運用上の優秀性の柱

AWS Well-Architected Framework

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

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: 2023 年 4 月 10 日 (改訂履歴 (p. 122))

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。

- 運用をコードとして実行する クラウドでは、アプリケーションコードに使用しているものと同じエンジニアリング原理を環境全体に適用できます。ワークロード全体 (アプリケーション、インフラストラクチャなど) をコードとして定義し、コードを使用して更新できます。運用手順をスクリプト化し、イベントに応じてスクリプトをトリガーすることで、実行を自動化できます。運用をコードとして実行することで、人為的なミスを抑制し、イベントへの一貫性のある対応を実現できます。
- 小規模かつ可逆的な変更を頻繁に行う ワークロードへの有益な変更の流れを増やすため、コンポーネントを定期的に更新できるようにワークロードを設計します。環境に導入された問題の特定と解決に役立たない場合に元に戻すことができるように、変更は小規模に行います (可能な場合は、顧客に影響がないようにします)。
- 運用手順を頻繁に改善する 運用手順を実施するときに、改善の機会を探します。ワークロードを改良するときに、手順もそれに応じて改良します。定期的なゲームデーを計画し、すべての手順が効果的であり、チームがその手順を熟知していることを確認および検証します。
- 障害を予想する 障害が起こる可能性を特定して除去または軽減できるように、「プレモータム」演習を実施します。障害シナリオをテストし、その影響に関する理解を検証します。対応手順をテストし、手順が有効であること、チームが手順を十分に理解していることを確認します。定期的にゲームデーを設定して、シミュレートされたイベントに対するワークロードとチームの反応をテストします。
- 運用上の障害すべてから学ぶ 運用上のイベントと障害すべてから教訓を学び、改善を促進します。チーム間と組織全体で 教訓を共有します。

定義

クラウドにおける「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 外部顧客のニーズを評価する

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

一般的なアンチパターン:

- あなたは、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。
- あなたは、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、顧客のままでいる可能性が高くなります。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を活用し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、運用チームを含む関係者全体で目標と理解を共有す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関係者と外部顧客の目標、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

関連するドキュメント:

- AWS Well-Architected Framework の概念 - フィードバックループ

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。

確立された優先順位を使用して、改善の努力を最も影響があるところに集中させます (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)。必要に応じて優先順位を更新します。

一般的なアンチパターン:

- あなたは、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。
- あなたは、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確立するメリット: 社内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、運用チームを含む関係者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関係者と連携し、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

関連するドキュメント:

- AWS Well-Architected Framework Concepts – Feedback loop (AWS Well-Architected Framework の概念 - フィードバックループ)

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 (p. 8) - 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-BP05 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。

それらの Well-Architected フレームワーク は、学習、測定、改善を重視しています。アーキテクチャを評価し、時間の経過とともにスケールアップ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 は AWS Well-Architected Tool を提供しており、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 をご利用のお客様は、AWS のベストプラクティスと照らし合わせ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をご利用のお客様は、 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に設計された運用レビューの対象となります。

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、優先順位を決定するのに役立ちます。 ビジネスおよびエンタープライズサポートのお客様 は、優先順位をさらに高めることができるセキュリティ、信頼性、パフォーマンス、コストの最適化に重点を置いた追加のチェックにアクセスできます。

一般的なアンチパターン:

- あなたは、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。あなたは、未対応のコンプライアンス要件への対応に優先順位を付けていません。

このベストプラクティスを確立するメリット: 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、どの脅威に対処すべきか、その優先度、およびそれに必要なリソースを判断できます。

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: ミディアム

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、ビジネスリスクと負債、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軽減策、またその優先順位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

OPS01-BP06 トレードオフを評価する

競合する利益または代替アプローチ間のトレードオフの影響を評価し、重点領域を決定するか、一連のアクションを選択する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、新しい機能の市場投入までの時間を短縮することは、コストの最適化よりも重視されることがあります。または、非リレーショナルデータ用にリレーショナルデータベースを選択すれば、データ型に合わせて最適化されたデータベースに移行してアプリケーションを更新するよりも、システムの移行が簡素化されます。

AWS Support 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。チームを教育するには、 AWS Support (AWS ナレッジセンター、 AWS ディスカッションフォーラム、および AWS Support センター) および AWS ドキュメント が提供するリソースを使用する必要があります。AWS に関しての質問に対する支援については、AWS Support センターを利用して AWS Support に連絡してください。

また、AWS の運用を通じて学んだベストプラクティスとパターンを Amazon Builders' Library で読み、学ぶことができます。その他のさまざまな有益な情報は、 AWS ブログ および 公式の AWS ポッドキャストで入手できます。

一般的なアンチパターン:

- あなたは、リレーショナルデータベースを使用して、時系列と非リレーショナルデータを管理しています。使用しているデータ型をサポートするように最適化されたデータベースオプションがありますが、あなたはソリューション間のトレードオフを評価していないため、そのメリットを認識していません。
- 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。あなたは、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。あなたの投資家は、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、あなたの会社に対する支援を停止します。

このベストプラクティスを活用するメリット: 選択した影響と結果を理解することで、選択肢に優先順位を付けることができます。

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: ミディアム

実装のガイダンス

- トレードオフの評価: 競合する利益間のトレードオフの影響を評価し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、新しい機能を市場に出す速度を上げることが、コストの最適化より重視されることがあります。
- AWS Support 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。チームを教育するには、AWS Support (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) および AWS ドキュメントが提供するリソースを使用する必要があります。AWS に関しての質問に対する支援については、AWS Support センターを利用して AWS Support に連絡してください。

- また、AWS は Amazon Builders' Library の AWS の運用を通じて学んだベストプラクティスとパターンも共有しています。AWS ブログと公式の AWS ポッドキャストでは、その他のさまざまな有益な情報を入手できます。

リソース

関連するドキュメント:

- AWS ブログ
- AWS クラウド コンプライアンス
- AWS ディスカッションフォーラム
- AWS ドキュメント
- AWS ナレッジセンター
- AWS Support
- AWS Support センター
- Amazon Builders' Library
- 公式の AWS ポッドキャストで入手できます

OPS01-BP07 メリットとリスクを管理する

メリットとリスクを管理し、重点領域を決定する際に十分な情報に基づいて意思決定を下せるようにします。たとえば、重要な新機能を顧客に公開できるように、未解決の問題を記録するワークロードをデプロイしておくと便利な場合があります。関連するリスクを軽減できる場合もあれば、リスクが残るのを容認できない場合もあります。その場合、リスクに対処するための措置を講じることになります。

ある時点で、優先順位の小さなサブセットに注力したい場合に遭遇するかもしれません。必要な機能の開発とリスクの管理を確実にするために、長期的にバランスのとれたアプローチを使用します。必要に応じて優先順位を更新します。

一般的なアンチパターン:

- あなたは、開発者の 1 人が「インターネットで見つけた」「あなたが必要とするすべてのこと」を行うライブラリを含めることにしました。あなたは、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。
- あなたは、既存の問題を修正する代わりに、新しい機能を開発およびデプロイすることに決めました。あなたは、この機能がデプロイされるまで問題をそのままにしておくことのリスクを評価しておらず、顧客への影響がわかりません。
- あなたは、コンプライアンスチームから詳細不明の懸念が寄せられたため、顧客から頻繁にリクエストされる機能をデプロイしないことに決めました。

このベストプラクティスを活用するメリット: 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、十分な情報に基づいて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を活用しない場合のリスクレベル: 低

実装のガイダンス

- メリットとリスクの管理: 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。

- 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、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。
- 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。
- リスクに対するメリットの評価と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。

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。

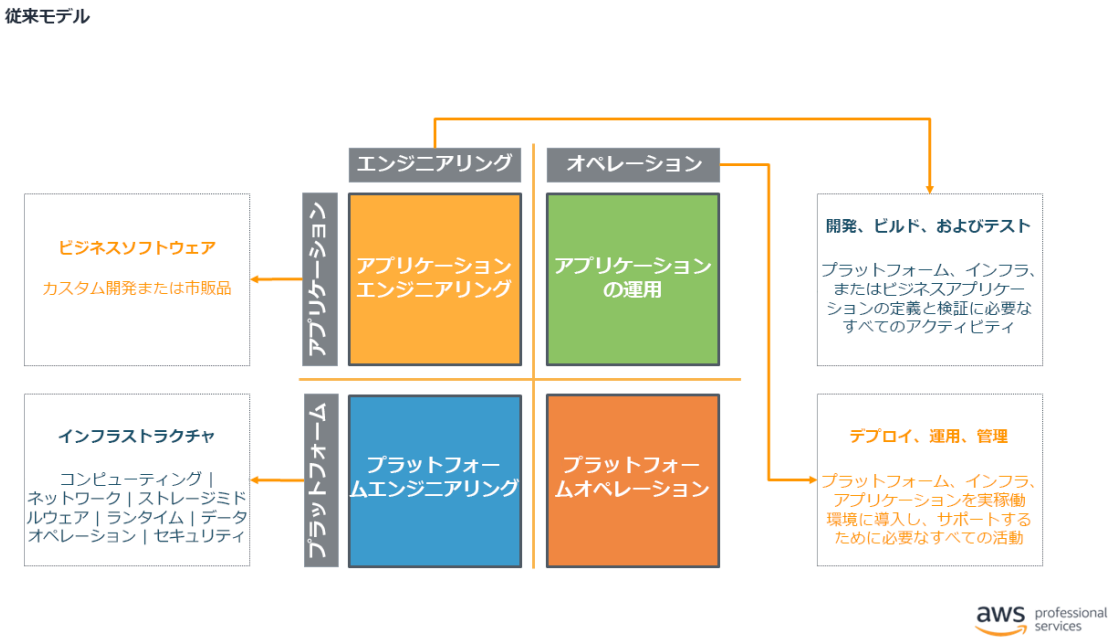
次の 4 つの各運用モデルを使用して成功を実現することができます。一部のモデルは、特定のユースケースや、開発における特定のポイントに適しています。これらのモデルによっては、現在の環境で使用しているモデルよりも利点が大きい場合があります。

トピック

完全に分離された運用モデル

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。

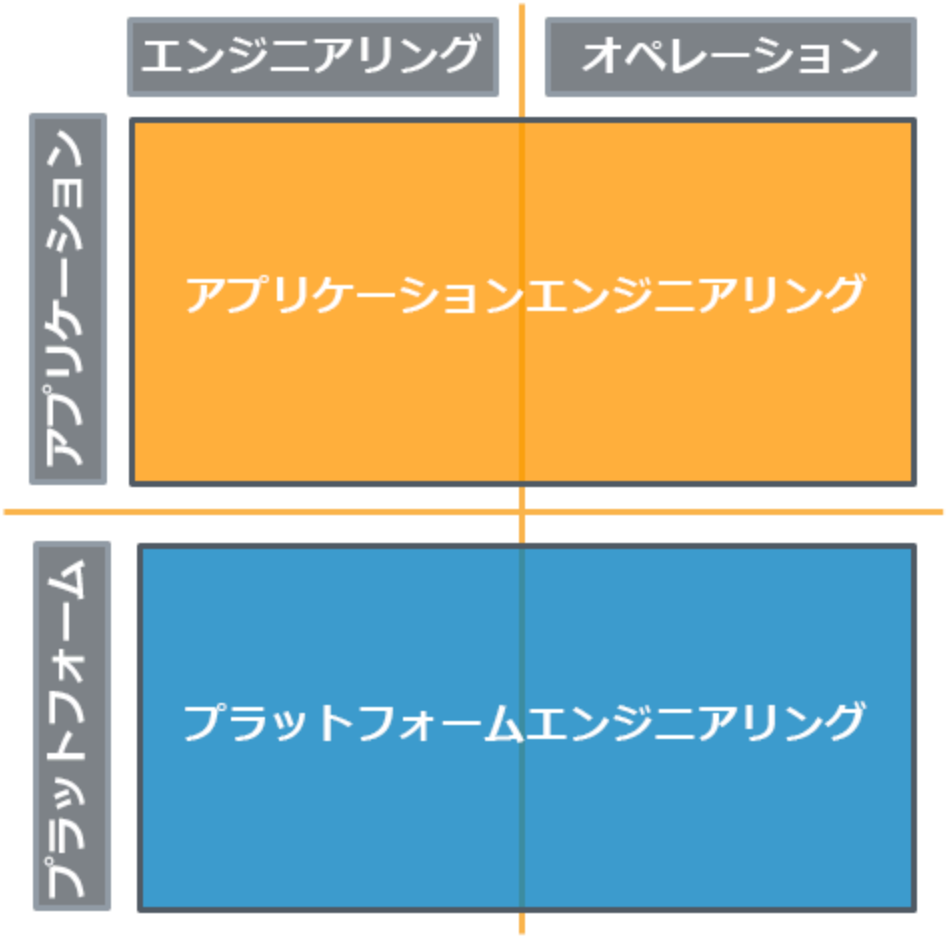
業務チーム、開発チーム、運用チームが、実行されているアクティビティや機能を中心に編成されている場合には、調整不良のリスクが高くなります。これでは、チームはビジネス成果の達成に集中するのでは

