッパーが協力してペアプログラミングを行うことを予定しているものもあります。すべてのプルリクエストは、インテグレーションとデプロイ前に、シニアデベロッパーによるコードレビューを受けます。

実装手順

1. テスト駆動型開発、コードレビュー、ペアプログラミングなどのコード品質プラクティスを、継続的インテグレーションと継続的デリバリープロセスに採用します。このような手法を使用して、ソフトウェアの品質を向上させます。
   a. Amazon Q Developer を使用します。これは、単体テストケース (境界条件を含む) の作成、コードやコメントを使った関数の生成、一般的なアルゴリズムの実装、コード内のセキュリティポリシー違反や脆弱性の検出、シークレットの検出、Infrastructure as Code (IaC) のスキャン、コードの文書化、サードパーティのコードライブラリの学習などに役立つ生成 AI ツールです。
   b. Amazon CodeGuru Reviewer は、機械学習を利用した Java と Python コードのプログラミングについてのレコメンデーションを提供します。
   c. AWS Cloud9 を使用して共有開発環境を作成すると、コードの共同開発ができます。

実装計画に必要な工数レベル: 中。ベストプラクティスを実施する方法は数多くありますが、組織全体での導入が難しい場合があります。

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer センター
- Amazon CodeWhisperer でアプリケーションをより速く構築する 10 の方法
- Amazon CodeWhisperer でコードカバレッジの先を見る
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Agile Software Guide
- My CI/CD pipeline is my release captain
- Automate code reviews with Amazon CodeGuru Reviewer
- Adopt a test-driven development approach
- How DevFactory builds better applications with Amazon CodeGuru
- On Pair Programming
- 株式会社レンガにおける Amazon CodeGuru を使ったコードレビューの自動化
- The Art of Agile Development: Test-Driven Development
- Why code reviews matter (and actually save time!)

関連動画:

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

関連サービス:

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler
- AWS Cloud9

OPS05-BP08 複数の環境を使用する

ワークロードの実験、開発、テストを行うには、複数の環境を使用します。本番環境に近い環境ほど使用するコントロールレベルを増大し、デプロイ時にはワークロードを意図したとおりに運用できるという確信を得ます。

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

一般的なアンチパターン:

- あなたは、共有開発環境で開発を実行しており、別のデベロッパーがあなたのコードの変更を上書きします。
- 共有開発環境の制限的なセキュリティ制御により、あなたは新しいサービスや機能を試すことができません。
- あなたは本稼働用システムで負荷テストを実行し、ユーザーの機能停止を引き起こします。
- データ損失につながる重大なエラーが本稼働環境で発生しました。あなたは、データ損失がどのように発生したかを特定し、これを再び発生させないようにするため、本稼働環境において、データ損失につながる条件を再現しようとします。テスト中のさらなるデータ損失を防ぐため、あなたは、ユーザーがアプリケーションを使用できないようにすることを強制されます。
- あなたは、マルチテナントサービスを運用していますが、専用環境を求める顧客のリクエストをサポートできません。

- テストは常に実行するとは限らず、テストを行う場合は本番環境でテストします。
- あなたは、単一環境というシンプルさが、環境内での変更の影響範囲に勝ると考えています。

このベストプラクティスを活用するメリット: デベロッパーやユーザーコミュニティ間で競合を生じさせることなく、複数の同時開発、テスト、本番環境をサポートできます。

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

複数の環境を使用して、実験を行える最小限のコントロールを備えたサンドボックス環境をデベロッパーに提供します。並行して作業が進められるように個別の開発環境を提供して、開発の俊敏性を高めます。デベロッパーがイノベーションを試せるように、本番環境に近い環境でより厳格なコントロールを実装します。コードとしてインフラストラクチャを使用したり、構成管理システムを使用したりして本番環境に存在するコントロールに準拠して設定された環境をデプロイし、システムがデプロイ時に予想どおりに動作することを確認します。環境を使用しない場合は、オフにして、アイドル状態のリソース (夜間や週末の開発システムなど) に関連するコストを避けることができます。テスト結果の有効性を向上させるためにロードテストを行う場合は、本番環境と同等の環境をデプロイします。

リソース

関連するドキュメント:

- AWS での Instance Scheduler
- AWS CloudFormation とは

OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う

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

一般的なアンチパターン:

- アプリケーションの新しいバージョンを変更期間を設けて四半期ごとにデプロイするが、変更期間中は、コアサービスがオフになる。

- 管理システムで変更を追跡せずに、データベーススキーマを頻繁に変更する。
- インプレースアップデートを手動で実行して、既存のインストールと設定を上書きし、明確なロールバック計画がない。

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

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

変更の範囲と影響を低減するために、頻繁かつ小規模で可逆的な変更を行います。これにより、トラブルシューティングが容易になり、迅速に修復できるようになります。また変更を元に戻すこともできます。また、ビジネスに価値をもたらす速度も向上します。

リソース

関連するベストプラクティス:

- OPS05-BP03 構成管理システムを使用する
- OPS05-BP04 構築およびデプロイ管理システムを使用する
- OPS06-BP04 テストとロールバックを自動化する

関連するドキュメント:

- AWS でのマイクロサービスの実装
- マイクロサービス - オブザーバビリティ

OPS05-BP10 統合とデプロイを完全自動化する

ワークロードのビルド、デプロイ、テストを自動化します。これにより、手動プロセスによって発生するエラーと、変更をデプロイする労力を減らすことができます。

一貫したタグ付け戦略に従って リソースタグ および AWS Resource Groups を使用して メタデータを適用し、 リソースの識別を可能にします。組織、原価計算、アクセスコントロールのリソースにタグを付け、自動化された運用アクティビティの実行に的を絞ります。

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

一般的なアンチパターン:

- 金曜日に、機能ブランチ用の新しいコードの作成を完了します。月曜日になって、コード品質テストスクリプトと各ユニットテストスクリプトを実行した後、予定された次のリリースに向けてコードをチェックインします。
- 本番環境の多数のお客様に影響を及ぼす重要な問題の修正のコーディング作業を担当することになります。この修正をテストした後、コードと E メールの変更管理をコミットして、本番環境でのデプロイに向けて承認をリクエストします。
- デベロッパーは、AWS Management Console にログインして、標準以外の方法やシステムを使用して新しい開発環境を作成します。

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

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

構築およびデプロイ管理システムを使用して、変更を追跡、実装し、手動プロセスが原因で発生するエラーと労力を低減できます。コードのチェックインから、ビルド、テスト、デプロイ、検証を通じて統合とデプロイのパイプラインを完全自動化します。これにより、リードタイムが短縮され、変更頻度が増加し、労力が軽減され、市場投入までの時間が短縮され、生産性が向上し、本番環境に移行する際のコードのセキュリティが強化されます。

リソース

関連するベストプラクティス:

- OPS05-BP03 構成管理システムを使用する
- OPS05-BP04 構築およびデプロイ管理システムを使用する

関連するドキュメント:

- AWS CodeBuild とは
- AWS CodeDeploy とは

関連動画:

- AWSre\: Invent 2022 - AWS の DevOps 向け AWS Well-Architected ベストプラクティス

デプロイのリスクを緩和する

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

ワークロードの設計には、デプロイ、更新、運用の方法が含まれている必要があります。欠陥の削減と迅速かつ安全な修正に対応するエンジニアリングのプラクティスの実装が必要になるでしょう。

ベストプラクティス

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

OPS06-BP01 変更の失敗に備える

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

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

一般的なアンチパターン:

- あなたがデプロイを実行したところ、アプリケーションが不安定になりましたが、システムにはアクティブなユーザーがいるように見えます。変更をロールバックしてアクティブなユーザーに影響を与えるか、または、いずれにしてもユーザーが影響を受ける可能性があることを考慮して、変更をロールバックするのを待つかを判断しなければなりません。
- ルーティンを変更すると、新しい環境はアクセスできますが、サブネットの 1 つにアクセスできなくなります。すべてをロールバックするか、アクセスできないサブネットを修正するかを判断しなければなりません。その判断がなされるまでの間、サブネットはアクセスできないままとなります。
- システムが、より小さなリリースで更新できるように設計されていません。その結果、デプロイが失敗した際に、これらの一括変更を取り消すことが困難になります。
- Infrastructure as Code (IaC) を使用せず、インフラストラクチャを手動で更新してきた結果、望ましくない構成が生じます。手動変更を効果的に追跡して元に戻すことができません。
- デプロイ頻度の増加については測定していないため、チームには変更の規模を縮小したり、変更のたびにロールバック計画を改善したりする動機付けがなされておらず、リスクも失敗率が高まることになります。
- 変更の失敗によるシステム停止の合計時間を測定していないため、チームは、デプロイプロセスや復旧計画の効果を優先順位付けして改善することができません。

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

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

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

実装手順

1. 特定の期間内に変更を元に戻すための効果的な計画を立てることをチームに要求するポリシーを文書化します。
   a. ポリシーには、フィックスフォワードが許可される状況を明記します。
   b. 関係者全員が文書化されたロールバック計画にアクセスできることを必須とします。
   c. ロールバックの要件 (許可されない変更がデプロイされたことが判明した場合など) を指定します。
2. ワークロードの各コンポーネントに関連するすべての変更の影響レベルを分析します。
   a. 反復可能な変更が変更のポリシーを実行する一貫したワークフローに従っていれば、こうした変更の標準化、テンプレート化、事前承認が許可されるようにします。
   b. 変更の規模を小さくすることで、変更による潜在的な影響を軽減し、復旧にかかる時間を短縮し、ビジネスへの影響を軽減します。
   c. 可能な限りインシデントを回避するために、ロールバック手順によってコードが確実に既知の良好な状態に戻るようにします。
3. ツールとワークフローを統合して、プログラムによってポリシーを適用します。
4. 変更に関するデータを他のワークロードオーナーにも見えるようにすることで、ロールバックができない変更の失敗の診断を迅速に行えるようにします。
   a. 目に見える変更データを使用することで、このプラクティスの成功を測定し、反復的な改善点を特定します。
5. モニタリングツールを使用してデプロイの成功または失敗を検証し、ロールバックに関する意思決定を加速します。
6. 変更の失敗時のシステム停止時間を測定して、復旧計画を継続的に改善します。

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

- OPS06-BP04 テストとロールバックを自動化する

関連するドキュメント:

- AWS Builders Library | デプロイ時におけるロールバックの安全性の確保

- AWS ホワイトペーパー | Change Management in the Cloud

関連動画:

- re:Invent 2019 | Amazon's approach to high-availability deployment

OPS06-BP02 デプロイをテストする

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

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット: デプロイ手順とデプロイによって生じる変更を本番稼働前に十分にテストすることで、デプロイ手順による本番環境への潜在的な影響を最小限に抑えるこ

とができます。これにより、変更の提供を遅らせることなく、本番リリースでの自信が高まり、運用サポートが最小限に抑えられます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

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

お客様事例

AnyCompany Retail は、継続的インテグレーションと継続的デリバリー (CI/CD) パイプラインの一環として、インフラストラクチャとソフトウェアの更新を顧客にリリースするために必要な定義済みの手順を本番環境に似た環境で実行します。このパイプラインは、デプロイ前にリソースのドリフトを検出する (IaC 外で実行されたリソースへの変更を検出する) 事前チェックと、IaC の開始時に実行されるアクションの検証で構成されます。ロードバランサーに再登録する前に、特定のファイルや設定が整っていること、サービスが実行中の状態にあって、ローカルホストでのヘルスチェックに正しく応答していることを確認するなど、デプロイ手順が検証されます。さらに、すべての変更は、機能テスト、セキュリティテスト、リグレッションテスト、統合テスト、負荷テストなど、多くの自動テストにフラグを立てます。

実装手順

1. インストール前のチェックを実行して、本番環境をミラーリングした本番稼働前の環境を設定します。
   a. ドリフト検出を 使用して、 AWS CloudFormation 外でリソースが変更された場合に検出します。
   b. 変更セットを 使用して、 スタック更新の意図が、変更セットの開始時に AWS CloudFormation が実行するアクションと一致することを検証します。
2. これにより、 AWS CodePipeline で、本番稼働前環境へのデプロイを承認するための手動承認手順がトリガーされます。
3. AWS CodeDeploy AppSpec ファイル などのデプロイ設定を使用して、 デプロイと検証の手順を定義します。
4. 該当する場合は、AWS CodeDeploy を他の AWS サービスと統合 もしくは AWS CodeDeploy をパートナー製品およびサービスと統合します。

5. Amazon CloudWatch、 AWS CloudTrail、Amazon SNS イベント通知を使用して、デプロイをモニタリングします。
6. 機能テスト、セキュリティテスト、リグレッションテスト、統合テスト、負荷テストなど、デプロイ後の自動テストを実行します。
7. デプロイに関する問題を トラブルシューティングします。
8. 上記の手順の検証が成功すると、本番環境へのデプロイを承認するための手動承認ワークフローが開始されます。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- OPS05-BP02 変更をテストし、検証する

関連するドキュメント:

- AWS Builders' Library | 安全なハンズオフデプロイメントの自動化 | デプロイテスト
- AWS ホワイトペーパー | Practicing Continuous Integration and Continuous Delivery on AWS
- The Story of Apollo - Amazon's Deployment Engine
- コードを送信する前に AWS CodeDeploy をローカルでテスト/デバッグする方法
- Integrating Network Connectivity Testing with Infrastructure Deployment

関連動画:

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

関連する例:

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

OPS06-BP03 安全なデプロイ戦略を使用する

安全な本番環境のロールアウトでは、変化による顧客への影響を最小限に抑えることを目的として、有益な変化の流れを管理します。安全管理は、期待される結果を検証し、変更やデプロイの失敗に

よって生じた不具合による影響の範囲を制限するための検査メカニズムを提供します。安全なロールアウトには、機能フラグ、ワンボックス、ローリング (canary リリース)、イミュータブル、トラフィック分割、ブルー/グリーンデプロイなどの戦略が含まれる場合があります。

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

一般的なアンチパターン:

- あなたは、失敗した変更を一度にすべての本稼働環境にデプロイします。その結果、すべての顧客に一斉に影響が及びます。
- 全システムへの同時デプロイで生じた不具合により、緊急リリースが必要となります。すべての顧客への影響を修正するには数日かかります。
- 本番リリースを管理するために、複数のチームの計画と参加が必要です。これにより、顧客のために頻繁に機能を更新する能力が制限されます。
- あなたは、既存のシステムを変更することにより、変更可能なデプロイを実行します。変更の失敗が判明した後、あなたは、システムを再度変更して古いバージョンを復元することを強いられ、これにより復旧にかかる時間が長くなります。

