Amazon CloudWatch を使用したロギングとモニタリングの設計と実装

AWS 規範ガイダンス

AWS 規範ガイダンス: Amazon CloudWatch を使用したロギングとモニタリングの設計と実装

Table of Contents

Amazon CloudWatch を使用したロギングとモニタリングの設計と実装です。

クーラム・ニザミ、Amazon Web Services (AWS)

2023 年 4 月 (ドキュメント履歴)

このガイドは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS)、AWS Lambda、およびオンプレミスサーバーを使用するワークロードに対して、Amazon CloudWatch と関連のAmazon Web Services (AWS) 管理、ガバナンスサービスによるロギングとモニタリングを設計、および実装することを支援するものです。このガイドは、 AWS クラウド上のワークロードを管理する運用チーム、DevOps エンジニア、アプリケーションエンジニアを対象としています。

ロギングとモニタリングのアプローチは、 AWS Well-Architected フレームワークの 6 つの柱に基づく必要があります。これらの柱は、運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化 です。Well-Architected モニタリングとアラームのソリューションは、インフラストラクチャをプロアクティブに分析し調整するのに役立ち、信頼性とパフォーマンスを向上させます。

このガイドでは、セキュリティやコスト最適化のためのロギングとモニタリングについては、詳細な評価が必要なトピックであるため、広範囲には説明しません。セキュリティログと監視をサポートする AWS サービスは、AWS CloudTrail、AWS Config、Amazon Inspector、Amazon Detective、Amazon Macie、Amazon GuardDuty、および AWS Security Hub CSPM など多岐にわたります。AWS Cost Explorer、AWS Budgets、および CloudWatch 請求メトリクスを使用してコストを最適化することもできます。

次の表に、ロギング、およびモニタリングソリューションが対応する 6 つの領域の概要を示します。

ログファイル、およびメトリクスの取得と取り込み	システム、およびアプリケーションのログとメトリクスを識別し、構成し、さまざまなソースから AWS サービスに送信します。
ログの検索と分析	運用管理、問題の特定、トラブルシューティング、およびアプリケーション分析のためのログを検索、および分析します。

メトリクスとアラームのモニタリング	ワークロードの観察と傾向を特定し、それに基づいて行動します。
アプリケーションとサービスの可用性のモニタリング	サービスの可用性を継続的にモニタリングすることで、ダウンタイムを削減し、サービスレベルの目標を達成する能力を向上させます。
アプリケーションをトレースする	システムと外部の依存関係にあるアプリケーションのリクエストを追跡して、パフォーマンスの微調整、根本原因の分析、および問題のトラブルシューティングを行います。
ダッシュボードとビジュアライゼーションの作成	システムおよびワークロードの関連メトリクスと観察結果に焦点を当てたダッシュボードを作成し、継続的な改善と問題の事前発見に役立てることができます。

CloudWatch は、ロギングとモニタリングのほとんどの要件を満たすことができ、信頼性、拡張性、柔軟性に優れたソリューションを提供します。多くの AWS サービスは、モニタリングと分析のための CloudWatch ログ記録統合に加えて、CloudWatch メトリクスを自動的に提供します。また、CloudWatch は、サーバ (クラウドとオンプレミスの両方) 、コンテナ、サーバーレスコンピューティングなどの様々な計算オプションをサポートするエージェントとログドライバを提供します。このガイドでは、ログ記録とモニタリングで使用される以下の AWS サービスについても説明します。

- EC2 インスタンスとオンプレミスサーバーの CloudWatch エージェントを自動化、設定、更新する AWS Systems Manager Distributor、Systems Manager State Manager、Systems Manager Automation https://docs.aws.amazon.com//systems-manager/latest/userguide/systems-manager-automation.html
- 高度なログ集約、検索、分析のための Amazon OpenSearch Service
- Amazon Route 53 ヘルスチェック そして CloudWatch Synthetics によるアプリケーションとサービスの可用性のモニタリングです。
- Amazon Managed Service for Prometheus によるコンテナ型アプリケーションの大規模なモニタリングです。
- AWS X-Ray はアプリケーションのトレースとランタイム解析のためのものです。

- 複数のソース (CloudWatch、Amazon OpenSearch Service、Amazon Timestream など) からのデータを視覚化および分析するための Amazon Managed Grafana

選択した AWS コンピューティングサービスは、ロギングおよびモニタリングソリューションの実装と設定にも影響します。例えば、Amazon EC2、Amazon ECS、Amazon EKS、Lambda では、CloudWatch の実装や設定が異なります。

アプリケーション、およびワークロードの所有者は、ロギングとモニタリングについて忘れてしまったり、一貫性のない構成や実装をしてしまったりすることがよくあります。つまり、ワークロードは観測性が制限された本番環境に入り、問題の特定に遅延が生じ、トラブルシューティングと解決に要する時間が長くなります。少なくとも、ロギングおよびモニタリングソリューションでは、アプリケーションログおよびメトリックのアプリケーション層に加えて、オペレーティングシステム（OS）レベルのログとメトリックのシステム層に対処する必要があります。このガイドでは、次の表で概説する 3 つのコンピューティングタイプを含む、異なるコンピューティングタイプでこれらの 2 つのレイヤーに対処するための推奨されるアプローチについて説明します。

長時間稼働するイミュータブル EC2 インスタンス	複数の AWS リージョンまたはアカウントの複数のオペレーティングシステム (OSs) にわたるシステムおよびアプリケーションのログとメトリクス。
コンテナ	Amazon ECS および Amazon EKS クラスターのシステムログとアプリケーションのログとメトリックス（さまざまな設定の例を含む）です。
サーバーレス	Lambda 関数のシステムログとアプリケーションのログとメトリクス、およびカスタマイズに関する考慮事項です。

このガイドでは、以下の分野における CloudWatch および関連 AWS サービスに対応するロギングおよびモニタリングソリューションを提供します。

- CloudWatch デプロイを計画する - CloudWatch デプロイを計画する際の考慮事項と、CloudWatch 設定の一元化に関するガイダンスです。
- EC2 インスタンスとオンプレミスサーバー用の CloudWatch エージェントの設定 - システムレベルおよびアプリケーションレベルのロギングとメトリクスの CloudWatch 設定の詳細です。

- Amazon EC2 およびオンプレミスサーバーに対する CloudWatch エージェントのインストールアプローチ - 複数のリージョンとアカウントにまたがる Systems Manager を使用した自動デプロイメントを含む、CloudWatch エージェントのインストール方法です。
- Amazon ECS でのログ記録とモニタリング - Amazon ECS でクラスターレベル、およびアプリケーションレベルのロギングとメトリクスに CloudWatch を設定するためのガイダンスです。
- Amazon EKS でのログ記録とモニタリング - Amazon EKS でクラスターレベル、およびアプリケーションレベルのロギングとメトリックスに CloudWatch を設定するためのガイダンスです。
- Amazon EKS における Prometheus モニタリング - Prometheus 向けアマゾンマネージドサービスと、Prometheus 向けCloudWatch コンテナインサイトモニタリングを紹介し、比較します。
- のログ記録とメトリクス AWS Lambda - Lambda 関数に CloudWatch を設定するためのガイダンスです。
- CloudWatch でのログの検索および分析 – Amazon CloudWatch Application Insights、CloudWatch Logs Insights を使用してログを分析し、ログ分析を Amazon OpenSearch Service に拡張する方法。
- CloudWatch によるアラームのオプション - CloudWatch アラームと CloudWatch Anomaly Detectionを導入し、アラームの作成とセットアップのガイダンスを提供します。
- アプリケーションとサービスの可用性のモニタリング - CloudWatch Synthetics と Route 53 ヘルスチェックを導入し、比較して、自動化された可用性モニタリングを行います。
- を使用したアプリケーションのトレース AWS X-Ray - Amazon EC2、Amazon ECS、Amazon EKS、および Lambda の X-Ray を使用したアプリケーショントレーシングの概要とセットアップです。
- CloudWatch を使用したダッシュボードとビジュアライゼーション – AWS ワークロード全体のオブザーバビリティを向上させる CloudWatch Dashboards の概要。
- CloudWatch と AWS サービスとの統合 – CloudWatch とさまざまな AWS サービスとの統合方法について説明します。
- ダッシュボードと可視化のための Amazon マネージド Grafana – ダッシュボードと視覚化のために Amazon Managed Grafana と CloudWatch を紹介し、比較します。

実装例は、これらの領域にわたってこのガイド全体で使用され、また AWS GitHub リポジトリ例 から入手できます。

ターゲットを絞ったビジネス成果

AWS クラウド向けに設計されたロギングおよびモニタリングソリューションを作成することは、クラウドコンピューティングの 6 つの利点を達成するために不可欠です。ロギングおよびモニタリングソリューションは、IT 組織がビジネスプロセス、ビジネスパートナー、従業員、顧客に利点をもたらすビジネス成果を達成するのに役立ちます。AWS Well-Architected フレームワークに沿ったロギングおよびモニタリングソリューションを実装すると、次の 4 つの結果が期待できます。

オペレーショナルレディネスの迅速化

ロギングおよびモニタリングソリューションを有効にすることは、本番環境のサポートおよび使用のためにワークロードを準備する上で重要な要素です。マニュアルでのプロセスに大きく依存しすぎると、オペレーショナルレディネスがすぐに障害となる可能性があり、お客様の IT 投資のタイムトゥバリュー (TTV) も短縮することにもなりかねません。効果のないアプローチでは、お客様のワークロードのオブザーバビリティが制限される結果にもなります。これにより、長期にわたるシステム停止、顧客不満、ビジネスプロセスの失敗のリスクが高まる可能性があります。

このガイドのアプローチを使用して、 AWS クラウドでのログ記録とモニタリングを標準化および自動化できます。新しいワークロードでは、本番環境のロギングとモニタリングのためのマニュアルでの準備と介入が最小限で済みます。これにより、複数のアカウントとリージョンにまたがるさまざまなワークロードのロギングおよびモニタリングのデフォルトを大規模に作成するのに必要な時間とステップが削減されます。

オペレーショナルエクセレンスの向上

このガイドでは、さまざまなワークロードがビジネス目標とオペレーショナルエクセレンスを達成するのに役立つロギングとモニタリングに関する複数のベストプラクティスを提供します。このガイドでは、詳細な例と、Infrastructure as Code (IaC) アプローチで使用できるオープンソースの再利用可能なテンプレートも提供し、 AWS サービスを使用して適切に設計されたロギングおよびモニタリングソリューションを実装します。 IaC オペレーショナルエクセレンスの向上は反復的であり、継続的な改善が必要です。このガイドでは、ロギングとモニタリングの実践を継続的に改善する方法について提案しています。

オペレーショナルビジビリティの向上

ビジネスプロセスとアプリケーションは、さまざまな IT リソースでサポートされ、オンプレミスまたは AWS クラウドのさまざまなコンピューティングタイプでホストされる場合があります。オペ

レーショナルビジビリティは、お客様のロギングおよびモニタリング戦略の不整合で不完全な実装によって制限される可能性があります。包括的なロギングとモニタリングアプローチを採用することで、ワークロード全体の問題を迅速に特定、診断、対応できます。このガイドは、全体的なオペレーショナルビジビリティを改善し、障害解決までの平均時間 (MTTR) の短縮のためのアプローチを設計および実装するのに役立ちます。また、包括的なロギングとモニタリングアプローチにより、お客様の組織のサービス品質の向上、エンドユーザーエクスペリエンスの向上、SLA (サービスレベルアグリーメント) の遵守にも役立ちます。

オペレーションの拡張とオーバーヘッドコストの削減

このガイドからロギングとモニタリングの実践を拡張して、複数のリージョンとアカウント、短期間のリソース、および複数の環境をサポートできます。このガイドでは、マニュアルのステップを自動化するためのアプローチと例について説明します (例えば、エージェントのインストールと設定、メトリクスのモニタリング、問題が発生した場合の通知またはアクションの実行など)。これらのアプローチは、クラウドの導入が成熟して成長し、クラウド管理アクティビティやリソースを増やすことなく運用能力を拡張する必要がある場合に役立ちます。

CloudWatch デプロイを計画する

ロギングとモニタリングのソリューションの複雑さと範囲は、以下のようないくつかの要因に依存します。

- 使用されている環境、リージョン、およびアカウントの数と、この数がどのように増加するかです。
- 既存のワークロードとアーキテクチャの多様性と種類。
- ログに記録、および監視する必要のあるコンピューティングタイプと OS です。
- オンプレミスの場所と AWS インフラストラクチャの両方があるかどうか。
- 複数のシステム、アプリケーションの集計、および分析要件です。
- ログやメトリクスの不正な流出を防ぐためのセキュリティ要件です。
- 運用プロセスをサポートするために、ロギングおよびモニタリングソリューションと統合する必要がある製品とソリューションです。

新規または更新されたワークロードの導入に伴い、ロギングおよびモニタリングソリューションを定期的に見直し、更新する必要があります。ロギング、モニタリング、およびアラームの更新は、問題が観察されたときに特定し、適用する必要があります。将来的に、このような問題はプロアクティブに特定され、防止することができます。

ログとメトリクスを取得· 取り込むためのソフトウェアとサービスを一貫してインストール、および構成していることを確認する必要があります。確立されたログ記録とモニタリングのアプローチでは、さまざまなドメイン (セキュリティ、パフォーマンス、ネットワーク、分析など) に対して複数の AWS 、または独立したソフトウェアベンダー (ISV) のサービスとソリューションを使用します。各ドメインには、独自の展開および構成要件があります。

CloudWatch を使用して、複数の OS とコンピューティングタイプのログとメトリクスを取得することをお勧めします。多くの AWS サービスでは、CloudWatch を使用してログとメトリクスをログに記録し、モニタリングし、公開します。さらに設定する必要はありません。CloudWatch は ソフトウェアエージェント を提供しており、様々な OS や環境に合わせてインストールや設定ができます。以下のセクションでは、複数のアカウント、リージョン、構成に対して CloudWatch エージェントをデプロイ、インストール、および設定する方法について説明します。

トピック

- 集中型、または分散型アカウントで CloudWatch を使用する

- CloudWatch エージェント設定ファイルの管理

集中型、または分散型アカウントで CloudWatch を使用する

CloudWatch は 1 つのアカウントとリージョンのサービス AWS またはリソースをモニタリングするように設計されていますが、中央アカウントを使用して複数のアカウントとリージョンからログとメトリクスをキャプチャできます。複数のアカウント、またはリージョンを使用する場合は、集中型アカウントのアプローチを使用するか、個々のアカウントを使用してログとメトリクスを取得するかを評価する必要があります。通常、セキュリティ、分析、運用、ワークロードの所有者の要件をサポートするために、マルチアカウント、およびマルチリージョンデプロイにはハイブリッドアプローチが必要です。