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

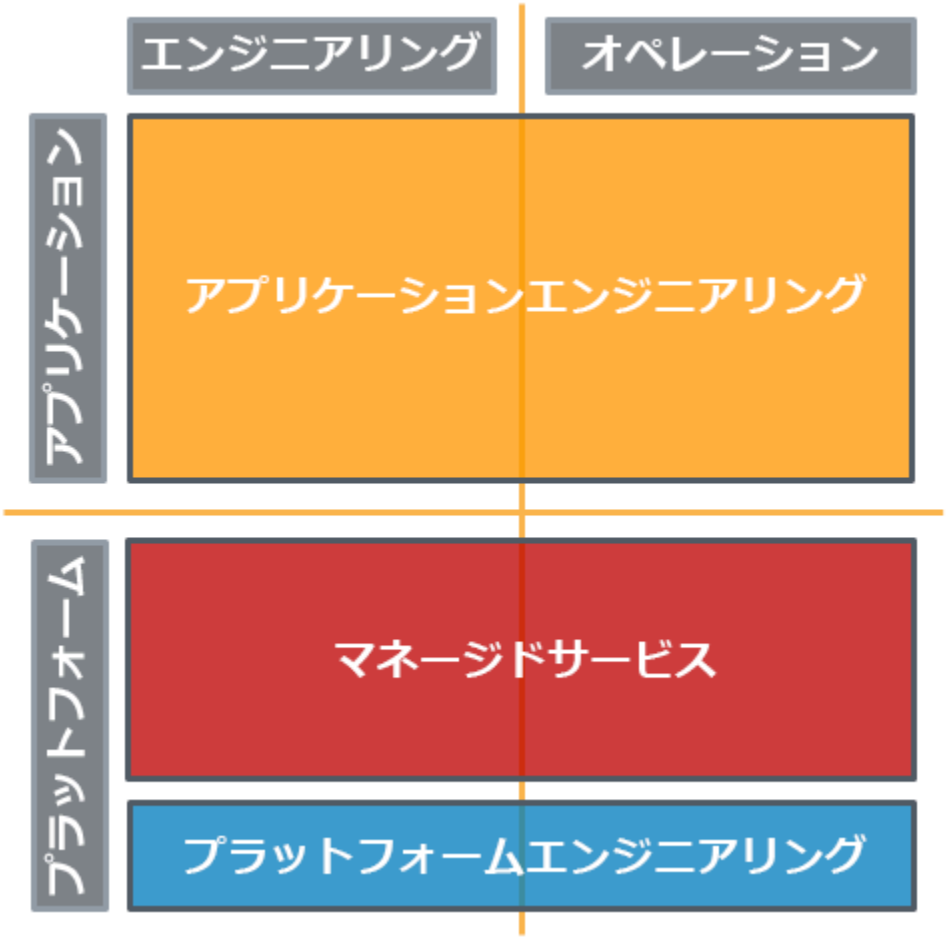
この「分離された AEO と IEO」モデルでは、「自分で構築して実行する」という方法論に従います。

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。

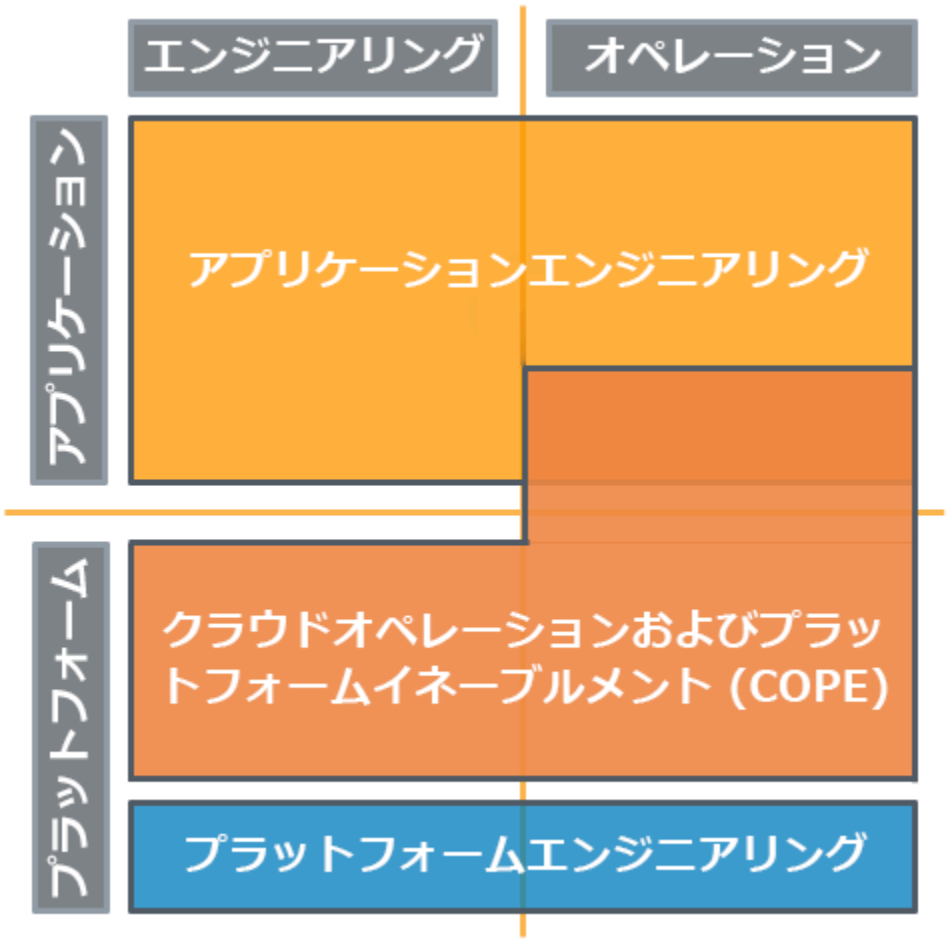
アプリケーションチームが採用できるように、プラットフォームエンジニアリングチームはこれらの標準に基づいたコアとなる共有プラットフォーム機能を構築します。リファレンスアーキテクチャ、およびセルフサービスメカニズムを介してアプリケーションチームに提供されるパターンをコード化します。AWS Service Catalog などのサービスを使用することで、アプリケーションチームは承認済のリファレンスアーキテクチャ、パターン、サービス、構成、一元化されたガバナンスおよびセキュリティ標準による既定のコンプライアンスなどをデプロイできます。

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。

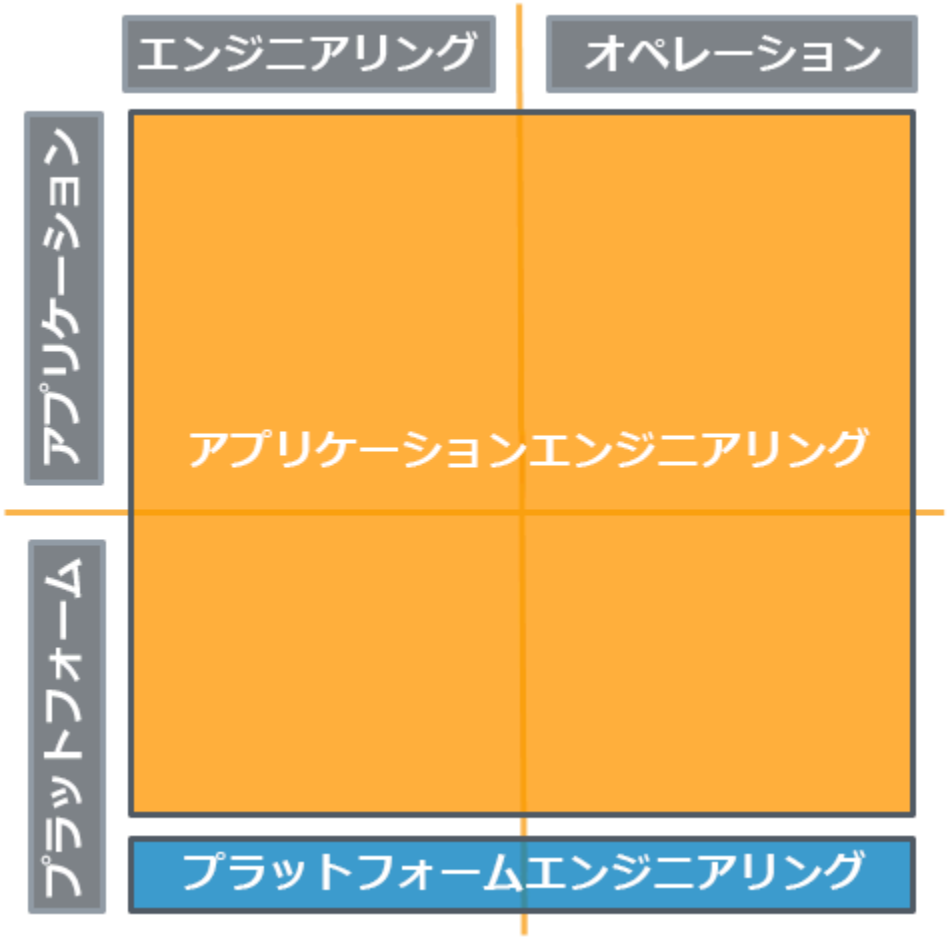
各チームがもたらす組織の標準、ベストプラクティス、プロセス、専門知識による恩恵を受けることができます。新しいチームがクラウドを採用したりモダナイゼーションを行ったりする際に、これらの成功パターンを繰り返せるようなメカニズムを確立します。このモデルでは、COPE チームによるアプリケーションチームの確立、知識とアーティファクトの継承を支援する能力を重視しています。アプリケーションチームによる広範囲の独立性の取得の失敗リスクに対処するために、アプリケーションチームの運用負荷を低減します。CCoE、COPE、アプリケーションチーム間の関係を確立し、進化と革新をさらに進めるためのフィードバックループを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

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

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いなくなったとしても、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。アカウントのその他の連絡先は、個人よりもグループを指定して設定します。
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 Tagging Best Practices whitepaper (AWS のタグ付けに関する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. 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 (p. 23) - リソースをサポートするプロセスと手順はリソースの所有権によって決まります。
- OPS02-BP04 チームメンバーが自らの責任範囲を把握する (p. 24) - チームメンバーは自分がどのリソースの所有者になっているかを理解する必要があります。
- 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 (p. 24) - 所有権はタグやアカウント連絡先などの仕組みを使用して確認できるようにする必要があります。

関連するドキュメント:

- AWS Account Management - Updating contact information (AWS アカウントの管理 - 連絡先情報の更新)
- AWS Config ルール - required-tags
- AWS Organizations - 組織の代替連絡先の更新
- AWS Tagging Best Practices whitepaper (AWS のタグ付けに関するベストプラクティスホワイトペーパー)

関連する例:

- AWS Config ルール - Amazon EC2 と必要なタグおよび有効な値

関連サービス:

- AWS Config
- 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、改善の機会を特定できます。

このベストプラクティスを確立するメリット: 所有権を理解すると、改善の承認者、改善を実装する者、またはその両方がわかります。

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

- プロセスや手順には、その定義に責任を持つ所有者が存在する: お客様の環境で使用されているプロセスや手順、およびその定義に責任を持つ個人またはチームを把握します。
  - プロセスと手順の特定: 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。
  - プロセスまたは手順の定義の所有者の定義: 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を持つ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。
  - 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のキャプチャ: AWS Systems Manager (ドキュメント)、および AWS Lambda (関数) などのサービスで自動化されている手続きは、メタデータ情報をタグとしてキャプチャするのをサポートしています。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がキャプチャされるようにします。

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

定義されたワークロードに対して特定のアクティビティを実行する責任を持つ者と、その責任が存在する理由を理解します。運用アクティビティを実行することに責任を負う者を理解すると、誰がアクティビティを実行し、結果を検証し、アクティビティの所有者にフィードバックを提供するかを通知します。

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負う者を理解すると、アクションが必要になったときに誰に通知し、誰がアクションを実行し、結果を検証し、アクティビティの所有者にフィードバックを提供するかがわかります。

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

- 運用アクティビティのパフォーマンスに責任を持つ所有者の存在: 環境で使用されるプロセスと手順を実行する責任を明確にする
  - プロセスと手順の特定: 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。
  - 各アクティビティを実行する責任者の定義: アクティビティに責任を有するチームを特定します。アクティビティの詳細と、アクティビティを実行するために必要なスキルと適切なアクセス許可、アクセス、ツールがあることを確認します。チームは、当該アクティビティが実行される条件 (イベントやスケジュールなど) を理解する必要があります。この情報を検出可能にして、組織のメンバーが、特定のニーズに合わせ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。