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

このベストプラクティスを活用しない場合のリスクレベル: 中

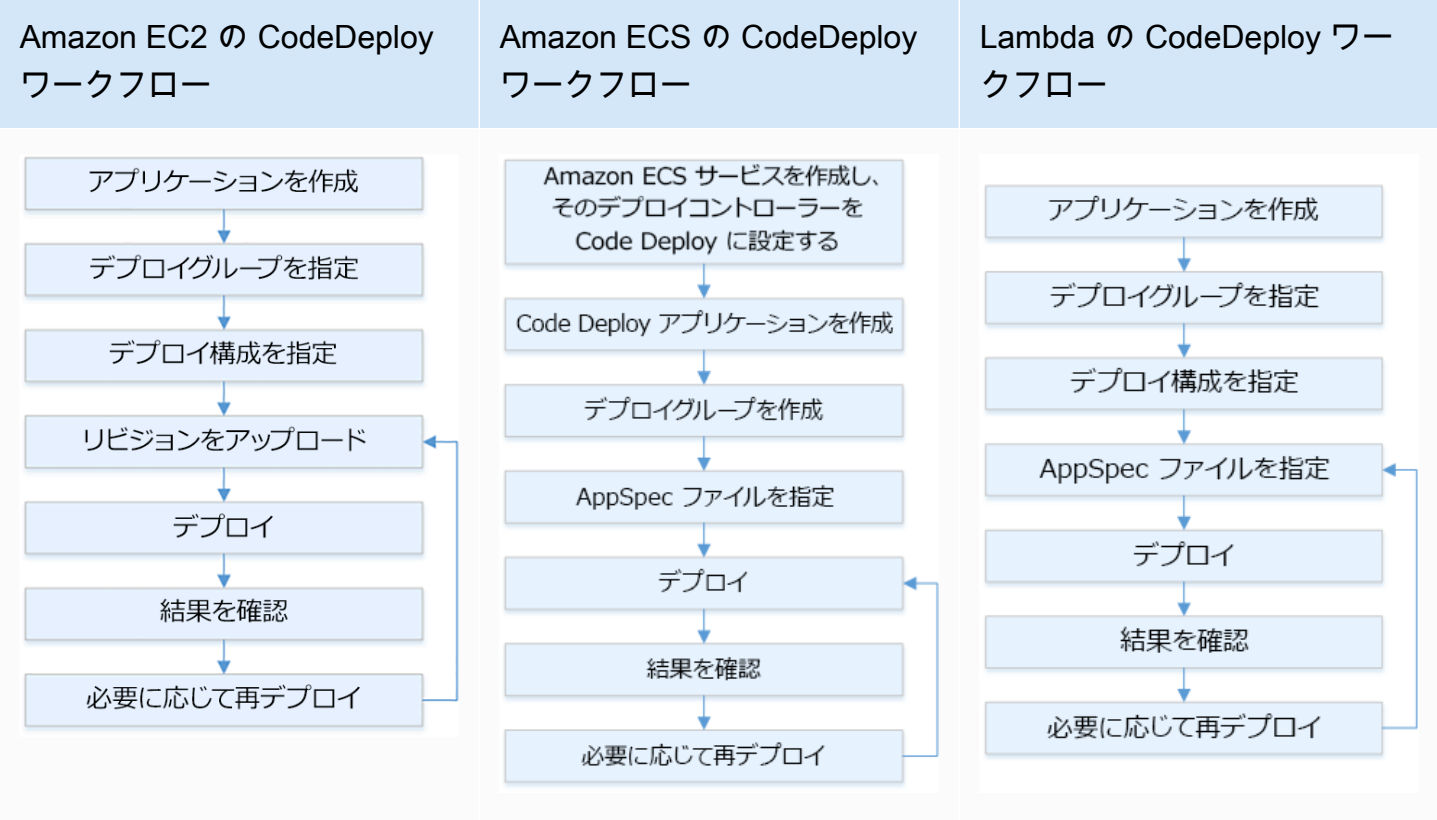
実装のガイダンス

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

お客様事例

AnyCompany Retail は、ダウンタイムを最小限またはゼロにすることを目指しています。これは、デプロイ中に認識されるユーザーへの影響がまったくないことを意味します。これを実現するために、同社はローリングデプロイやブルー/グリーンデプロイなどのデプロイパターン (次のワークフ

ロー図を参照) を確立しました。すべてのチームが、CI/CD パイプラインでこれらのパターンを 1 つ以上採用しています。



実装手順

1. 承認ワークフローを使用して、本番環境に移行する際に、一連の本番環境のロールアウト手順を開始します。
2. AWS CodeDeploy などの 自動デプロイシステムを使用します。AWS CodeDeploy の デプロイオプション には、EC2/オンプレミス向けのインプレースデプロイと、EC2/オンプレミス、AWS Lambda、Amazon ECS 向けのブルー/グリーンデプロイが含まれています (上のワークフロー図を参照)。
   a. 該当する場合は、AWS CodeDeploy を他の AWS サービスと統合 もしくは AWS CodeDeploy をパートナー製品およびサービスと統合します。
3. Amazon Aurora や Amazon RDS の などのデータベースでは ブルー/グリーンデプロイを使用します。
4. デプロイをモニタリング します。それには、Amazon CloudWatch、AWS CloudTrail、Amazon Simple Notification Service (Amazon SNS) イベント通知を使用します。

5. 機能テスト、セキュリティテスト、リグレッションテスト、統合テスト、負荷テストなど、デプロイ後の自動テストを実行します。
6. デプロイに関する問題を トラブルシューティングします。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS05-BP02 変更をテストし、検証する
- OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う
- OPS05-BP10 統合とデプロイを完全自動化する

関連するドキュメント:

- AWS Builders' Library | 安全なハンズオフデプロイメントの自動化 | 本番デプロイメント
- AWS Builders Library | My CI/CD pipeline is my release captain | Safe, automatic production releases
- AWS ホワイトペーパー | Practicing Continuous Integration and Continuous Delivery on AWS | Deployment methods
- AWS CodeDeploy ユーザーガイド
- AWS CodeDeploy でのデプロイ構成の操作
- API Gateway Canary リリースデプロイの設定
- Amazon ECS Deployment Types
- Fully Managed Blue/Green Deployments in Amazon Aurora and Amazon RDS
- AWS Elastic Beanstalk を使用したブルー/グリーンデプロイ

関連動画:

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

関連する例:

- AWS CodeDeploy でブルー/グリーンデプロイのサンプルを試す
- Workshop | Buiding CI/CD pipelines for Lambda canary deployments using AWS CDK
- Workshop | Blue/Green and Canary Deployment for EKS and ECS
- Workshop | Building a Cross-account CI/CD Pipeline

OPS06-BP04 テストとロールバックを自動化する

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

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

一般的なアンチパターン:

- システムが、より小さなリリースで更新できるように設計されていません。その結果、デプロイが失敗した際に、これらの一括変更を取り消すことが困難になります。
- デプロイプロセスが一連の手動のステップで構成されています。ワークロードに変更をデプロイした後に、デプロイ後のテストを開始します。テスト後、ワークロードが操作できず、顧客の接続が切断されたことに気付きます。あなたはその後、以前のバージョンへのロールバックを開始します。こうした手動の手順すべてが、システム復旧を遅らせるだけでなく、顧客への影響も長引く原因となります。
- アプリケーションで使用頻度の低い機能に対する自動テストケースを時間をかけて構築したことで、自動テスト機能の投資利益率が最小化されています。
- リリースには、アプリケーション、インフラストラクチャ、パッチ、および設定の更新が含まれ、これらは互いに独立しています。ただし、単一の CI/CD パイプラインを使用して、すべての変更

を一度に提供しています。1 つのコンポーネントで失敗が発生すると、すべての変更を元に戻すことを強いられることになり、ロールバックが複雑で非効率なものになります。

- あなたのチームは、スプリント 1 でコード作業を完了し、スプリント 2 の作業を開始しますが、計画にはスプリント 3 まではテストが含まれていません。その結果、自動テストによって、スプリント 2 の成果物のテストを開始する前に解決が必要な障害がスプリント 1 で検出されたため、リリース全体が遅れ、あなたの自動テストの評価が下がってしまいます。
- 本番リリースに対する自動リグレッションテストケースは完了していますが、ワークロードの状態はモニタリングしていません。サービスが再起動したかどうかを確認できないため、あなたはロールバックが必要なのか、ロールバックが実行済みなのかがわかりません。

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

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

1. 要件の計画から、テストケースの作成、ツールの設定、自動テスト、テストケースの完了に至る、テストプロセスのあらゆる段階を定義する、開発ライフサイクルのテストライフサイクルを確立します。
   a. 全体的なテスト戦略から、ワークロード固有のテストアプローチを作成します。
   b. 開発ライフサイクル全体を通じて、必要に応じて継続的なテスト戦略を検討します。
2. ビジネス要件とパイプラインへの投資に基づいて、テストとロールバック向けの自動ツールを選択します。

3. どのテストケースを自動化し、どのテストケースを手動で実行するかを決めます。これは、テスト対象の機能に対するビジネス価値の優先順位に基づいて定義できます。チームメンバー全員にこの計画を浸透させて、手動テストを実施する責任を確認します。
   a. 反復可能なケースや頻繁に実行されるケース、反復的なタスクが必要なケース、複数の構成で必要なケースなど、自動化に適した特定のテストケースに自動テスト機能を適用します。
   b. 自動化ツールでテスト自動化スクリプトと成功基準を定義して、特定のケースが失敗した場合に継続的なワークフローの自動化が開始されるようにします。
   c. 自動ロールバックの具体的な失敗基準を定義します。
4. テスト自動化を優先させ、複雑さと人間の操作によって失敗のリスクが高まる部分で、綿密なテストケースにより一貫した結果が達成されるようにします。
5. 自動テストツールとロールバックツールを CI/CD パイプラインに統合します。
   a. 変更の明確な成功基準を策定します。
   b. モニタリングと観察によってこうした基準を検出し、特定のロールバック基準を満たす場合は自動的に変更を元に戻します。
6. 次のようなさまざまなタイプの自動の本番環境テストを実施します。
   a. 2 つのユーザーテストグループ間の現在のバージョンとの比較結果を示す A/B テスト。
   b. すべてのユーザーにリリースする前に、変更を一部のユーザーにロールアウトできる canary テスト。
   c. アプリケーションの外部から新しいバージョンの機能に一度に 1 つずつフラグを付け/外し、新しい機能を 1 つずつ検証することが可能な機能フラグテスト。
   d. 相互に関連する既存のコンポーネントで新機能を検証するリグレッションテスト。
7. アプリケーションの運用、トランザクション、他のアプリケーションやコンポーネントとのやり取りをモニタリングします。ワークロードごとに変更の成功を示すレポートを作成して、自動化とワークフローでさらに最適化の余地がある部分を特定できるようにします。
   a. ロールバック手順を呼び出すべきかどうかについて迅速な判断を可能にする、テスト結果レポートを作成します。
   b. 1 つまたは複数のテスト方法を基に事前定義された失敗条件に基づいて自動ロールバックを許可する戦略を実装します。
8. 将来の反復可能な変更で再利用が可能な自動テストケースを作成します。

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS Builders Library | デプロイ時におけるロールバックの安全性の確保
- Redeploy and rollback a deployment with AWS CodeDeploy
- 8 best practices when automating your deployments with AWS CloudFormation

関連する例:

- Serverless UI testing using Selenium, AWS Lambda, AWS Fargate (Fargate), and AWS Developer Tools

関連動画:

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

運用準備状況と変更管理

ワークロード、プロセス、手順、および従業員の運用準備状況を評価し、ワークロードに関連する運用上のリスクを理解します。環境での変更フローを管理します。

ワークロードや変更を本番稼働する準備が整うタイミングを明らかにするために、一貫性のあるプロセス (手作業または自動化によるチェックリストを含む) を使用します。これにより、対処計画を策定する必要がある領域も明らかにすることができます。日常業務を文書化したランブックと、問題解決のプロセスのガイドとなるプレイブックを利用します。ビジネスバリューの提供をサポートし、変更に関連するリスクの緩和を支援する変更管理メカニズムを使用します。

ベストプラクティス

- OPS07-BP01 人材能力の確保

- OPS07-BP02 運用準備状況の継続的な確認を実現する
- OPS07-BP03 ランブックを使用して手順を実行する
- OPS07-BP04 プレイブックを使用して問題を調査する
- OPS07-BP05 システムや変更をデプロイするために十分な情報に基づいて決定を下す
- OPS07-BP06 本稼働ワークロード用のサポートプランを有効にする

OPS07-BP01 人材能力の確保

トレーニングを受けた、ワークロードをサポートするための適切な人数の従業員が配置されていることを検証するメカニズムを導入します。担当者は、ワークロードを構成するプラットフォームとサービスについてのトレーニングを受けている必要があります。ワークロードのオペレーションに必要となるナレッジを提供します。ワークロードの通常の運用サポートと発生したインシデントのトラブルシューティングを行うために、十分な人数のトレーニングを受けた人材が必要です。人員の疲弊を避けるため、オンコール対応と休暇を考慮に入れたローテーションを組むうえで十分な人材を配置します。

期待される成果:

- ワークロードが利用可能な間、ワークロードのサポートを担当する、十分なトレーニングを受けた人材が確保されています。
- ワークロードを構成するソフトウェアとサービスについて、担当者にトレーニングを提供しています。

一般的なアンチパターン:

- 使用中のプラットフォームとサービスを運用するにあたって、トレーニングを受けたチームメンバーなしでワークロードをデプロイします。
- オンコール対応と人材の休暇を考慮したローテーションを行ううえで十分な人材が不足しています。

このベストプラクティスを活用するメリット:

- スキルのあるチームメンバーを持つことで、ワークロードを効果的にサポートできます。
- チームメンバーが十分に配置されていれば、ワークロードをサポートでき、人員の疲弊を引き起こすリスクを軽減しつつ、オンコールローテーションを行うことができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードをサポートするために、十分にトレーニングを受けた担当者がいることを確認します。オンコール対応を含め、通常の運用アクティビティに対応するうえで十分なチームメンバーが配置されていることを確認します。

お客様事例

AnyCompany Retail では、ワークロードをサポートするチームが適切に配置され、トレーニングを受けていることを確認しており、オンコールローテーションをサポートするうえで十分な人数のエンジニアがいます。担当者は、ワークロード構築の基盤となっているソフトウェアとプラットフォームについてのトレーニングを受けており、認定資格の取得が奨励されています。十分な人材が配置されているため、ワークロードをサポートし、オンコールローテーションを組みつつ、担当者は休暇を取ることができます。

実装手順

1. オンコール業務を含め、ワークロードの運用とサポートに十分な人数の人材を割り当てます。
2. ワークロードを構成するソフトウェアとプラットフォームについてのトレーニングを人材に提供します。
   a. AWS トレーニングと認定 には、AWS についてのコースライブラリがあり、無料および有料のコース、オンラインコース、クラスルーム形式のコースが提供されています。
   b. AWS では、イベントやオンラインセミナーを開催しており、AWS のエキスパートから学ぶことができます。
3. 運用状況とワークロードの変化に応じて、チームの規模とスキルを定期的に評価します。運用要件に合わせてチームの規模とスキルを調整します。

実装計画に必要な工数レベル: 高。ワークロードをサポートするチームを雇用し、トレーニングするには、多大な労力が必要になる場合がありますが、長期的に多大な利点があります。

リソース

関連するベストプラクティス:

- OPS11-BP04 ナレッジ管理を実施する - チームメンバーは、ワークロードの運用とサポートを行ううえで必要となる情報を持っている必要があります。それを提供する鍵となるのが、ナレッジ管理です。

関連するドキュメント:

- AWS イベントとオンラインセミナー
- AWS トレーニングと認定

OPS07-BP02 運用準備状況の継続的な確認を実現する

運用準備状況レビュー (ORR) を使用して、組織のワークロードを運用できることを検証します。ORR は Amazon が開発した仕組みの 1 つで、チームがワークロードを安全に運用できることを検証します。ORR は、要件のチェックリストを使用したレビューおよび検証プロセスです。ORR は、ワークロードの検証をチームが自分たちで行うことができるセルフサービスエクスペリエンスです。ORR には、Amazon がソフトウェアを開発する中で学んだ知識や経験に基づくベストプラクティスが含まれます。