次の表は、集中型、分散型、またはハイブリッドアプローチを選択する際に考慮すべき事項を示しています。

アカウント構造	組織では、特定の環境内の単一のアプリケーションに対して複数の個別のアカウント（例えば、非本番ワークロードと本番ワークロードのアカウントなど）または数千のアカウントがある場合があります。ワークロードが実行されるアカウントでアプリケーションのログとメトリクスを管理し、ワークロードの所有者がログとメトリクスにアクセスできるようにすることを推奨します。これにより、ロギングとモニタリングでアクティブなロールを持つことができます。また、分析、集計、傾向、および集中操作のために、すべてのワークロードログを集約する別のログ用アカウントを使用することを推奨します。個別のログ記録アカウントは、セキュリティ、アーカイブとモニタリング、および分析にも使用できます。
アクセス要件	チームメンバー（たとえば、ワークロード所有者や開発者）は、トラブルシューティングや改善のためにログやメトリクスにアクセスする必要があります。ログは、アクセスやトラブルシューティングを容易にするために、ワークロードのアカウントで管理する必要があります。ログと測定基準をワークロードとは別のアカウントで管理する場合、ユーザーは、定期的にアカウントを交互に切り替える必要があるかもしれません。

	集中型アカウントを使用すると、ワークロードアカウントへのアクセスを許可せずに、承認されたユーザーにログ情報が提供されます。これにより、複数のアカウントで実行されているワークロードから集約が必要な分析ワークロードのアクセス要件を簡素化できます。集中型ログ記録アカウントには、Amazon OpenSearch Service クラスターなどの代替の検索および集約オプションも使用できます。Amazon OpenSearch Service は、ログのフィールドレベルまできめ細かなアクセスコントロールを提供します。特殊なアクセスや許可を必要とする機密データを扱う場合、きめ細かなアクセス制御が重要になります。
オペレーション	多くの組織では、集中管理された運用・セキュリティチームや、運用支援のための外部組織があり、モニタリングのためのログへのアクセスが必要です。ログとモニタリングを一元化することで、すべてのアカウントとワークロードの傾向の把握、検索、集計、分析の実行が容易になります。組織が DevOps のために「構築し、実行する」アプローチを使用している場合、ワークロードの所有者は自分のアカウントでログとモニタリング情報を必要とします。分散したワークロードの所有権に加えて、中央の運用と分析を満足させるには、ハイブリッドなアプローチが必要になるかもしれません。
環境	セキュリティ要件とアカウントのアーキテクチャに応じて、本番用アカウントではログとメトリクスを中央でホストし、その他の環境（例えば、開発またはテスト）ではログとメトリクスを同じアカウントまたは別のアカウントに保持することを選択できます。これにより、本番環境で作成された機密データが、より多くのユーザーによってアクセスされることを防ぐことができます。

CloudWatch は、CloudWatch サブスクリプションフィルタでリアルタイムにログを処理するための複数のオプション を提供します。サブスクリプションフィルターを使用して、ログを AWS サービスにリアルタイムでストリーミングし、カスタム処理、分析、他のシステムにロードできます。これは、集中管理されたアカウントとリージョンだけでなく、個々のアカウントとリージョンでもログと測定基準を利用できるハイブリッドアプローチをとっている場合に特に役立ちます。次のリストは、このために使用できる AWS サービスの例を示しています。

- Amazon Data Firehose – Firehose は、生成されるデータボリュームに基づいて自動的にスケーリングおよびサイズ変更するストリーミングソリューションを提供します。Amazon Kinesis データストリーム内のシャードの数を管理する必要はなく、追加のコーディングなしで Amazon Simple Storage Service (Amazon S3)、Amazon OpenSearch Service、または Amazon Redshift に直接接続できます。Firehose は、これらの AWS サービスにログを一元化する場合に効果的なソリューションです。
- Amazon Kinesis Data Streams – Firehose がサポートしていないサービスと統合し、追加の処理ロジックを実装する必要がある場合、Kinesis Data Streams は適切なソリューションです。中央アカウントで Kinesis データストリームを指定するアカウントとリージョンに Amazon CloudWatch Logs 送信先を作成し、ストリームにレコードを配置するアクセス許可を付与する AWS Identity and Access Management (IAM) ロールを作成できます。Kinesis Data Streams は、ログデータのための柔軟でオープンエンドなランディングゾーンを提供し、その後、さまざまなオプションで使用することができます。Kinesis Data Streams のログデータをアカウントに読み込み、前処理を行い、選択した宛先にデータを送信することができます。

  ただし、生成されるログデータに合わせて適切なサイズになるように、ストリーミングのシャードを構成する必要があります。Kinesis Data Streams は、ログデータの一時的な仲介またはキューとして機能し、データを Kinesis ストリーム内に 1 ~ 365 日間保存できます。Kinesis Data Streams は再生機能もサポートしており、使用されなかったデータを再生することができます。
- Amazon OpenSearch Service – CloudWatch Logs は、ロググループのログを、個別または一元化されたアカウントの OpenSearch クラスターにストリーミングできます。OpenSearch クラスターにデータをストリーミングするようにロググループを設定すると、ロググループと同じアカウントとリージョンに Lambda 関数が作成されます。Lambda 関数には、OpenSearch クラスターとのネットワーク接続が必要です。Lambda 関数をカスタマイズして、Amazon OpenSearch Service への取り込みをカスタマイズするだけでなく、追加の前処理を実行することもできます。Amazon OpenSearch Service による一元的なログ記録により、クラウドアーキテクチャの複数のコンポーネントにわたる問題の分析、検索、トラブルシューティングが容易になります。
- Lambda - Kinesis Data Streams を使用する場合、ストリーミングからデータを消費するコンピューティングリソースをプロビジョニングおよび管理する必要があります。これを回避するには、処理のためにログデータを直接 Lambda にストリーミングし、ロジックに基づいて送信先に送信します。つまり、受信データをを処理するためのコンピューティングリソースをプロビジョニングして管理する必要がありません。Lambda を使用することを選択した場合、ソリューションが Lambda クォータ と互換性があることを確認してください。

CloudWatch Logs にファイル形式で保存されているログデータを処理または共有する必要がある場合があります。特定の日付または時間範囲のロググループを Amazon S3 にエクスポートする エク

スポートタスクを作成することができます。例えば、分析や監査のために、Amazon S3 に毎日ログをエクスポートすることを選択することができます。Lambda を使用すると、このソリューションを自動化することができます。また、このソリューションを Amazon S3 レプリケーションと組み合わせることで、複数のアカウントやリージョンから 1 つの集中アカウントやリージョンにログを出荷し、一元管理することができます。

CloudWatch エージェントの設定では、`credentials`セクション に `agent` フィールドを指定することも可能です。これは、メトリクスやログを別のアカウントに送信する際に使用する IAM ロールを指定するものです。指定した場合、このフィールドは `role_arn` パラメータが含まれています。このフィールドは、特定の集中型アカウントおよびリージョンで集中ロギングとモニタリングのみが必要な場合に使用できます。

AWS SDK を使用して、任意の言語で独自のカスタム処理アプリケーションを記述したり、アカウントからログやメトリクスを読み取ったり、データを一元管理されたアカウントやその他の送信先に送信して、さらなる処理やモニタリングを行ったりすることもできます。

CloudWatch エージェント設定ファイルの管理

すべての Amazon Elastic Compute Cloud (Amazon EC2) インスタンスとオンプレミスサーバーでキャプチャするシステムログとメトリクスを含む、標準の Amazon CloudWatch エージェント設定を作成することをお勧めします。Amazon EC2 CloudWatch エージェント 設定ファイルウィザード を使用すると、設定ファイルの作成に役立ちます。構成ウィザードを複数回実行することで、異なるシステムや環境に対して固有の構成を生成することができます。また、設定ファイルのスキーマ を使用して、設定ファイルを変更したり、バリエーションを作成したりすることもできます。CloudWatch エージェント設定ファイルは、AWS Systems Manager パラメータストアパラメータに保存できます。 複数の CloudWatch エージェント設定ファイルがある場合は、個別の Parameter Store パラメータを作成できます。複数の AWS アカウントまたは AWS リージョンを使用している場合は、各アカウントとリージョンで Parameter Store パラメータを管理および更新する必要があります。または、CloudWatch 設定を Amazon S3 または任意のバージョン管理ツールのファイルとして一元管理することもできます。

CloudWatch エージェントに含まれる `amazon-cloudwatch-agent-ctl` スクリプトでは、設定ファイル、Parameter Store パラメータ、またはエージェントのデフォルト設定を指定することができます。デフォルトの設定は、ベーシックな事前定義されたメトリクスセットに合わせ、CloudWatch にメモリとディスクスペースのメトリクスを報告するようにエージェントを構成しています。ただし、ログファイルの設定は含まれません。CloudWatch エージェントに Systems Manager Quick Setup を使用した場合にも、デフォルトの設定が適用されます。

デフォルト設定には口グ記録が含まれず、要件に合わせてカスタマイズされていないため、要件に合わせてカスタマイズされた独自の CloudWatch 設定を作成して適用することをお勧めします。

CloudWatch 設定の管理

デフォルトでは、CloudWatch 設定は Parameter Store パラメータまたは CloudWatch 設定ファイルとして保存および適用できます。最適な選択肢は、要件によって異なります。このセクションでは、これら 2 つのオプションの長所と短所について説明します。また、複数の AWS アカウントと AWS リージョンの CloudWatch 設定ファイルを管理するための代表的なソリューションについても詳しく説明します。

Systems Manager パラメータストアのパラメータ

パラメータストアパラメータを使用して CloudWatch 設定を管理するのは、少数の AWS アカウントとリージョンに適用して管理する単一の標準 CloudWatch エージェント設定ファイルがある場合に適しています。CloudWatch 設定をパラメータストアパラメータとして保存すると、CloudWatch エージェント設定ツール (`amazon-cloudwatch-agent-ctl`Linux の場合) を使用して、設定ファイルをインスタンスにコピーすることなく、パラメータストアから設定を読み取って適用できます。AmazonCloudWatch-ManageAgent Systems Manager コマンドドキュメントを使用して、1 回の実行で複数の EC2 インスタンスの CloudWatch 設定を更新できます。Parameter Store パラメータはリージョン別であるため、各 AWS リージョンと AWS アカウントで CloudWatch Parameter Store パラメータを更新および維持する必要があります。各インスタンスに適用する複数の CloudWatch 設定がある場合は、AmazonCloudWatch-ManageAgent コマンドドキュメントをカスタマイズ して、これらのパラメータを含める必要があります。

CloudWatch 設定ファイル

ファイルとしての CloudWatch 設定の管理は、AWS アカウントとリージョンが多数あり、複数の CloudWatch 設定ファイルを管理している場合にうまく機能する可能性があります。このアプローチを使用して、フォルダ構造で参照、整理、管理できます。 セキュリティルールを個々のフォルダまたはファイルに適用して、更新や読み取りのアクセス許可などのアクセスを制限および付与できます。コラボレーションのために AWS の外部で共有および転送できます。ファイルをバージョン管理して、変更を追跡および管理できます。各設定ファイルを個別に適用せずに、設定ファイルを CloudWatch エージェント設定ディレクトリにコピーすることで、CloudWatch 設定をまとめて適用できます。 CloudWatch Linux の場合、CloudWatch 設定ディレクトリは `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d` にあります。。Windows の場合、設定ディレクトリは `C:\ProgramData\Amazon\AmazonCloudWatchAgent\Configs` にあります。

CloudWatch エージェントを起動すると、エージェントは自動的にこれらのディレクトリで見つかった各ファイルを追加し、CloudWatch コンポジットコンフィギュレーションファイルを作成します。設定ファイルは、必要なアカウントとリージョンがアクセスできる中央ロケーション (例えば、S3 バケット) に保存する必要があります。このアプローチを使用したソリューションの例を示します。

CloudWatch 設定の整理

CloudWatch 設定の管理に使用されるアプローチに関係なく、CloudWatch 設定を整理します。次のようなアプローチを使用して、設定をファイルまたは Parameter Store パスに整理できます。

/config/standard/windows/ec2	Amazon EC2 用の Windows 標準の CloudWatch 設定ファイルを保存します。このフォルダでは、さまざまな Windows バージョン、EC2 インスタンスタイプ、環境の標準オペレーティングシステム (OS) 設定をさらに分類できます。
/config/standard/windows/onpremises	オンプレミスサーバー用の Windows 標準の CloudWatch 設定ファイルを保存します。また、このフォルダのさまざまな Windows バージョン、サーバータイプ、環境の標準 OS 設定をさらに分類します。
/config/standard/linux/ec2	Amazon EC2 用の Linux 標準の CloudWatch 設定ファイルを保存します。このフォルダでは、さまざまな Linux ディストリビューション、EC2 インスタンスタイプ、環境の標準 OS 設定をさらに分類できます。
/config/standard/linux/onpremises	オンプレミスサーバー用の Linux 固有の標準的な CloudWatch 設定ファイルを保存します。このフォルダでは、さまざまな Linux ディストリビューション、サーバータイプ、環境の標準 OS 設定をさらに分類できます。
/config/ecs	Amazon ECS コンテナインスタンスを使用する場合は、Amazon Elastic Container Service (Amazon ECS) に固有の CloudWatch 設定ファイルを保存します。これらの設定は、Amazon ECS 固有のシステムレベルのロギングとモニ

	タリングのために、Amazon EC2 の標準設定に追加することができます。
/config/ <application_name>	アプリケーション固有の CloudWatch 設定ファイルを保存します。環境とバージョンの追加のフォルダとプレフィックスを使用して、アプリケーションをさらに分類できます。

例: CloudWatch 設定ファイルを S3 バケットに保存する

このセクションでは、Amazon S3 を使用して CloudWatch 設定ファイルを保存し、カスタム Systems Manager ランブックを使用して CloudWatch 設定ファイルを取得して適用する例を示します。このアプローチでは、CloudWatch 設定に Systems Manager パラメータストアパラメータを大規模に使用するという課題の一部に対処できます。

- 複数のリージョンを使用する場合は、各リージョンの Parameter Store で CloudWatch 設定の更新を同期する必要があります。パラメータストアはリージョンのサービスであり、CloudWatch エージェントを使用する各リージョンで同じパラメータを更新する必要があります。
- 複数の CloudWatch 設定がある場合は、各パラメータストア設定の取得と適用を開始する必要があります。パラメータストアから各 CloudWatch 設定を個別に取得し、新しい設定を追加するたびに取得メソッドを更新する必要があります。対照的に、CloudWatch は、設定ファイルを保存するための設定ディレクトリを提供し、個別に指定する必要なく、ディレクトリ内の各設定を適用します。
- 複数のアカウントを使用する場合は、新しい各アカウントのパラメータストアに必要な CloudWatch 設定があることを確認する必要があります。また、構成の変更が今後これらのアカウントとそのリージョンに適用されていることを確認する必要があります。