OPS02-BP04 チームメンバーが自らの責任範囲を把握する

役割の責任と、ビジネスの成果にどのように貢献するかを理解することで、タスクの優先順位付けと役割が重要である理由を知ることができます。これにより、チームメンバーはニーズを認識し、適切に対応できます。

このベストプラクティスを活用するメリット: 責任を理解すると、決定する事項、実行するアクション、および適切な所有者に引き渡すべきアクティビティがわかります。

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

- チームメンバーに自らの役割と責任を確実に理解させる: チームメンバーの役割と責任を特定し、そのチームメンバーが自らの役割に期待されている事項を理解できるようにします。この情報を検出可能にして、組織のメンバーが、特定のニーズに合わせ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。

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

個人またはチームが特定されない場合、対処される必要がある事項についての所有権または計画を割り当てる権限を持つ者へのエスカレーションパスが定義されています。

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を理解すると、適切なチームまたはチームメンバーに連絡して、リクエストし、またはタスクを移行することができます。責任や所有権を割り当て、またはニーズに対処する計画を立てる権限を持つ個人を特定することで、不作為のリスクや対応されないままとなるリスクを軽減できます。

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

- 責任と所有権を特定するためのメカニズムの存在: 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。これらのメカニズムにより、メンバーは、特定のニーズについて、連絡すべきチームまたは個人を特定できます。

OPS02-BP06 追加、変更、例外をリクエストするメカニズムが存在する

プロセス、手順、およびリソースの所有者にリクエストを送信できます。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 (p. 21)、OPS02-BP02 プロセスと手順には特定の所有者が存在する (p. 23)、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 (p. 23) 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 (p. 21) - 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 (p. 23) - 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 (p. 23) - 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-BP07 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 (p. 23) - 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 (p. 23) - 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 (p. 28)
- 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 (p. 28)

OPS03-BP01 エグゼクティブスポンサーシップ

シニアリーダーシップは、組織に対する期待を明確に設定し、成功を評価します。シニアリーダーシップは、ベストプラクティスの採用と組織の進化の協賛者、支持者、および推進者です。

このベストプラクティスを確立するメリット: 熱心なリーダーシップ、期待事項の明確な伝達、目標の共有により、チームメンバーは、何が期待されているのかを知ることができます。成功を評価することで、成功に至るまでの障壁を特定でき、支持者や権限を与えられた者の介入を通じて対処できます。

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

- エグゼクティブスポンサーシップ: シニアリーダーシップは、組織に対する期待値を明確に設定し、成功を評価します。シニアリーダーシップは、ベストプラクティスの採用と組織の進化の協賛者、支持者、および推進者です。
  - 期待値の設定: 測定方法を含め、組織の目標を定義し、公開します。
  - 目標の達成の追跡: 目標の段階的な達成度を定期的に測定し、結果を共有して、結果にリスクがある場合に適切なアクションが講じられるようにします。
  - 目標達成に必要なリソースの提供: 新しい情報、目標、責任、ビジネス環境の変化などに基づいて、リソースが適切かどうか、また追加のリソースが必要であるかどうかを定期的に見直します。
  - チームの支援: チームと常に関わり、彼らがどのような状態にあるのか、また彼らに影響を与える外的要因があるのかを理解します。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。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。
  - ベストプラクティスの導入の促進: 定量的な利益をもたらした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。
  - チームの進化の推進: 継続的な改善の文化を生み出します。個人と組織の両者の成長と発展を奨励します。時間の経過とともに段階的な達成を求める長期的なターゲットを提供します。ニーズ、ビジネス目標、ビジネス環境の変化に応じて、これらを補完するために、このビジョンを調整します。

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

ワークロード所有者は、結果にリスクがあるときにチームメンバーが対応できるようにするためのガイダンスと範囲を定義しています。エスカレーションメカニズムは、イベントが定義された範囲外にある場合に対応の方向性を取得するために使用されます。

このベストプラクティスを活用するメリット: 変更を早期にテストして検証することで、コストを最小限に抑えて問題に対処し、顧客への影響を抑えることができます。デプロイ前にテストすることで、エラーの発生を最小限に抑えることができます。

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: 高

実装のガイダンス

- 結果にリスクがあるときにアクションを実行する権限が、チームメンバーに付与されている: 効果的に対応するために必要なスキルを実践するためのアクセス許可、ツール、機会をチームメンバーに提供します。
  - 対応するために必要なスキルを訓練する機会をチームメンバーに提供する: プロセスと手順のテストとトレーニングを安全に実行できる、安全な代替環境を提供します。ゲームデーを実施して、チームメンバーがシミュレートされた安全な環境で現実世界のインシデントに対応する経験を積むことができるようにします。
  - アクションを実行するチームメンバーの権限を定義および認識する: 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具体的に定義します。チームメンバーが、結果にリスクがあるときにアクションを実行する権限を付与されていることを認識します。

OPS03-BP03 エスカレーションが推奨されている

チームメンバーにはメカニズムがあり、結果にリスクがあると思われる場合は、意思決定者や利害関係者に懸念をエスカレートすることが推奨されます。エスカレーションは、リスクを特定し、インシデントの発生を防ぐために、早く、頻繁に実行する必要があります。

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

- 早期かつ頻繁なエスカレーションを奨励する: 早期かつ頻繁なエスカレーションがベストプラクティスであることを組織的に認識します。エスカレーションに理由がないとされる場合があること、そして、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よいことを組織的に認識し、受け入れます。
  - エスカレーションメカニズムの存在: 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。アクションを実行したり、アクションを承認したりする強い権限を持つ人々とその連絡先情報を文書化します。リスクに対処できる人物に当該リスクが引き渡されたとチームメンバーが満足するまで、またはワークロードの運用に関するリスクと責任の所有者に連絡するまで、エスカレーションを続行する必要があります。当該者がワークロードに関するすべての決定を最終的に所有します。エスカレーションには、リスクの性質、ワークロードの重要度、影響を受ける者、影響の内容、緊急性 (予想される影響の時期など) を含める必要があります。
  - エスカレーションする従業員の保護: 無策の意思決定や利害関係者などにエスカレーションする場合にチームメンバーを報復から保護するポリシーを備えます。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。

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

メカニズムが存在し、チームメンバーに既知のリスクや計画されたイベントをタイムリーに通知するために使用されます。アクションが必要かどうか、どのようなアクションが必要かを判断し、タイミングよくそのアクションを実行するために、必要なコンテキスト、詳細、および時間 (可能な場合) が提供されます。たとえば、パッチ適用を迅速に行えるようにソフトウェアの脆弱性を通知したり、サービス中断のリスクを回避するために変更のフリーズを実装できるように計画された販売プロモーションの通知を提供したりします。計画されたイベントは変更カレンダーまたはメンテナンススケジュールに記録できるため、チームメンバーが保留中のアクティビティを特定できます。

期待される成果:

- コミュニケーションでコンテクスト、詳細、目標時間が提供されます。

- チームメンバーが、コミュニケーションに応答して、いつどのように行動すべきかを明確に理解できます。
- 変更カレンダーを活用して、想定される変更についての通知が提供されます。

一般的なアンチパターン:

- 毎週何度も誤検知でアラートが発生する。発生するたびに通知をミュートにしている。
- セキュリティグループの変更を依頼されたが、いつ行うべきかを指示されていない。
- システムがスケールアップしたがアクションは不要という通知を、何度もチャットで受け取る。チャットチャネルを避けていたため、重要な通知を見逃した。
- 本稼働環境に対する変更が、オペレーションチームに通知せずに行われた。変更によりアラートが発報され、チームに対するオンコールが発生した。

このベストプラクティスを活用するメリット:

- 組織がアラートによる疲弊を回避できます。
- チームメンバーは必要なコンテキストと目標を得て行動できます。
- 変更は変更ウィンドウ中に行われるため、リスクを軽減できます。

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

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。誤検知や常にオンのアラートを特定して排除します。変更カレンダーを活用して、アクションが実行されるタイミングや保留中のアクティビティをチームメンバーが把握できるようにします。コミュニケーションによって、必要なコンテクストが提供され、アクションが明確になることを検証します。

お客様事例

AnyCompany Retail は、主なコミュニケーションメディアとしてチャットを使用しています。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。変更カレンダーを使用して本稼働システムに対する大きな変更をスケジュールしています。

実装手順

1. 誤検知のアラートや継続的に発報されるアラートを分析します。それらを排除するか、人による介入が必要なときに発報されるように変更します。アラートが発報される場合は、ランブックまたはプレイブックを指定します。
   a. アラート用のプレイブックやランブックの作成には、AWS Systems Manager ドキュメントを使用できます。
2. 2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 を使用して、組織のメッセージプラットフォーム内でアラートを送信したりイベントに対応したりできます。
3. 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 を使用して、変更を行う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。

4. 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。

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 (p. 74) - 成果がわかっている場合は、ランブックを提供してコミュニケーションを実行可能なものにします。
- OPS07-BP04 プレイブックを使用して問題を調査する (p. 76) - 成果が不明の場合は、プレイブックによってコミュニケーションを実行可能なものにできます。

関連するドキュメント:

- AWS セキュリティ速報
- Open CVE

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 (p. 109) - 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 (p. 109) - 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。チームメンバーが強化している自らのスキルを検証および認識し、業界認証を追求および維持できるように支援します。組織の知識とスキルを持ち、熟練したチームメンバーを失った場合は、クロストレーニングによって知識の伝達を促進し、重大な影響のリスクを緩和します。学習のために専用の時間を割り当てます。

AWS は、 AWS ご利用開始のためのリソースセンター、 AWS ブログ、 AWS オンラインテックトーク、 AWS イベントとオンラインセミナー、 AWS Well-Architected ラボなどのリソースを提供しており、チームを教育するためのガイダンス、事例、詳細なチュートリアルを提供しています。

また、AWS では、 Amazon Builders' Library で AWS の運用を通じて学んだベストプラクティスと パターン、 AWS ブログ や 公式 AWS ポッドキャストを通じて幅広い有益な教材を共有しています。.

AWS が提供する Well-Architected ラボ、 AWS Support (AWS ナレッジセンター、 AWS ディスカッションフォーラム、および AWS Support センター) および AWS ドキュメント などのリソースを活用して、チームの教育に役立ててください。AWS に関しての質問については、AWS Support センターを利用して AWS Support に連絡してください。

AWS トレーニング と認定 では、AWS の基礎に関するセルフペースデジタルコースを通じて無料のトレーニングを提供しています。また、インストラクターが実施するトレーニングに登録して、チームの AWS スキルの開発をさらにサポートすることもできます。

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: ミディアム

実装のガイダンス

- チームメンバーはスキルセットの維持と成長が可能で推奨されている: 新しいテクノロジーを採用し、イノベーションをサポートし、ワークロードのサポートにおける需要と責任の変化に対応するためには、継続的な教育が必要です。
  - 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。下級チームのメンバーが、上級チームのメンバーをメンターとするためにアクセスできるようにしたり、上級チームのメンバーの業務をシャドーイングして、自らの手法やスキルを評価してもらえるようにしたりします。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。
  - チーム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合った計画を立てます。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する
  - 業界認証の追求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および維持するのをサポートします。

リソース

関連するドキュメント:

- AWS 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 ディスカッションフォーラム
- AWS ドキュメント
- 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 ナレッジセンター
- AWS Support
- AWS トレーニング と認定
- AWS Well-Architected ラボ、
- Amazon Builders' Library
- 公式 AWS ポッドキャストを通じて幅広い有益な教材を共有しています。.