ORR チェックリストは、アーキテクチャレコメンデーション、運用プロセス、イベント管理、リリース品質によって構成されます。Amazon のエラーの修正 (CoE) プロセスは、主にこれらの項目によって推進されます。組織の ORR の発展を推進するには、独自のインシデント後の分析を使用する必要があります。ORR はベストプラクティスに従うためだけでなく、過去に経験したイベントの再発を防ぐためのものです。また、セキュリティ、ガバナンス、コンプライアンスの各要件も ORR に含めることができます。

ワークロードの一般提供前に ORR を実施し、その後はソフトウェア開発ライフサイクルをとおして実施し続けます。ワークロードのローンチ前に ORR を実施することで、ワークロードをより安全に運用することができます。ORR をワークロードで定期的に実施することで、ベストプラクティスからの逸脱を検知することができます。ORR チェックリストは、新しいサービスのローンチや、ORR の定期的なレビューに使用できます。そうすることで、新しいベストプラクティスに沿って更新したり、インシデント後の分析で学んだ知識や経験を反映したりできます。クラウドの使用に慣れていくにしたがって、組織のアーキテクチャのデフォルトの要件として ORR を組み込むことができます。

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

一般的なアンチパターン:

- 運用できるかどうか不明なままワークロードをローンチする。
- ガバナンスおよびセキュリティ要件は、ワークロードのローンチ要件に含まれていない。
- ワークロードは定期的に評価されていない。

- 必要な手続きなしでワークロードがローンチされる。
- 複数のワークロードで同じ根本原因の故障が繰り返される。

このベストプラクティスを活用するメリット:

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

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

顧客の事例

いくつかの製造インシデントが発生した後、AnyCompany Retail は ORR プロセスを導入することを決めました。彼らはベストプラクティス、ガバナンスおよびコンプライアンスの要件、故障から学んだ知識や経験で構成されたチェックリストを作成しました。新しいワークロードのローンチ前には、ORR が実施されます。すべてのワークロードでは、ベストプラクティスのサブセットを使用して年次 ORR が実施され、ORR チェックリストに追加されたベストプラクティスや要件が反映されます。時間の経過とともに、AnyCompany Retail は AWS Config を使用して、ベストプラクティスを検出し ORR プロセスを迅速化しました。

実装手順

ORR の詳細については、 運用準備状況の確認 (ORR) に関するホワイトペーパーをご覧ください。このドキュメントでは、ORR プロセスの歴史、独自の ORR プラクティスの構築方法、ORR チェッ

クリストの作成方法に関する詳細な情報を提供しています。以下の手順は、このドキュメントからの抜粋です。ORR および独自の ORR の構築方法の詳細については、このホワイトペーパーをご覧ください。

1. セキュリティ、運用、開発の代表者を含む、主要な関係者を集めます。
2. 各関係者に少なくとも 1 つの要件を提供してもらいます。初回に提供される要件は、30 項目以下に制限します。
   - 付録 B: 運用準備状況の確認 (ORR) に関する ホワイトペーパーの ORR 質問の例には、使用できるいくつかの質問の例が含まれています。
3. 要件をスプレッドシートにまとめます。
   - ここでは AWS Well-Architected Tool の カスタムレンズを 使用して ORR を作成し、 アカウントや AWS 組織全体で共有することができます。
4. ORR を実施するワークロードを 1 つ選びます。ローンチ前のワークロード、または内部ワークロードが理想的です。
5. ORR チェックリストを実施し、発見した事柄をメモします。定められた緩和がある場合、発見は問題になる可能性があります。緩和が定められていない発見については、対応予定の項目に追加して、ローンチ前に対応を実施します。
6. 時間の経過とともに、ベストプラクティスや要件を ORR に継続的に追加します。

エンタープライズサポートのある AWS Support の顧客は、 運用準備状況の確認に関するワークショップ をテクニカルアカウントマネージャーからリクエストできます。このワークショップでは、 顧客の視点から ORR チェックリストの作成を行います。

実装計画に必要な工数レベル: 高。組織で ORR プラクティスを採用するには、エグゼクティブスポンサーと関係者の同意が必要です。組織全体からのインプットを含めてチェックリストを作成し更新します。

リソース

関連するベストプラクティス:

- OPS01-BP03 ガバナンス要件を評価する - ガバナンス要件は ORR チェックリストに適しています。
- OPS01-BP04 コンプライアンス要件を評価する - コンプライアンス要件は ORR チェックリストに含まれることがあります。別のプロセスに含まれる場合もあります。

- OPS03-BP07 チームに適正なリソースを提供する - チームキャパシティは ORR 要件の良い候補です。
- OPS06-BP01 変更の失敗に備える - ワークロードをローンチする前に、ロールバックプランまたはロールフォワードプランを確立する必要があります。
- OPS07-BP01 人材能力の確保 - ワークロードをサポートするために、必要な人材を確保する必要があります。
- SEC01-BP03 管理目標を特定および検証する - セキュリティ管理目標は ORR 要件に最適の項目です。
- REL13-BP01 ダウンタイムやデータ消失に関する復旧目標を定義する - ディザスタリカバリプランは ORR 要件に適しています。
- COST02-BP01 組織の要件に基づいてポリシーを策定する - コスト管理ポリシーは ORR チェックリストの項目として適しています。

関連するドキュメント:

- AWS Control Tower - AWS Control Tower のガードレール
- AWS Well-Architected Tool - カスタムレンズ
- Adrian Hornsby による運用準備状況レビューテンプレート
- 運用準備状況の確認 (ORR) に関するホワイトペーパー

関連動画:

- あなたをサポートする AWS | 効果的な運用準備状況レビュー (ORR) の構築

関連サンプル:

- 運用準備状況レビュー (ORR) レンズの例

関連サービス:

- AWS Config
- AWS Control Tower
- AWS Security Hub
- ORR を作成し、

OPS07-BP03 ランブックを使用して手順を実行する

ランブックは、特定の成果を達成するために文書化されたプロセスです。ランブックは一連のステップから成り、それをたどることでプロセスを完了できます。ランブックは、飛行機の黎明期から運用に使用されてきました。クラウド運用では、ランブックを使用してリスクを削減し、望ましい成果を達成します。端的に言うと、ランブックはタスクを完了するためのチェックリストです。

ランブックは、ワークロードを運用するための不可欠の一部です。新しいチームメンバーのオンボーディングからメジャーリリースのデプロイまで、ランブックは、使用者に関係なく、一定の成果をもたらすように成文化されたプロセスです。ランブックの更新は変更管理プロセスの重要な要素であるため、ランブックは一箇所で公開し、プロセスの進化に合わせて更新する必要があります。また、エラー処理、ツール、アクセス許可、例外、問題発生時のエスカレーションに関するガイダンスを含める必要があります。

組織が成熟してきたら、ランブックの自動化を始めましょう。短く、頻繁に使用されるランブックから始めます。スクリプト言語を使用して、ステップを自動化するか、ステップを実行しやすくします。最初のいくつかのランブックを自動化したら、より複雑なランブックを自動化するために時間を割くようにします。やがて、ほとんどのランブックが何らかの方法で自動化されるはずです。

期待される成果: チームには、ワークロードのタスクを実行するためのステップバイステップのガイド集があります。ランブックには、期待される成果、必要なツールとアクセス許可、エラー処理に関する指示が含まれています。一箇所 (バージョン管理システム) に保管され、頻繁に更新されます。例えば、ランブックは、アプリケーションアラーム、運用上の問題、計画されたライフサイクルイベントの発生時に、重要なアカウントの AWS Health イベントをモニタリング、通知、対応するための機能をチームに提供します。

一般的なアンチパターン:

- プロセスの各ステップの完了を記憶に頼る。
- チェックリストなしで、変更を手動でデプロイする。
- 異なるチームメンバーが同じプロセスを実行しても、手順や結果が異なる。
- システムの変更や自動化に伴い、ランブックの同期が取れなくなる

このベストプラクティスを活用するメリット:

- 手動タスクのエラー率を削減します。
- 運用が一貫した方法で実行されます。
- 新しいチームメンバーがタスクの実行をすぐに始められます。

- ランブックの自動化により、苦労を減らすことができます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

ランブックは、組織の成熟度に応じて、いくつかの形態をとります。少なくとも、ステップバイステップのテキスト文書で構成されている必要があります。期待される成果が明確に示されている必要があります。必要な特殊なアクセス許可やツールを明確に文書化します。問題発生時にエラー処理とエスカレーションに関する詳細なガイダンスを提供します。ランブックの所有者をリストアップし、一元的な場所で公開します。ランブックが文書化されたら、チームの別のメンバーに使用してもらって検証します。プロセスの進化につれて、変更管理プロセスに従ってランブックを更新します。

組織が成熟するにつれて、テキストのランブックは自動化されるはずです。AWS Systems Manager Automation などのサービスを使用すると、ワークロードに対して実行可能な自動化にフラットテキストを変換できます。これらの自動化はイベントに反応して実行でき、ワークロードを保守する運用上の負担が軽減されます。AWS Systems Manager Automation は、ローコードのビジュアルデザインエクスペリエンスも提供し、自動化ランブックをより簡単に作成できます。

お客様事例

AnyCompany Retail は、ソフトウェアのデプロイ時にデータベーススキーマの更新を行う必要があります。クラウド運用チームはデータベース管理チームと協力して、これらの変更を手動でデプロイするためのランブックを作成しました。ランブックには、プロセスの各ステップがチェックリスト形式で記載されました。問題発生時のエラー処理のセクションも含まれています。このランブックは、他のランブックとともに社内 Wiki で公開されました。クラウド運用チームは、将来のスプリントでランブックを自動化する予定です。

実装手順

既存のドキュメントリポジトリがない場合、バージョン管理リポジトリはランブックライブラリの構築を始める場所として最適です。ランブックは Markdown を使用して作成できます。ランブック作成の開始に使用できるサンプルのランブックテンプレートを提供しています。

```
# Runbook Title
## Runbook Info
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last
 Updated | Escalation POC |
|-------|-------|-------|-------|-------|-------|-------|
```

```
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions
 | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
2. Step two
```

1. 既存のドキュメントリポジトリや Wiki がない場合は、バージョン管理システムに新しいバージョン管理リポジトリを作成します。
2. ランブックがないプロセスを特定します。理想的なプロセスは、半定期的に実施され、ステップ数が少なく、失敗の影響が少ないプロセスです。
3. ドキュメントリポジトリに、テンプレートを使用して新しいドラフト Markdown ドキュメントを作成します。[ランブックのタイトル] を入力して、[ランブック情報] の下にある必須フィールドを入力します。
4. 最初のステップから開始して、ランブックのステップ部分を入力します。
5. ランブックをチームメンバーに渡します。ランブックを使用してもらって、ステップを検証します。不足しているものや明確化が必要なものがあれば、ランブックを更新します。
6. ランブックを社内ドキュメントストアに公開します。公開したら、チームや他の関係者に伝えましょう。
7. 時間が経てば、ランブックのライブラリが構築されますライブラリが大きくなったら、ランブックを自動化する作業を開始します。

実装計画に必要な工数レベル: 低。ランブックの最低基準は、ステップバイステップのテキストガイドです。ランブックの自動化は、導入の手間を増やす可能性があります。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する
- OPS07-BP04 プレイブックを使用して問題を調査する
- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する
- OPS10-BP02 アラートごとにプロセスを用意する
- OPS11-BP04 ナレッジ管理を実施する

関連するドキュメント:

- AWS Well-Architected Framework: Concepts: Runbook development
- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Migration playbook for AWS large migrations - Task 4: Improving your migration runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

関連動画:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

関連する例:

- Well-Architected ラボ: プレイブックとランブックによるオペレーションの自動化
- AWS ブログ投稿: Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Systems Manager: オートメーションチュートリアル
- AWS Systems Manager: 最新のスナップショットランブックからルートボリュームを復元する
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Gitlab - ランブック
- Rubix - A Python library for building runbooks in Jupyter Notebooks
- ランブック作成のためのドキュメントビルダーの使用

関連サービス:

- AWS Systems Manager Automation

OPS07-BP04 プレイブックを使用して問題を調査する

プレイブックは、インシデントの調査に使用するステップバイステップガイドです。インシデントが発生した際は、プレイブックを使用して調査を行い、影響の範囲と根本原因を特定します。プレイブックは、デプロイの失敗からセキュリティインシデントに至るまで、さまざまなシナリオで使用さ

れます。ランブックを使用して緩和する根本原因は、多くの場合プレイブックによって特定します。プレイブックは、組織のインシデント対応計画の基幹的なコンポーネントです。

優れたプレイブックには、いくつかの重要な特徴があります。プレイブックは検出プロセスにおける各手順をユーザーに示します。外側から内側への思考を使って、インシデントの診断に必要な手順を示します。 特別なツールやより高い権限が必要な場合は、プレイブックで明確に定義します。インシデント調査のステータスを関係者と共有するためのコミュニケーションプランの策定は重要なコンポーネントです。根本原因を特定できない場合に備え、プレイブックにはエスカレーションプランが必要です。根本原因を特定できる場合、プレイブックは問題の解決方法が記載されているランブックを示す必要があります。プレイブックは一元的に保管し、定期的に更新する必要があります。特定のアラートにプレイブックを使用する場合、使用すべきプレイブックをアラート内でチームに示します。

組織が成熟するにしたがって、プレイブックを自動化します。最初に、低リスクインシデント用のプレイブックを作成します。スクリプトを使用して検出手順を自動化します。一般的な根本原因を緩和するための関連するランブックも作成します。

期待される成果: 組織には一般的なインシデントに対するプレイブックがあります。プレイブックは一元的に保管され、チームメンバーに提供されます。プレイブックは頻繁に更新されます。既知の根本原因については、関連するランブックが作成されています。

一般的なアンチパターン:

- インシデントを調査する標準的な方法がない。
- チームメンバーは過去の経験や社内で蓄積した知識に基づいて、失敗したデプロイの問題を解決している。
- 新しいチームメンバーは、トライアンドエラーを通じて問題の調査方法を学んでいる。
- 問題調査のベストプラクティスがチーム間で共有されていない。

このベストプラクティスを活用するメリット:

- プレイブックはインシデント緩和の工数を削減します。
- さまざまなチームメンバーが同じプレイブックを使って、一貫した方法で根本原因の特定を行えます。
- 既知の根本原因にはランブックが用意されており、復旧時間を短縮できます。
- プレイブックによって、新しいチームメンバーはすぐにチームに貢献できるようになります。

- 繰り返し使用可能なプレイブックを持つことで、チームはプロセスをスケールすることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

プレイブックの作成方法と使用方法は、組織の成熟度によって異なります。組織がクラウドに慣れていない場合、文章によるプレイブックを作成し、中央ドキュメントリポジトリに保管します。組織が成熟するにしたがって、Python などのスクリプト言語を使用して、プレイブックを半自動化できます。これらのスクリプトは Jupyter Notebook 内で実行でき、復旧を迅速化します。高度な組織では、一般的な問題のプレイブックを完全に自動化し、ランブックを使用して自動的に問題を緩和します。

プレイブックの作成は、組織のワークロードで発生する一般的なインシデントを一覧化することから始めます。最初に、根本原因がいくつかの問題に絞られている、低リスクインシデント用のプレイブックを作成します。シンプルなシナリオ用のプレイブックの作成後、高リスクシナリオや根本原因があまり知られていないシナリオ用のプレイブックを作成します。

組織が成熟するにつれて、文章によるプレイブックを自動化します。AWS Systems Manager Automations などのサービスを使用すると、フラットテキストを自動化に変換できます。これらの自動化を組織のワークロードで実行し、調査を迅速化できます。これらの自動化はイベントへの応答としてアクティブ化され、インシデントの検出と解決の平均時間を短縮します。