CloudWatch 設定は、すべてのアカウントとリージョンからアクセス可能な S3 バケットに保存することができます。その後、Systems Manager オートメーションランブックと Systems Manager、ステートマネージャーを使用して、S3 バケットから CloudWatch 設定ディレクトリにこれらの設定をコピーできます。cloudwatch-config-s3-bucket.yamlAWS CloudFormation テンプレートを使用して、AWS Organizations 内の組織内の複数のアカウントからアクセスできる S3 バケットを作成できます。このテンプレートには、組織 内のすべてのアカウントに読み取りアクセスを許可する `OrganizationID` パラメータが含まれています。

このガイドの「Set up State Manager and Distributor for CloudWatch agent deployment and configuration」セクションにある拡張サンプル Systems Manager ランブックは、cloudwatch-config-s3-bucket.yamlAWS CloudFormation テンプレートによって作成された S3 バケットを使用してファイルを取得するように設定されています。https://github.com/aws-samples/logging-monitoring-apg-guide-examples/blob/main/cloudwatch-config-s3-bucket.yaml

または、バージョン管理システム (GitHub など) を使用して設定ファイルを保存することもできます。バージョン管理システムに保存されている設定ファイルを自動的に取得する場合は、認証情報ストレージを管理または一元化し、アカウントと 全体の認証情報を取得するために使用される Systems Manager Automation ランブックを更新する必要があります AWS リージョン。

EC2 インスタンスとオンプレミスサーバー用の CloudWatch エージェントの設定

多くの組織は、物理的なサーバーと仮想マシン (VM) の両方でワークロードを実行します。通常、これらのワークロードは、メトリクスのキャプチャと取り込みに関する固有のインストールおよび構成要件を持つ異なる OS 上で実行されます。

EC2 インスタンスを使用することを選択した場合、インスタンスと OS の設定を高いレベルで制御できます。ただし、この高いレベルの制御と責任では、より効率的な使用を実現するために、構成をモニタリングおよび調整する必要があります。ロギングとモニタリングの標準を確立し、ログとメトリクスのキャプチャと取り込みのための標準的なインストールと構成のアプローチを適用することで、運用効率を向上させることができます。

IT 投資を AWS クラウドに移行または拡張する組織は、CloudWatch を活用して、統合されたログ記録とモニタリングソリューションを実現できます。CloudWatch 料金は、取得するメトリクスとログに対して段階的に料金を支払うことを意味します。Amazon EC2 の場合と同様の CloudWatch エージェントインストールプロセスを使用して、オンプレミスサーバーのログとメトリクスをキャプチャすることもできます。

CloudWatch のインストールとデプロイを開始する前に、システムとアプリケーションのロギングとメトリクス設定を必ず評価してください。使用する OS のキャプチャに必要な標準ログとメトリクスを定義していることを確認します。システムログとメトリクスは、OS によって生成され、Linux と Windows では異なるため、ロギングおよびモニタリングソリューションの基盤および標準です。Linux バージョンまたはディストリビューションに固有のメトリクスとログファイルに加えて、Linux ディストリビューション全体で利用できる重要なメトリクスとログファイルがあります。この差異は、異なる Windows バージョン間でも発生します。

CloudWatch エージェントを設定します。

CloudWatch は、各 OS に固有の CloudWatch エージェントとエージェント設定ファイル を使用して Amazon EC2 サーバーとオンプレミスサーバーのメトリクスとログをキャプチャします。CloudWatch エージェントをアカウントに大規模にインストールする前に、組織の標準メトリクスとログキャプチャ設定を定義することをお勧めします。

複数の CloudWatch エージェント設定を組み合わせて、複合 CloudWatch エージェント設定を構成できます。推奨されるアプローチの 1 つは、システムレベルとアプリケーションレベルでログとメト

リクスの構成を定義して分割することです。次の図は、異なる要件に対する複数の CloudWatch 設定ファイルタイプを組み合わせて、複合 CloudWatch 設定を形成する方法を示しています。

これらのログとメトリクスは、特定の環境や要件に合わせてさらに分類して構成することもできます。例えば、規制されていない開発環境では低い精度でログとメトリクスの小さなサブセットを定義し、規制された本番環境ではより精度の高い大規模で完全なセットを定義できます。

EC2 インスタンスのログキャプチャの設定

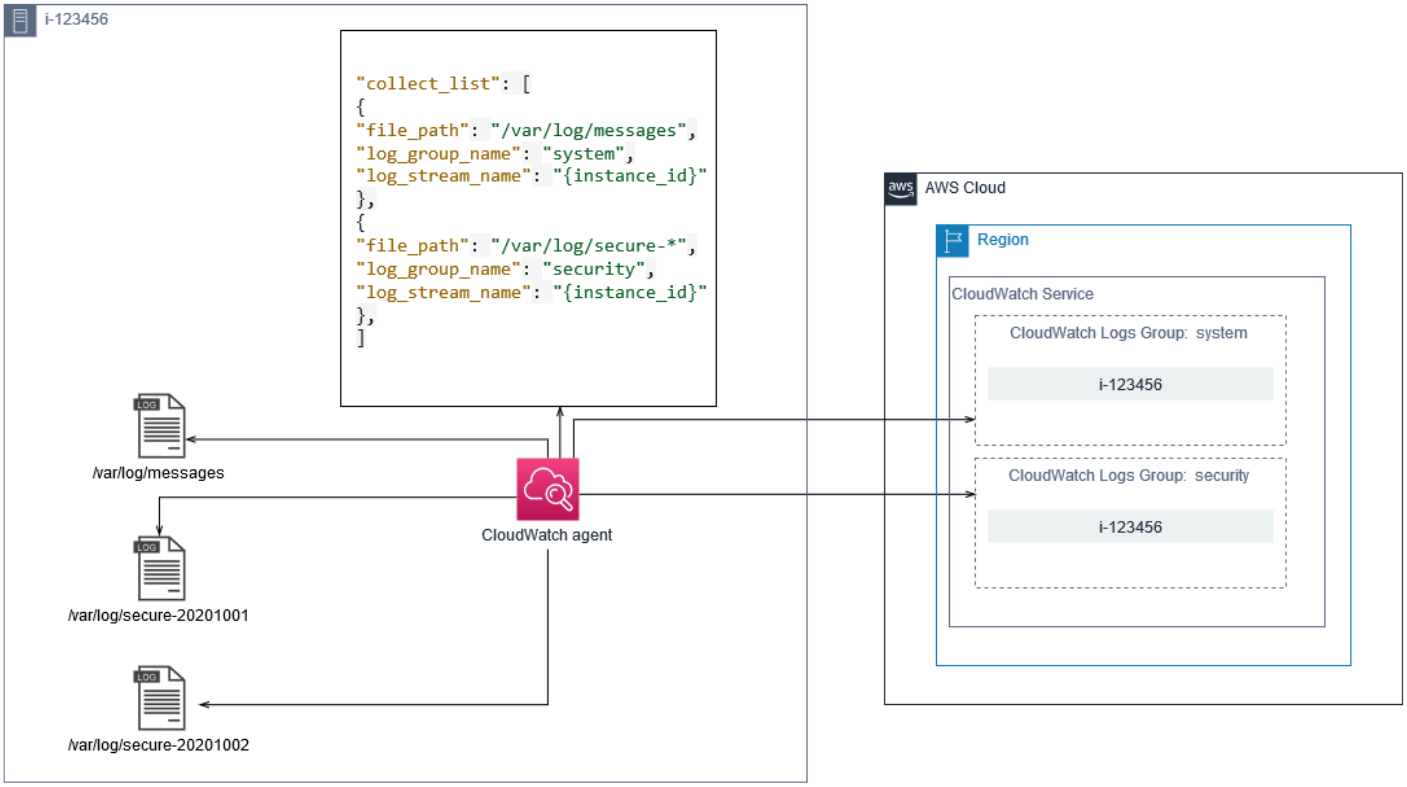
デフォルトでは、Amazon EC2 はログファイルをモニタリングまたはキャプチャしません。代わりに、EC2 インスタンス、 AWS API、または AWS Command Line Interface () にインストールされた CloudWatch エージェントソフトウェアによってログファイルがキャプチャされ、CloudWatch Logs に取り込まれますAWS CLI。CloudWatch エージェントを使用して、Amazon EC2 とオンプレミスサーバーの CloudWatch Logs にログファイルを取り込むことをお勧めします。

CloudWatch のログファイルからのパターンパッチ適用に基づいて、ログを検索してフィルタリングしたり、メトリクスを抽出したり、自動化を実行したりできます。CloudWatch は、プレーンテキスト、スペース区切り、および JSON 形式のフィルタおよびパターン構文オプションをサポートし、JSON 形式のログが最も柔軟になります。フィルタリングと分析オプションを増やすには、プレーンテキストではなくフォーマットされたログ出力を使用する必要があります。

CloudWatch エージェントは、CloudWatch に送信するログとメトリクスを定義する設定ファイルを使用します。CloudWatch は、各ログファイルを ログストリーム としてキャプチャし、これらのログストリームを ロググループ にグループ化します。これにより、一致する文字列の検索など、EC2 インスタンスからのログ間でオペレーションを実行できます。

デフォルトのログストリーム名は EC2 インスタンス ID と同じで、デフォルトのロググループ名はログファイルのパスと同じです。ログストリーム名は CloudWatch ロググループ内で一意である必要があります。ログストリームとロググループ名の動的置換の場合、`instance_id`、`hostname`、`local_hostname`、または `ip_address` を使用できます。つまり、複数の EC2 インスタンス間で同じ CloudWatch エージェント設定ファイルを使用できます。

次の図は、ログをキャプチャするための CloudWatch エージェント設定を示しています。ロググループは、キャプチャされたログファイルによって定義され、EC2 インスタンスごとに個別のログストリームが含まれます。`{instance_id}` 変数はログストリーム名に使用され、EC2 インスタンス ID は一意です。



ロググループは、それらに含まれるログストリームの保存期間、タグ、セキュリティ、メトリクスフィルタ、および検索範囲を定義します。ログファイル名に基づくデフォルトのグループ化動作は、アカウントとリージョン内の EC2 インスタンス間でログファイルに固有のデータの検索、メトリクスの作成、およびアラームに役立ちます。ロググループの細分化が必要かどうかを評価する必要があります。例えば、アカウントが複数のビジネスユニットによって共有され、異なる技術所有者またはオペレーションの所有者がいる場合があります。つまり、分離と所有権を反映するように、ロググループ名をさらに絞り込む必要があります。このアプローチにより、関連する EC2 インスタンスに分析とトラブルシューティングを集中させることができます。

複数の環境で 1 つのアカウントを使用する場合は、各環境で実行されるワークロードのログ記録を分けることができます。次の表に、ビジネスユニット、プロジェクトまたはアプリケーション、および環境を含むロググループの命名規則を示します。

ロググループ名	/<Business unit>/<Project or application name>/<Environment>/<Log file name>
ログストリーム名	<EC2 instance ID>

EC2 インスタンスのすべてのログファイルを同じロググループにグループ化することもできます。これにより、単一の EC2 インスタンスについて一連のログファイルを検索および分析することが容易になります。これは、ほとんどの EC2 インスタンスが 1 つのアプリケーションまたはワークロードを処理し、各 EC2 インスタンスが特定の目的を果たす場合に便利です。次の表に、この方法をサポートするようにロググループとログストリームの名前をフォーマットする方法を示します。

ロググループ名	`/<Business unit>/<Project or application name>/<Environment>/<EC2 instance ID>`
ログストリーム名	`<Log file name>`

EC2 インスタンスのメトリクスキャプチャの設定

デフォルトでは、EC2 インスタンスで基本モニタリングが有効になり、標準メトリクスセット (CPU、ネットワーク、ストレージ関連のメトリクスなど) は、5 分ごとに自動的に CloudWatch に送信されます。CloudWatch メトリクスは、インスタンスファミリーによって異なる場合があります。例えば、バーストパフォーマンスインスタンス は CPU クレジットのメトリクスがあります。Amazon EC2 標準メトリクスは、インスタンス料金に含まれます。EC2 インスタンスで 詳細モニタリング を有効にすると、1 分間隔でデータを受信できます。期間の頻度が CloudWatch のコストに影響するため、EC2 インスタンスのすべてまたは一部のみに詳細モニタリングが必要かどうかを評価してください。例えば、実稼働ワークロードの詳細モニタリングを有効にし、非運用ワークロードには基本モニタリングを使用できます。

オンプレミスサーバーには CloudWatch のデフォルトのメトリクスが含まれていないため AWS CLI、CloudWatch エージェント、、または AWS SDK を使用してメトリクスをキャプチャする必要があります。つまり、CloudWatch 設定ファイルでキャプチャするメトリクス (CPU 使用率など) を定義する必要があります。オンプレミスサーバーの標準 EC2 インスタンスメトリクスを含む一意の CloudWatch 設定ファイルを作成し、標準の CloudWatch 設定に加えて適用できます。

CloudWatch の メトリクス では、メトリクス名とゼロ以上のディメンションで一意に定義され、メトリクス名前空間に一意にグループ化されます。 AWS サービスによって提供されるメトリクスには、 で始まる名前空間 `AWS` (などAWS/EC2) があり、非AWS メトリクスはカスタムメトリクスと見なされます。CloudWatch エージェントで設定してキャプチャするメトリクスは、すべてカスタムメトリクスと見なされます。作成されたメトリクスの数は CloudWatch のコストに影響を与えるため、各メトリクスがすべての EC2 インスタンスまたは一部にのみ必要かどうかを評価する必要がありま

す。例えば、実稼働ワークロードのメトリクスの完全なセットを定義し、非運用ワークロードにはこれらのメトリクスの小さなサブセットを使用できます。

CWAgent は、CloudWatch エージェントによって公開されるメトリクスのデフォルトの名前空間です。ロググループと同様に、メトリクス名前空間は一連のメトリクスを整理して、それらを 1 か所でまとめて見つけることができます。ビジネスユニット、プロジェクト、アプリケーション、および環境 (例えば、/<Business unit>/<Project or application name>/<Environment>) を反映するように名前空間を変更する必要があります。このアプローチは、複数の無関係なワークロードが同じアカウントを使用する場合に便利です。また、名前空間の命名規則を CloudWatch ロググループの命名規則に関連付けることもできます。

指標はディメンションによって識別され、一連の条件に対して分析するのに役立ち、観測値が記録されるプロパティです。Amazon EC2 には EC2 インスタンス用の 個別のメトリクス が InstanceId およびAutoScalingGroupName ディメンションで含まれます。また、詳細モニタリングを有効にする場合、ImageId および InstanceType ディメンションでメトリクスを受け取ります。例えば、Amazon EC2 は、InstanceId ディメンション用の別の CPU 使用率メトリクスに加えて、InstanceType ディメンション CPU 使用率に関する別個の EC2 インスタンスメトリクスを提供します。これにより、固有のインスタンスタイプ のすべての EC2 インスタンスに加えて、一意の EC2 インスタンスの CPU 使用率を分析できます。