OPS03-BP07 チームに適正なリソースを提供する

チームメンバーのキャパシティを維持し、ワークロードのニーズを満たすツールとリソースを提供します。チームメンバーに過剰な負荷がかかることは、人為的ミスに起因するインシデントのリスクを高めま

す。ツールやリソースへの投資 (頻繁に実行されるアクティビティのオートメーションなど) によって、チームの有効性を高め、より多くの活動をサポートすることを可能にします。

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: ミディアム

実装のガイダンス

- チームに適正なリソースを提供する: チームの成功、および成功または失敗の要因を確実に把握します。チームに適正なリソースを提供してサポートします。
  - チームのパフォーマンスを理解する: チームによる業務成果の達成度や アセット開発の成果を測定します。時間の経過とともに成果とエラー率の変化を追跡します。チームと協力して、チームに影響する業務に関する課題 (責任の増加、テクノロジーの変化、人員の喪失、サポート対象の顧客の増加など) を理解します。
  - チームのパフォーマンスへの影響を理解する: チームと常に関わり、彼らがどのような状態にあるのか、また彼らに影響を与える外的要因があるのかを理解します。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。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。
  - チームの成功に必要なリソースを提供する: リソースが適切かどうか、追加リソースが必要かどうかを定期的に検証し、チームをサポートするために適切な調整を行います。

OPS03-BP08 チーム内やチーム間でさまざまな意見が推奨され、求められる

組織間の多様性を活用して、複数のユニークな視点を追求します。この視点を使用して、イノベーションを高め、想定に挑み、確証バイアスに傾くリスクを軽減します。チーム内のインクルージョン、多様性、アクセシビリティを向上させ、有益な視点を得ます。

組織文化は、チームメンバーのジョブに対する満足度と定着率に直接影響します。チームメンバーのやる気と能力を引き出して、ビジネスの成功につなげます。

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

- 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。立場の弱いグループの意見に耳を傾けます。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。
  - 役割と責任を拡張する: チームメンバーが通常引き受けることのないであろう役割を引き受ける機会を提供します。その役割から、そして、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。また自分の経験と視点を、やり取りをする新しい役割やチームメンバーにもたらします。視点が増えるにつれて、追加のビジネスの機会が得られたり、改善のための新しい機会が見出されたりすることがあります。チーム内のメンバーに、他のメンバーが通常実行する一般的なタスクを交替で担当してもらい、当該タスクの実行の需要と影響を理解してもらいます。
  - 安全で温かい環境を提供する: 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統制を備えます。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心し、温かい気持ちになると、当事者意識が高まり、生産性が向上する可能性が高くなります。組織がより多様になるほど、顧客を含め、サポートする人々に対する理解が深まります。チームのメンバーが快適で、自由に話し、自分の話を聞いてもらえることを確信すると、自らの貴重な洞察 (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) を共有する可能性が高まります。
  - チームメンバーが完全に参加できるようにする: 従業員がすべての業務関連活動に完全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、それを回避するためのスキルを身に付けています。これらの独自に開発したスキルは、組織に大きなメリットをもたらしま

す。チームメンバーに必要な配慮をしてサポートすることで、貢献から得られるメリットが大きくなります。

準備

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

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

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

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” (p. 39)
- 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” (p. 41)
- the section called “OPS04-BP04 依存関係のテレメトリーを実装する” (p. 43)
- the section called “OPS04-BP05 分散トレースを実装する” (p. 45)

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積極的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

ワークロードのアプリケーションテレメトリーを実装するには、 Amazon CloudWatch および 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 内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 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. ビジネス KPI の定義とモニタリング: ビジネス成果に沿った カスタムメトリクス を確立 します。
4. 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できるよう 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。
5. アプリケーション全体にわたるデータ収集の標準化: アプリケーション全体にわたり、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。
6. データの分析とデータに基づく対処: データ収集と正規化が完了したら、 Amazon CloudWatch をメトリクスとログの分析に、 AWS X-Ray をトレース分析に使用します。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 主要業績評価指標を特定する (p. 37)
- 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 (p. 41)
- OPS04-BP04 依存関係のテレメトリーを実装する (p. 43)
- OPS04-BP05 分散トレースを実装する (p. 45)

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 オブザーバビリティ 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 - Amazon におけるオブザーバビリティのベストプラクティス
- AWS re:Invent 2022 - 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
- 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 (p. 37)
- OPS04-BP02 アプリケーションテレメトリーを実装する (p. 39)
- OPS04-BP04 依存関係のテレメトリーを実装する (p. 43)
- OPS04-BP05 分散トレースを実装する (p. 45)

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、サードパーティー 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:

- 外部依存の見落し: 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、依存関係のあ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: 関係者と協力して、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、サードパーティー 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. Amazon CloudWatch Internet Monitor の活用: Internet Monitor を使用すると、インターネットに関してグローバルなインサイトが得られ、外部依存に影響を及ぼす可能性のある障害や中断の把握につながります。
4. AWS Health Dashboard を使用して常に情報を入手: Health Dashboard を使用すると、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が得られます。
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. 定期的なモニタリング: 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 (p. 37)
- OPS04-BP02 アプリケーションテレメトリーを実装する (p. 39)
- 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 (p. 41)
- OPS04-BP05 分散トレースを実装する (p. 45)

関連するドキュメント:

- AWS Health とは
- Amazon CloudWatch Internet Monitor の使用
- AWS X-Ray デベロッパーガイド
- Amazon DevOps Guru ユーザーガイド

関連動画:

- アプリケーションのパフォーマンスに影響を及ぼすインターネットの問題の可視化
- Amazon DevOps Guru の紹介

関連する例:

- Amazon DevOps Guru を使用した AIOps で運用上のインサイトを得る
- AWS Health Aware

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 (p. 37)
- OPS04-BP02 アプリケーションテレメトリーを実装する (p. 39)
- 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 (p. 41)
- OPS04-BP04 依存関係のテレメトリーを実装する (p. 43)

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 (p. 47)
- OPS05-BP02 変更をテストし、検証する (p. 48)

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 (p. 51)

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ています。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、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を活用し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。

お客様事例

AnyCompany Retail では、継続的インテグレーションパイプラインの一環として、すべてのソフトウェアアーティファクトに対して複数の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。 テスト完了時に通知が必要となるのは誰ですか。

a. AWS デプロイパイプラインリファレンスアーキテクチャは、 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施できる、さまざまな種類のテストの信頼できるリストを提供しています。

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 CodeGuru Reviewer を使用すると、アプリケーションコードをテストして欠陥を検出できます。

b. AWS CodeBuild を使用して、ソフトウェアアーティファクトのテストを実施できます。

c. AWS CodePipeline を使用すると、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 (p. 47)
- OPS05-BP06 設計標準を共有する (p. 55)
- OPS05-BP10 統合とデプロイを完全自動化する (p. 60)

関連するドキュメント:

- テスト駆動の開発アプローチを導入する
- TaskCat および CodePipeline を使用した自動化された AWS CloudFormation テストパイプライン
- オープンソースの SCA、SAST、DAST ツールを使用してエンドツーエンドの AWS DevSecOps CI/CD パイプラインを構築する
- サーバーレスアプリケーションのテスト開始方法
- CI/CD パイプラインをリリースキャプテンとして活用する
- 「AWS における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:

- AWS re:Invent 2020: テスト可能なインフラストラクチャ: AWS での統合テスト
- AWS Summit ANZ 2021 - CDK とテスト駆動開発を活用してテストファースト戦略を促進する
- AWS CDK を使用して Infrastructure as Code をテストする

関連リソース:

- 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
- AWS CodeBuild を使用して GitHub から Node.js アプリケーションの単体テストを実行する
- インフラストラクチャコードのデータ駆動開発に Serverspec を使用する

関連サービス:

- 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 (p. 61)
- OPS06-BP02 デプロイをテストする (p. 63)
- OPS06-BP03 安全なデプロイ戦略を使用する (p. 65)
- OPS06-BP08 テストとロールバックを自動化する (p. 67)

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。

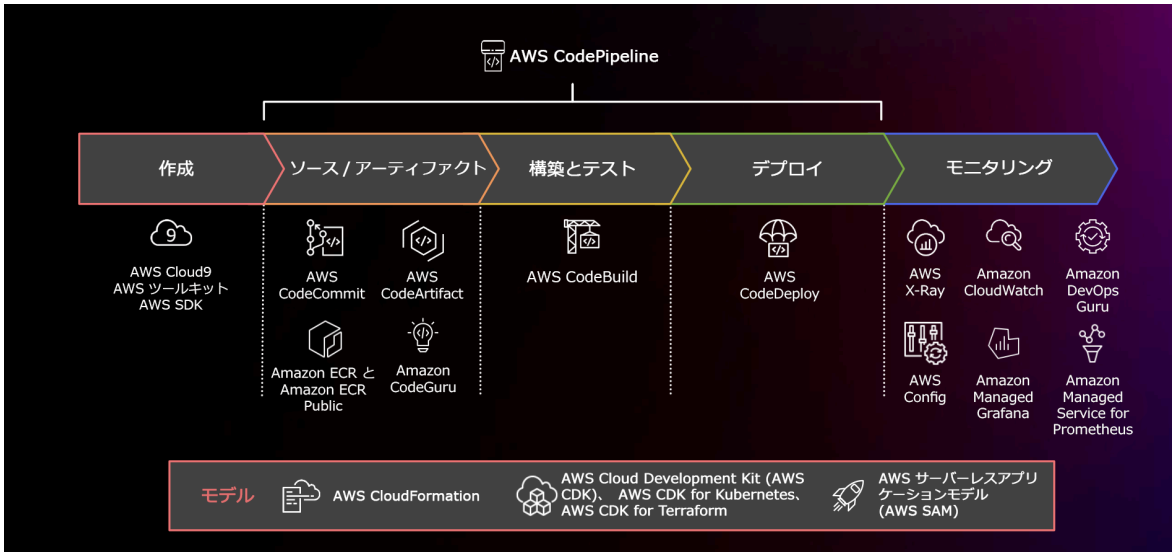
このベストプラクティスを活用するメリット: ビルドとデプロイのアクティビティを管理するメカニズムを提供することで、反復的なタスクを実行するための労力の程度を減らし、チームメンバーは高価値のクリエイティブなタスクに専念し、手動の手順によるエラーの発生を抑制できます。

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-BP08 テストとロールバックを自動化する (p. 67)

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-BP08 テストとロールバックを自動化する (p. 67)

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 (p. 6) - 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 (p. 8) - 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 (p. 71) - 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 (p. 107) - 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 (p. 112) - 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な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は、本稼働環境に発生する問題を最小限に抑えることに役立ちます。ペアプログラミングやコードレビューなどのベストプラクティスを使用すると、コード品質が向上します。

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: 中程度

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。

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 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。
   b. AWS Cloud9 を使用すると、共同開発環境を作成して、コードの開発を共同で進めることができます。

実装計画に必要な工数レベル: 中程度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-BP06 設計標準を共有する (p. 55) - コード品質プラクティスの一環として、設計標準を共有できます。

関連するドキュメント:

- アジャイルソフトウェアガイド
- CI/CD パイプラインをリリースキャプテンとして活用する
- Amazon CodeGuru Reviewer を使用したコードレビューの自動化
- テスト駆動の開発アプローチを導入する
- DevFactory が Amazon CodeGuru を使用してアプリケーション構築を向上
- ペアプログラミングについて
- RENGA Inc. が Amazon CodeGuru を使用してコードレビューを自動化
- The Art of Agile Development: テスト駆動開発
- コードレビューが重要である理由 (そして実際に時間の節約になる理由)

関連動画:

- AWS re:Invent 2020: Amazon CodeGuru を使用したコード品質の継続的改善
- AWS Summit ANZ 2021 - CDK とテスト駆動開発を活用してテストファースト戦略を促進する

関連サービス:

- 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 (p. 50)
- OPS05-BP04 構築およびデプロイ管理システムを使用する (p. 51)
- OPS06-BP08 テストとロールバックを自動化する (p. 67)

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 (p. 50)
- OPS05-BP04 構築およびデプロイ管理システムを使用する (p. 51)

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 (p. 61)
- OPS06-BP02 デプロイをテストする (p. 63)
- OPS06-BP03 安全なデプロイ戦略を使用する (p. 65)
- OPS06-BP08 テストとロールバックを自動化する (p. 67)

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。

期待される結果: あなたは、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。あなたは、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。あなたは、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。
- あなたは Infrastructure as Code を使用しておらず、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てしまいます。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確立するメリット: 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

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

リソース

関連するベストプラクティス:

- OPS06-BP08 テストとロールバックを自動化する (p. 67)

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。

期待される結果: あなたの組織は、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は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繁に発生するため、運用チームは本番リリースについて不安を感じています。
- 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。あなたは、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の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確立するメリット: 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

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を検出します。
   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 CodeDeployAppSpec 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. 状況に応じて、AWS CodeDeploy を他の AWS サービスと統合するか、AWS CodeDeploy をパートナーの製品や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トリガーされます。

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 (p. 48)

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 | 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
- The Story of Apollo - Amazon's Deployment Engine
- How to test and debug AWS CodeDeploy locally before you ship your code
- 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。

期待される結果: 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確立するメリット: 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。