お客様は、AWS Systems Manager Incident Manager を使用してインシデントに対応できます。このサービスは、インシデントのトリアージを行い、インシデントの検出中および緩和中に関係者に情報を提供し、インシデントを通してコラボレーションを行うための単一のインターフェイスを提供します。このサービスは AWS Systems Manager Automations を使用して検出と復旧を迅速化します。

お客様事例

AnyCompany Retail で製造上の問題が発生しました。オンコールエンジニアは、プレイブックを使用して問題を調査しました。調査を進める中で、AnyCompany Retail はプレイブックに記載されている主要な関係者と情報を共有し続けました。エンジニアは、根本原因がバックエンドサービス内の競合状態であることを特定しました。エンジニアはランブックを使用してサービスを再起動し、AnyCompany Retail をオンライン状態に戻しました。

実装手順

既存のドキュメントリポジトリがない場合、プレイブックライブラリ用のバージョン管理リポジトリを作成することをお勧めします。プレイブックは Markdown を使用して作成できます。Markdown は、ほとんどのプレイブック自動化システムとの互換性を持っています。プレイブックを一から作成する場合、以下のプレイブックテンプレートの例を使用します。

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

1. 既存のドキュメントリポジトリや Wiki がない場合は、バージョン管理システムにプレイブック用の新しいバージョン管理リポジトリを作成します。
2. 調査が必要な一般的な問題を特定します。根本原因がいくつかの問題に絞られており、解決策が低リスクであるシナリオを選んでください。
3. Markdown テンプレートを使用して、[プレイブック名] セクションと [プレイブック情報] の下のフィールドに入力します。
4. トラブルシューティング手順を入力します。実行すべきアクション、または調査すべき領域をできるだけ明確に記載します。
5. プレイブックをチームメンバーに渡して、内容を確認してもらいます。記載漏れや不明瞭な記載がある場合、プレイブックを更新します。
6. プレイブックをドキュメントリポジトリに公開し、チームと関係者に通知します。
7. このプレイブックライブラリは、追加のプレイブックによって拡大します。いくつかのプレイブックを作成したら、AWS Systems Manager Automations などのツールを使用して自動化を開始し、自動化とプレイブックの同期を維持します。

実装計画に必要な工数レベル: 低。プレイブックは、一元的に保管されるテキストドキュメントとして作成します。組織が成熟するにしたがって、プレイブックの自動化に移行します。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する
- OPS07-BP03 ランブックを使用して手順を実行する
- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する
- OPS10-BP02 アラートごとにプロセスを用意する
- OPS11-BP04 ナレッジ管理を実施する

関連するドキュメント:

- AWS Well-Architected Framework: Concepts: Playbook development
- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

関連動画:

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

関連する例:

- AWS Customer Playbook Framework
- AWS Systems Manager: オートメーションチュートリアル
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix – A Python library for building runbooks in Jupyter Notebooks
- ランブック作成のためのドキュメントビルダーの使用
- Well-Architected ラボ: プレイブックとランブックによるオペレーションの自動化
- Well-Architected ラボ: Jupyter を使用したインシデント対応プレイブック

関連サービス:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 システムや変更をデプロイするために十分な情報に基づいて決定を下す

ワークロードに対する変更が正常に行われた場合のプロセスと正常に行われなかった場合のプロセスを施行します。プレモータムは、チームが行う演習で、ここでは軽減戦略を策定するために障害のシミュレーションを行います。プレモータムを使用して、障害を予測し、必要に応じて手順を作成します。ワークロードに対する変更をデプロイする利点とリスクを評価します。すべての変更がガバナンスに準拠していることを確認します。

期待される成果:

- ワークロードに変更をデプロイする際に、情報に基づく意思決定を行います。
- 変更は、ガバナンスに準拠しています。

一般的なアンチパターン:

- デプロイが正常に行われなかった場合に対応するプロセスなしで、変更をワークロードにデプロイします。
- ガバナンス要件に準拠していない変更を本番環境に加えます。
- リソース使用率のベースラインを設定することなく、ワークロードの新しいバージョンをデプロイします。

このベストプラクティスを活用するメリット:

- ワークロードへの変更が正常に行われなかった場合の準備が整っています。
- ワークロードへの変更は、ガバナンスポリシーに準拠しています。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

プレモータムを使用して、変更が正常に行われなかった場合のプロセスを開発します。変更が正常に行われなかった場合のプロセスを文書化します。すべての変更がガバナンスに準拠していることを確認します。ワークロードに対する変更をデプロイする利点とリスクを評価します。

お客様事例

AnyCompany Retail では、変更が正常に行われなかった場合のプロセスの検証のために、定期的にプレモータムを実施しています。このプロセスは文書化され、共有の Wiki で公開され、頻繁に更新されています。すべての変更がガバナンスに準拠しています。

実装手順

1. ワークロードに変更をデプロイする際に、情報に基づく意思決定を行います。デプロイの正常完了基準を設定し、レビューを行います。変更のロールバックをトリガーするシナリオまたは基準を作成します。変更をデプロイする利点と、変更が正常に実行されないリスクを比較検討します。
2. すべての変更がガバナンスポリシーに準拠していることを確認します。
3. 変更が正常に実行されない場合に備え、また軽減戦略を文書化するために、プレモータムを使用します。机上演習を行って、正常に完了しない変更をモデル化して、ロールバック手順を検証します。

実装計画に必要な工数レベル: 中。プレモータム演習の実施には、組織全体にわたるステークホルダーの調整と尽力が必要となります。

リソース

関連するベストプラクティス:

- OPS01-BP03 ガバナンス要件を評価する - ガバナンス要件は、変更をデプロイするかを決定するうえでの重要な要素となります。
- OPS06-BP01 変更の失敗に備える - 障害が発生したデプロイの軽減策を設定し、プレモータムを使用して軽減策を検証します。
- OPS06-BP02 デプロイをテストする - 本番環境でのエラーの低減に向けて、すべてのソフトウェア変更について、デプロイ前に適切なテストを行う必要があります。

- OPS07-BP01 人材能力の確保 - システム変更をデプロイする際に情報に基づく決定を行うには、トレーニングを受けたワークロードサポート担当の人材が十分に配置されていることが不可欠です。

関連するドキュメント:

- Amazon Web Services: Risk and Compliance
- AWS 責任共有モデル
- Governance in the AWS クラウド: The Right Balance Between Agility and Safety (AWS クラウドのガバナンス: 俊敏性と安全性の適切なバランス)

OPS07-BP06 本稼働ワークロード用のサポートプランを有効にする

本稼働ワークロードが依存しているあらゆるソフトウェアやサービスのサポートを有効にします。本稼働のサービスレベルのニーズに合わせて、適切なサポートレベルを選択します。このような依存関係のためのサポートプランは、サービスの停止時やソフトウェアに問題が発生した場合に必要です。すべてのサービスおよびソフトウェアのベンダーについて、サポートプランやサービスのリクエスト方法を文書化します。サポートの連絡先が最新の状態に保たれていることを検証する仕組みを実装します。

期待される成果:

- 本稼働ワークロードが依存しているソフトウェアやサービスのサポートプランを実装します。
- サービスレベルのニーズに基づいて適切なサポートプランを選択します。
- サポートプラン、サポートレベル、サポートのリクエスト方法を文書化します。

一般的なアンチパターン:

- 重要なソフトウェアベンダーのサポートプランがない。ワークロードがその影響を受けたが、修正を急がせる手段もなければ、ベンダーからタイムリーに最新情報を得ることもできない。
- ソフトウェアベンダーの主要連絡先だった開発者が退社した。ベンダーのサポートに直接連絡できなくなった。時間をかけて汎用の問い合わせシステムを検索し移動しなければならず、必要なときに対応してもらうための時間が増えた。
- ソフトウェアベンダーに起因する本稼働の停止が発生した。サポートケースの記録方法に関するドキュメントがない。

このベストプラクティスを活用するメリット:

- 適切なサポートレベルを受けていると、サービスレベルのニーズを満たすのに必要な時間内で対応を得ることができます。
- サポートを受ける顧客として、本稼働で問題があればエスカレーションできます。
- インシデント発生時にソフトウェアやサービスのベンダーがトラブルシューティングを支援します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

本稼働ワークロードが依存しているあらゆるソフトウェアやサービスのベンダーのサポートプランを有効にします。サービスレベルのニーズに合わせた適切なサポートプランをセットアップします。AWS のお客様の場合は、本稼働ワークロードがある任意のアカウントで AWS Business Support 以上を有効にすることを意味します。サポートベンダーと定期的に打ち合わせ、サポートのオファー、プロセス、連絡先に関する最新情報を入手します。ソフトウェアやサービスのベンダーにサポートをリクエストする方法を、停止が発生した場合のエスカレーション方法を含めて文書化します。サポートの連絡先を最新の状態に保つ仕組みを実装します。

お客様事例

AnyCompany Retail では、すべての商用ソフトウェアおよびサービスの依存関係がサポートプランを備えています。例えば、本稼働ワークロードがあるすべてのアカウントで AWS Enterprise Support が有効になっています。問題が発生した場合は、開発者が誰でもサポートケースを作成できます。サポートのリクエスト方法、通知を受ける担当者、ケースを迅速化するベストプラクティスに関する情報を掲載した wiki ページがあります。

実装手順

1. 組織の関係者と協力して、ワークロードが依存しているソフトウェアやサービスのベンダーを特定します。これらの依存関係を文書化します。
2. ワークロードに必要なサービスレベルを判断します。それらに合うサポートプランを選択します。
3. 商用のソフトウェアやサービスの場合は、ベンダーとサポートプランを締結します。
    a. すべての本稼働稼働用アカウントで AWS Business Support 以上を契約すると、AWS Support からの応答時間が短縮されるため、これを強くお勧めします。プレミアムサポートがない場合は、問題に対処するアクションプランが必要となり、これには AWS Support からの支援が必

要です。AWS Support が、さまざまなツール、テクノロジー、人、プログラムを組み合わせて提供します。これらは、パフォーマンスの最適化、コスト削減、より迅速なイノベーションの実現を積極的に支援するために設計されたものです。AWS Business Support には追加の利点があります。AWS Trusted Advisor や AWS Personal Health Dashboard へのアクセスや、応答時間の短縮などです。

4. ナレッジマネジメントツールにサポートプランを記録します。サポートのリクエスト方法、サポートケースが記録された場合の通知先、インシデント中のエスカレーション方法を含めます。wiki は、サポートプロセスや連絡先の変更に気付いた人が誰でも、ドキュメントに必要な更新を行うことができるため、良い仕組みです。

実装計画に必要な工数レベル: 低。ソフトウェアやサービスのほとんどのベンダーは、サポートプランの登録を提供しています。ナレッジマネジメントシステムにサポートのベストプラクティスを記録して共有すると、本稼働環境に問題が発生した場合にどうすべきかをチームが確実に把握できます。

リソース

関連するベストプラクティス:

- OPS02-BP02 プロセスと手順には特定の所有者が存在する

関連するドキュメント:

- AWS Support プラン

関連サービス:

- AWS Business Support
- AWS Enterprise Support

運用

成功とは、定義したメトリクスによって測定されるビジネス成果を達成することです。ワークロードと運用の状態を理解することで、組織やビジネス成果がいつリスクにさらされるか、または現在リスクにさらされているかを特定して適切に対応することができます。

成功につなげるには、以下ができる必要があります。

トピック

- ワークロードのオブザーバビリティの活用
- 運用状態の把握
- イベントへの対応

ワークロードのオブザーバビリティの活用

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

オブザーバビリティにより、意義あるデータに集中して取り組み、ワークロードの相互作用と出力を把握できます。重要なインサイトに重点的に取り組み、不要なデータを排除することで、ワークロードのパフォーマンスを把握するうえで明快なアプローチを維持できます。

データの収集のみでなく、データを正しく解釈することも不可欠です。明確なベースラインを定義して、適切なアラートのしきい値を設定し、逸脱がないかを積極的にモニタリングします。主要なメトリクスの変化は、特に他のデータと相関している場合、特定の問題領域を指し示すことができます。

オブザーバビリティを使用すると、潜在的な課題の予測や対処がしやすくなり、ワークロードを円滑に動作させ、ビジネスニーズを満たせるようになります。

AWS では、モニタリングとロギングには Amazon CloudWatch、 分散トレースには AWS X-Ray などの特定のツールを提供しています。これらのサービスはさまざまな AWS のリソースと簡単に統合でき、効率的なデータ収集、事前定義されたしきい値に基づくアラートの設定、理解しやすいダッシュボードでのデータの閲覧を可能にします。このようなインサイトを活用することで、運用目標に沿って、十分な情報に基づいたデータ主導の意思決定を行うことができます。

ベストプラクティス

OPS08-BP01 ワークロードメトリクスを分析する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

Amazon CloudWatch などのツールを活用してメトリクス分析を行います。特に静的なしきい値が明らかでない場合や動作パターンがより異常検出に適している場合、AWS Cost Anomaly Detection や Amazon DevOps Guru などの AWS サービスを異常検出に使用できます。

実装手順

1. 分析とレビュー: ワークロードメトリクスを定期的に見直して解析します。
   a. 純粋に技術的なメトリクスよりもビジネス成果メトリクスを優先します。
   b. データ内のスパイク、ドロップ、パターンの重要性を理解します。
2. Amazon CloudWatch の利用: Amazon CloudWatch を一元化されたビューと詳細な分析に使用します。
   a. メトリクスを可視化して時系列で比較できるように CloudWatch ダッシュボードを設定します。
   b. CloudWatch のパーセンタイルを使用すると、 メトリクスの分布を明確に把握できるため、SLA の定義や外れ値を把握できます。
   c. 静的なしきい値に依存せずに異常パターンを特定するように AWS Cost Anomaly Detection を設定します。
   d. CloudWatch クロスアカウントオブザーバビリティ を実装して、リージョン内の複数のアカウントにわたるアプリケーションのモニタリングとトラブルシューティングを行います。
   e. CloudWatch Metric Insights を使用して、アカウントやリージョンのメトリクスデータをクエリして分析し、傾向や異常を特定します。
   f. CloudWatch Metric Math を適用すると、メトリクスの変換、集計、または計算を実行して、より深いインサイトが得られます。
3. Amazon DevOps Guru の採用: Amazon DevOps Guru の機械学習を強化した異常検出機能と連携して、サーバーレスアプリケーションの運用上の問題の兆候を早期に特定し、顧客に影響が及ぶ前に修正します。
4. インサイトに基づく最適化: メトリクス分析を基盤に情報に基づいた意思決定を行い、ワークロードを調整して改善します。

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

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

関連するドキュメント:

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

関連動画:

- Amazon CloudWatch でクロスアカウントオブザーバビリティを有効にする
- Amazon DevOps Guru の紹介
- AWS Cost Anomaly Detection を使用してメトリクスを継続的に分析する

関連する例:

- One Observability ワークショップ
- Amazon DevOps Guru を使用した AIOps で運用上のインサイトを得る

OPS08-BP02 ワークロードログを分析する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- 運用上のボトルネック、セキュリティ上の脅威、その他の潜在的な問題を事前に特定できます。

- ログデータを効率的に利用して、アプリケーションを継続的に最適化できます。
- アプリケーションの動作に関してより詳細に把握できるようになり、デバッグとトラブルシューティングに役立ちます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