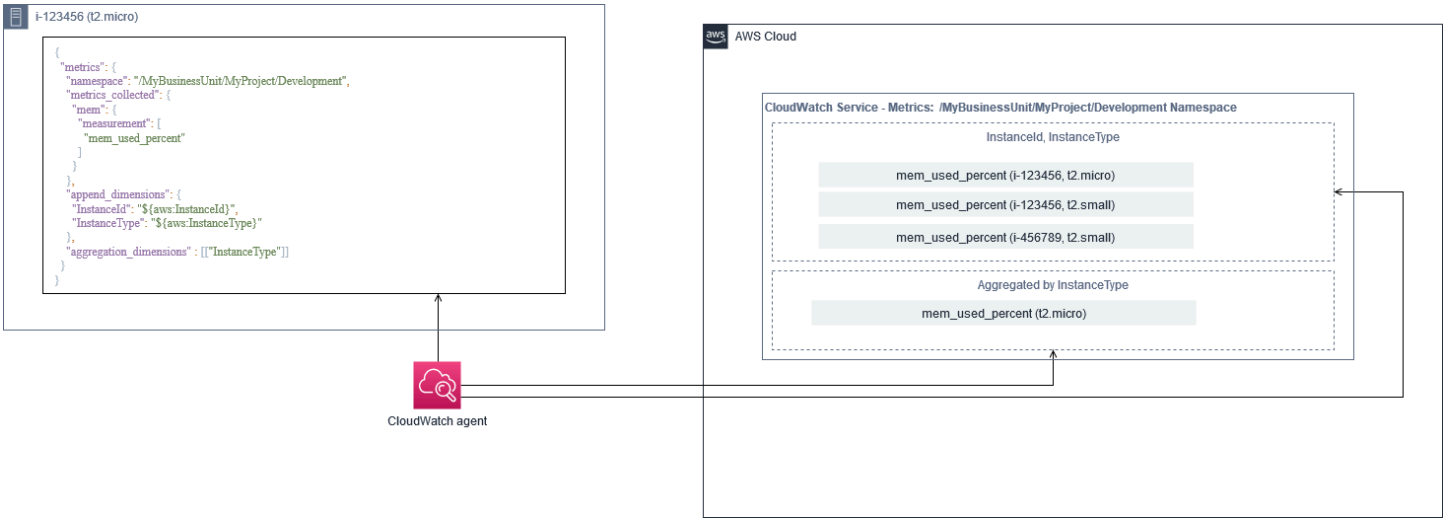
ディメンションを追加すると、分析能力は向上しますが、全体的なコストも増加します。これは、各メトリクスと固有のディメンション値の組み合わせによって新しいメトリクスが作成されるためです。例えば、InstanceId ディメンションに対してメモリ使用率のメトリクスを作成した場合、これは各 EC2 インスタンスの新しいメトリクスです。組織が数千の EC2 インスタンスを実行している場合、これにより数千のメトリクスが発生し、コストが高くなります。コストを制御および予測するには、メトリクスのカーディナリティと、最も価値の高いディメンションを決定する必要があります。例えば、実稼働ワークロードメトリクスのディメンションの完全なセットを定義し、非本番ワークロードではこれらのディメンションの小さなサブセットを定義できます。

CloudWatch 設定で定義された 1 つまたはすべてのメトリクスにディメンションを追加する append_dimensions プロパティを使用できます。また、ImageId、InstanceId、InstanceType、および AutoScalingGroupName を CloudWatch 設定のすべてのメトリクスに動的に追加することもできます。または、append_dimensions そのメトリクスのプロパティを使用することで、特定のメトリクスに任意のディメンション名と値を追加することもできます。。CloudWatch は、aggregation_dimensions プロパティで定義したメトリクスディメンションに関する統計を集計することもできます。

例えば、InstanceType ディメンションに使用されたメモリを集計できるので、インスタンスタイプごとにすべての EC2 インスタンスが使用している平均メモリを確認します。t2.micro リージョ

ンで実行されているインスタンスを使用すると、t2.micro クラスを使用しているワークロードが提供されたメモリを過度に利用している、または過小利用しているかを判断できます。使用率の低下は、不要なメモリ容量を持つ EC2 クラスを使用するワークロードの兆候である可能性があります。対照的に、過剰使用率は、メモリ不足の Amazon EC2 クラスを使用するワークロードの兆候である可能性があります。

次の図は、InstanceType によってカスタム名前空間、追加されたディメンション、および集計を使用する CloudWatch メトリクス設定のサンプルを示しています。



CloudWatch システムレベルの設定

システムレベルのメトリクスとログは、モニタリングおよびロギングソリューションの中心的なコンポーネントであり、CloudWatch エージェントには Windows および Linux 用の特定の設定オプションがあります。

サポートする予定の各 OS の CloudWatch エージェント設定ファイルを定義するために、CloudWatch 設定ファイルウィザード または構成ファイルスキーマを使用することをお勧めします。追加のワークロード固有の OS レベルのログとメトリクスを個別の CloudWatch 設定ファイルで定義し、標準設定に追加できます。これらの固有の設定ファイルは、EC2 インスタンスで取得できる S3 バケットに別々に保存する必要があります。この目的のための S3 バケットの設定の例については、本ガイドの「CloudWatch 設定の管理 」セクションを参照してください。State Manager とディストリビューターを使用して、これらの構成を自動的に取得して適用できます。

システムレベルのログの設定

システムレベルのログは、オンプレミスまたは AWS クラウドの問題の診断とトラブルシューティングに不可欠です。ログキャプチャアプローチには、OS によって生成されたすべてのシステムログ

とセキュリティログを含める必要があります。OS が生成するログファイルは、OS のバージョンによって異なる場合があります。

CloudWatch エージェントは、イベントログ名を指定して Windows イベントログのモニタリングをサポートします。モニタリングする Windows イベントログ (例えば、`System`、`Application`、または`Security`) を選択できます。

Linux システムのシステム、アプリケーション、およびセキュリティログは、通常 `/var/log` ディレクトリに保存されます。次の表は、モニタリングする必要がある一般的なデフォルトログファイルを定義していますが、`/etc/rsyslog.conf` または `/etc/syslog.conf` ファイルを使用して、システムのログファイルの特定の設定を判別します。

Fedora ディストリビューション (Amazon Linux、CentOS、Red Hat Enterprise Linux の場合)	`/var/log/boot.log*` — ブートアップログ
	`/var/log/dmesg` — カーネルログ
	`/var/log/secure` – セキュリティと認証
	`/var/log/messages` — 一般的なシステムログ
	`/var/log/cron*` — Cron ログ
	`/var/log/cloud-init-output.log` — Userdata スタートアップスクリプトからの出力
Debian (Ubuntu)	`/var/log/syslog` — ブートアップログ
	`/var/log/cloud-init-output.log` — Userdata スタートアップスクリプトからの出力
	`/var/log/auth.log` – セキュリティと認証
	`/var/log/kern.log` — カーネルログ

組織には、モニタリングするログを生成する他のエージェントまたはシステムコンポーネントがある場合もあります。これらのエージェントまたはアプリケーションによって生成されるログファイルを評価して決定し、ファイルの場所を特定して構成に含める必要があります。例えば、Systems Manager と CloudWatch エージェントログを設定に含める必要があります。次の表に、Windows および Linux のこれらのエージェントログの場所を示します。

Windows	CloudWatch エージェント	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log`

	Systems Manager Agent	`%PROGRAMDATA%\Amazon\SSM\Logs\amazon-ssm-agent.log`
		
		`%PROGRAMDATA%\Amazon\SSM\Logs\errors.log`
		
		`%PROGRAMDATA%\Amazon\SSM\Logs\audits\amazon-ssm-agent-audit-YYYY-MM-DD`

Linux	CloudWatch エージェント	`/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`

	Systems Manager Agent	`/var/log/amazon/ssm/amazon-ssm-agent.log`
		
		`/var/log/amazon/ssm/errors.log`
		
		`/var/log/amazon/ssm/audits/amazon-ssm-`

		agent-audit-YYYY-MM-DD

CloudWatch は、ログファイルが CloudWatch エージェント設定で定義されていても見つからない場合、ログファイルを無視します。これは、ディストリビューションごとに個別の構成ではなく、Linux 用の単一のログ構成を維持する場合に便利です。また、エージェントまたはソフトウェアアプリケーションの実行が開始されるまでログファイルが存在しない場合にも役立ちます。

システムレベルのメトリクスを設定する

メモリとディスク領域の使用率は、Amazon EC2 が提供する標準メトリクスには含まれません。これらのメトリクスを含めるには、EC2 インスタンスに CloudWatch エージェントをインストールして設定する必要があります。CloudWatch エージェント設定ウィザードは 事前定義されたメトリクス で CloudWatch 設定を作成し、必要に応じて、メトリクスを追加または削除することができます。事前に定義されたメトリクスセットを確認して、必要な適切なレベルを決定してください。

エンドユーザーとワークロード所有者は、サーバーまたは EC2 インスタンスの特定の要件に基づいて、追加のシステムメトリクスを公開する必要があります。これらのメトリクス定義は、別の CloudWatch エージェント設定ファイルに保存、バージョン管理、維持され、再利用と自動化のために中央の場所 (Amazon S3 など) で共有される必要があります。

標準の Amazon EC2 メトリクスは、オンプレミスサーバーでは自動的にキャプチャされません。これらのメトリクスは、オンプレミスインスタンスで使用される CloudWatch エージェント設定ファイルに定義される必要があります。CPU 使用率などのメトリクスを使用して、オンプレミスインスタンス用に個別のメトリクス設定ファイルを作成し、これらのメトリクスを標準のメトリクス設定ファイルに追加できます。

アプリケーションレベルの CloudWatch 設定

アプリケーションログとメトリクスは、実行中のアプリケーションによって生成され、アプリケーション固有です。組織で定期的に使用されるアプリケーションを適切にモニタリングするために必要なログとメトリクスを定義してください。例えば、組織が Web ベースのアプリケーション用の Microsoft インターネットインフォメーションサーバー (IIS) で標準化している場合があります。IIS の標準的なログとメトリクス CloudWatch 構成を作成して、組織全体で使用することもできます。アプリケーション固有の設定ファイルは、一元化された場所 (S3 バケットなど) に格納され、ワークロード所有者または自動取得によってアクセスされ、CloudWatch 設定ディレクトリにコピーされます。CloudWatch エージェントは、各 EC2 インスタンスまたはサーバーの設定ファイルディレ

クトリにある CloudWatch 設定ファイルをコンポジット CloudWatch 設定に自動的に結合します。その結果、組織の標準的なシステムレベルの設定と、関連するすべてのアプリケーションレベルの CloudWatch 設定を含む CloudWatch 設定が作成されます。

ワークロードの所有者は、すべての重要なアプリケーションおよびコンポーネントのログファイルとメトリクスを特定して構成する必要があります。

アプリケーションレベルのログの設定

アプリケーションレベルのロギングは、アプリケーションが商用オフザシェルフ (COTS) またはカスタム開発アプリケーションのどちらであるかによって異なります。COTS アプリケーションとそのコンポーネントは、ログの詳細レベル、ログファイル形式、ログファイルの場所など、ログの構成と出力に関するいくつかのオプションを提供します。ただし、ほとんどの COTS またはサードパーティアプリケーションは、ロギング (例えば、アプリケーションのコードを更新して、構成できない追加のログステートメントまたは形式を含めるなど) を根本的に変更することを許可しません。少なくとも、警告およびエラーレベルの情報 (できれば JSON 形式) をログに記録するには、COTS またはサードパーティアプリケーションのロギングオプションを設定する必要があります。

CloudWatch 設定にアプリケーションのログファイルを含めることで、カスタム開発したアプリケーションを CloudWatch Logs と統合できます。カスタムアプリケーションを使用すると、ログ出力形式をカスタマイズしたり、コンポーネント出力を個別のログファイルに分類したり、必要な詳細を追加したりできるため、ログの品質と制御が向上します。分析と処理が簡単になるように、ロギングライブラリと組織に必要なデータと書式を必ず確認して標準化してください。

CloudWatch Logs `PutLogEvents` API コールまたは AWS SDK を使用して CloudWatch ログストリームに書き込むこともできます。API または SDK は、分散した一連のコンポーネントおよびサーバーにわたる単一のログストリームへのロギングを調整するなど、カスタムロギング要件に使用できます。ただし、保守が最も簡単で広く適用可能なソリューションは、ログファイルに書き込むようにアプリケーションを設定し、CloudWatch エージェントを使用してログファイルを読み取り、CloudWatch にストリーミングすることです。

また、アプリケーションログファイルから測定するメトリクスの種類も考慮する必要があります。メトリクスフィルターを使用して、CloudWatch ロググループでこのデータを測定、グラフ化、アラームできます。例えば、メトリクスフィルターを使用して、ログで失敗したログイン試行を特定してカウントできます。

アプリケーションログファイルで CloudWatch の埋め込みメトリクス 形式 を使用して、独自のアプリケーションメトリクスを作成することも可能です。

アプリケーションレベルのメトリクスを設定する

カスタムメトリクスは、 AWS サービスによって CloudWatch に直接提供されず、CloudWatch メトリクスのカスタム名前空間で公開されるメトリクスです。すべてのアプリケーションメトリクスは、カスタム CloudWatch メトリクスと見なされます。アプリケーションメトリクスは、EC2 インスタンス、アプリケーションコンポーネント、API コール、またはビジネス関数に揃える場合があります。また、メトリクスで選択したディメンションの重要性とカーディナリティも考慮する必要があります。カーディナリティの高いディメンションは、多数のカスタムメトリクスを生成し、CloudWatch のコストを増加させる可能性があります。

CloudWatch は、以下を含む複数の方法でアプリケーションレベルのメトリクスをキャプチャするのに役立ちます。

- procstat プラグイン からキャプチャする個々のプロセスを定義して、プロセスレベルのメトリクスを取得します。
- アプリケーションは Windows パフォーマンスモニターにメトリクスを公開し、このメトリクスは CloudWatch の設定で定義されます。
- メトリクスフィルターとパターンは CloudWatch 内のアプリケーションのログに適用されます。
- アプリケーションは、CloudWatch 組み込みメトリクス形式を使用して CloudWatch ログに書き込みます。
- アプリケーションは API または AWS SDK を介して CloudWatch にメトリクスを送信します。
- アプリケーションは、CloudWatch エージェントが設定された 収集された または statsD デーモンにメトリクスを送信します。

procstat を使用して、CloudWatch エージェントで重要なアプリケーションプロセスをモニタリングおよび測定できます。これにより、アプリケーションで重要なプロセスが実行されなくなった場合に、アラームを発生させ、アクション (通知や再起動プロセスなど) を実行するのに役立ちます。また、アプリケーションプロセスのパフォーマンス特性を測定し、特定のプロセスが異常動作している場合にアラームを発生させることもできます。