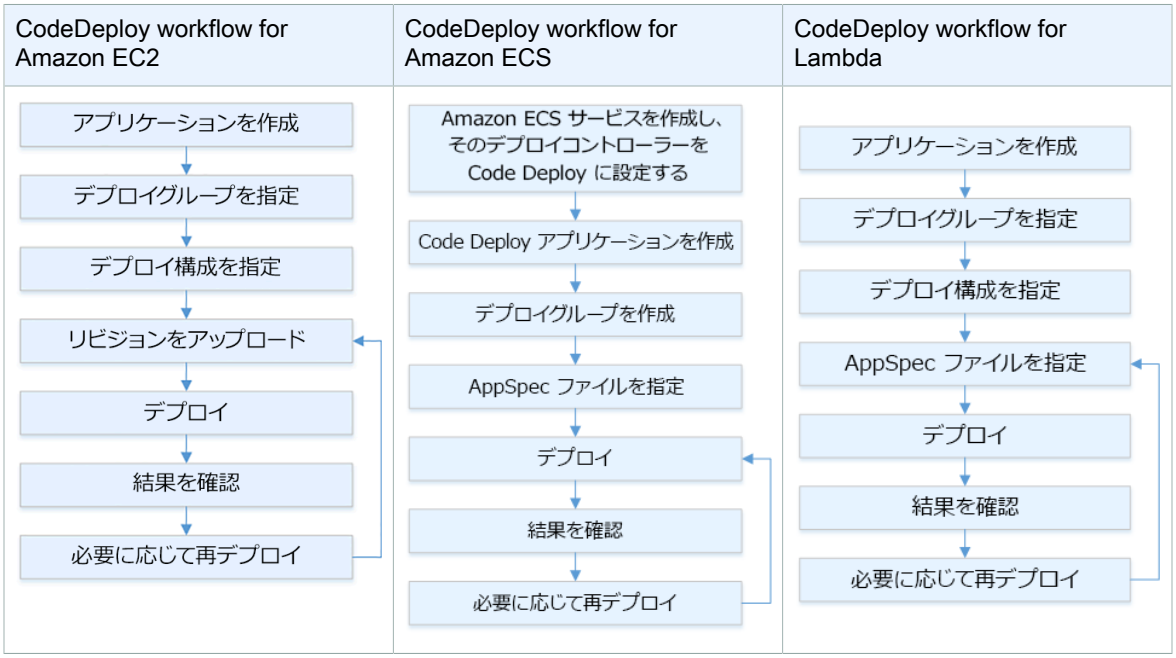
このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

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. 本番環境への移行に対する承認ワークフローによって、一連の本番環境のロールアウト手順がトリガーされます。
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. 状況に応じて、AWS CodeDeploy を他の AWS サービスと統合するか、、AWS CodeDeploy をパートナーの製品やサービスと統合します。
3. Amazon Aurora や Amazon RDS 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. 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。
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 (p. 48)

- OPS05-BP09 小規模かつ可逆的な変更を頻繁に行う (p. 59)
- OPS05-BP10 統合とデプロイを完全自動化する (p. 60)

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 | AWS における継続的インテグレーションと継続的デリバリーの実践 | デプロイ方法
- AWS CodeDeploy ユーザーガイド
- Working with deployment configurations in AWS CodeDeploy
- API Gateway の Canary リリースデプロイの設定
- Amazon ECS デプロイタイプ
- 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:

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- 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-BP08 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。

期待される結果: 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確立するメリット: 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

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

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 (p. 61)
- OPS06-BP02 デプロイをテストする (p. 63)

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 roll back 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

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 (p. 112) - 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 (p. 6) - 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 (p. 8) - 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 (p. 34) - 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 (p. 61) - 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 (p. 70) - 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

A 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。一箇所に保管され、頻繁に更新されます。

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とれなくなる

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を活用しない場合のリスクレベル: ミディアム

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。例えば、 AWS Systems Manager オートメーションなどのサービスを使用すると、フラットなテキストを、ワークロードに対して実行可能なオートメーションに変換できます。これらのオートメーション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。

顧客の事例

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。

```
# ######### ## ####### | ##### ID | ## | ##### | ######### | ######## | ##### | ########
 POC | |-------|-------|-------|-------|-------|-------|-------| | RUN001 | ########## ####
### | ### | ###### | ##### | 2022 # 9 # 21 # | ######### | ## #### 1##### 1 2##### 2
```

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。その際、 `##########` と `######################`.
4. 最初のステップから開始して、ランブックの `####` 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 (p. 23): ランブックには、保守を担当する所有者が必要です。
- OPS07-BP04 プレイブックを使用して問題を調査する (p. 76): ランブックとプレイブックは似ていますが、1 つだけ違うのは、ランブックには期待される成果があることです。多くの場合、プレイブックが根本原因を特定すると、ランブックがトリガーさ れます。
- 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 (p. 97)ランブックは、適切なイベント、インシデント、および問題管理の実践の一部です。
- OPS10-BP02 アラートごとにプロセスを用意する (p. 100): ランブックやプレイブックは、アラートに対応するために使用する必要があります。時間の経過とともに、これらの対応を自動化する必要があります。
- OPS11-BP04 ナレッジ管理を実施する (p. 112): ランブックの保守は、ナレッジマネジメントの重要な一部です。

関連するドキュメント:

- 自動化されたプレイブックとランブックを使用して運用上の優秀性を達成する
- AWS Systems Manager: ランブックの操作
- AWS の大規模移行のための移行プレイブック - タスク 4: 移行ランブックの改良
- AWS Systems Manager Automation ランブックを使用して、運用タスクを解決する

関連動画:

- AWS re:Invent 2019: ランブック、インシデントレポート、インシデント対応の DIY ガイド (SEC318-R1)
- AWS | Amazon Web Services での IT 運用を自動化する方法
- スクリプトを AWS Systems Manager に統合する

関連する例:

- 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
- Jupyter Notebook と CloudTrail Lake を使用して、AWS インシデント対応ランブックを作成する
- Gitlab - ランブック
- Rubix - Jupyter Notebook でランブックを作成するための Python ライブラリ
- Document Builder を使用してカスタムランブックを作成する
- 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

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はない。
- 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は、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。例えば、 AWS Systems Manager Automations などのサービスを使用して、テキストを自動化に変換することができます。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。

顧客は 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、インシデントをとお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。

顧客の事例

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。

```
# ########## ## ######## | ###### ID | ## | ######## | #####
 | ########## | ##### | ######## POC | ### | ############ |
 |-------|-------|-------|-------|-------|-------|-------|-------|-------| | RUN001 | #####
#######? ###############? | ### | ## | ##### | 2022-09-21 | ######### | #### | ##############
#? | ## ## 1.## 1 2## 2
```

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、 ############## プレイブック情報以下の ############。
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 (p. 23): プレイブックには、保守を担当する所有者が必要です。
- OPS07-BP03 ランブックを使用して手順を実行する (p. 74): ランブックとプレイブックは似ていますが、重要な違いが 1 つあり、それはランブックには期待される成果があることです。多くの場合、ランブックは、プレイブックで根本原因を特定したときに使用されます。
- 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 (p. 97): プレイブックは、イベント、インシデント、および問題管理の適切な実践の一部です。
- OPS10-BP02 アラートごとにプロセスを用意する (p. 100): ランブックとプレイブックは、アラートへの応答として使用されます。時間の経過とともに、これらの応答を自動化します。
- OPS11-BP04 ナレッジ管理を実施する (p. 112): プレイブックの保守は、ナレッジマネジメントの重要な一部です。

関連するドキュメント:

- 自動化されたプレイブックとランブックを使用して運用上の優秀性を達成する
- AWS Systems Manager: ランブックの操作
- AWS Systems Manager Automation ランブックを使用して、運用タスクを解決する

関連動画:

- AWS re:Invent 2019: ランブック、インシデントレポート、インシデント対応の DIY ガイド (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS 仮想ワークショップ
- スクリプトを AWS Systems Manager に統合する

関連サンプル:

- AWS カスタムプレイブックフレームワーク
- AWS Systems Manager: オートメーションチュートリアル
- Jupyter Notebook と CloudTrail Lake を使用して、AWS インシデント対応ランブックを作成する
- Rubix - Jupyter Notebook でランブックを作成するための Python ライブラリ
- Document Builder を使用してカスタムランブックを作成する
- 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 を使用して

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 (p. 6) - 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 (p. 61) - 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 (p. 63) - 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 (p. 70) - 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 (p. 23)

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 (p. 83)
- 運用状態の把握 (p. 92)
- イベントへの対応 (p. 96)

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 (p. 83)
- OPS08-BP02 ワークロードログを分析する (p. 85)
- OPS08-BP03 ワークロードのトレースを分析する (p. 86)
- OPS08-BP04 実践的なアラートを作成する (p. 88)
- OPS08-BP05 ダッシュボードを作成する (p. 90)

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 (p. 37)

- OPS04-BP02 アプリケーションテレメトリーを実装する (p. 39)

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 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. CloudWatch Logs Insights の設定: CloudWatch Logs Insights を利用して、 ログデータをインタラクティブに検索して分析します。
   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。
3. Contributor Insights の活用 CloudWatch Contributor Insights を使用して、 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。
4. CloudWatch Logs メトリクススフィルターの実装: CloudWatch Logs のメトリクスフィルターを設定して、 ログデータを実践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。
5. 定期的なレビューと改善: 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 (p. 37)
- OPS04-BP02 アプリケーションテレメトリーを実装する (p. 39)
- OPS08-BP01 ワークロードメトリクスを分析する (p. 83)

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 Logs ログのメトリクススフィルターの作成と管理

関連動画:

- CloudWatch Logsインサイトを使用してログデータを分析する
- CloudWatch Contributor Insights を使用して高カーディナリティデータを分析する

関連する例:

- CloudWatch Logs サンプルクエリ
- One Observability ワークショップ

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

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ています。

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の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 から 取得できるメトリクスを 詳細に確認します。
3. ServiceLens の使用: ServiceLens マップを活用すると、 サービスやアプリケーションのオブザーバビリティを向上できます。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 のトレース内の自動異常検出を有効にします。
   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 を使用すると、トレースデータを徹底的に調査してパターンを特定し、インサイトを抽出できます。
6. X-Ray 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 の統合: Amazon DevOps Guru を採用すると、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できます。
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、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口グとの関連付け: トレースデータ を X-Ray トレースビュー内の関連口グと関連付けると、アプリケーションの動作を詳細に把握できます。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口グイベントを確認できます。

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:

- OPS08-BP01 ワークロードメトリクスを分析する (p. 83)
- OPS08-BP02 ワークロードログを分析する (p. 85)

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ーリング
- 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:

- Amazon CloudWatch Synthetics と AWS X-Ray を使用してアプリケーションを分析してデバッグを行う
- AWS X-Ray Insights を使用する

関連する例:

- One Observability ワークショップ
- X-Ray を AWS Lambda と実装する
- 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を活用し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) を定義します。 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:
   - AWS Cost Anomaly Detection の使用: AWS Cost Anomaly Detection を 異常なパターンを自動的に検出し、正当な異常に対してのみアラートが生成されるように設定します。
   - X-Ray Insights の使用:
     a. X-Ray Insights を トレースデータの異常を検出するように設定します。
     b. 問題が検出された場合に X-Ray Insights にアラートを送信する ように通知を設定します。
   - 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. https://docs.aws.amazon.com/devops-guru/latest/userguide/update-notifications.html#navigate-to-notification-settings 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. 実践的なアラートの実装: すぐに行動に移せるように、適切な情報を提供するアラートを設計します。
4. アラーム疲労の軽減: 重大ではないアラートは最小限に抑えます。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 の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などのツールと統合します。
7. AWS Chatbot との連携: AWS Chatbotと統合して、Chime、Microsoft Teams、Slack にアラートを中継します。
8. ログに基づくアラート: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/MonitoringLogData.html CloudWatch ログの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くアラームを作成します。
9. レビューと反復: アラート設定を定期的に見直して調整します。

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 (p. 37)
- OPS04-BP02 アプリケーションテレメトリーを実装する (p. 39)
- 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 (p. 41)

- OPS04-BP04 依存関係のテレメトリーを実装する (p. 43)
- OPS04-BP05 分散トレースを実装する (p. 45)
- OPS08-BP01 ワークロードメトリクスを分析する (p. 83)
- OPS08-BP02 ワークロードログを分析する (p. 85)
- OPS08-BP03 ワークロードのトレースを分析する (p. 86)

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 の通知
- 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 統合ガイド | PagerDuty
- OpsGenie を Amazon CloudWatch と統合する

関連動画:

- Amazon CloudWatch で複合アラームを作成する
- AWS Chatbot の概要
- AWS on Air ft.AWS Chatbot の変異型コマンド

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 ワークショップ

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関係者に提供できます。

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を活用するメリット:

- 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

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、幅広い関係者のエンゲージメントを向上します。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、単一のビューでリソースのモニタリングができます。

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て、識別しやすい名前を付けます。
2. Markdown ウィジェットの使用: メトリクスの詳細に取り掛かる前に、 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の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: 必要に応じてダッシュボード変数を組み込み、 動的で柔軟なダッシュボードビューを提供します。
4. メトリクスウィジェットの作成: メトリクス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のヘルスとビジネス成果を効果的に表示します。
5. ログのインサイトのクエリ: CloudWatch Logs Insights を利用して、 ログから実践的なメトリクスを導き出し、その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 アラームを 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、リソースの最大の原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されない特定のニーズについて、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. 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 (p. 37)
- OPS08-BP01 ワークロードメトリクスを分析する (p. 83)
- OPS08-BP02 ワークロードログを分析する (p. 85)
- OPS08-BP03 ワークロードのトレースを分析する (p. 86)
- OPS08-BP04 実践的なアラートを作成する (p. 88)