1. CloudWatch Logs の設定: ログを CloudWatch Logs に送信するようにアプリケーションとサービスを設定します。
2. ログ異常検出の使用: Amazon CloudWatch Logs 異常検出を使用して、異常なログパターンを自動的に識別して警告します。このツールを使用すると、ログの異常を積極的に管理し、潜在的な問題を早期に検出できます。
3. CloudWatch Logs Insights の設定: CloudWatch Logs Insights を使用して、ログデータのインタラクティブ検索と分析を行います。
    a. クエリを作成してパターンを抽出し、ログデータを可視化して、実践的なインサイトを導き出します。
    b. CloudWatch Logs Insights のパターン分析を使用して、頻繁に発生するログパターンを分析および視覚化します。この機能は、ログデータの一般的な運用傾向と潜在的な外れ値を理解するのに役立ちます。
    c. CloudWatch Logs 差分を使用して、異なる期間または異なるロググループの間の差分分析を実行します。この機能を使用すると、変更点を特定し、システムのパフォーマンスや動作への影響を評価できます。
4. Live Tail を使用したリアルタイムのログモニタリング: Amazon CloudWatch Logs Live Tail を使用して、ログデータをリアルタイムで表示します。アプリケーションの運用アクティビティを発生時に積極的にモニタリングできるため、システムパフォーマンスと潜在的な問題を即座に把握できます。
5. Contributor Insights の活用: CloudWatch Contributor Insights を活用して、IP アドレスやユーザーエージェントなどの高カーディナリティディメンションでトップのトーカーを特定します。

6. CloudWatch Logs メトリクスフィルターの実装: CloudWatch Logs メトリクスフィルターを設定して、ログデータを実用的なメトリクスに変換します。これにより、アラームを設定したり、パターンをさらに詳細に分析したりできます。
7. CloudWatch クロスアカウントオブザーバビリティの実装: リージョン内の複数のアカウントにわたるアプリケーションのモニタリングとトラブルシューティングを行います。
8. 定期的なレビューと調整: ログ分析戦略を定期的に確認して、すべての関連情報を収集し、アプリケーションのパフォーマンスを継続的に最適化します。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

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

関連するドキュメント:

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

関連動画:

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

関連する例:

- CloudWatch Logs サンプルクエリ
- One Observability Workshop

OPS08-BP03 ワークロードのトレースを分析する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

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

1. AWS X-Ray の統合: トレースデータをキャプチャするために、X-Ray をアプリケーションと統合することが必要です。
2. X-Ray メトリクスの分析: サービスマップを使用してアプリケーションヘルスをモニタリングするために、レイテンシー、リクエスト率、障害率、応答時間の分布などの X-Ray トレースから取得できるメトリクスを詳細に分析します。
3. ServiceLens の使用: ServiceLens マップを活用して、サービスとアプリケーションのオブザーバビリティを強化します。これにより、トレース、メトリクス、ログ、アラーム、その他のヘルス情報を総合的に確認できます。
4. X-Ray Insights の有効化:
   a. X-Ray Insights 有効化して、トレースの異常を自動的に検出します。
   b. インサイトを調べてパターンを特定し、障害率の増加やレイテンシーの増大などについての根本原因を突き止めます。
   c. 検出された問題を時系列で分析するには、インサイトタイムラインを参照します。
5. X-Ray Analytics の使用: X-Ray Analytics を使用して、トレースデータを徹底的に調査してパターンを特定し、インサイトを抽出します。
6. X-Ray でのグループの使用: X-Ray でグループを作成して、高レイテンシーなどの条件に基づいてトレースをフィルタリングすると、より的を絞った分析につながります。
7. Amazon DevOps Guru の活用: Amazon DevOps Guru を使用して、トレース内の運用上の異常を正確に特定する機械学習モデルの利点を活用します。
8. CloudWatch Synthetics の使用: CloudWatch Synthetics を使用して、エンドポイントとワークフローを継続的にモニタリングするための Canary を作成します。Canary は X-Ray と統合でき、テスト対象のアプリケーションを詳細に分析するためのトレースデータを提供できます。
9. リアルユーザーモニタリング (RUM) の使用: AWS X-Ray と CloudWatch RUM を使用すると、アプリケーションのエンドユーザーからダウンストリームの AWS マネージドサービスまでのリクエストパスを分析してデバッグできます。これにより、エンドユーザーに影響を及ぼすレイテンシーの傾向やエラーを特定できます。
10. ログとの関連付け:X-Ray トレースビュー内でトレースデータをログに関連付けて、アプリケーションの動作に関する詳細な情報を入手します。これにより、トレース対象のトランザクションに直接関連するログイベントを確認できます。
11. CloudWatch クロスアカウントオブザーバビリティの実装: リージョン内の複数のアカウントにわたるアプリケーションのモニタリングとトラブルシューティングを行います。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

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

関連するドキュメント:

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

関連動画:

- Analyze and Debug Applications Using Amazon CloudWatch Synthetics & AWS X-Ray
- Use AWS X-Ray Insights

関連する例:

- One Observability Workshop
- AWS Lambda で X-Ray を実装する
- CloudWatch Synthetics Canary テンプレート

OPS08-BP04 実践的なアラートを作成する

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

期待される成果: 特に KPI の結果がリスクにさらされている場合に、潜在的な問題を迅速に特定して緩和するために、関連性が高く、実践的なアラートをタイムリーに受信できます。

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

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

実装手順

1. 主要業績評価指標 (KPI) の決定: アプリケーションの KPI を特定します。正確に業務への影響を反映するには、アラートをこのような KPI に関連付ける必要があります。
2. 異常検出の実装:
   - Amazon CloudWatch 異常検出の使用: Amazon CloudWatch 異常検出を設定して、異常なパターンを自動的に検出するようにすると、正当な異常に対してのみアラートが生成されるようになります。
   - AWS X-Ray Insights の使用:
     a. X-Ray Insights を設定して、トレースデータの異常を検出します。
     b. 問題が検出されたときにアラートを受け取るようにX-Ray Insights の通知を設定します。
   - Amazon DevOps Guru との統合:
     a. Amazon DevOps Guru の機械学習機能を活用して、既存のデータの運用上の異常を検出します。

   b. DevOps Guru の通知設定に移動して、異常アラートを設定します。

3. 実用的なアラートの実装: すぐに行動を起こすための適切な情報を提供するアラートを設計します。
   1. Amazon EventBridge ルールを使用して AWS Health イベントをモニタリングしたり、プログラムで AWS Health API と統合して AWS Health イベントを受信したときのアクションを自動化したりできます。これらのアクションには、計画されたすべてのライフサイクルイベントメッセージをチャットインターフェイスに送信するなどの一般的なアクションや、IT サービス管理ツールでのワークフローの開始などの特定のアクションがあります。
4. アラート疲労の軽減: 重要でないアラートを最小限に抑えます。多数の重要でないアラートによりチームに負担がかかると、重大な問題の見落としにつながり、アラートメカニズムの全体的な有効性が低下する場合があります。
5. 複合アラームの設定: Amazon CloudWatch 複合アラームを使用して複数のアラームを統合します。
6. アラートツールとの統合: Ops Genie や PagerDuty などのツールを組み込みます。
7. AWS Chatbot との統合: AWS Chatbot を統合して、Amazon Chime、Microsoft Teams、Slack にアラートを転送します。
8. ログに基づくアラート: CloudWatch でログメトリクスフィルターを使用して、特定のログイベントに基づいてアラームを生成します。
9. 確認と反復: アラート設定を定期的に見直し、調整します。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

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

- OPS08-BP03 ワークロードのトレースを分析する

関連するドキュメント:

- Amazon CloudWatch でのアラームの使用
- 複合アラームを作成する
- 異常検出に基づいて CloudWatch アラームを作成する
- DevOps Guru 通知
- X-ray Insights の通知
- インタラクティブな ChatOps による AWS リソースのモニタリング、運用、トラブルシューティング
- Amazon CloudWatch インテグレーションガイド | PagerDuty
- Integrate Opsgenie with Amazon CloudWatch

関連動画:

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

関連する例:

- Amazon CloudWatch を使用したクラウドでのアラーム、インシデント管理、修復
- チュートリアル: AWS Chatbot に通知を送信する Amazon EventBridge ルールの作成
- One Observability Workshop

OPS08-BP05 ダッシュボードを作成する

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

期待される成果:

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

一般的なアンチパターン:

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

このベストプラクティスのメリット:

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

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

ビジネス視点のダッシュボード

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

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

Amazon CloudWatch ダッシュボードは、CloudWatch コンソール内のカスタマイズ可能なホームページであり、さまざまな AWS リージョン リージョンにまたがるリソースであっても単一のビューでモニタリングできます。

実装手順

1. 基本的なダッシュボードの作成: CloudWatch で新しいダッシュボードを作成し、わかりやすい名前を付けます。
2. Markdown ウィジェットの使用: メトリクスを使用し始める前に、Markdown ウィジェットを使用して、ダッシュボードの上部にテキストコンテキストを追加します。これにより、ダッシュボードの内容、表示されるメトリクスの重要性を説明できます。説明には、その他のダッシュボードやトラブルシューティングツールへのリンクも記載できます。
3. ダッシュボード変数の作成: 動的で柔軟なダッシュボードビューを可能にするために、必要に応じてダッシュボード変数を組み込みます。
4. メトリクスウィジェットの作成: ウィジェットを追加して、アプリケーションが出力するさまざまなメトリクスを可視化し、ウィジェットを調整してシステムヘルスとビジネス成果を効果的に表示します。
5. Log Insights クエリ: CloudWatch Log Insights を使用してログから実用的なメトリクスを導き出し、インサイトをダッシュボードに表示します。
6. アラームの設定: CloudWatch Alarms をダッシュボードに統合して、しきい値を超えているメトリクスを簡単に確認できるビューを提供します。
7. Contributor Insights の使用: CloudWatch Contributor Insights を組み込んで、高カーディナリティフィールドを分析し、リソースの最上位の要因をより明確に把握します。
8. カスタムウィジェットの設計: 標準のウィジェットでは満たせない特定のニーズについては、カスタムウィジェットの作成を検討します。カスタムウィジェットを使用すると、さまざまなデータソースからデータを引き出したり、独自の方法でデータを表示したりできます。
9. AWS Health Dashboard の使用: AWS Health Dashboard を使用して、アカウントヘルス、イベント、サービスやリソースに影響する可能性のある予定されている変更についての詳細なインサイトを取得します。また、AWS Organizations でヘルスイベントを一元表示したり、独自のカスタムダッシュボードを作成したりすることもできます (詳細については、「関連する例」を参照してください)。
10. 反復と改良: アプリケーションの進化に応じて、定期的にダッシュボードを見直し、関連性を確認します。

リソース

関連するベストプラクティス:

- OPS04-BP01 主要業績評価指標を特定する

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

関連するドキュメント:

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

関連動画:

- Create Cross Account & Cross Region CloudWatch Dashboards
- AWS re:Invent 2021 - Gain enterprise visibility with AWS クラウド operation dashboards)

関連する例:

- One Observability Workshop
- Amazon CloudWatch を使用したアプリケーションモニタリング
- AWS Health Events Intelligence Dashboards and Insights
- Visualize AWS Health events using Amazon Managed Grafana

運用状態の把握

適切な措置をとれるように、運用メトリクスを定義、取得、分析して運用チームのアクティビティの可視性を高めます。

組織は、運用状態を容易に把握できる必要があります。有用なインサイトを得るには、運用チームのビジネス上の目標を定義し、ビジネス上の目標を反映する主要業績評価メトリクスを特定して、運用成果に基づきメトリクスを使用し開発します。このようなメトリクスを使用して、リーダーや関係者が情報に基づいた意思決定を行うのに役立つ、ビジネスと技術上の観点の両方を提供するダッシュボードを実装する必要があります。

AWS では、オペレーションログの統合と分析が簡単にできるため、メトリクスの生成、運用状況の把握、経時的な運用のインサイトを得ることができます。

ベストプラクティス

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

OPS09-BP01 メトリクスを使用して業務目標と KPI を測定する

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

期待される成果:

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

一般的なアンチパターン:

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

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

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

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

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

リソース

関連するドキュメント:

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

OPS09-BP02 ステータスと傾向を伝達して運用の可視性を確保する

運用のステータスと傾向の方向性を把握することとは、その結果がリスクにさらされる可能性がある時期、追加の作業をサポートできるかどうか、または変更がチームに及ぼす影響を特定するために必

要です。運用イベント中に、ユーザーや運用チームが情報を参照できるステータスページを提供することにより、コミュニケーションチャネルの負担を軽減し、情報を積極的に広めることができます。

期待される成果:

- 運用リーダーは、チームがどのような種類のコール数に対応して業務を行っているのか、デプロイなど、どのような取り組みが進行中であるかを一目で把握できます。
- 通常の運用に影響が及ぶ場合、アラートが関係者やユーザーコミュニティに配信されます。
- 組織のリーダーや関係者は、アラートや影響に応じてステータスページを確認したり、連絡先、チケット情報、推定復旧時間など、運用上のイベントに関する情報を取得したりすることができます。
- 経営陣やその他の関係者には、特定期間のコール数、ユーザー満足度スコア、未処理のチケット数、チケットの経過時間などの運用に関する統計値を表示するレポートが提供されます。

一般的なアンチパターン:

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

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

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

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

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

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

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

リソース

関連するドキュメント:

- 進捗状況を測定する
- 運用を可視化するためのダッシュボードの構築

関連するソリューション:

- データオペレーション

OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け

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

期待される成果:

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

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

一般的なアンチパターン:

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

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

このベストプラクティスを活用しない場合のリスクレベル: 中程度

実装のガイダンス

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

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

リソース

関連するドキュメント:

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

イベントへの対応

計画 (販売促進、デプロイメント、障害テストなど) と計画外 (稼働率の急増やコンポーネントの障害など) の両方の運用イベントを予測する必要があります。既存のランブックとプレイブックを使用して、アラートに対応するときに一貫した結果を提供する必要があります。定義されたアラートは、応答とエスカレーションに責任を負う役割またはチームが所有する必要があります。また、システムコンポーネントのビジネスへの影響を把握し、必要に応じてこれを活用して作業の的を絞ることもできます。イベントの後に根本原因の分析 (RCA) を実行し、失敗の再発を防止したり、回避策を文書化したりする必要があります。

AWS は、ワークロードと運用のすべての側面をコードとしてサポートするツールを提供することで、イベント対応を簡素化します。このようなツールを使用すると、運用イベントへの対応をスクリプト化し、モニタリングデータに対応して実行をトリガーできます。

AWS では、障害が発生したコンポーネントを修復しようとするよりも、既知の正常なバージョンに置き換えることで、復旧時間を短縮できます。その後、障害が発生したリソースの分析を実行できます。

ベストプラクティス

- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する
- OPS10-BP02 アラートごとにプロセスを用意する
- OPS10-BP03 ビジネスへの影響に基づいて運用上のイベントの優先度を決定する
- OPS10-BP04 エスカレーション経路を決定する
- OPS10-BP05 サービスに影響するイベント発生時の顧客コミュニケーション計画を定義する
- OPS10-BP06 ダッシュボードでステータスを知らせる
- OPS10-BP07 イベントへの対応を自動化する

OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- 対応プロセスが合理化され、標準化されます。
- インシデントがサービスや顧客に与える影響を軽減します。

- 問題解決を早めます。
- 運用プロセスが継続的に改善されます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

このベストプラクティスを実装すると、ワークロードイベントを追跡することになります。インシデントと問題を扱うためのプロセスができます。プロセスは文書化され、共有され、頻繁に更新されます。問題が特定され、優先順位が付けられ、修正されます。

イベント、インシデント、問題の理解

- イベント: イ ベント とは、ある行動、出来事、または状態の変化を観察したものを指します。イベントは計画的な場合も計画外の場合もあり、ワークロードの内部または外部から発生する可能性があります。
- インシデント: インシデント とは、予定外の中断やサービス品質の低下など、対応が必要なイベントのことです。これらは、ワークロードを通常運用に復旧するために早急な対応を迫られる障害です。
- 問題: 問題 は、1 つ以上のインシデントの根本原因です。問題を特定して解決するには、再発防止のため、インシデントを掘り下げて調査することなどが必要です。