Procstat モニタリングは、追加のカスタムメトリクスを使用して COTS アプリケーションを更新できない場合にも役立ちます。例えば、`my_process` を測定しカスタム `cpu_time` ディメンションを含む `application_version` メトリクスを作成できます。メトリクスごとに異なるディメンションがある場合は、アプリケーションに対して複数の CloudWatch エージェント設定ファイルを使用することもできます。

アプリケーションが Windows で実行されている場合は、すでに Windows パフォーマンスモニターにメトリクスを公開しているかどうかを評価する必要があります。多くの COTS アプリケーションは Windows パフォーマンスモニタと統合されているため、アプリケーションのメトリクスを簡単にモニタリングできます。CloudWatch は Windows パフォーマンスモニターとも統合され、すでに利用可能なメトリクスをキャプチャできます。

アプリケーションによって提供されるロギング形式とログ情報を確認して、メトリクス・フィルタを使用して抽出できるメトリクスを判別してください。アプリケーションの履歴ログを確認して、エラーメッセージと異常シャットダウンがどのように表示されるかを判断できます。また、以前に報告された問題を確認して、問題が繰り返し発生しないようにメトリクスを取得できるかどうかを判断する必要があります。また、アプリケーションのドキュメントを確認し、アプリケーション開発者にエラーメッセージの識別方法を確認するよう依頼する必要があります。

カスタム開発アプリケーションの場合は、アプリケーションの開発者と協力して、CloudWatch 埋め込みメトリクス形式、 AWS SDK、または AWS API を使用して実装できる重要なメトリクスを定義します。推奨されるアプローチは、埋め込みメトリクスフォーマットを使用することです。必要な形式でステートメントを記述するのに役立つように、 AWS 提供されたオープンソースの組み込みメトリクス形式のライブラリを使用できます。また、アプリケーション固有の CloudWatch 設定 を更新して、埋め込みメトリクスフォーマットエージェントを含める必要があります。これにより、EC2 インスタンスで実行されているエージェントは、組み込みメトリクス形式のメトリクスを CloudWatch に送信するローカル埋め込みメトリクス形式のエンドポイントとして機能します。

アプリケーションが collectd または statsd へのメトリクスの公開をすでにサポートしている場合は、それらを活用して CloudWatch にメトリクスを取り込むことができます。

Amazon EC2 およびオンプレミスサーバーに対する CloudWatch エージェントのインストールアプローチ

CloudWatch エージェントのインストールプロセスを自動化することで、迅速かつ一貫してデプロイし、必要なログとメトリクスをキャプチャできます。CloudWatch エージェントのインストールを自動化するには、マルチアカウントやマルチリージョンのサポートなど、いくつかのアプローチがあります。次の自動インストール方法について説明します。

- Systems Manager ディストリビューターおよび Systems Manager State Manager を使用して CloudWatch エージェントをインストールする - EC2 インスタンスとオンプレミスサーバーで Systems Manager Agent を実行している場合は、この方法を使用することをお勧めします。これにより、CloudWatch エージェントが最新の状態に保たれ、CloudWatch エージェントを持たないサーバーについてレポートして修正できるようになります。このアプローチは、複数のアカウントとリージョンをサポートするように拡張することもできます。
- EC2 インスタンスのプロビジョニング中にユーザーデータスクリプトの一部として CloudWatch エージェントをデプロイする - Amazon EC2 では、初回起動または再起動時に実行される起動スクリプトを定義できます。スクリプトを定義して、エージェントのダウンロードおよびインストールプロセスを自動化できます。これは、 CloudFormation スクリプトと AWS Service Catalog 製品に含めることもできます。このアプローチは、標準とは異なる特定のワークロードに対してカスタマイズされたエージェントのインストールと構成のアプローチがある場合は、必要に応じて適切です。
- CloudWatch エージェントを Amazon マシンイメージ (AMI) に含める - Amazon EC2 のカスタム AMI に CloudWatch エージェントをインストールできます。AMI を使用する EC2 インスタンスには、エージェントが自動的にインストールされ、開始されます。ただし、エージェントとその構成が定期的に更新されていることを確認する必要があります。

Systems Manager ディストリビューターとステートマネージャーを使用して CloudWatch エージェントをインストールする

Systems Manager のステートマネージャーを Systems Manager のディストリビューターとともに使用して、サーバーおよび EC2 インスタンスに CloudWatch エージェントを自動的にインストールおよび更新できます。Distributor には、最新の CloudWatch エージェントバージョンをインストールする `AmazonCloudWatchAgent` AWS マネージドパッケージが含まれています。

このインストール方法には、次のような前提条件があります。

- Systems Manager エージェントは、サーバーまたは EC2 インスタンスにインストールして実行されている必要があります。Systems Manager エージェントは Amazon Linux、Amazon Linux 2、および一部の AMI にプレインストールされています。エージェントは、他のイメージまたはオンプレミスの仮想マシンおよびサーバーにインストールして構成する必要があります。

  > **Note**
  >
  > Amazon Linux 2 のサポート終了が近づいています。詳細については、「Amazon Linux 2 のFAQs」を参照してください。

- IAM ロールまたは 必要な CloudWatch と Systems Manager アクセス許可 を持つ認証情報は、EC2 インスタンスにアタッチされるか、オンプレミスサーバーの認証情報ファイルに定義される必要があります。たとえば、`AmazonSSMManagedInstanceCore`Systems Manager 用と CloudWatch `CloudWatchAgentServerPolicy`用の AWS マネージドポリシーを含む IAM ロールを作成できます。ssm-cloudwatch-instance-role.yaml CloudFormation テンプレートを使用して、これらのポリシーの両方を含む IAM ロールとインスタンスプロファイルをデプロイできます。このテンプレートを変更して、EC2 インスタンスに対する他の標準 IAM アクセス権限を含めることもできます。オンプレミスサーバーまたは VM の場合は、オンプレミスサーバー用に構成されている Systems Manager のサービスロール を使用して、CloudWatch エージェントを設定する必要があります。詳細については、 AWS ナレッジセンターの「Systems Manager Agent と統合 CloudWatch エージェントを使用するオンプレミスサーバーで一時的な認証情報のみを使用するように設定するにはどうすればよいですか?」を参照してください。

次のリストに、Systems Manager ディストリビュータおよびステートマネージャのアプローチを使用して CloudWatch エージェントをインストールおよび保守する場合に、いくつかの利点があります。

- 複数の OS の自動インストール - CloudWatch エージェントをダウンロードしてインストールするために、OS ごとにスクリプトを作成して保守する必要はありません。
- 自動更新チェック - State Manager は、各 EC2 インスタンスに最新の CloudWatch バージョンがあることを、自動的かつ定期的にチェックします。
- コンプライアンスレポート - Systems Manager コンプライアンスダッシュボードには、ディストリビューターパッケージのインストールに失敗した EC2 インスタンスが示されます。
- 新しく起動された EC2 インスタンスの自動インストール - アカウントに起動された新しい EC2 インスタンスは CloudWatch エージェントを自動的に受け取ります。

ただし、この方法を選択する前に、次の 3 つの領域も考慮する必要があります。

- 既存の関連付けとの衝突 - 別のアソシエーションがすでに CloudWatch エージェントをインストールまたは設定している場合、2 つの関連付けが相互に干渉し、問題を引き起こす可能性があります。この方法を使用する場合は、CloudWatch エージェントと設定をインストールまたは更新する既存の関連付けを削除する必要があります。
- カスタムエージェント設定ファイルの更新 - ディストリビュータは、デフォルトの設定ファイルを使用してインストールを実行します。カスタム設定ファイルまたは複数の CloudWatch 設定ファイルを使用する場合は、インストール後に設定を更新する必要があります。
- マルチリージョンまたはマルチアカウントの設定 - ステートマネージャーの関連付けは、各アカウントとリージョンで設定する必要があります。マルチアカウント環境の新しいアカウントは、ステートマネージャーの関連付けを含めるように更新する必要があります。複数のアカウントとリージョンが必要な標準を取得して適用できるように、CloudWatch 設定を集中化または同期する必要があります。

CloudWatch エージェントのデプロイと設定のステートマネージャーとディストリビューターをセットアップする

Systems Manager 高速セットアップ を使用して、EC2 インスタンスに CloudWatch エージェントを自動的にインストールおよび更新するなど、Systems Manager 機能をすばやく設定することができます。高速セットアップは、選択した内容に基づいて Systems Manager リソースをデプロイして設定する CloudFormation スタックをデプロイします。

次のリストに、CloudWatch エージェントの自動インストールと更新のために Quick Setup によって実行される 2 つの重要なアクションを示します。

1. Systems Manager のカスタムドキュメントを作成する – クイックセットアップは、ステートマネージャで使用するために、次の Systems Manager ドキュメントを作成します。ドキュメント名は異なる場合がありますが、内容は同じままです。
   - `CreateAndAttachIAMToInstance` - それらが存在しない場合は、`AmazonSSMRoleForInstancesQuickSetup` ロールとインスタンスプロファイルを作成し、ロールに `AmazonSSMManagedInstanceCore` ポリシーを付与します。これには、必要な `CloudWatchAgentServerPolicy` IAM ポリシーは含まれません。このポリシーを更新し、次のセクションで説明するように、このポリシーを含めるようにこの Systems Manager ドキュメントを更新する必要があります。

- `InstallAndManageCloudWatchDocument` - CloudWatch エージェントをディストリビューターとともにインストールし、`AWS-ConfigureAWSPackage` Systems Manager のドキュメントを使用して、デフォルトの CloudWatch エージェント設定で各 EC2 インスタンスを 1 回設定します。
- `UpdateCloudWatchDocument` - `AWS-ConfigureAWSPackage` Systems Manager のドキュメントを使用して、最新の CloudWatch エージェントをインストールして CloudWatch エージェントを更新します。エージェントを更新またはアンインストールしても、EC2 インスタンスから既存の CloudWatch 設定ファイルは削除されません。

2. ステートマネージャーの関連付けを作成する - ステートマネージャの関連付けは、カスタム作成された Systems Manager ドキュメントを使用するように作成および構成されます。ステートマネージャーの関連付け名は異なる場合がありますが、設定は同じままです。
   - `ManageCloudWatchAgent` - EC2 インスタンスごとに `InstallAndManageCloudWatchDocument` Systems Manager ドキュメントを 1 回ずつ実行します。
   - `UpdateCloudWatchAgent` - 各 EC2 インスタンスについて 30 日ごとに `UpdateCloudWatchDocument` Systems Manager ドキュメントを実行します。
   - EC2 インスタンスごとに 1 回ずつ `CreateAndAttachIAMToInstance` Systems Manager を実行します。

CloudWatch アクセス許可を含めて、カスタム CloudWatch 設定をサポートするには、完了したクイックセットアップ設定を拡張およびカスタマイズする必要があります。特に、`CreateAndAttachIAMToInstance` と `InstallAndManageCloudWatchDocument` ドキュメントを更新する必要があります。クイックセットアップで作成された Systems Manager ドキュメントを手動で更新できます。または、独自の CloudFormation テンプレートを使用して、必要な更新を使用して同じリソースをプロビジョニングし、他の Systems Manager リソースを設定してデプロイし、クイックセットアップを使用しないこともできます。

> **Important**
>
> 高速セットアップは、選択した内容に基づいて Systems Manager リソースをデプロイおよび設定する CloudFormation スタックを作成します。クイックセットアップの選択肢を更新する場合は、Systems Manager のドキュメントを手動で再更新する必要がある場合があります。

以下のセクションでは、クイックセットアップによって作成された Systems Manager リソースを手動で更新する方法と、独自の CloudFormation テンプレートを使用して更新されたクイックセットアップを実行する方法について説明します。クイックセットアップと によって作成されたリソースを手動で更新しないように、独自の CloudFormation テンプレートを使用することをお勧めします CloudFormation。

Systems Manager のクイックセットアップを使用して、作成した Systems Manager のリソースを手動で更新します。

クイックセットアップアプローチで作成された Systems Manager リソースを更新して、必要な CloudWatch エージェントのアクセス許可を含めて、複数の CloudWatch 設定ファイルをサポートする必要があります。このセクションでは、IAM ロールと Systems Manager ドキュメントを更新して、複数のアカウントからアクセスできる CloudWatch 設定を含む一元化された S3 バケットを使用する方法について説明します。CloudWatch 設定ファイルを保存するための S3 バケットの作成については、このガイドの「CloudWatch 設定の管理」セクションを参照してください。

CreateAndAttachIAMToInstance Systems Manager ドキュメントの更新

Quick Setup によって作成されたこの Systems Manager ドキュメントは、EC2 インスタンスに既存の IAM インスタンスプロファイルがアタッチされているかどうかをチェックします。もしそうなら、それは既存のロールに `AmazonSSMManagedInstanceCore` ポリシーを付与します。これにより、既存の EC2 インスタンスは、既存のインスタンスプロファイルを通じて割り当てられる可能性のある AWS アクセス許可を失うことがなくなります。インスタンスプロファイルが既にアタッチされている EC2 インスタンスに対する `CloudWatchAgentServerPolicy` IAM ポリシーをアタッチするには、このドキュメントにステップを追加する必要があります。Systems Manager ドキュメントでは、IAM ロールが存在せず、EC2 インスタンスにインスタンスプロファイルがアタッチされていない場合、IAM ロールも作成されます。お客様は `CloudWatchAgentServerPolicy` IAM ポリシーも含むためにドキュメントのこのセクションを更新する必要があります。

完了した CreateAndAttachIAMToInstance.yaml サンプルドキュメントを確認し、クイックセットアップで作成されたドキュメントと比較します。既存のドキュメントを編集して、必要な手順と変更を含めます。クイックセットアップの選択肢に基づいて、クイックセットアップで作成されたドキュメントは、提供されたサンプルドキュメントとは異なる場合があります。そのため、必要な調整を行ってください。サンプルドキュメントには、欠落しているパッチを毎日スキャンするためのクイックセットアップオプションの選択肢が含まれています。そのため、Systems Manager パッチマネージャーのポリシーが含まれています。

`InstallAndManageCloudWatchDocument` Systems Manager のドキュメントの更新

クイックセットアップによって作成されたこの Systems Manager ドキュメントは CloudWatch エージェントをインストールし、デフォルトの CloudWatch エージェント設定で設定します。デフォルトの CloudWatch 設定は、基本的な事前定義されたメトリクスセットにアライメントされます。デフォルトの設定ステップを置き換え、CloudWatch 設定 S3 バケットから CloudWatch 設定ファイルをダウンロードするステップを追加する必要があります。