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:

- クロスアカウントとクロスリージョンの CloudWatch ダッシュボードを作成する
- AWS re:Invent 2021 - AWS クラウド オペレーションダッシュボードを使用してエンタープライズレベルの可視化を実現する

関連する例:

- One Observability ワークショップ
- Amazon CloudWatch を使用したアプリケーションモニタリング

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 (p. 92)
- 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 (p. 94)
- OPS09-BP03 運用メトリクスのレビューと改善の優先順位付け (p. 95)

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 (p. 97)
- OPS10-BP02 アラートごとにプロセスを用意する (p. 100)
- 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 (p. 100)

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

組織には、イベント、インシデント、問題を扱うためのプロセスがあります。イベント は、ワークロードで発生しますが、必ずしも介入を必要としない出来事です。インシデント は、介入を必要とするイベントです。 問題 は、介入しなければ解決できない、繰り返し発生するイベントです。これらのイベントがビジネスに与える影響を軽減し、適切に対応できるようにするためのプロセスが必要です。

ワークロードにインシデントや問題が発生したとき、それらを処理するためのプロセスが必要です。イベントの状況を関係者にどのように伝えますか。 対応をだれが監督しますか。 イベントの緩和に使用するツールは何ですか。 これらは、確実な対応策を講じるために必要な質問の一例です。

プロセスは一元的に文書化し、ワークロードに関わる誰もが利用できるようにする必要があります。もし、一元的な Wiki や ドキュメントの保管場所がない場合は、バージョン管理リポジトリを使用することができます。プロセスの進化につれて、これらのプランを最新に保ちます。

問題は、オートメーションの候補です。これらのイベントが発生すると、イノベーションにかける時間が奪われます。問題を軽減するための繰り返し可能なプロセスから始めましょう。時間をかけて、軽減策のオートメーション化または根本的な問題の解決に注力します。そうすることで、ワークロードの改善に充てる時間を確保することができます。

期待される成果: 組織には、イベント、インシデント、問題を扱うためのプロセスがあります。これらのプロセスは文書化され、一元的に保管されます。プロセスの変化につれて更新されます。

一般的なアンチパターン:

- インシデントが週末に発生すると、オンコールエンジニアはどうしてよいかわかりません。
- 顧客からアプリケーションがダウンしたという E メールが送られてきます。修正するためにサーバーを再起動します。これが頻繁に起きます。
- 複数のチームが解決のために個別に取り組んでいるインシデントがあります。
- ワークロードで、記録されることなくデプロイが行われます。

このベストプラクティスを活用するメリット:

- ワークロードのイベントの監査証跡ができます。
- インシデントからの復旧時間が短縮されます。
- チームメンバーは一貫した方法でインシデントと問題を解決できます。
- インシデントを調査するときに、より総合的に取り組むことができます。

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: 高

実装のガイダンス

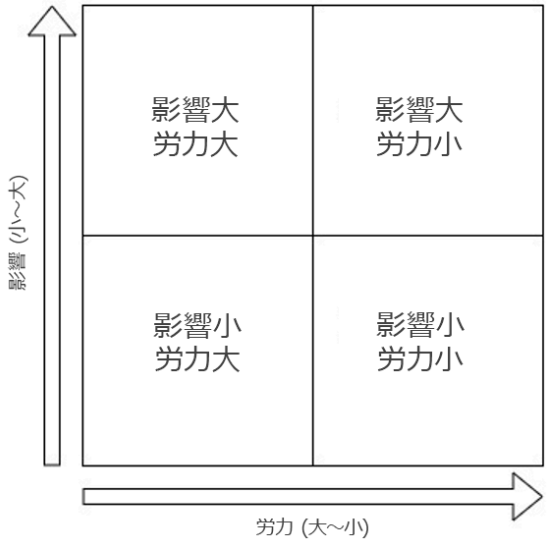
このベストプラクティスを実装すると、ワークロードイベントを追跡することになります。インシデントと問題を扱うためのプロセスができます。プロセスは文書化され、共有され、頻繁に更新されます。問題が特定され、優先順位が付けられ、修正されます。

顧客の事例

AnyCompany Retail では、社内 Wiki の一部がイベント、インシデント、問題管理のためのプロセス専用になっています。 すべてのイベントは以下に送信されます : Amazon EventBridge.問題は AWS Systems Manager OpsCenter で OpsItem として特定され、優先的に修正されるため、未分化な労力を削減することができます。プロセスが変化すると、社内 Wiki で更新されます。同社では、 AWS Systems Manager Incident Manager を使用してインシデントを管理し、緩和の取り組みを調整しています。

実装手順

1. イベント
   - 人間の介入を必要としない場合でも、ワークロードで発生するイベントを追跡します。
   - ワークロードの関係者と協力して、追跡すべきイベントのリストを作成します。例えば、デプロイの完了やパッチ適用の成功などです。
   - また、 Amazon EventBridge または Amazon Simple Notification Service などのサービスを使用して、追跡するカスタムイベントを生成することができます。
2. インシデント
   - インシデント発生時の情報伝達プランを明確にすることから始めましょう。どのような関係者に通知する必要がありますか。 どのようにして情報を共有しますか。 誰が調整作業を監督しますか。 コミュニケーションと調整のために、社内チャットチャネルを立ち上げることをお勧めします。
   - 特にオンコールのローテーションがないチームの場合は、ワークロードをサポートするチームのエスカレーションパスを定義しておきましょう。サポートレベルに基づいて、AWS Support に申請を行うことも可能です。
   - インシデントを調査するためのプレイブックを作成します。これには、情報伝達プランや詳細な調査手順が含まれている必要があります。調査には、 AWS Health Dashboard の確認を含めます。
   - インシデント対応計画を文書化します。インシデント管理計画を伝達し、社内外の顧客がエンゲージメントのルールと何を期待されているのかを理解できるようにします。使用方法について、チームメンバーをトレーニングします。
   - 顧客は Incident Manager を使用してインシデント対応プランを設定し、管理できます。
   - 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。
3. 問題
   - ITSM システムで問題を特定して、追跡する必要があります。
   - 既知の問題をすべて特定し、修正作業とワークロードへの影響から優先順位を付けます。



- 影響が大きく、労力が少なくて済む問題から解決します。そのような問題が解決したら、影響が少なく、労力が少なくて済む象限の問題に移ります。
- 専用のインフラストラクチャで Systems Manager OpsCenter を使用して、これらの問題を特定し、ランブックをアタッチして、追跡します。

実装計画に必要な工数レベル: 中程度。このベストプラクティスを実装するには、プロセスとツールの両方が必要です。プロセスを文書化し、ワークロードに関わる誰もが使用できるようにします。頻繁に更新します。問題を管理し、問題を緩和または修正するためのプロセスがあります。

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 (p. 74): 既知の問題には、緩和作業に一貫性を持たせるために、関連するランブックが必要です。
- OPS07-BP04 プレイブックを使用して問題を調査する (p. 76): インシデントはプレイブックを使用して調査する必要があります。
- OPS11-BP02 インシデント後の分析を実行する (p. 109): インシデントからの復旧後は、必ず事後分析を実施します。

関連するドキュメント:

- Atlassian - DevOps 時代のインシデント管理
- AWS セキュリティインシデント対応ガイド
- 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:

- AWS re:Invent 2020: 分散組織におけるインシデント管理
- AWS re:Invent 2021: イベント駆動型アーキテクチャによる次世代アプリケーションの構築
- AWS Supports You | インシデント管理の机上演習を試す
- AWS Systems Manager Incident Manager - AWS 仮想ワークショップ
- AWS What's Next ft. Incident Manager | AWS イベント

関連する例:

- AWS 管理とガバナンスツールワークショップ - OpsCenter
- 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
- Amazon EventBridge によるイベント駆動型アプリケーションの構築
- AWS でのイベント駆動型アーキテクチャの構築

関連サービス:

- Amazon EventBridge
- Amazon SNS
- AWS Health Dashboard
- AWS Systems Manager Incident Manager

- AWS Systems Manager OpsCenter

OPS10-BP02 アラートごとにプロセスを用意する

アラートを発生させるイベントすべてに対して具体的な対応策 (ランブックやプレイブック) を定め、所有者を明確に指定しておくようにします。こうすることで、運用上のイベントに対する効果的で迅速な対応が可能になり、アクションの必要なイベントが重要度の低い通知に埋もれてしまうことを避けられます。

一般的なアンチパターン:

- モニタリングシステムは、他のメッセージとともに、承認された接続のストリームを表示します。メッセージの量が非常に大きいため、あなたは、介入を必要とする定期的なエラーメッセージを見逃します。
- あなたは、ウェブサイトがダウンしている旨のアラートを受け取ります。このような状況が発生した場合のプロセスは定義されていません。あなたは、アドホックのアプローチで問題を診断して解決しなければなりません。対応に当たりながらこのプロセスを開発すると、復旧までの時間が長くなります。

このベストプラクティスを確立するメリット: アクションが必要な場合にのみアラートを出すことで、低い価値のアラートによって高い価値のアラートが見過ごされるのを防ぎます。アクション可能なアラートごとにプロセスを設定することで、環境内のイベントに対して一貫した迅速な対応を可能にします。

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: 高

実装のガイダンス

- アラートを発するイベントには、明確に定義された対応策 (ランブックまたはプレイブック) が必要であり、成功裏に完了する責任を負う所有者 (例えば、個人、チーム、役割) が明確に特定されていなければなりません。対応策が実際には自動で、または別のチームによって実行される場合でも、プロセスによって期待される成果を実現させる責任は所有者が持ちます。こうしたプロセスによって、運用上のイベントに対する効果的で迅速な対応が可能になり、アクションの必要なイベントが重要度の低い通知に埋もれてしまうことを避けられます。例えば、ウェブのフロントエンドをスケールする際にスケーリングが自動的に適用される場合でも、自動スケーリングのルールや制限をワークロードのニーズに適したものにすることは運用チームの責任になります。

リソース

関連するドキュメント:

- Amazon CloudWatch の特徴
- Amazon CloudWatch Events とは

関連動画:

- モニタリング計画を立てる

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

介入を必要とする複数のイベントが発生したときに、ビジネスにとって最重要なものから対応できるようにしておきます。影響の例として、死亡や傷害、経済的損失、評判や信用の低下などがあります。

一般的なアンチパターン:

- あなたは、ユーザーのプリンター設定を追加するサポートリクエストを受け取ります。問題に対応している間に、あなたは、小売サイトがダウンしている旨のサポートリクエストを受け取ります。ユーザーのプリンター設定が完了したら、あなたは、ウェブサイトの問題の対応を開始します。
- あなたには、小売ウェブサイトと給与システムの両方が停止していることが通知されます。あなたは、どちらを優先すべきかわかりません。

このベストプラクティスを活用するメリット: ビジネスに最も大きな影響を与えるインシデントへの対応に優先順位を付けることで、その影響を管理できます。

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: ミディアム

実装のガイダンス

- ビジネスへの影響に基づき、運用イベントの優先度を決定する: 介入を必要とする複数のイベントが発生したときに、ビジネスにとって最も重要なものから対応できるようにしておきます。影響の例として、死亡や傷害、経済的損失、規定違反、評判や信用の低下などがあります。

OPS10-BP04 エスカレーション経路を決定する

ランブックとプレイブックで、エスカレーションをトリガーするものとエスカレーションの手順を含むエスカレーション経路を決定します。特に、各アクションの所有者を特定し、運用イベントに効果的かつすばやく対応できるようにします。

アクションを行う前に、人間による決定が必要なタイミングを特定します。意思決定者と協力して事前に決定を行い、アクションを事前に承認することで、MTTR が応答を長時間待機しないようにします。

一般的なアンチパターン:

- あなたの小売サイトがダウンしています。あなたは、サイトを復旧するためのランブックを理解していません。あなたは、誰かがサポートしてくれることを願いながら、同僚に電話をかけ始めます。
- あなたは、アクセス不能なアプリケーションのサポートケースを受け取ります。あなたには、システムを管理するためのアクセス許可がありません。あなたは、誰が管理しているかを知りません。ケースを開いたシステム所有者に連絡しようとしましたが、応答がありません。あなたはシステムに関する問い合わせ先を知らず、同僚はそれになじみがありません。

このベストプラクティスを活用するメリット: エスカレーション、エスカレーションのトリガー、エスカレーションの手順を定義することで、影響に対して適切な割合で、インシデントへの体系的なリソースの追加が可能となります。