実装手順

イベント

1. イベントを監視する:
   - オブザーバビリティを実装 し、 ワークロードオブザーバビリティを活用します。
   - ユーザー、ロール、AWS サービスによって実行されたアクションを監視します。これらのアクションは、イベントとして AWS CloudTrailで記録されます。
   - アプリケーションで運用上の変更にリアルタイムに対応します。それには Amazon EventBridge を使用します。
   - リソース構成の変更を AWS Config で継続的に評価、監視、記録します。
2. プロセスを作成する:
   - どのイベントが重要でモニタリングが必要かを評価するプロセスを考案します。正常なアクティビティと異常なアクティビティのしきい値やパラメータの設定などを行います。

- イベントをインシデントにエスカレートする基準を決定します。これは、重大度やユーザーへの影響、想定される動作から逸脱しているかどうかなどに基づいて行います。
- イベントの監視と対応のプロセスを定期的に見直します。例えば、過去のインシデントの分析、しきい値の調整、警告メカニズムの改善などを行います。

インシデント

1. インシデントに対応する:
   - オブザーバビリティツールから得たインサイトを活用して、インシデントを迅速に特定し、対応します。
   - AWS Systems Manager OpsCenter を実装して、運用上の問題とインシデントを集約して整理し、優先順位を付けます。
   - より詳細な分析とトラブルシューティングを行うため、 Amazon CloudWatch や AWS X-Ray などのサービスを利用します。
   - インシデント管理の強化のため、 AWS Managed Services (AMS) の積極的、予防的、検出的な機能を利用することを検討します。AMS は、モニタリング、インシデントの検出と対応、セキュリティ管理などのサービスで運用サポートを拡充します。
   - エンタープライズサポートのお客様は AWS Incident Detection and Response を利用できます。本番ワークロードを継続的かつ予防的に監視し、インシデント管理を担うサービスです。
2. インシデント管理プロセスを作成する:
   - 役割、コミュニケーションプロトコル、解決手順などを明確に定義した、構造化されたインシデント管理プロセスを確立します。
   - 対応と調整を効率化するため、 AWS Chatbot などのツールをインシデント管理に統合します。
   - 重大度別にインシデントを分類し、各カテゴリの インシデント対応計画 をあらかじめ定義しておきます。
3. 学習して改善する:
   - 根本原因と解決効果を理解するため、 インシデント後の分析 を実施します。
   - 見直しと変化する慣行に基づいて、対応計画を継続的に更新および改善します。
   - 学んだ教訓を文書化し、チーム全体で共有することで、業務のレジリエンスを強化します。
   - エンタープライズサポートのお客様は インシデント管理ワークショップ をテクニカルアカウントマネージャーからリクエストできます。このガイド付きワークショップでは、既存のインシデント対応計画をテストし、改善すべき点を明らかにすることができます。

問題

1. 問題を特定する:
   - 過去のインシデントからのデータを活用して、システム上の深層の問題を示唆している可能性のある、反復的なパターンを洗い出します。
   - ツール (AWS CloudTrail や Amazon CloudWatch など) を利用して傾向を分析し、根本的な問題を明らかにします。
   - 運用、開発、ビジネスユニットなど、部門横断的なチームを組織し、多様な視点から根本原因を探ります。
2. 問題管理プロセスを作成する:
   - 構造化された問題管理プロセスを開発し、その場しのぎの修正ではなく長期的な解決策に焦点を当てます。
   - 根本原因分析 (RCA) 手法を取り入れて、インシデントの根本原因を調査し、理解します。
   - 検出結果に基づいて運用ポリシー、手順、インフラストラクチャを更新し、再発を防ぎます。
3. 継続的に改善する:
   - 絶え間ない学習と改善の文化を育み、潜在的な問題を先回りして特定し、対処することをチームに奨励します。
   - ビジネスとテクノロジーにおける環境の変化に応じて、問題管理のプロセスとツールを定期的に見直し、改訂します。
   - 組織全体でインサイトとベストプラクティスを共有して、よりレジリエントで効率的な運用環境を構築します。
4. AWS Support と連携する:
   - AWS のサポートリソース (AWS Trusted Advisor など) を活用し、先を見据えたガイダンスと最適化の推奨事項を確認します。
   - エンタープライズサポートのお客様は、重大イベントの実施中のサポートを行う AWS Countdown など、専門的なプログラムを利用できます。
   -

実装計画に必要な工数レベル: 中程度

リソース

関連するベストプラクティス:

- OPS04-BP01 主要業績評価指標を特定する

- OPS04-BP02 アプリケーションテレメトリーを実装する
- OPS07-BP03 ランブックを使用して手順を実行する
- OPS07-BP04 プレイブックを使用して問題を調査する
- OPS08-BP01 ワークロードメトリクスを分析する
- OPS11-BP02 インシデント後の分析を実行する

関連するドキュメント:

- AWS セキュリティインシデント対応ガイド
- AWS Incident Detection and Response
- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- DevOps および SRE 時代のインシデント管理
- PagerDuty - インシデント管理とは

関連動画:

- Top incident response tips from AWS
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 yrs of Amazon operational excellence
- AWS re:Invent 2022 - AWS Incident Detection and Response (SUP201)
- Introducing Incident Manager from AWS Systems Manager

関連する例:

- AWS プロアクティブサービス - インシデント管理ワークショップ
- How to Automate Incident Response with PagerDuty and AWS Systems Manager Incident Manager
- Engage Incident Responders with the On-Call Schedules in AWS Systems Manager Incident Manager
- Improve the Visibility and Collaboration during Incident Handling in AWS Systems Manager Incident Manager
- Incident reports and service requests in AMS

関連サービス:

- Amazon EventBridge

OPS10-BP02 アラートごとにプロセスを用意する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

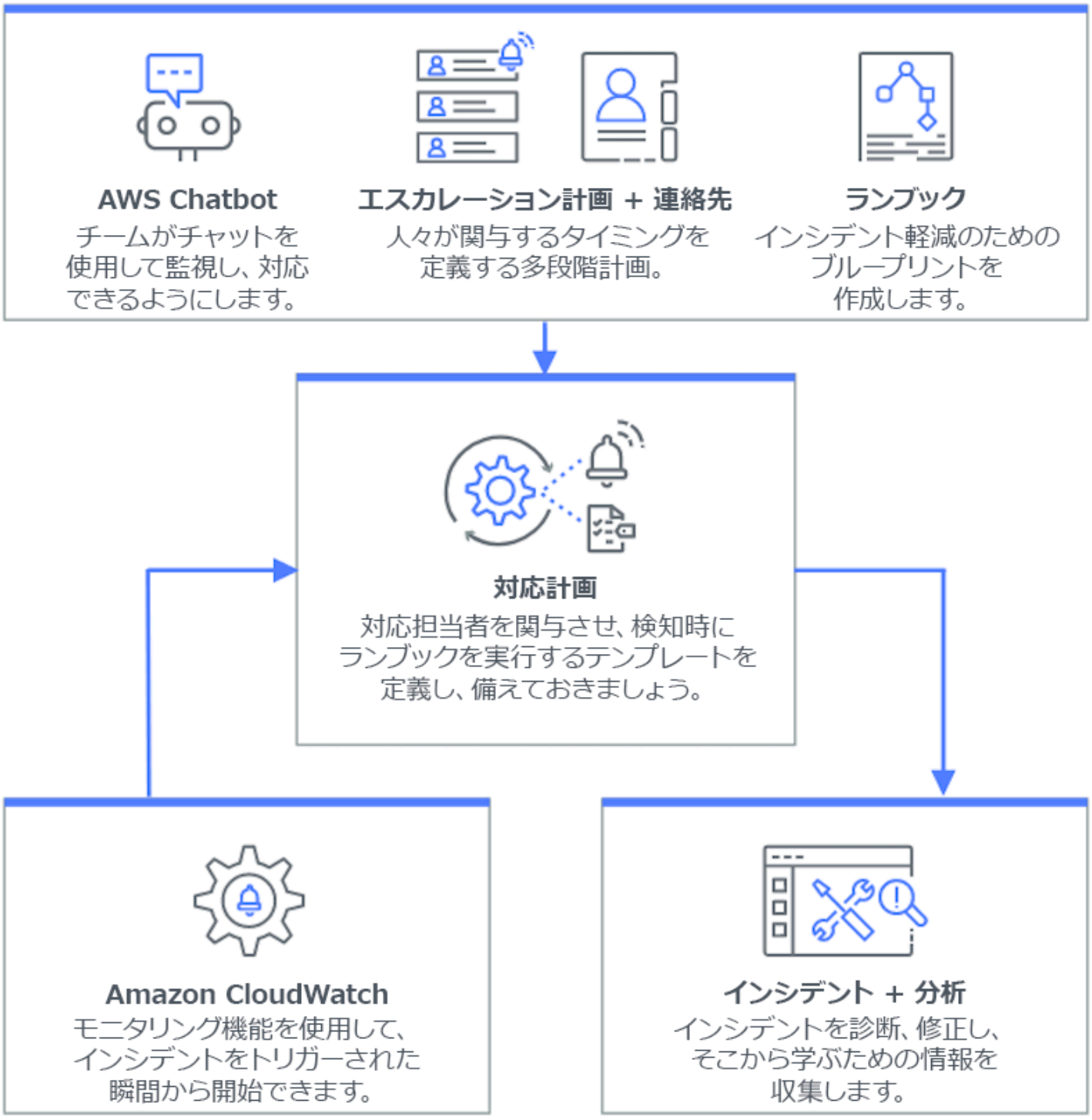
このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

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

実装手順

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



1. 複合アラームを使用する: CloudWatch で 複合アラーム を作成し、関連するアラームをグループ化します。ノイズが減り、より有意義な対応が可能になります。
2. Amazon CloudWatch アラームを Incident Manager と統合する: CloudWatch アラームを設定して、 AWS Systems Manager Incident Manager でインシデントを自動的に作成します。

3. Amazon EventBridge を Incident Manager と統合する: 定義済みの対応計画にそってイベントに対応し、インシデントを作成する EventBridge ルール を作成します。
4. Incident Manager でインシデントに備える:
   - アラートの種類ごとに、詳細な 対応計画 を Incident Manager で策定します。
   - チャットチャネルを AWS Chatbot を通じて確立します。このチャネルは Incident Manager の対応計画に接続され、インシデント発生時の Slack 、Microsoft Teams、Amazon Chime などのプラットフォーム間でのリアルタイムコミュニケーションを促進します。
   - Incident Manager 内で Systems Manager オートメーションランブック を統合し、インシデントへの自動対応を実現します。

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Amazon CloudWatch でのアラームの使用
- Setting up AWS Systems Manager Incident Manager
- Preparing for incidents in Incident Manager

関連動画:

- Top incident response tips from AWS

関連する例:

- AWS Workshops - AWS Systems Manager Incident Manager - Automate incident response to security events

OPS10-BP03 ビジネスへの影響に基づいて運用上のイベントの優先度を決定する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

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

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

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

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

3. 優先順位付けのマトリクスを作成する:

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

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

緊急性と影響	高	中	低
高	重要	緊急	高
中	緊急	高	通常
低	高	通常	低

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

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Atlassian - インシデントの重大度レベルの把握
- IT Process Map - Checklist Incident Priority

OPS10-BP04 エスカレーション経路を決定する

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

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

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

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

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS08-BP04 実践的なアラートを作成する

- OPS10-BP02 アラートごとにプロセスを用意する
- OPS11-BP02 インシデント後の分析を実行する

関連するドキュメント:

- AWS Systems Manager Incident Manager Escalation Plans
- Working with on-call schedules in Incident Manager
- Creating and Managing Runbooks
- Temporary elevated access management with AWS IAM Identity Center
- Atlassian - 効果的なインシデント管理のためのエスカレーション ポリシー

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

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

期待される成果:

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

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

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

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

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

- OPS07-BP03 ランブックを使用して手順を実行する
- OPS10-BP06 ダッシュボードでステータスを知らせる

- OPS11-BP02 インシデント後の分析を実行する

関連するドキュメント:

- Atlassian - インシデント コミュニケーションのベスト プラクティス
- Atlassian - How to write a good status update
- PagerDuty - A Guide to Incident Communications

関連動画:

- Atlassian - Create your own incident communication plan: Incident templates

関連する例:

- AWS Health Dashboard
- AWS のステータスの最新情報の例

OPS10-BP06 ダッシュボードでステータスを知らせる

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

期待される成果:

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

一般的なアンチパターン:

- インシデント管理の会議に参加するエンジニアが、最新状況を把握するために、状況確認のリクエストをしなければならない。

- 管理面は手作業による報告に頼っているため、遅延が起きたり正確さを欠いたりする可能性がある。
- インシデント発生時に、運用チームが最新の状況確認のために頻繁に中断される。

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

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

- ダッシュボードに重要なメトリクスのしきい値と目標を示す注釈を付け、すぐに視認できるようにします。

4. データソースを統合する:
   - さまざまな AWS サービスからのメトリクスを集約して表示するため Amazon CloudWatch を使用し、さらに 他のデータソースからのメトリクスを照会して、システムの状態とビジネスのメトリクスをまとめたビューを作成できます。
   - さまざまなアプリケーションやサービスからログデータをクエリして視覚化するため、CloudWatch Logs Insights のような機能を使用します。
5. セルフサービスアクセスを可能にする:
   - 関連するステークホルダーと CloudWatch ダッシュボードを共有し、セルフサービスで情報にアクセスできるようにします。これには、 ダッシュボード共有機能を使用します。
   - ダッシュボードに簡単にアクセスできるようにし、リアルタイムで最新情報が提供されるようにします。
6. 定期的に更新して改良する:
   - 進化するビジネスニーズとステークホルダーのフィードバックに応じて、ダッシュボードを継続的に更新し、改良していきます。
   - ダッシュボードを定期的に見直し、必要な情報を伝えるために適切かつ効果的であり続けるようにします。

リソース

関連するベストプラクティス:

- OPS08-BP05 ダッシュボードを作成する

関連するドキュメント:

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

関連する例:

- One Observability Workshop - ダッシュボード

OPS10-BP07 イベントへの対応を自動化する

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

期待される成果:

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

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

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

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

実装手順

1. 自動化の機会を見極める: 問題の修正、チケットの強化、容量管理、スケーリング、デプロイ、テストなど、自動化の余地がある反復的なタスクを判断します。

2. 自動化のプロンプトを特定する:
    - 自動応答の引き金となる特定の条件やメトリクスを Amazon CloudWatch アラームアクションを使用して評価し、定義します。
    - Amazon EventBridge を使用して AWS サービス、カスタムワークロード、SaaS アプリケーションでイベントに対応します。
    - AWS リソースで 特定のログエントリ、パフォーマンスメトリクスのしきい値、 状態の変更 などの開始イベントを検討します。
3. イベント駆動型の自動化を実装する:
    - AWS Systems Manager オートメーションランブックを使用して、メンテナンス、デプロイ、修正のタスクを簡素化します。
    - Incident Manager でインシデントを作成 して、関係する AWS リソースの詳細を自動的に収集し、インシデントに追加します。
    - AWS での クォータモニタを使用してクォータをプロアクティブにモニタリングします。
    - 可用性とパフォーマンスを維持するため、 AWS Auto Scaling で容量を自動的に調節します。
    - 開発パイプラインを Amazon CodeCatalystを使用して自動化します。
    - エンドポイントと API のスモークテストまたは継続的な監視に 合成モニタリングを使用します。
4. 自動化によるリスク軽減を実行する:
    - リスクに迅速に対処するため、 自動化されたセキュリティ対応 を実装します。
    - 設定ドリフトを軽減するため、 AWS Systems Manager ステートマネージャー を使用ます。
    - AWS Config ルール で非準拠のリソースを修復します。