完了した InstallAndManageCloudWatchDocument.yaml の更新されたドキュメントを確認し、クイックセットアップで作成されたドキュメントと比較します。クイックセットアップで作成されたドキュメントは異なる場合があるため、必要な調整を行っていることを確認してください。既存のドキュメントを編集して、必要な手順と変更を含めます。

高速セットアップ CloudFormation の代わりに を使用する

高速セットアップを使用する代わりに、 CloudFormation を使用して Systems Manager を設定できます。この方法では、特定の要件に従って Systems Manager の設定をカスタマイズできます。この方法では、カスタムの CloudWatch 設定をサポートするために Quick Setup によって作成された設定済みの Systems Manager リソースを手動で更新することも回避できます。

高速セットアップ機能では CloudFormation 、選択に基づいて Systems Manager リソースをデプロイおよび設定するための CloudFormation スタックセットも使用および作成します。CloudFormation スタックセットを使用する前に、複数のアカウントまたはリージョンへのデプロイをサポートするために、 CloudFormation StackSets で使用される IAM ロールを作成する必要があります。高速セットアップは、 CloudFormation StackSets でマルチリージョンまたはマルチアカウントデプロイをサポートするために必要なロールを作成します。Systems Manager リソースを複数のリージョンまたは単一のアカウントとリージョンから複数のアカウントで構成してデプロイしたい場合、 CloudFormation スタックセットの前提条件を満たしている必要があります。詳細については、 CloudFormation ドキュメントの「スタックセットオペレーションの前提条件」を参照してください。

カスタマイズされたクイックセットアップについては、AWS-QuickSetup-SSMHostMgmt.yaml CloudFormation テンプレートを確認してください。

CloudFormation テンプレートのリソースと機能を確認し、要件に応じて調整する必要があります。使用する CloudFormation テンプレートをバージョン管理し、変更を段階的にテストして、必要な結果を確認する必要があります。さらに、クラウドセキュリティレビューを実行して、組織の要件に基づいて必要なポリシー調整があるかどうかを判断する必要があります。

CloudFormation スタックを単一のテストアカウントとリージョンにデプロイし、必要なテストケースを実行して、目的の結果をカスタマイズして確認する必要があります。その後、1 つのアカウントの複数のリージョンに、複数のアカウントと複数のリージョンにデプロイを段階させることができます。

CloudFormation スタックを使用して 1 つのアカウントとリージョンでカスタマイズされた高速セットアップ

単一のアカウントとリージョンのみを使用している場合は、完全な例を CloudFormation スタックセットではなく CloudFormation スタックとしてデプロイできます。ただし、可能な場合は、単一のアカウントとリージョンのみを使用する場合でも、マルチアカウント、マルチリージョンスタックセットのアプローチを使用することをお勧めします。 CloudFormation StackSets を使用すると、将来的に追加のアカウントやリージョンに拡張することが容易になります。

次の手順を使用して、AWS-QuickSetup-SSMHostMgmt.yaml CloudFormation テンプレートを 1 つのアカウントに CloudFormation スタックとしてデプロイします AWS リージョン。

1. テンプレートをダウンロードし、任意のバージョン管理システム (GitHub など) にチェックインします。
2. 組織の要件に基づいてデフォルトの CloudFormation パラメータ値をカスタマイズします。
3. ステートマネージャーの関連付けスケジュールをカスタマイズします。
4. Systems Manager のドキュメントを `InstallAndManageCloudWatchDocument` 論理 ID でカスタマイズします。S3 バケットプレフィックスが、CloudWatch 設定を含む S3 バケットのプレフィックスと一致していることを確認します。
5. CloudWatch 設定を含む S3 バケットの Amazon リソースネーム (ARN) を取得して記録します。詳細については、このガイドの CloudWatch 設定の管理 セクションを参照してください。 AWS Organizations アカウントへの読み取りアクセスを提供するバケットポリシーを含むサンプル cloudwatch-config-s3-bucket.yaml CloudFormation テンプレートを利用できます。
6. カスタマイズされたクイックセットアップ CloudFormation テンプレートを S3 バケットと同じアカウントにデプロイします。
   - `CloudWatchConfigBucketARN` パラメータ向けのの場合は、S3 バケットの ARN を入力します。
   - Systems Manager で有効にする機能に応じて、パラメーターオプションを調整します。

7. IAM ロールの有無にかかわらずテスト EC2 インスタンスをデプロイして、EC2 インスタンスが CloudWatch で動作することを確認します。

- `AttachIAMToInstance` ステートマネージャーの関連付けを適用します。これは、スケジュールに従って実行するように構成された Systems Manager Runbook です。Runbook を使用する State Manager の関連付けは、新しい EC2 インスタンスに自動的に適用されず、スケジュールに基づいて実行するように設定できます。詳細については、Systems Manager のドキュメントでステートマネージャーを使用したトリガーによるオートメーションの実行を参照してください。
- EC2 インスタンスに必要な IAM ロールがアタッチされていることを確認します。
- EC2 インスタンスが Systems Manager に表示されていることを確認して、Systems Manager エージェントが正しく動作していることを確認します。
- S3 バケットの CloudWatch 設定に基づいて CloudWatch Logs とメトリクスを表示して、CloudWatch エージェントが正しく動作していることを確認します。

Stack CloudFormation StackSets を使用して、複数のリージョンと複数のアカウントでカスタマイズされた高速セットアップ

複数のアカウントとリージョンを使用している場合は、AWS-QuickSetup-SSMHostMgmt.yaml CloudFormation テンプレートをスタックセットとしてデプロイできます。スタックセットを使用する前に CloudFormation スタックセットの前提条件 を完了する必要があります。要件は、自己管理型または サービスマネージド型 アクセス許可 でスタックセットをデプロイするかどうかによって異なります。

新しいアカウントが自動的にカスタマイズされた Quick Setup を受け取るように、サービス管理アクセス許可でスタックセットをデプロイすることをお勧めします。サービスマネージドスタックセットは、 AWS Organizations 管理アカウントまたは委任管理者アカウントからデプロイする必要があります。スタックセットは、 AWS Organizations 管理アカウントではなく、委任された管理者権限を持つ自動化に使用される一元化されたアカウントからデプロイする必要があります。また、1 つのリージョンに 1 つまたは少数のアカウントを持つテスト組織単位 (OU) をターゲットにして、スタックセットのデプロイをテストすることをお勧めします。

1. このガイドの CloudFormation スタックを使用して 1 つのアカウントとリージョンでカスタマイズされた高速セットアップ セクションを参照してステップ 1 ~ 5 を完了します。
2. にサインインし AWS マネジメントコンソール、 CloudFormation コンソールを開き、StackSet の作成を選択します。
   - Template is ready (テンプレートの準備ができています) と Upload a template file (テンプレートファイルのアップロード) を選択します。要件に合わせてカスタマイズした CloudFormation テンプレートをアップロードします。
   - スタックセットの詳細を指定します:

- 例えば、StackSet-SSM-QuickSetup のようにスタックセット名を入力します。
- Systems Manager で有効にする機能に応じて、パラメーターオプションを調整します。
- CloudWatchConfigBucketARN パラメータ向けに、CloudWatch 設定の S3 バケットの ARN を入力します。
- スタックセットオプションを指定し、 でサービスマネージドアクセス許可を使用するか、セルフマネージドアクセス許可を使用する AWS Organizations かを選択します。
  - 自己管理型のアクセス許可を選択する場合は、AWSCloudFormationStackSetAdministrationRole および AWSCloudFormationStackSetExecutionRole IAM ロールの詳細を入力します。管理者ロールはアカウントに存在し、実行ロールは各ターゲットアカウントに存在する必要があります。
- AWS Organizationsを使用して サービスマネージド型 アクセス許可の場合は、組織全体ではなくテスト OU に最初にデプロイすることをお勧めします。
  - 自動デプロイメントを有効にするかどうかを選択します。有効 を選択することをお勧めします。アカウントの削除動作では、推奨される設定は スタックの削除 です。
- 自己管理型 のアクセス許可を使用する場合、 AWS 設定するアカウントのアカウント ID を入力します。自己管理型のアクセス許可を使用する場合は、新しいアカウントごとにこのプロセスを繰り返す必要があります。
- CloudWatch および Systems Manager を使用するリージョンを入力します。
- スタックセット向けの オペレーション および スタックインスタンス タブでステータスを表示して、デプロイが成功したことを確認します。
- デプロイされたアカウントで Systems Manager と CloudWatch が正しく動作していることを確認するには、このガイドの CloudFormation スタックを使用して 1 つのアカウントとリージョンでカスタマイズされた高速セットアップ セクションを参照してください。

オンプレミスサーバーを構成する際の考慮事項

オンプレミスサーバーおよび VM 用の CloudWatch エージェントは、EC2 インスタンスの場合と同様のアプローチを使用してインストールおよび構成されます。ただし、次の表では、CloudWatch エージェントをオンプレミスサーバーおよび VM にインストールして構成するときに評価する必要がある考慮事項を示します。

CloudWatch エージェントに、Systems Manager で使用されるのと同じ一時的な認証情報を指定します。	オンプレミスサーバーを含むハイブリッド環境で Systems Manager をセットアップすると、IAM ロールを使用して Systems Manager をアクティブ化できます。`CloudWatchAgentServerPolicy` および `AmazonSSMManagedInstanceCore` ポリシー を含む EC2 インスタンス用に作成されたロールを使用する必要があります。 これにより、Systems Manager エージェントは一時的な資格情報をローカル認証情報ファイルに取得して書き込みます。CloudWatch エージェントの設定を同じファイルに指定することができます。このプロセスは、Systems Manager エージェントと統合 CloudWatch エージェントを使用するオンプレミスサーバーを設定して、ナレッジセンターで一時的な認証情報のみを使用できます。 AWS このプロセスは、別の Systems Manager Automation Runbook と State Manager の関連付けを定義し、タグを使用してオンプレミスインスタンスをターゲットにすることで、このプロセスを自動化することもできます。オンプレミスインスタンス向けに Systems Manager のアクティブ化 を作成するとき、インスタンスをオンプレミスインスタンスとして識別するタグを含める必要があります。
VPN または Direct Connect アクセスと AWS PrivateLink を持つアカウントとリージョンの使用を検討してください。	AWS Direct Connect または AWS Virtual Private Network (Site-to-Site VPN) を使用して、オンプレミスネットワークと仮想プライベートクラウド (VPC) 間のプライベート接続を確立できます。 AWS PrivateLink は、インターフェイス VPC エンドポイントを使用して CloudWatch Logs へのプライベート接続を確立します。この方法は、パブリックインターネッ

	ト経由でパブリックサービスエンドポイントへのデータの送信を妨げる制限がある場合に便利です。
CloudWatch 設定ファイルにすべてのメトリクスを含める必要があります。	Amazon EC2 には標準のメトリクス (CPU 使用率など) が含まれていますが、これらのメトリクスはオンプレミスインスタンスに対して定義される必要があります。個別のプラットフォーム設定ファイルを使用して、オンプレミスサーバーに対してこれらのメトリクスを定義し、プラットフォームの標準の CloudWatch メトリクス設定にその設定を追加できます。

エフェメラル EC2 インスタンスに関する考慮事項

Amazon EC2 Auto Scaling、Amazon EMR、Amazon EC2 スポットインスタンス、または AWS Batchによってプロビジョニングされている場合、EC2 インスタンスは、一時的または 一過性 です。エフェメラル EC2 インスタンスは、ランタイムオリジンに関する追加情報なしで、共通のロググループの下に非常に多くの CloudWatch ストリームを引き起こす可能性があります。

エフェメラル EC2 インスタンスを使用する場合は、ロググループとログストリーム名に動的なコンテキスト情報を追加することを検討してください。たとえば、スポットインスタンスリクエスト ID、Amazon EMR クラスター名、または Amazon EC2 Auto Scaling グループ名を含めることができます。この情報は、新しく起動した EC2 インスタンスでは異なる場合があり、ランタイムに取得して設定する必要がある場合があります。これを行うには、ブート時に CloudWatch エージェント設定ファイルを記述し、エージェントを再起動して更新された設定ファイルを含めます。これにより、動的なランタイム情報を使用して CloudWatch にログとメトリクスを配信できるようになります。

また、エフェメラル EC2 インスタンスが終了する前に、メトリクスとログが CloudWatch エージェントによって送信されていることを確認する必要があります。CloudWatch エージェントは、`flush_interval` ログバッファとメトリクスバッファのフラッシュイングの時間間隔を定義するように構成できるパラメータを含みます。ワークロードに基づいてこの値を下げて CloudWatch エージェントを停止し、EC2 インスタンスが終了する前にバッファを強制的にフラッシュすることができます。

CloudWatch エージェントをデプロイするための自動化ソリューションの使用

自動化ソリューション (Ansible や Chef など) を使用する場合は、それを活用して CloudWatch エージェントを自動的にインストールおよび更新できます。この方法を使用する場合は、次の考慮事項を評価する必要があります。

- 自動化が OS とサポートしている OS のバージョンをカバーしていることを確認します。自動化スクリプトが組織の OS をすべてサポートしていない場合は、サポートされていない OS の代替ソリューションを定義する必要があります。
- 自動化ソリューションが CloudWatch エージェントのアップデートとアップグレードを定期的にチェックすることを検証する。自動化ソリューションでは、CloudWatch エージェントの更新を定期的にチェックするか、定期的にエージェントをアンインストールして再インストールする必要があります。スケジューラまたはオートメーションソリューション機能を使用して、エージェントを定期的にチェックおよび更新できます。
- エージェントのインストールと構成のコンプライアンスを確認できることを確認します。自動化ソリューションでは、システムにエージェントがインストールされていない場合やエージェントが動作していない時期を判断できるはずです。オートメーションソリューションに通知またはアラームを実装して、失敗したインストールと構成を追跡できます。

ユーザーデータスクリプトを使用したインスタンスのプロビジョニング中に CloudWatch エージェントをデプロイする

Systems Manager を使用する予定がなく、EC2 インスタンスに CloudWatch を選択的に使用する場合は、この方法を使用できます。通常、このアプローチは 1 回限り、または特殊な設定が必要な場合に使用されます。 は、起動スクリプトまたはユーザーデータスクリプトでダウンロードできる CloudWatch エージェント用の直接リンク AWS を提供します。エージェントインストールパッケージは、ユーザー操作なしでサイレントで実行できます。つまり、自動デプロイで使用できます。この方法を使用する場合は、次の考慮事項を評価する必要があります。

- ユーザーがエージェントをインストールしたり、標準メトリクスを構成したりしないリスクが増大する。ユーザーは、CloudWatch エージェントをインストールするために必要な手順を含めずに、インスタンスをプロビジョニングできます。また、エージェントの設定を誤って、ロギングとモニタリングの不整合が発生する可能性があります。