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: ミディアム

実装のガイダンス

- エスカレーションパスを定義する: ランブックとプレイブックで、何がエスカレーションをトリガーするかやエスカレーションの手順を含む、エスカレーション経路を決定します。例えば、ランブックで問題が解決できない場合や、一定期間が経過した場合にサポートエンジニアからシニアサポートエンジニアに向けた問題のエスカレーションがあります。また、プレイブックでは修正経路が特定できない場合や、一定期間が経過した場合に、ワークロードについてシニアサポートエンジニアから開発チームに向けたエスカレーションなども例として挙げられます。特に、各アクションの所有者を特定し、運用イベントに効果的かつすばやく対応できるようにします。エスカレーションには第三者が入る場合があります。例えば、ネットワーク接続プロバイダーまたはソフトウェアベンダーです。エスカレーションには、影響するシステムについて承認を受けた特定の意思決定者を含めることができます。

OPS10-BP05 システム停止時の顧客コミュニケーション計画を定義する

システム停止時に顧客やステークホルダーに継続して情報を提供するうえで、信頼性の高いシステム停止時向けのコミュニケーション計画を定義し、テストします。ユーザーが利用するサービスが影響を受けたタイミングと、サービスが正常に戻ったタイミングの両方で直接ユーザーにコミュニケーションを行います。

期待される成果:

- 予定されたメンテナンスから、ディザスタリカバリ計画が発動されるような予期せぬ大規模な障害まで、さまざまな状況に対応するコミュニケーション計画が準備されています。
- コミュニケーションでは、システムの問題に関して、明確かつ透明性の高い情報を提供し、システムのパフォーマンスについての顧客の憶測を回避します。
- カスタムのエラーメッセージとステータスページを使用して、ヘルプデスクへのリクエストのスパイクを低減し、ユーザーに継続的に情報提供を行います。
- コミュニケーション計画は定期的にテストし、実際にシステム停止が発生した際に、意図したとおりに機能することを確認します。

一般的なアンチパターン:

- ワークロードの停止が発生しても、コミュニケーション計画がありません。ユーザーには停止に関する情報が提供されていないため、トラブルチケットシステムにリクエストが殺到します。
- システム停止中にユーザーに E メール通知を送信しますが、サービス復旧のタイムラインが記載されていないため、ユーザーは停止を回避する計画を立てることができません。
- システム停止の際のコミュニケーション計画はあっても、テストしたことがなく、テストしていれば明らかになっていた可能性のある重要なステップを見逃していたため、システム停止が発生すると、コミュニケーション計画は失敗に終わります。
- システム停止中にユーザーに送信する通知には、過剰な技術的な詳細と AWS の NDA に基づく情報が記載されています。

このベストプラクティスを活用するメリット:

- システム停止中にコミュニケーションを継続すると、問題に関する進捗状況と解決までの推定時間を顧客は確実に把握できます。
- 明確に定義されたコミュニケーション計画を策定すると、顧客とエンドユーザーは、確実に十分な情報を入手でき、停止の影響を軽減するために必要となる追加の手順を実行できます。
- 適切なコミュニケーションを行うと、計画されたシステム停止と計画外の停止に関する意識が向上するため、顧客満足度が向上し、意図せぬ反応を制限して、顧客維持を促進できます。
- システム停止に関して、タイムリーかつ透明性が高いコミュニケーションを行うことにより、信頼性が高まり、顧客との関係を維持するうえで必要となる信頼が築かれることになります。
- システム停止の際や危機的状況下で実証済みのコミュニケーション戦略を採用することで、復旧能力の妨害につながる憶測やゴシップを低減できます。

このベストプラクティスを確立しない場合のリスクレベル: 中

実装のガイダンス

システム停止中に顧客に情報を提供するコミュニケーション計画は包括的なものであり、顧客に表示されるエラーページ、カスタム API エラーメッセージ、システムステータスバナー、正常性ステータスページ

など、複数のインターフェイスをカバーしています。システムに登録ユーザーが含まれている場合は、E メール、SMS、プッシュ通知などのメッセージチャネル経由でコミュニケーションを行い、パーソナライズしたメッセージコンテンツを顧客に送信することができます。

顧客コミュニケーションツール

防御の最前線として、ウェブアプリケーションとモバイルアプリケーションは、システム停止中にわかりやすく情報豊富なエラーメッセージを提供し、トラフィックをステータスページにリダイレクトする機能を備えている必要があります。Amazon CloudFront は、カスタムのエラーコンテンツを定義して提供する機能を含む、フルマネージドコンテンツ配信ネットワークです。CloudFront のカスタムエラーページは、コンポーネントレベルの停止に関する顧客メッセージの最初のレイヤーとして適しています。CloudFront は、ステータスページの管理とアクティベーションを簡素化し、計画された停止や計画外の停止中にすべてのリクエストをインターセプトすることもできます。

カスタム API エラーメッセージを使用すると、個別のサービスに分離されている場合に停止が起こった際に、影響を検出して軽減するうえで役立ちます。Amazon API Gateway を使用すると、REST API のカスタムレスポンスを設定できます。これにより、API Gateway がバックエンドサービスに到達できない場合に、API の利用者に明確かつ意味あるメッセージを提供できます。カスタムメッセージは、サービスレイヤーの停止により特定のシステム機能が低下した場合に、停止バナーコンテンツと通知のサポートにも使用できます。

ダイレクトメッセージは、最もパーソナライズされたタイプの顧客メッセージです。Amazon Pinpoint は、スケーラブルなマルチチャネルコミュニケーション向けのマネージドサービスです。Amazon Pinpoint を使用すると、SMS、E メール、音声、プッシュ通知、またはユーザー定義のカスタムチャネルを介して、影響を受ける顧客ベースに広範囲にわたりメッセージをブロードキャストするキャンペーンを構築できます。Amazon Pinpoint を使用してメッセージングを管理すると、メッセージキャンペーンを明確に定義し、テストでき、ターゲットを絞った顧客セグメントにインテリジェントに適用できます。設定したキャンペーンは、スケジュールしたり、イベントでトリガーしたりすることができ、簡単にテストを行うことができます。

お客様事例

AnyCompany Retail では、ワークロードで障害が発生すると、ユーザーに E メール通知を送信します。この E メールには、どのビジネス機能で障害が発生しているを説明し、サービスが復旧するタイミングについての現実的な見積りを提供します。さらに、ワークロードの正常性に関するリアルタイムの情報を表示するステータスページも提供しています。このコミュニケーション計画は、開発環境において年に 2 度テストし、効果的であることを確認しています。

実装手順

1. メッセージング戦略のコミュニケーションチャネルを決定します。アプリケーションのアーキテクチャの側面を考慮に入れて、顧客にフィードバックを提供するための最適な戦略を決定します。これには、エラーページとステータスページ、カスタム API エラーレスポンス、ダイレクトメッセージなど、概説したガイダンス戦略の 1 つ以上の戦略が含まれる場合があります。
2. アプリケーションのステータスページを設計します。ステータスページまたはカスタムエラーページが顧客に適していると判断した場合は、それらのページのコンテンツとメッセージを設計する必要があります。エラーページでは、アプリケーションが利用できない理由、再び利用できるようになるタイミング、その間にできることをユーザーに説明します。アプリケーションが Amazon CloudFront を利用している場合、カスタムエラーレスポンスを提供するか、Lambda at Edge を使用してエラーを翻訳したり、ページのコンテンツを書き換えたりすることができます。CloudFront を使用すると、宛先をアプリケーションコンテンツからメンテナンスや停止のステータスページを含む静的な Amazon S3 コンテンツのオリジンに置換することもできます。
3. サービスの適正な API エラーステータスセットを設計します。バックエンドサービスにアクセスできない場合に API Gateway が生成するエラーメッセージやサービスティアの例外には、エンドユーザーへの表示に適したユーザーフレンドリーなメッセージが含まれていない場合があります。バックエンドサービスのコードを変更する必要なく、API Gateway カスタムエラーレスポンスを設定して、キュレートした API エラーメッセージにHTTP レスポンスコードをマップできます。

4. システムのエンドユーザーに対して関連性が高い内容とし、過度に技術的な詳細が含まれないよう、ビジネスの観点からメッセージを設計します。対象ユーザーを考慮してメッセージを調整します。例えば、社内ユーザーに対しては、代替システムを活用する回避策や手動のプロセスに誘導することができます。外部ユーザーに対しては、システム復旧まで待つか、システム復旧時に通知を受け取るためのアップデートにサブスクライブするようお願いすることができます。予期せぬシステム停止、計画されたメンテナンス、特定機能の障害や利用できなくなるといった部分的なシステム障害など、複数のシナリオについて、事前承認されたメッセージングを定義します。
5. 顧客メッセージをテンプレート化して自動化します。メッセージのコンテンツを設定したら、Amazon Pinpoint またはその他のツールを使用して、メッセージングキャンペーンを自動化することができます。Amazon Pinpoint を使用すると、影響を受けた特定のユーザー向けに顧客ターゲットセグメントを作成し、メッセージをテンプレートに変換できます。メッセージングキャンペーンの設定方法については、「Amazon Pinpoint チュートリアル」を確認してください。
6. メッセージング機能を顧客が使用するシステムに密結合することは避けてください。停止発生時にメッセージが正常に送信できることを確認するためには、メッセージング戦略がシステムデータストアやサービスに対して強制依存関係を持つべきではありません。メッセージングの可用性のために、複数のアベイラビリティーゾーンまたはリージョンからメッセージを送信する機能を構築することを検討してください。AWS サービスを使用してメッセージを送信している場合は、コントロールプレーンのオペレーションではなくデータプレーンのオペレーションを活用して、メッセージを呼び出します。

実装計画に必要な工数レベル: 高。コミュニケーション計画とコミュニケーションを送信するメカニズムの開発には、かなりの労力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:

- OPS07-BP03 ランブックを使用して手順を実行する (p. 74) - 担当者が対応方法を把握しておけるように、コミュニケーション計画にはランブックが関連付けられている必要があります。
- OPS11-BP02 インシデント後の分析を実行する (p. 109) - 停止後にはインシデント後分析を実施し、今後の停止を防ぐメカニズムを特定します。

関連するドキュメント:

- Error Handling Patterns in Amazon API Gateway and AWS Lambda (Amazon API Gateway と AWS Lambda のエラー処理パターン)
- Amazon API Gateway responses (Amazon API Gateway のレスポンス)

関連する例:

- AWS Health Dashboard
- Summary of the AWS Service Event in the Northern Virginia (US-EAST-1) Region (バージニア北部 (US-EAST-1) リージョンにおける AWS サービスイベントの概要)

関連サービス:

- AWS Support
- AWS カスタマーアグリーメント
- Amazon CloudFront
- Amazon API Gateway
- Amazon Pinpoint
- Amazon S3

OPS10-BP06 ダッシュボードでステータスを知らせる

対象となる利用者 (内部技術チーム、指導部、顧客など) に合わせたダッシュボードを用意して、現在の業務の運用状況と、相手が関心を持つメトリクスを知らせます。

コンソールのカスタマイズ可能なホームページで Amazon CloudWatch ダッシュボードを使用して コンソール CloudWatch でダッシュボードを作成できます。Amazon QuickSight のようなビジネスインテリジェンスサービスを 使用すると、 ワークロードと運用状態 (注文率、接続ユーザー、トランザクション時間など) のインタラクティブなダッシュボードを作成して公開できます。メトリクスのシステムレベルおよびビジネスレベルのビューを表示するダッシュボードを作成します。

一般的なアンチパターン:

- リクエストに応じて、あなたは、管理のためのアプリケーションの現在の使用状況に関するレポートを実行します。
- インシデント中、あなたには、心配なシステム所有者から「修正状況」を知りたいという連絡が 20 分ごとにあります。

このベストプラクティスを活用するメリット: ダッシュボードを作成することで、情報へのセルフサービスアクセスが可能になり、顧客自身が情報を確認し、アクションが必要かどうかを判断できるようになります。

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: ミディアム

実装のガイダンス

- ダッシュボードで状態を知らせる: 対象となる利用者 (内部技術チーム、経営陣、顧客など) に合わせたダッシュボードを用意して、現在の業務の運用状況と、相手が関心を持つメトリクスを知らせます。ステータス情報のセルフサービスオプションによって、ステータスのリクエスト処理による運用チームの中断を減らすことができます。例として、Amazon CloudWatch ダッシュボードと AWS Health Dashboard が挙げられます。
  - CloudWatch ダッシュボードを使ったカスタムメトリクスビューの作成と使用

リソース

関連するドキュメント:

- Amazon QuickSight
- CloudWatch ダッシュボードを使ったカスタムメトリクスビューの作成と使用

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し、手動プロセスによって発生するエラーを減らして、迅速かつ一貫した対応を実現します。

AWS には、ランブックやプレイブックのアクションを自動的に実行する複数の方法があります。AWS リソースの状態変化や独自のカスタムイベントからのイベントに対応するには、 CloudWatch Events ルールを作成し、 CloudWatch のターゲット (Lambda 関数、Amazon Simple Notification Service (Amazon SNS) トピック、Amazon ECS タスク、AWS Systems Manager Automation など) を通じて対応を起動します。