実装計画に必要な工数レベル: 高

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Using Systems Manager Automation runbooks with Incident Manager

- Creating incidents in Incident Manager
- AWS Service Quotas
- Monitor resource usage and send notifications when approaching quotas
- AWS Auto Scaling
- What is Amazon CodeCatalyst?
- Amazon CloudWatch でのアラームの使用
- Amazon CloudWatch アラームアクションの使用
- Remediating Noncompliant Resources with AWS Config ルール
- フィルターを使用したログイベントからのメトリクスの作成
- AWS Systems Manager ステートマネージャー

関連動画:

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

関連する例:

- Amazon CodeCatalyst Tutorial: Creating a project with the Modern three-tier web application blueprint
- One Observability ワークショップ
- Respond to incidents using Incident Manager

進化

進歩とは、時間をかけて改善を繰り返す連続的なサイクルです。運用アクティビティから学んだ教訓に基づいて、頻繁に生じる小さな増分変更を実装し、改善された場合の成功を評価します。

経時的に運用を進化させるには、以下のことを実行する必要があります。

トピック

- 学習、共有、改善

学習、共有、改善

定期的に運用活動の分析、失敗の分析、実験、改善のための時間を用意することが不可欠です。失敗した場合には、チームだけでなく、より大規模なエンジニアリングコミュニティでも、それらの失敗から学習できるようにする必要があります。失敗を分析して、教訓を特定し、改善を計画する必要があります。学んだ教訓を定期的に他のチームと共に見直し、インサイトを検証する必要があります。

ベストプラクティス

- OPS11-BP01 継続的改善のプロセスを用意する
- OPS11-BP02 インシデント後の分析を実行する
- OPS11-BP03 フィードバックループを実装する
- OPS11-BP04 ナレッジ管理を実施する
- OPS11-BP05 改善の推進要因を定義する
- OPS11-BP06 インサイトを検証する
- OPS11-BP07 オペレーションメトリクスのレビューを実行する
- OPS11-BP08 教訓を文書化して共有する
- OPS11-BP09 改善を行うための時間を割り当てる

OPS11-BP01 継続的改善のプロセスを用意する

ワークロードを社内外のアーキテクチャのベストプラクティスに対して評価します。頻繁かつ意図的なワークロードレビューを実施します。ソフトウェア開発サイクルの中で改善の機会を優先事項にします。

期待される成果:

- アーキテクチャのベストプラクティスに対してワークロードを頻繁に分析します。
- ソフトウェア開発プロセスにおいて、新機能の開発と改善の機会に同程度の優先順位を与えます。

一般的なアンチパターン:

- ワークロードを数年前にデプロイして以来、アーキテクチャレビューを実施していない。
- 改善の機会の優先順位が低い。新機能と比較して、これらの機会は未処理のままである。
- 組織のベストプラクティスに対する変更の実装について基準がない。

このベストプラクティスを活用するメリット:

- ワークロードがアーキテクチャのベストプラクティスに準拠した最新の状態に保たれます。
- ワークロードを意図を持って進化させることができます。
- 組織のベストプラクティスを活用して、すべてのワークロードを改善できます。
- わずかなメリットが累積的な影響をもたらし、効率性の向上につながります。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ワークロードの構造的レビューを頻繁に実施します。社内外のベストプラクティスを使用してワークロードを評価し、改善の機会を特定します。ソフトウェア開発サイクルの中で改善の機会を優先事項にします。

実装手順

1. 合意された頻度で、本稼働ワークロードの定期的なアーキテクチャレビューを実施します。AWS 固有のベストプラクティスを含む文書化された構造基準を使用します。
   a. これらのレビューには、社内で定義された基準を使用します。社内基準がない場合は、AWS Well-Architected フレームワークを使用します。
   b. AWS Well-Architected Tool を使用して社内ベストプラクティスのカスタムレンズを作成し、アーキテクチャレビューを実施します。
   c. AWS ソリューションアーキテクトまたはテクニカルアカウントマネージャーに連絡して、ワークロードのガイド付き Well-Architected Framework レビューを実施します。
2. レビュー中に特定された改善機会を、ソフトウェア開発プロセスの中で優先事項に設定します。

実装計画に必要な工数レベル: 低。AWS Well-Architected フレームワークを使用して年次のアーキテクチャレビューを実施できます。

リソース

関連するベストプラクティス:

- OPS11-BP02 インシデント後の分析を実行する
- OPS11-BP08 教訓を文書化して共有する
- OPS04 オブザーバビリティを実装する

関連するドキュメント:

- AWS Well-Architected Tool - カスタムレンズ
- AWS Well-Architected ホワイトペーパー - レビュープロセス
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization

関連動画:

- Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool
- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

関連する例:

- AWS Well-Architected Tool

OPS11-BP02 インシデント後の分析を実行する

顧客に影響を与えるイベントを確認し、寄与する要因と予防措置を特定します。この情報を使用して、再発を制限または回避するための緩和策を開発します。迅速で効果的な対応のための手順を開発します。対象者に合わせて調整された、寄与因子と是正措置を必要に応じて伝えます。

期待される成果:

- インシデント後の分析を含むインシデント管理プロセスが確立されます。
- イベントに関するデータを収集するためのオブザーバビリティ計画が整います。

- このデータから、インシデント後の分析プロセスを支えるメトリクスを理解し、収集できます。
- インシデントから学び、その後の成果の向上につなげることができます。

一般的なアンチパターン:

- アプリケーションサーバーを管理しています。約 23 時間 55 分ごとに、すべてのアクティブなセッションが終了します。あなたは、アプリケーションサーバーで何が問題なのかを特定しようとしました。あなたは、これがネットワークの問題である可能性があることを疑っていますが、ネットワークチームが忙しすぎてサポートを提供できないため、当該チームから協力を得ることができません。あなたには、サポートを得て、何が起こっているかを判断するために必要な情報を収集するための事前定義されたプロセスがありません。
- あなたは、ワークロード内でデータを失ってしまいました。このような問題が発生したのはこれが最初であり、原因は明らかではありません。あなたは、データを再作成できるため、これが重要ではないと判断しています。データ損失は、顧客に影響するほどの高い頻度で発生し始めます。また、これにより、失われたデータの復元に際して、追加の運用上の負担も発生します。

このベストプラクティスを活用するメリット:

- インシデントの原因となったコンポーネント、条件、アクション、イベントを決定する事前定義されたプロセスを持つことで、改善の機会を把握できます。
- インシデント後の分析のデータを改善に役立てます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

プロセスを使用して、寄与した要因を判断します。顧客に影響を与えるすべてのインシデントを確認します。インシデントに寄与した要因を特定してドキュメント化するためのプロセスを用意しておき、再発を抑制または防止する緩和策と、迅速で効果的な対応手順を展開できるようにしておきます。インシデントの根本原因を適宜伝達し、伝える相手に合わせて伝え方を調整します。教訓を組織内で広く共有します。

実装手順

1. デプロイの変更、構成変更、インシデントの開始時刻、アラーム時刻、エンゲージメント時間、緩和開始時刻、インシデント解決時刻などのメトリクスを収集します。
2. タイムライン上で重要な時点を特定し、インシデントの該当時点のイベントを把握します。

3. 次の質問について検討します。
    a. 検出までの時間を短縮できますか?
    b. メトリクスとアラームについて、インシデントの検出を早めるための改善点はありますか?
    c. 診断までの時間を短縮できますか?
    d. 対応計画やエスカレーション計画について、適切な対応担当者をより早く関与させるための改善点はありますか?
    e. 緩和までの時間を短縮できますか?
    f. ランブックやプレイブックに追加または改善できる手順はありますか?
    g. 今後のインシデントの発生を防止できますか?
4. チェックリストとアクションを作成します。すべてのアクションを追跡し、実行します。

実装計画に必要な工数レベル: 中

リソース

関連するベストプラクティス:

- OPS11-BP01 継続的改善のプロセスを用意する
- OPS 4 - オブザーバビリティを実装する

関連するドキュメント:

- Performing a post-incident analysis in Incident Manager
- Operational Readiness Review

OPS11-BP03 フィードバックループを実装する

フィードバックループは、意思決定を推進するための実行可能なインサイトを提供します。フィードバックループを手順やワークロードに組み込みます。そうすることで、問題および改善すべき領域を特定することができます。またフィードバックループは、改善への投資を検証することもできます。これらのフィードバックループは、ワークロードの継続的な改善の基盤となります。

フィードバックループは、 即時フィードバック および 遡及分析の 2 つのカテゴリに分類されます。即時フィードバックは、オペレーションアクティビティのパフォーマンスと結果のレビューをとおして収集されます。このフィードバックは、チームメンバー、顧客、またはアクティビティの自動出

力から得られます。即時フィードバックは A/B テストや新機能のリリースなどからも得ることができ、フェイルファストにおいて不可欠なものです。

遡及分析は定期的に実行され、オペレーションの結果とメトリクスの長期間にわたるレビューからフィードバックを取得します。これらの遡及分析は、スプリント、サイクル、またはメジャーリリースやイベントの完了時に行われます。このタイプのフィードバックループは、オペレーションまたはワークロードへの投資を検証でき、成果と戦略の計測に役立ちます。

期待される成果: 即時フィードバックと遡及分析を使用して、改善を推進します。ユーザーやチームメンバーからのフィードバックを取得する仕組みがあります。遡及分析を使用して、改善を推進する傾向を特定します。

一般的なアンチパターン:

- 新しい機能をローンチしたが、顧客からのフィードバックを得る方法はない。
- オペレーションの改善に投資した後、遡及分析を行って投資を検証していない。
- 顧客からのフィードバックを収集しているが、定期的にレビューしていない。
- フィードバックループに基づいて提案されたアクション項目があるが、それらはソフトウェア開発プロセスに含まれていない。
- 顧客からの改善提案に対するフィードバックを行っていない。

このベストプラクティスを活用するメリット:

- 顧客の視点から新しい機能を推進することができる。
- 組織の文化をより迅速に変化させることができる。
- 傾向をレビューすることで、改善の機会を特定できる。
- 遡及分析によって、ワークロードやオペレーションへの投資を検証できる。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

このベストプラクティスを採用すると、即時フィードバックと遡及分析の両方を使用することになります。これらのフィードバックループによって改善を推進します。即時フィードバックには、調査、顧客へのアンケート、フィードバックフォーラムなど、さまざまな仕組みがあります。また組織は、遡及分析を使用して改善の機会を特定し、取り組みを検証できます。

顧客の事例

AnyCompany Retail は、顧客がフィードバックを投稿したり、問題を報告したりすることができるウェブフォーラムを作成しました。週次会議では、ソフトウェア開発チームがユーザーからのフィードバックを評価します。プラットフォームの改善方針の決定のために、フィードバックは定期的に使用されます。各スプリントの完了時に遡及分析を実施して、改善する項目を特定します。

実装手順

1. 即時フィードバック
   - 顧客やチームメンバーからフィードバックを得るための仕組みが必要です。また、オペレーションアクティビティを構成して、自動的にフィードバックを受信することもできます。
   - 組織にはフィードバックをレビューし、改善点を決定して、改善のスケジュールを策定するプロセスが必要です。
   - フィードバックはソフトウェア開発プロセスに追加する必要があります。
   - 改善を進めるとともに、改善の提案者にフォローアップのフィードバックを行います。
     - AWS Systems Manager OpsCenterを使用して、 これらの改善を OpsItems として作成し 追跡できます。
2. 遡及分析
   - 開発サイクル、定められたサイクル、またはメジャーリリースの完了時に遡及分析を実施します。
   - ワークロードの関係者を集めて、遡及分析会議を行います。
   - ホワイトボードまたはスプレッドシートに、停止、開始、維持の 3 つの列を作成します。
     - 停止は、 チームの活動を停止する項目を指します。
     - 開始は、 アイデアへの取り組みを開始する項目を指します。
     - 維持は、 取り組みを維持する項目を指します。
   - 会議室内の関係者からフィードバックを収集します。
   - フィードバックに優先順位を付けます。アクションと関係者を開始項目または維持項目に割り当てます。
   - アクションをソフトウェア開発プロセスに追加し、改善を進めながら更新されたステータスを関係者に通知します。

実装計画に必要な工数レベル: 中。このベストプラクティスを採用するには、即時フィードバックを収集し分析するプロセスが必要です。また、遡及分析プロセスを確立する必要もあります。

リソース

関連するベストプラクティス:

- OPS01-BP01 顧客のニーズを評価する: フィードバックループは、外部顧客のニーズを収集する仕組みです。
- OPS01-BP02 内部顧客のニーズを評価する: 内部関係者は、フィードバックループを使用して、ニーズや要件を伝えることができます。
- OPS11-BP02 インシデント後の分析を実行する: 事後分析は、インシデント後に実施される重要な遡及分析の 1 つです。
- OPS11-BP07 オペレーションメトリクスのレビューを実行する: オペレーションメトリクスレビューでは、傾向および改善の領域を特定します。

関連するドキュメント:

- CCOE を構築するときに回避すべき 7 つの落し穴
- Atlassian チームプレイブック - 振り返り
- E メールの定義: フィードバックループ
- AWS Well-Architected フレームワークレビューに基づくフィードバックループの確立
- IBM ガレージメソドロジー - 振り返りの保留
- Investopedia - PDCS サイクル
- 開発者の有効性を最大化する (Tim Cochran 著)
- 運用準備状況の確認 (ORR) に関するホワイトペーパー - イテレーション
- TIL CSI - 継続的なサービスの改善
- トヨタでの e コマースの採用: Amazon での無駄のない管理

関連動画:

- 効果的な顧客フィードバックループの構築

関連サンプル:

- Astuto - オープンソースの顧客フィードバックツール
- AWS ソリューション - AWS の QnABot

- Fider - 顧客フィードバックの管理プラットフォーム

関連サービス:

- これらの改善を

OPS11-BP04 ナレッジ管理を実施する

ナレッジ管理は、チームメンバーが業務を遂行するために情報を検索する際に役立ちます。従業員の学びが促進される組織では、個人を支援する情報が自由に共有されています。情報は探索したり検索したりできます。情報は正確かつ最新の内容です。新しい情報を作成し、既存の情報を更新し、古い情報をアーカイブするメカニズムが存在します。ナレッジ管理プラットフォームの最も一般的な例は、wiki などのコンテンツ管理システムです。

期待される成果:

- チームメンバーはタイムリーで正確な情報にアクセスできます。
- 情報は検索できます。
- 情報を追加、更新、アーカイブするメカニズムが導入されています。

一般的なアンチパターン:

- 一元化されたナレッジストレージがありません。チームメンバーは、個人のローカルマシンで自分用のメモを管理しています。
- 組織でホストする Wiki はあっても、情報を管理するメカニズムがないため、情報が古くなっています。
- 不足する情報が特定されても、チームの wiki にその情報の追加を要請するプロセスがありません。チームが独自に情報を追加しても、重要なステップを見逃してしまい、使用停止につながります。

このベストプラクティスを活用するメリット:

- 情報が自由に共有されるため、チームメンバーに支援が行き届きます。
- ドキュメントは最新の内容で検索可能であるため、新しいチームメンバーのオンボーディングがより迅速になります。
- 情報はタイムリーな内容で正確かつ実用的です。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ナレッジ管理は、従業員の学びが促進される組織の重要な側面です。まず、ナレッジを保存する中央リポジトリが必要です (一般的な例には、自己ホスト型の wiki があります)。ナレッジを追加、更新、アーカイブするためのプロセスを開発する必要があります。文書化すべき対象の基準を策定して、全チームメンバーが貢献できるプロセスを導入します。

お客様事例

AnyCompany Retail では、社内 Wiki をホストして、すべてのナレッジを保存しています。チームメンバーには、日常業務を遂行する際にナレッジベースに情報を追加することが推奨されています。四半期ごとに、部門横断的なチームが、更新が最も少ないページを評価し、アーカイブまたは更新する必要があるかを判断しています。

実装手順

1. まず、ナレッジを保存するコンテンツ管理システムを特定します。組織全体にわたるステークホルダーからの賛同を得ます。
   a. 既存のコンテンツ管理システムがない場合は、自己ホスト型の wiki を導入するか、バージョン管理リポジトリの導入から始めるかを検討します。
2. 情報を追加、更新、アーカイブするためのランブックを作成します。チームにこのプロセスについての教育を提供します。
3. コンテンツ管理システムに保存すべきナレッジを特定します。チームメンバーが実行する日常業務のアクティビティ (ランブックとプレイブック) から始めます。ステークホルダーと協力して、追加するナレッジに優先順位を付けます。
4. ステークホルダーと協力し、定期的に古い情報を特定し、アーカイブするか、最新の状態に更新します。

実装計画に必要な工数レベル: 中。既存のコンテンツ管理システムがない場合は、自己ホスト型の wiki またはバージョン管理されたドキュメントリポジトリを設定することができます。

リソース

関連するベストプラクティス:

- OPS11-BP08 教訓を文書化して共有する - ナレッジ管理を行うと、学んだ教訓の情報共有が容易になります。

関連するドキュメント:

- Atlassian - ナレッジマネジメント

関連する例:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 改善の推進要因を定義する

データとフィードバックループに基づいて機会を評価して優先順位を設定できるよう、改善の推進要因を特定します。システムやプロセスの改善機会を探り、適切な場合は自動化します。

期待される成果:

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- 改善の基準を決定することで、イベントベースのモチベーションや感情的投資の影響を最小限に抑えることができます。
- 技術的なイベントだけでなく、ビジネスイベントにも対応できます。
- 環境を測定して、改善すべき領域を特定します。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

- 改善の推進要因を理解する: システムに変更を加えるのは、望まれている成果がサポートされているときだけにしてください。
  - 望まれている機能: 改善の機会を評価する際は、望まれている機能を評価してください。
    - AWS の最新情報
  - 許容できない問題: 改善の機会を評価する際は、許容できない問題、バグ、脆弱性を評価してください。適切なサイジングオプションを追跡し、最適化の機会を探します。
    - AWS セキュリティ速報
    - AWS Trusted Advisor
    - Cloud Intelligence Dashboards
  - コンプライアンスの要件: 改善の機会を確認する際は、規制/ポリシー遵守の維持、またはサードパーティによるサポートの維持に必要な更新と変更を評価します。
    - AWS コンプライアンス
    - AWS コンプライアンスプログラム
    - AWS コンプライアンスの最新情報

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Amazon Athena
- Amazon QuickSight
- AWS コンプライアンス
- AWS コンプライアンスの最新情報
- AWS コンプライアンスプログラム
- AWS Glue
- AWS セキュリティ速報
- AWS Trusted Advisor
- ログデータを Amazon S3 にエクスポートする
- AWS の最新情報
- 顧客中心のイノベーションの必要性
- Digital Transformation: Hype or a Strategic Necessity?

関連動画

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

OPS11-BP06 インサイトを検証する

分析結果を確認してクロスな役割を持つチームやビジネスオーナーで応答します。これらのレビューに基づいて共通の理解を確立し、追加的な影響を特定するとともに、一連のアクションを決定します。必要に応じて対応を調整してください。

期待される成果:

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

一般的なアンチパターン:

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

このベストプラクティスを活用するメリット:

- ビジネスオーナーや各分野のエキスパートとインサイトを検証することで、共通の理解を確立し、より効果的に改善につなげることができます。
- 隠れた問題を発見し、それを将来の意思決定に取り入れることができます。
- 技術的な成果からビジネスの成果にフォーカスを移します。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

- インサイトを検証する: ビジネスオーナーや各分野のエキスパートと協力して、収集したデータの意味について共通の理解と合意があることを確認します。追加の懸念事項や潜在的な影響を特定し、一連のアクションを判断します。

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Designing a Cloud Center of Excellence (CCOE)

関連動画:

- Building observability to increase resiliency

OPS11-BP07 オペレーションメトリクスのレビューを実行する

ビジネスのさまざまな分野のチームメンバー間でオペレーションメトリクスの遡及分析を定期的に実施します。これらのレビューに基づいて、改善の機会と取り得る一連のアクションを特定するとともに、教訓を共有します。すべての環境 (開発、テスト、本番など) で改善する機会を探します。

期待される成果:

- ビジネスに影響するメトリクスを頻繁に確認する
- オブザーバビリティ機能を通じて異常を検出し確認する
- データをビジネスの成果と目標の裏付けに使用する

一般的なアンチパターン:

- 大規模な販促活動によってメンテナンスウィンドウが中断されます。ビジネスに影響する他のイベントがある場合、標準メンテナンスウィンドウが延期される可能性があることが認識されていません。
- 組織で古いライブラリを頻繁に使用していたため、長い時間システムが停止しました。その後、サポートされているライブラリに移行しました。組織内の他のチームは、自身がリスクにさらされているかはわかっていません。
- 顧客の SLA の達成状況を定期的に確認していません。顧客の SLA に適合しない傾向があります。顧客の SLA に適合しない場合は、金銭的ペナルティが発生します。

このベストプラクティスを活用するメリット:

- 運用メトリクス、イベント、インシデントを定期的に確認することで、チーム間の共通理解を維持します。

- チームは定期的にミーティングを行い、メトリクスやインシデントを確認します。これにより、リスクに対処し、顧客の SLA を確認できます。
- 学んだ教訓を共有することで、ビジネス成果の優先順位付けや目標とする改善のためのデータが得られます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

- ビジネスのさまざまな分野のチームメンバー間で運用メトリクスの遡及分析を定期的に実施します。
- ビジネス、開発、オペレーションチームを含むステークホルダーを参加させて、即時フィードバックと遡及分析から得られた結果を検証し、教訓を共有します。
- それらのインサイトに基づいて、改善の機会と取り得る一連のアクションを特定します。

リソース

関連するベストプラクティス:

- OPS08-BP05 ダッシュボードを作成する
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け
- OPS10-BP01 イベント、インシデント、問題管理のプロセスを使用する

関連するドキュメント:

- Amazon CloudWatch
- Amazon CloudWatch のメトリクスとディメンションのリファレンス
- カスタムメトリクスをパブリッシュする
- Amazon CloudWatch メトリクスを使用する
- Dashboards and visualizations with CloudWatch

OPS11-BP08 教訓を文書化して共有する

運用アクティビティから学んだ教訓を文書化して共有し、社内とチーム全体で利用できるようにします。チームが学んだことを共有して、組織全体のメリットを増やす必要があります。情報とリソース

を共有して、回避可能なエラーを防止し、開発作業を容易にして、期待される機能の提供にフォーカスします。

AWS Identity and Access Management (IAM) を使用して、アカウント内またはアカウント間で共有するリソースへのコントロールされたアクセスを可能にするアクセス許可を定義します。

期待される成果:

- バージョン管理されたリポジトリを使用して、アプリケーションライブラリ、スクリプト化された手順、手順のドキュメント、その他のシステムドキュメントを共有します。
- インフラストラクチャ標準は、バージョン管理された AWS CloudFormation テンプレートとして共有します。
- チーム全体で学んだ教訓を確認します。

一般的なアンチパターン:

- 組織でバグが含まれているライブラリを頻繁に使用していたため、長い時間システムが停止しました。その後、チームは信頼性の高いライブラリに移行しました。組織内の他のチームは、自身がリスクにさらされているかはわかっていません。このライブラリでの経験が文書化や共有されていないため、誰もリスクに気づいていません。
- あるユーザーが、セッションがドロップする原因となる内部共有マイクロサービスのエッジケースを特定しました。そのユーザーは、このエッジケースを回避するために、サービスへの自分の呼び出しを更新しました。組織内の他のチームは、自身がリスクにさらされているかはわかっていません。
- マイクロサービスの 1 つについて、CPU 使用率要件を大幅に削減する方法が見つかりました。他のチームがこの手法を利用できるかどうかはわかりません。

このベストプラクティスを活用するメリット: 教訓を共有して改善をサポートし、経験から得られるメリットを最大化します。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 教訓を文書化して共有する: 運用アクティビティと遡及分析の実行から学習した教訓を文書化する手順を決めて、ほかのチームが使用できるようにします。

- 教訓を共有する: 教訓と関連するアーティファクトをチーム全体で共有する手順を決めます。例えば、アクセス可能な Wiki を使用して手順の更新、ガイダンス、ガバナンス、ベストプラクティスを共有します。共通のリポジトリを使用してスクリプト、コード、ライブラリを共有します。
  - AWS 環境へのアクセスの委任
  - AWS CodeCommit リポジトリを共有する

リソース

関連するベストプラクティス:

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

関連するドキュメント:

- Reduce project delays with a docs-as-code solution

関連動画:

- AWS 環境へのアクセスの委任
- AWS Supports You | Exploring the Incident Management Tabletop Exercise

OPS11-BP09 改善を行うための時間を割り当てる

漸進的な継続的改善を可能にする時間とリソースをプロセス内に設けます。

期待される成果:

- 一時的に重複する環境を作成することで、実験やテストのリスク、労力、コストを削減できます。
- こうした重複する環境を使用して、分析、実験からの結論をテストし、計画した改善を開発してテストできます。
- ゲームデーを実施し、Fault Injection Service (FIS) を使用して、チームが本番環境に似た環境で実験を行うために必要な制御とガードレールを提供します。

一般的なアンチパターン:

- アプリケーションサーバーに既知のパフォーマンスの問題があります。当該問題は、すべての計画された機能実装の背後にあるバックログに追加されます。計画された機能が一定の割合で追加され続ければ、パフォーマンスの問題は解決しません。
- 継続的な改善をサポートするために、管理者と開発者が改善の選択と実装にすべての余分な時間を費やすことを承認します。改善は完了しません。
- 運用上の承認が完了した後は、運用プラクティスの再テストを行っていません。

このベストプラクティスを活用するメリット: 時間とリソースをプロセス内に設けることで、漸進的な継続的改善が可能となります。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

- 改善を行うための時間を割り当てる: 継続的な漸進的改善のために、プロセス内に時間とリソースを割り当てます。
- 改善のための変更を加えて結果を評価し、成功を判断します。
- 結果が目標に達しておらず、今後も改善が優先事項である場合は、アクションの代替案を検討します。
- ゲームデーを通して本番環境のワークロードをシミュレートし、これらのシミュレーションから学んだことを改善に生かします。

リソース

関連するベストプラクティス:

- OPS05-BP08 複数の環境を使用する

関連動画:

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

まとめ

運用上の優秀性は、継続的かつ反復的な取り組みです。

目標を共有することで、組織が成功するように設定します。ビジネス成果を達成する上での役割と、成功のための他者の能力にどのような影響を与えるかを全員に理解してもらいます。チームメンバーがビジネス成果をサポートできるように、チームメンバーにサポートを提供します。

すべての運用イベントや失敗は、アーキテクチャの運用を改善する機会として扱う必要があります。ワークロードのニーズを理解し、日常的な活動のためのランブックを事前定義し、問題解決の指針となるプレイブックを作成し、運用をAWSのコード機能として使用し、状況認識を維持することで運用の準備がより整い、インシデントが発生しても、より効率的に対応できるようになります。

変化する優先順位に基づく段階的な改善と、イベント対応や遡及的分析から学んだ教訓に焦点を当てることで、活動の効率と効果を高め、ビジネスを成功させることが可能になります。

AWS は、応答性と適応性の高いデプロイを構築し、効率を最大化するアーキテクチャの構築と運用を支援できるよう努めています。ワークロードの運用上の優秀性を高めるには、このホワイトペーパーで説明されているベストプラクティスを使用する必要があります。

寄稿者

- Amazon Web Services、Well-Architected 部門、Operational Excellence Pillar Lead、Rich Boyd
- Amazon Web Services、Well-Architected 部門、Solutions Architect、Jon Steele
- <!--ATMS sidestep. Remove this-->Amazon Web Services、Sr. Technical Program Manager、Ryan King
- Amazon Web Services、Advisory Consultant、Chris Kunselman
- Amazon Web Services、Advisory Consultant、Peter Mullen
- (<!--ATMS sidestep. Remove this-->)Amazon Web Services、Sr. Advisory Consultant、Brian Quinn
- Amazon Web Services、Cloud Operating Model Lead、David Stanley
- Amazon Web Services、Enterprise Support 部門、Senior Specialist Technical Account Manager、Chris Kozlowski
- Amazon Web Services、Cloud Operations 部門、Principal Specialist Solutions Architect、Alex Livingstone
- Amazon Web Services、Enterprise Support 部門、Principal Technologist、Paul Moran
- Amazon Web Services、Professional Services 部門、Advisory Consultant、Peter Mullen
- Amazon Web Services、Enterprise Support 部門、Senior Specialist Technical Account Manager、Chris Pates
- Amazon Web Services、Enterprise Support 部門、Principal Specialist Technical Account Manager、Arvind Raghunathan
- Amazon Web Services、Senior Cost Lead Solutions Architect、Ben Mergen

その他の資料

追加ガイダンスについては、次の資料を参照してください。

- AWS Well-Architected Framework
- AWS アーキテクチャセンター

改訂履歴

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

変更	説明	日付
ホワイトペーパーの更新	新しい実装ガイダンスを使用してベストプラクティスを更新。	June 27, 2024
メジャーなコンテンツの更新と統合	コンテンツを更新し、複数のベストプラクティス領域に統合。2 つのベストプラクティス領域 (OPS 04 と OPS 08) を書き直し、新しいコンテンツと重要点を追加。 次の領域のベストプラクティスを更新し、統合。 運用のための設計、デプロイのリスクを緩和する、 運用状態の把握。ベストプラクティス領域 OPS 04 を「 オブザーバビリティを実装する」に更新。ベストプラクティス領域 OPS 08 を「 ワークロードのオブザーバビリティの活用」に更新。	October 3, 2023
新しいフレームワークの更新	規範ガイダンスを使用してベストプラクティスを更新、および新しいベストプラクティスを追加。	April 10, 2023

ホワイトペーパーの更新	新しい実装ガイダンスを使用してベストプラクティスを更新。	December 15, 2022
ホワイトペーパーの更新	ベストプラクティスに加筆し、改善計画を追加。	October 20, 2022
マイナーな更新	編集上の微小な修正	August 8, 2022
ホワイトペーパーの更新	新しい AWS のサービスと機能、最新のベストプラクティスを反映する更新。	February 2, 2022
マイナーな更新	イントロダクションに持続可能性の柱を追加。	December 2, 2021
新しいフレームワークの更新	新しい AWS のサービスと機能、最新のベストプラクティスを反映する更新。	July 8, 2020
ホワイトペーパーの更新	新しい AWS のサービスと機能を反映するための更新とリファレンスの更新。	July 1, 2018
初版発行	運用上の優秀性の柱 – AWS Well-Architected フレームワークを公開。	November 1, 2017