- インストールスクリプトは OS 固有で、異なる OS バージョンに適したものである必要があります。Windows と Linux の両方を使用する場合は、別個のスクリプトが必要です。Linux スクリプトは、ディストリビューションに基づいて異なるインストール手順を設定する必要があります。
- 利用可能な場合は、CloudWatch エージェントを新しいバージョンで定期的に更新する必要があります。State Manager で Systems Manager を使用する場合、これは自動化できますが、インスタンスの起動時に再実行するようにユーザーデータスクリプトを構成することもできます。CloudWatch エージェントは、再起動のたびに更新され、再インストールされます。
- 標準の CloudWatch 設定の取得と適用を自動化する必要があります。State Manager で Systems Manager を使用する場合、これは自動化できますが、ブート時に設定ファイルを取得し、CloudWatch エージェントを再起動するようにユーザーデータスクリプトを設定することもできます。

AMI に CloudWatch エージェントを含める

この方法を使用する利点は、CloudWatch エージェントがインストールおよび設定されるのを待つ必要がなく、すぐにロギングとモニタリングを開始できることです。これにより、インスタンスの起動に失敗した場合に備えて、インスタンスのプロビジョニングと起動ステップをより適切に監視できます。このアプローチは、Systems Manager エージェントを使用する予定がない場合にも適しています。この方法を使用する場合は、次の考慮事項を評価する必要があります。

- AMI に最新の CloudWatch エージェントバージョンが含まれていない可能性があるため、更新プロセスが存在する必要があります。AMI にインストールされた CloudWatch エージェントは、AMI が最後に作成された時点でのみ最新です。EC2 インスタンスがプロビジョニングされるときに、エージェントを定期的に更新するための追加の方法を含める必要があります。Systems Manager を使用する場合は、Systems Manager ディストリビューターとステートマネージャーを使用して CloudWatch エージェントをインストールする このガイドで提供されているソリューションを使用できます。Systems Manager を使用しない場合は、ユーザーデータスクリプトを使用して、インスタンスの起動時および再起動時にエージェントを更新できます。
- CloudWatch エージェントの設定ファイルは、インスタンスの起動時に取得する必要があります。Systems Manager を使用しない場合は、起動時に設定ファイルを取得し、CloudWatch エージェントを再起動するようにユーザーデータスクリプトを設定できます。
- CloudWatch 設定が更新された後、CloudWatch エージェントを再起動する必要があります。
- AWS 認証情報を AMI に保存しないでください。ローカル AWS 認証情報が AMI に保存されていないことを確認します。Amazon EC2 を使用する場合は、必要な IAM ロールをインスタンスに適用し、ローカル認証情報を避けることができます。オンプレミスインスタンスを使用する場合

は、CloudWatch エージェントを起動する前に、インスタンスの認証情報を自動化または手動で更新する必要があります。

Amazon ECS でのログ記録とモニタリング

Amazon Elastic Container Service (Amazon ECS) には、コンテナを実行するための 2 つの起動タイプがあり、タスクとサービスをホストするインフラストラクチャのタイプを決定します。これらの起動タイプは AWS Fargate と Amazon EC2 です。どちらの起動タイプも CloudWatch と統合されますが、設定とサポートは異なります。

以下のセクションでは、CloudWatch を使用して Amazon ECS でのログ記録とモニタリングを行う方法を理解するのに役立ちます。

トピック

- EC2 起動タイプの CloudWatch の設定
- EC2 および Fargate 起動タイプの Amazon ECS コンテナログ
- Amazon ECS 用 FireLens でカスタムログルーティングを使用する
- Amazon ECS のメトリクス

EC2 起動タイプの CloudWatch の設定

EC2 起動タイプでは、ログとモニタリングに CloudWatch エージェントを使用する EC2 インスタンスの Amazon ECS クラスターをプロビジョニングします。Amazon ECS 最適化 AMI には、Amazon ECS コンテナエージェント およびは、Amazon ECS クラスターの CloudWatch メトリクスを提供します。

これらのデフォルトのメトリクスは Amazon ECS のコストに含まれますが、Amazon ECS のデフォルト設定では、ログファイルや追加のメトリック (空きディスク容量など) は監視されません。を使用して AWS マネジメントコンソール 、EC2 起動タイプで Amazon ECS クラスターをプロビジョニングできます。これにより、起動設定で Amazon EC2 Auto Scaling グループをデプロイする CloudFormation スタックが作成されます。ただし、この方法では、カスタム AMI を選択したり、異なる設定や追加の起動スクリプトを使用して起動設定をカスタマイズしたりすることはできません。

追加のログとメトリクスを監視するには、Amazon ECS コンテナインスタンスに CloudWatch エージェントをインストールする必要があります。EC2 インスタンスのインストールアプローチは、Systems Manager ディストリビューターとステートマネージャーを使用して CloudWatch エージェントをインストールする ガイドのセクションを参照してください。ただし、Amazon ECS AMI には、必要な Systems Manager Agent は含まれていません。Amazon ECS クラスターの作成時に

Systems Manager Agent をインストールするユーザーデータスクリプトを使用して、カスタム起動設定を使用する必要があります。これにより、コンテナインスタンスを Systems Manager に登録し、ステートマネージャの関連付けを適用して CloudWatch エージェントをインストール、設定、および更新できます。ステートマネージャーは CloudWatch エージェントの設定を実行し、更新するときに、Amazon EC2 用の標準化されたシステムレベルの CloudWatch 設定も適用されます。また、Amazon ECS の標準化された CloudWatch 設定を CloudWatch 設定の S3 バケットに保存し、ステートマネージャーで自動的に適用することもできます。

Amazon ECS コンテナインスタンスに適用された IAM ロールまたはインスタンスプロファイルに `CloudWatchAgentServerPolicy` そして `AmazonSSMManagedInstanceCore` ポリシーに必要なものが含まれているか確認する必要があります。ecs_cluster_with_cloudwatch_linux.yaml CloudFormation テンプレートを使用して、Linux ベースの Amazon ECS クラスターをプロビジョニングできます。このテンプレートは、Systems Manager をインストールし、カスタムの CloudWatch 設定をデプロイして Amazon ECS 固有のログファイルを監視するカスタム起動設定を持つ Amazon ECS クラスターを作成します。

Amazon ECS コンテナインスタンスおよび標準 EC2 インスタンスログについて、次のログをキャプチャする必要があります。

- Amazon ECS エージェントの起動出力–`/var/log/ecs/ecs-init.log`
- Amazon ECS エージェントの出力–`/var/log/ecs/ecs-agent.log`
- IAM 認証情報プロバイダーのリクエストログ–`/var/log/ecs/audit.log`

出力レベル、フォーマット、および追加設定オプションの詳細については、Amazon ECS のドキュメント内の「Amazon ECS ログファイルの場所」を参照してください。

> **Important**
>
> EC2 コンテナインスタンスを実行したり管理したりしないため、Fargate 起動タイプにはエージェントのインストールや設定は必要ありません。

Amazon ECS コンテナインスタンスでは、最新の Amazon ECS で最適化された AMI とコンテナエージェントを使用する必要があります。 AWS は、AMI ID を含む Amazon ECS で最適化された AMI 情報とともに、パブリックSystems Manager パラメータストアパラメータを格納します。パラメータストアから、Amazon ECS で最適化された AMI の パラメータストアパラメータ形式 から最適化された最新の AMI を取得することができます。お客様の CloudFormation テンプレートの最新

の AMI または特定の AMI リリースを参照するパブリックパラメータストアパラメータを参照できます。

AWS は、サポートされている各リージョンで同じ Parameter Store パラメータを提供します。つまり、これらのパラメータを参照する CloudFormation テンプレートは、AMI を更新せずにリージョンとアカウント間で再利用できます。特定のリリースを参照して、新しい Amazon ECS AMI の組織へのデプロイを制御できます。これにより、新しい Amazon ECS で最適化された AMI がテストされるまで使用されなくなります。

EC2 および Fargate 起動タイプの Amazon ECS コンテナログ

Amazon ECS はタスク定義を使用して、コンテナをタスクとサービスとしてデプロイおよび管理します。タスク定義内で Amazon ECS クラスターに起動するコンテナを設定します。ログは、コンテナレベルでログドライバーを使用して構成されます。複数のログドライバオプションは、コンテナに起動タイプが EC2 か Fargate のどちらを使用するかによって異なるロギングシステム（ 例えば、awslogs,fluentd,gelf,json-file,journald,logentries,splunk,syslog または awsfirelens)を提供します Fargate 起動タイプは、、awslogs、splunkおよび のログドライバーオプションのサブセットを提供しますawsfirelens。 は、コンテナ出力をキャプチャして CloudWatch Logs に送信するawslogsログドライバー AWS を提供します。ログドライバの設定では、ロググループ、リージョン、ログストリームのプレフィックスを他の多くのオプションとともにカスタマイズできます。

ロググループのデフォルトの名前付けの CloudWatch Logs の自動設定 オプションは、 AWS マネジメントコンソール では /ecs/<task_name> です 。Amazon ECS で使用されるログストリーム名には、<awslogs-stream-prefix>/<container_name>/<task_id> の形式で設定。組織の要件に基づいてログをグループ化するグループ名を使用することをお勧めします。以下の表では、image_name そして image_tag がログストリームの名前に含まれます。

ロググループ名	/<Business unit>/<Project or application name>/<Environment>/<Cluster name>/<Task name>
ログストリーム名のプレフィクス	/<image_name>/<image_tag>

この情報は、タスク定義でも使用できます。ただし、タスクは新しいリビジョンで定期的に更新されます。つまり、タスク定義で現在使用しているものよりも、image_name そして image_tag タス

ク定義が別のリビジョンが使用されている可能性があります。詳細や名前付けの例については、このガイドの「CloudWatch デプロイを計画する」を参照してください。

継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインまたは自動プロセスを使用する場合は、新しい Docker イメージビルドごとにアプリケーションの新しいタスク定義リビジョンを作成できます。例えば、CI/CD プロセスの一部として、Docker イメージ名、イメージタグ、GitHub リビジョン、またはその他の重要な情報をタスク定義のリビジョンとロギング設定に含めることができます。

Amazon ECS 用 FireLens でカスタムログルーティングを使用する

FireLens for Amazon ECS は、Fluentd または Fluent Bit にログをルーティングするのに役立ちます。これにより、コンテナログを AWS サービスや AWS パートナーネットワーク (APN) の送信先に直接送信したり、CloudWatch Logs へのログ配信をサポートしたりできます。

AWS は、Amazon Kinesis Data Streams、Amazon Data Firehose、CloudWatch Logs 用のプラグインがプリインストールされた Fluent Bit 用の Docker イメージを提供します。FireLens ログドライバーは、CloudWatch Logs に送信されるログをさらにカスタマイズして制御するための `awslogs` ログドライバーを利用できます。

例えば、FireLens ログドライバを使用して、ログ形式の出力を制御できます。つまり、Amazon ECS コンテナの CloudWatch ログは JSON オブジェクトとして自動的にフォーマットされ、次の `ecs_cluster`,`ecs_task_arn`,`ecs_task_definition`,`container_id`,`container_name`, および `ec2_instance_id` のJSON 形式のプロパティが含まれます。Fluent ホストは、`FLUENT_HOST` そして `FLUENT_PORT` 環境変数は、`awsfirelens` ドライバーを手特定するときに露出します。つまり、流暢なロガーライブラリを使用して、コードからログルーターに直接ログを記録できます。例えば、アプリケーションに `fluent-logger-python` 環境変数の値を使用して Fluent Bit に記録するライブラリが含まれます。

Amazon ECS に FireLens を使用することを選択した場合、`awslogs` ログドライバー 他の設定も使って と同じ設定を構成できます。例えば、CloudWatch へのロギングに FireLens を使用するように設定された NGINX サーバーを起動する Amazon ECS タスク定義 ecs-Task-nginx-firelense.json を利用できます。また、ロギング用のサイドカーとして FireLens Fluent Bit コンテナを起動します。

Amazon ECS のメトリクス

Amazon ECS コンテナエージェントを使用したクラスターおよびサービスレベルでの EC2 および Fargate 起動タイプ (CPU およびメモリ使用率など)で、Amazon ECS は標準の CloudWatch メトリ

クスを提供します。CloudWatch Container Insights を使用してサービス、タスク、およびコンテナのメトリクスをキャプチャしたり、組み込みメトリクス形式を使用して独自のカスタムコンテナメトリクスをキャプチャすることもできます。

Container Insights はクラスターレベル、コンテナインスタンス、サービス、タスクレベルで CPU 使用率、メモリ使用率、ネットワークトラフィック、ストレージなどのメトリックスを提供する CloudWatch 機能です。Container Insights では、サービスとタスクを分析し、コンテナレベルで平均メモリまたは CPU 使用率を確認するのに役立つ自動ダッシュボードも作成します。コンテナインサイトは、カスタム指標を `ECS/ContainerInsights` カスタム名前空間 グラフ、アラーム、およびダッシュボードに使用できます。

個々の Amazon ECS クラスターでコンテナインサイトを有効にすることで、コンテナインサイトメトリクスを有効にすることができます。コンテナインスタンスレベルでメトリクスを表示する場合は、Amazon ECS クラスターで CloudWatch エージェントをデーモンコンテナとして起動します。cwagent-ecs-instance-metric-cfn.yaml CloudFormation テンプレートを使用して、CloudWatch エージェントを Amazon ECS サービスとしてデプロイできます。重要なのは、この例では、適切なカスタム CloudWatch エージェント設定を作成し、`ecs-cwagent-daemon-service` キーを使用してパラメータストアに保存したことを前提としています。

CloudWatch エージェント は、CloudWatch Container Insights のデーモンコンテナとしてデプロイされるには、次のような追加のディスク、メモリ、`instance_cpu_reserved_capacity` そして `instance_memory_reserved_capacity`と `ClusterName,ContainerInstanceId,InstanceId` ディメンションなどのCPU メトリクスが含まれます。コンテナインスタンスレベルのメトリクスは、CloudWatch 組み込みメトリックス形式を使用して Container Insights によって実装されます。Amazon ECS コンテナインスタンスにからのガイドの CloudWatch エージェントのデプロイと設定のステートマネージャーとディストリビューターをセットアップする セクションを参照したアプローチを使用して、追加のシステムレベルのメトリクスを設定できます。

Amazon ECS でカスタムアプリケーションメトリクスを作成する

CloudWatch の埋め込みメトリクス形式 を使用して、独自のアプリケーションメトリクスを作成、取得することも可能です。`awslogs` ログドライバーは、CloudWatch 埋め込みメトリクス形式のステートメントを解釈できます。