リソースのしきい値を超えるメトリクス (待機時間など) に応答するには、 CloudWatch アラームを作成して Amazon EC2 アクションや Auto Scaling アクションを使用して 1 つ以上のアクションを実行するか、Amazon SNS トピックに通知を送信します。アラームへの応答でカスタムアクションを実行する必要

がある場合は、Amazon SNS 通知を通じて Lambda を呼びだします。Amazon SNS を使用して、イベント通知とエスカレーションメッセージを発行し、ユーザーに情報を提供します。

また、AWS は、AWS のサービス API と SDK を通じてサードパーティーシステムもサポートしています。AWS パートナーやサードパーティーでは、モニタリング、通知、応答を可能にするモニタリングツールを多数提供しています。これらのツールには、New Relic、Splunk、Loggly、SumoLogic、Datadog などがあります。

自動化された手順が失敗した場合に、手動でも重要な手順を実施できるようにしておく必要があります。

一般的なアンチパターン:

- 開発者が自分のコードをチェックインします。このイベントは、ビルドを開始し、テストを実行するために使用することもできましたが、結局、何にも使用されていません。
- アプリケーションが動作を停止する前に、特定のエラーをログ記録します。アプリケーションを再起動する手順はよく理解されており、スクリプト化することもできました。あなたは、ログイベントを使用して、スクリプトを呼び出し、アプリケーションを再起動することもできました。しかし、それらの対応を行わなかったため、日曜日の午前 3 時にエラーが発生し、あなたは、オンコールでシステムを修正する責任者として起こされます。

このベストプラクティスを確立するメリット: イベントへの対応を自動化することで、対応にかかる時間を短縮し、手作業によるエラーの発生を抑制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: 低

実装のガイダンス

- イベントへの対応を自動化する: イベントへの対応を自動化し、手動プロセスによって発生するエラーを減らして、迅速かつ一貫した対応を実現します。
  - Amazon CloudWatch Events とは
  - イベントでトリガーする CloudWatch Events のルールの作成
  - AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成
  - サポートされているサービスからの CloudWatch Events イベントの例

リソース

関連するドキュメント:

- Amazon CloudWatch の特徴
- サポートされているサービスからの CloudWatch Events イベントの例
- AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch Events ルールの作成
- イベントでトリガーする CloudWatch Events のルールの作成
- Amazon CloudWatch Events とは

関連動画:

- モニタリング計画を立てる

関連する例:

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 (p. 107)

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 (p. 107)
- OPS11-BP02 インシデント後の分析を実行する (p. 109)
- OPS11-BP03 フィードバックループを実装する (p. 109)
- OPS11-BP04 ナレッジ管理を実施する (p. 112)
- OPS11-BP05 改善の推進要因を定義する (p. 113)
- OPS11-BP06 インサイトを検証する (p. 114)
- OPS11-BP07 オペレーションメトリクスのレビューを実行する (p. 115)
- OPS11-BP08 教訓を文書化して共有する (p. 116)
- OPS11-BP09 改善を行うための時間を割り当てる (p. 117)

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。1 年に 1 回以上ワークロードの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:

- 年 1 回以上、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分析します。
- ソフトウェア開発プロセスにおいて改善の機会が等しく優先されます。

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。
- 改善の機会の優先度が低く、ずっと後回しにされている。
- 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。
- ワークロードの進化を熟考した方法で実現できます。

- 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。

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

1 年に 1 回以上ベースで、ワークロードの構造的レビューを実施します。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。

お客様事例

AnyCompany Retail のすべてのワークロードは、毎年のアーキテクチャレビュープロセスを経由します。ベストプラクティスについての独自のチェックリストを作成し、すべてのワークロードに適用しています。AWS Well-Architected Tool のカスタムレンズ機能を活用し、ツールやベストプラクティスのカスタムレンズを使ってレビューを実施しています。レビューで見い出された改善の機会は、ソフトウェア開発サイクルの中で優先事項になっています。

実装手順

1. 1 年間に 1 回以上、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、社内で独自に定義した基準を使用することをお勧めします。社内基準がない場合は、AWS Well-Architected フレームワークを使用することをお勧めします。
   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できます。
   c. お客様は AWS ソリューションアーキテクトに連絡して、ワークロードのガイド付き Well-Architected フレームワークレビューを実施できます。
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 (p. 109) - インシデント後の分析は、改善項目を洗い出すもう一つの機会です。学んだ教訓で、アーキテクチャのベストプラクティスの社内リストを豊かにできます。
- OPS11-BP08 教訓を文書化して共有する (p. 116) - 独自のアーキテクチャのベストプラクティスを作成したら、組織全体で共有します。

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(Well-Architected レビューを、カスタムレンズと AWS Well-Architected ツールを使用してカスタマイズする)
- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization (AWS Well-Architected カスタムレンズのライフサイクルを組織に実装する)

関連動画:

- Well-Architected Labs - Level 100: Custom Lenses on AWS Well-Architected Tool(レベル 100: AWS WELL-ARCHITECTED ツールのカスタムレンズ)

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。

一般的なアンチパターン:

- あなたは、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。

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

- プロセスを使用して寄与した要因を判断する: 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。必要に応じ、対象者に合わせて根本原因を通知します。

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、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 外部顧客のニーズを評価する (p. 4): 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 (p. 5): 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 (p. 109): 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 (p. 115): 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 (p. 116) - 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

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。

AWS では、すべての運用アクティビティ、ワークロード、インフラストラクチャのログを集約して詳細なアクティビティ履歴を作成できます。AWS ツールを使用して長期的に運用とワークロードの状態を分析し (トレンドの特定、イベントやアクティビティの成果への関連付け、環境間やシステム全体の比較や対比など)、推進要因に基づいて改善の機会を見つけることができます。

CloudTrail を使用し、AWS Management Console、CLI、SDK、API を介して API アクティビティを追跡し、アカウント全体で何が起きているかを把握する必要があります。CloudTrail および CloudWatch を使用して、AWS デベロッパーツールのデプロイアクティビティを追跡します。これにより、デプロイの詳細なアクティビティ履歴と結果が CloudWatch Logs Logs のログデータに追加されます。

長期保管用の Amazon S3 にログデータを エクスポートします。分析のために、 AWS Glueを使用して、Amazon S3 のログデータを検出して準備します。AWS Glue とネイティブで統合された Amazon Athena を使用して、ログデータを分析します。Amazon QuickSight のような ビジネスインテリジェンスツール を使用して、データを可視化、調査、分析することができます。

一般的なアンチパターン:

- あるスクリプトは、機能はするものの、洗練されてはいません。あなたは、書き換えに時間を費やします。現在は素晴らしいスクリプトです。

- スタートアップは、ベンチャーキャピタリストから別の資金を調達しようとしています。そのスタートアップは、PCI DSS へのコンプライアンスを実証することをあなたに求めています。あなたは、ベンチャーキャピタリストを満足させたいと考え、コンプライアンスを文書化し、ある顧客の納期に遅れ、その顧客を失います。それをするのは間違ったことではありませんでしたが、今では正しいことだったのかどうかを疑問に思います。

このベストプラクティスを活用するメリット: 改善に使用する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。

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: ミディアム

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。
    - AWS セキュリティ速報
    - AWS Trusted Advisor
  - 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ー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

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。

一般的なアンチパターン:

- あなたは、CPU 使用率がシステムで 95% であることを確認し、システムの負荷を軽減する方法を見つけることを優先します。あなたは、最適なアクションがスケールアップであると判断します。システムはトランスコーダーであり、いつでも 95% の CPU 使用率で実行するようにスケールされています。あなたがシステム所有者に連絡していれば、状況を説明してもらえたかもしれません。時間を無駄にしてしまいました。
- システム所有者は、システムがミッションクリティカルであると主張しています。システムは強固なセキュリティ環境に配置されていませんでした。セキュリティの向上のため、あなたは、ミッションクリティカルなシステムに要求される追加の発見的統制および予防統制を実装します。あなたは、作業が完了し、追加のリソースに対して課金される旨をシステム所有者に通知します。この通知の後の話し合いにおいて、システム所有者は、ミッションクリティカルという用語について正式な定義があり、自身のシステムがこれに適合していないことを知ります。

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。

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: ミディアム

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。

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、生産など) で改善する機会を探します。

一般的なアンチパターン:

- 大々的な販促活動が行われていましたが、メンテナンスウィンドウによって中断されました。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。
- あなたは、組織で一般的に使用されているバグのあるライブラリを使用しているため、停止時間が長くなり、困っていました。その後、あなたは、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。あなたが定期的にミーティングを行い、このインシデントを確認していれば、彼らはリスクを認識していたでしょう。
- トランスコーダーのパフォーマンスは着実に低下しており、メディアチームに影響を及ぼしています。まだひどい状態であるとまでは言えません。インシデントの原因となるほど悪くなるまで気付く機会はありません。メディアチームと一緒にオペレーションメトリクスを見直すことで、メトリクスの変化や彼らの経験を認識し、問題に対処する機会が生まれるはずです。
- あなたは、顧客の SLA の満足度を確認していません。あなたは、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。これらの SLA のメトリクスを確認するためのミーティングを定期的に開催していれば、問題を認識して対処する機会が得られたはずです。

このベストプラクティスを確立するメリット: 定期的にミーティングを行い、オペレーションメトリクス、イベント、インシデントを確認することで、チーム全体で共通の理解を維持し、学んだ教訓を共有し、改善を優先順位付けして目標を設定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: ミディアム

実装のガイダンス

- オペレーションメトリクスのレビュー: 定期的に、さまざまな分野のチームメンバーとともに、オペレーションメトリクスの遡及分析を行います。ビジネス、開発、オペレーションチームを含む関係者を参加させて、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。それらの洞察に基づいて、改善の機会と取り得る一連のアクションを特定します。
  - Amazon CloudWatch
  - Amazon CloudWatch メトリクスを使用する
  - カスタムメトリクスをパブリッシュする
  - Amazon CloudWatch のメトリクスとディメンションのリファレンス

リソース

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

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、開発作業を容易にする必要があります。これにより、望まれる機能の提供に集中できます。

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。その後、バージョン管理された AWS CodeCommit 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する必要があります。AMI へのアクセスを共有し、Lambda 関数の使用をアカウント間で許可することで、コンピューティング標準を共有します。また、インフラストラクチャ標準を AWS CloudFormation のテンプレートとして共有する必要もあります。

AWS API と SDK を使用すると、外部ツールやサードパーティーのツールやリポジトリ (GitHub、BitBucket、SourceForge など) を統合できます。学んだことや開発したことを共有するときは、共有リポジトリの完全性を保証するためにアクセス許可を構造化することに注意してください。

一般的なアンチパターン:

- あなたは、組織で一般的に使用されているバグのあるライブラリを使用しているため、停止時間が長くなり、困っていました。その後、あなたは、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。このライブラリの経験を文書化および共有することとしていれば、これらのチームはリスクを認識していたでしょう。
- あなたは、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。このエッジケースを回避するために、サービスへ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。このライブラリの経験を文書化および共有することとしていれば、これらのチームはリスクを認識していたでしょう。

- 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を見つけました。あなたは、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。このライブラリで経験を文書化および共有することとしていれば、これらのチームは利用する機会を得ていたでしょう。

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、経験から得られる恩恵を最大化します。

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

- 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。たとえば、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
    - AWS Lambda 関数の簡単な承認
    - 特定の AWS アカウントと AMI を共有する
    - AWS CloudFormation デザイナー URL によるテンプレート共有の高速化
    - Amazon SNS での AWS Lambda の使用

リソース

関連するドキュメント:

- AWS Lambda 関数の簡単な承認
- AWS CodeCommit リポジトリを共有する
- 特定の AWS アカウントと AMI を共有する
- AWS CloudFormation デザイナー URL によるテンプレート共有の高速化
- Amazon SNS での AWS Lambda の使用

関連動画:

- AWS 環境へのアクセスの委任

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。

AWS では、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。

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。この問題は、計画されたすべての機能実装の後に実施されるバックログに追加されます。計画的に追加される機能の割合が一定であると、パフォーマンスの問題が解決されることはありません。
- 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。

このベストプラクティスを確立するメリット: 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を可能にするために、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追及してください。

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
- Amazon Web Services、Sr. Technical Program Manager、Ryan King
- Amazon Web Services、Advisory Consultant、Chris Kunselman
- Amazon Web Services、Advisory Consultant、Peter Mullen
- 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 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、Senior Specialist Technical Account Manager、Arvind Raghunathan

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
メジャーなコンテンツの更新と統合 (p. 122)	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 (p. 122)	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 (p. 122)	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 (p. 122)	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 (p. 122)	編集上の微小な修正	August 8, 2022
ホワイトペーパーの更新 (p. 122)	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 (p. 1)	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 (p. 122)	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 (p. 122)	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 (p. 122)	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