`CW_CONFIG_CONTENT` 次の例の環境変数は、`cwagentconfig` Systems Manager パラメータストアパラメータに設定されています。この基本構成でエージェントを実行して、組み込みメトリック形式のエンドポイントとして構成できます。ただし、不要になりました。

```
  {
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

複数のアカウントとリージョンにまたがって Amazon ECS デプロイメントがある場合は、 AWS Secrets Manager Secret は CloudWatch 設定を保存し、組織と共有するようにシークレットポリシーを設定します。タスク定義で secrets オプションを使用して、CW_CONFIG_CONTENT 変数を設定します。

アプリケーションで AWS 提供されているオープンソースの埋め込みメトリクス形式ライブラリを使用し、 AWS_EMF_AGENT_ENDPOINT環境変数を指定して、埋め込みメトリクス形式のエンドポイントとして機能する CloudWatch エージェントサイドカーコンテナに接続できます。例えば、ecs_cw_emf_example 埋め込みメトリックス形式のエンドポイントとして設定された CloudWatch エージェントサイドカーコンテナに埋め込みメトリックス形式のメトリクスを送信する Python アプリケーションのサンプルを使用します。

CloudWatch 用 Fluent Bit プラグインを使用して、埋め込みメトリクス形式のメッセージを送信することもできます。また、 ecs_firelense_emf_example Amazon ECS サイドカーコンテナに FireLens 組み込みメトリクス形式のメトリクスを送信する Python アプリケーションのサンプルを使用することもできます。

埋め込みメトリクス形式を使用しない場合は、 AWS API または AWS SDK を使用して CloudWatch メトリクスを作成および更新できます。特定のユースケースがない限り、このアプローチは推奨されません。これは、コードにメンテナンスと管理オーバーヘッドを追加するためです。

Amazon EKS でのログ記録とモニタリング

Amazon Elastic Kubernetes Service (Amazon EKS) は、Kubernetes コントロールプレーン用の CloudWatch Logs と統合されます。コントロールプレーンは Amazon EKS によってマネージドサービスとして提供され、CloudWatch エージェントをインストールせずにロギングを有効にする。CloudWatch エージェントをデプロイして Amazon EKS ノードとコンテナログをキャプチャすることもできます。Fluent Bit と Fluentd は、CloudWatch Logs へのコンテナログの送信にもサポートされています。

CloudWatch Container Insights は、クラスター、ノード、ポッド、タスク、サービスのレベルで Amazon EKS の包括的なメトリクスモニタリングソリューションを提供します。Amazon EKS では、Prometheus とメトリックスキャプチャの複数のオプションもサポートしています。Amazon EKS コントロールプレーン は、Prometheus 形式で公開されたメトリクスで メトリックエンドポイントを提供します。Prometheus を Amazon EKS クラスターにデプロイして、これらのメトリクスを消費することができます。

また、Prometheus メトリクスをスクレイプするように CloudWatch エージェントをセットアップ することで、他の Prometheus エンドポイントを使用するだけでなく、CloudWatch メトリクスを作成します。コンテナインサイトで Prometheus をモニタリングすると、コンテナ化されたシステムとワークロードからの Prometheus メトリクスの検出が自動化されます。

Amazon EKS ノードに CloudWatch エージェントをインストールして設定し、ディストリビューターおよびステートマネージャーを使用した Amazon EC2 で使用されるアプローチと同様に、Amazon EKS ノードを標準のシステムロギングおよびモニタリング設定に合わせることができます。

Amazon EKS のログ記録

Kubernetes ロギングは、コントロールプレーンのロギング、ノードロギング、およびアプリケーションロギングに分けることができます。Kubernetes コントロールプレーン は、Kubernetes クラスタを管理し、監査および診断に使用されるログを生成するコンポーネントのセットです。Amazon EKSで、さまざまなコントロールプレーンコンポーネントのログをオンにし、CloudWatch に送信します。

Kubernetes は、`kubelet` そして `kube-proxy` ポッドを実行する各 Kubernetes ノードで次のようなシステムコンポーネントも実行します。これらのコンポーネントは各ノード内にログを書き込み、各 Amazon EKS ノードでこれらのログをキャプチャするように CloudWatch とコンテナインサイトを設定できます。

コンテナは、Kubernetes クラスタ内でポッドとしてグループ化され、また Kubernetes ノードで実行するようにスケジュールされています。ほとんどのコンテナ化されたアプリケーションは、標準出力と標準エラーに書き込み、コンテナエンジンはその出力をロギングドライバにリダイレクトします。Kubernetes では、コンテナログは /var/log/pods ノード上のディレクトリで見つかります。CloudWatch とコンテナインサイトを設定して、各 Amazon EKS ポッドのこれらのログをキャプチャできます。

Amazon EKS コントロールプレーンのログ記録

Amazon EKS クラスターは、Kubernetes クラスターの高可用性のシングルテナントコントロールプレーンと、コンテナを実行する Amazon EKS ノードで構成されます。コントロールプレーンノードは、 が管理するアカウントで実行されます AWS。Amazon EKS クラスターコントロールプレーンノードは CloudWatch に統合され、特定のコントロールプレーンコンポーネントのロギングを有効にできます。

ログは、Kubernetes コントロールプレーンコンポーネントインスタンスごとに提供されます。 は、コントロールプレーンノードの状態 AWS を管理し、Kubernetes エンドポイントのサービスレベルアグリーメント (SLA) を提供します。

Amazon EKS ノードとアプリケーションのログ記録

CloudWatch コンテナインサイト Amazon EKS のログとメトリクスをキャプチャをを使用することをお勧めします。コンテナインサイトは CloudWatch エージェントでクラスター、ノード、ポッドレベルのメトリクスを実装し、CloudWatch へのログキャプチャ用のフルエントビットまたは Fluentd を実装します。Container Insights は、キャプチャされた CloudWatch メトリクスをレイヤー化したビューを含む自動ダッシュボードも提供します。コンテナインサイトは、すべての Amazon EKS ノードで実行される CloudWatch DaemonSet および Fluent Bit DaemonSet としてデプロイされます。Fargate ノードは、DaemonSets によって管理 AWS され、サポートされていないため、Container Insights ではサポートされません。Amazon EKS の Fargate ロギングについては、このガイドで個別に説明しています。

次の表に、デフォルトの Fluentd または Fluent Bit ログキャプチャ設定 し、Amazon EKS の場合でキャプチャされた CloudWatch ロググループとログを示します。

/aws/containerinsights/Cluster_Name/application	/var/log/containers のすべてのログファイル このディレクトリは、/var/log/pods ディレクトリ構造内のすべての Kubernetes コンテナログへ

	のシンボリックリンクを提供します。これにより、アプリケーションコンテナのログへの書き込みが stdout または stderr にキャプチャされます。また、次のようなKubernetes システムコンテナの aws-vpc-cni-init ,kube-proxy , および coreDNS のログも含まれます。
/aws/containerinsights/Cluster_Name/host	/var/log/dmesg 、/var/log/secure 、および /var/log/messages からのログ
/aws/containerinsights/Cluster_Name/dataplane	/var/log/journal 、kubelet.service 、およびkubeproxy .service に対する docker.service のログ。

ログに Fluent Bit または Fluentd でコンテナインサイトを使用したくない場合は、Amazon EKS ノードにインストールされた CloudWatch エージェントを使用してノードとコンテナログをキャプチャできます。Amazon EKS ノードは EC2 インスタンスです。つまり、Amazon EC2 の標準システムレベルのロギングアプローチにそれらを含める必要があります。ディストリビューターとステートマネージャーを使用して CloudWatch エージェントをインストールすると、Amazon EKS ノードも CloudWatch エージェントのインストール、設定、および更新に含まれます。

次の表に、Kubernetes に固有のログを示し、ログに Fluent Bit または Fluentd でコンテナインサイトを使用していない場合にキャプチャする必要があるログを示します。

/var/log/containers	このディレクトリは、/var/log/pods ディレクトリ構造内のすべての Kubernetes コンテナログへのシンボリックリンクを提供します。これにより、アプリケーションコンテナのログへの書き込みが stdout または stderr にキャプチャされます。また、次のような Kubernetes システムコンテナの aws-vpc-cni-init ,kube-proxy , および coreDNS のログも含まれます。重要: コンテナインサイト

	を使用している場合、これは必須ではありません。

`var/log/aws-routed-eni/ipamd.log`
 	
`/var/log/aws-routed-eni/plugin.log`
	L-IPAM デーモンの口グはここにあります

Amazon EKS ノードが適切なシステムレベルのログとメトリクスを送信するように CloudWatch エージェントをインストールして設定する必要があります。ただし、AMI 解析された Amazon EKS は、必要な Systems Manager Agent は含まれていません。起動テンプレート を使用して、Systems Manager エージェントのインストールと、ユーザーデータセクションを通じて実装された起動スクリプトを使用して、重要な Amazon EKS 固有の口グをキャプチャするデフォルトの CloudWatch 設定を自動化できます。Amazon EKS ノードは、Auto Scaling グループを マネージド型ノードグループ または セルフマネージド型ノード としてデプロイされています。

管理対象ノードグループでは、起動テンプレート には、Systems Manager エージェントのインストールと CloudWatch 設定を自動化するためのユーザーデータセクションが含まれています。amazon_eks_managed_node_group_launch_config.yaml CloudFormation テンプレートをカスタマイズして使用すると、Systems Manager エージェント、CloudWatch エージェントをインストールし、Amazon EKS 固有の口グ記録設定を CloudWatch 設定ディレクトリに追加する起動テンプレートを作成できます。このテンプレートを使用して、Amazon EKS マネージドノードグループの起動テンプレートをコードとしてのインフラストラクチャ (IaC) アプローチで更新できます。 CloudFormation テンプレートの更新ごとに、起動テンプレートの新しいバージョンがプロビジョニングされます。次に、ノードグループを更新して新しいテンプレートバージョンを使用し、マネージド型ライフサイクルプロセス をダウンタイムなしでノードを更新できます。マネージドノードグループに適用される IAM ロールとインスタンスプロファイルに、`CloudWatchAgentServerPolicy`および `AmazonSSMManagedInstanceCore` AWS マネージドポリシーが含まれていることを確認します。

セルフマネージドノードを使用すると、Amazon EKS ノードのライフサイクルと更新戦略を直接プロビジョニングおよび管理できます。自己管理ノードを使用すると、お客様のAmazon EKS クラスターで Windows ノードを実行し、Bottlerocketと併せて 他のオプション を許可します。CloudFormation を使用して、Amazon EKS クラスターにセルフマネージド型ノードをデプロイできます。つまり、Amazon EKS クラスターに IaC および マネージド型変更アプローチを使用できます。 AWS には、そのまま使用したりカスタマイズしたりできる amazon-eks-nodegroup.yaml CloudFormation テンプレートが用意されています。テンプレートは、クラスター内の Amazon

EKS ノードに必要なすべてのリソースをプロビジョニングします (たとえば、個別の IAM ロール、セキュリティグループ、Amazon EC2 Auto Scaling グループ、起動テンプレート)。amazon-eks-nodegroup.yaml CloudFormation テンプレートは、必要な Systems Manager エージェント、CloudWatch エージェントをインストールし、Amazon EKS 固有のログ記録設定を CloudWatch 設定ディレクトリに追加する更新バージョンです。

Fargate での Amazon EKS のログ記録

Fargate で Amazon EKS を使用すると、Kubernetes ノードを割り当てたり管理したりすることなく、ポッドをデプロイできます。これにより、Kubernetes ノードのシステムレベルのログをキャプチャする必要がなくなります。Fargate ポッドからログをキャプチャするには、Fluent Bit を使用してログを CloudWatch に直接転送できます。これにより、Fargate 上の Amazon EKS ポッドの追加設定やサイドカーコンテナを使用せずに、ログを CloudWatch に自動的にルーティングできます。詳細については、Amazon EKS ドキュメントの「Fargate logging」および AWS ブログの「Fluent Bit for Amazon EKS」を参照してください。このソリューションでは、STDOUT そして STDERR の入出力 (I/O) は、Fargate 上の Amazon EKS クラスター用に確立されたフルエントビット設定に基づいて、コンテナからストリームし、Fluent Bit を介して CloudWatch に送信します。

Amazon EKS および Kubernetes のメトリクス

Kubernetes には、リソース使用量メトリクス (ノードおよびポッドの CPU およびメモリ使用量など) にアクセスできるメトリクス API が提供されますが、API はポイントインタイム情報のみを提供し、履歴メトリックは提供しません。Kubernetes メトリックスサーバー は、通常、Amazon EKS および Kubernetes のデプロイで、メトリクスの集約、メトリクスに関する短期的な履歴情報の提供、Horizontal Pod Autoscaler のような機能をサポートするために使用されます。

Amazon EKS は Prometheus 形式で Kubernetes API サーバーを介してコントロールプレーンのメトリクスを公開しており、CloudWatch は、これらのメトリクスをキャプチャして取り込むことができます。CloudWatch とコンテナインサイトは、Amazon EKS ノードとポッドの包括的なメトリクスのキャプチャ、分析、およびアラームを提供するように構成することもできます。

Kubernetes コントロールプレーンのメトリクス

/metrics HTTP API エンドポイントを使用し Kubernetes は、Prometheus 形式でコントロールプレーンのメトリックを公開します。Kubernetes クラスターに Prometheus インストールし、これらのメトリクスをウェブブラウザーでグラフ化して表示します。また、Kubernetes API サーバーによって CloudWatch により、公開されたメトリクスを取り込めます。

Kubernetes のノードとシステムメトリック

Kubernetes が Prometheus を提供する メトリクスサーバー できるポッド デプロイして実行する クラスター、ノード、ポッドレベルの CPU およびメモリの統計情報については、Kubernetes クラスタ上で実行します。これらのメトリクスは、Horizontal Pod Autoscaler、そして Vertical Pod Autoscaler に使用されます。CloudWatch はこれらのメトリクスを提供することもできます。

Kubernetes Kubernetes ダッシュボード または、水平および垂直ポッドオートスケーラーを使用する場合は、Kubernetes メトリクスサーバをインストールする必要があります。Kubernetes ダッシュボードは、Kubernetes クラスター、ノード、ポッド、および関連する構成を参照して構成し、Kubernetes メトリクスサーバーから CPU およびメモリのメトリックを表示するのに役立ちます。

Kubernetes メトリクスサーバーによって提供されるメトリックは、Auto Scaling 以外の目的 (モニタリングなど) には使用できません。メトリックは、過去の分析ではなく、ポイントインタイム分析を対象としています。Kubernetes ダッシュボードは、`dashboard-metrics-scraper` Kubernetes メトリクスサーバーからのメトリックを短時間ウィンドウに保存します。

Container Insights は、Kubernetes DaemCloudWatch set で実行中のすべてのコンテナを検出し、ノードレベルのメトリクスを提供します。次に、パフォーマンススタックの各レイヤーでパフォーマンスデータを収集します。クイックスタートから AWS クイックスタートを使用するか、Container Insights を個別に設定できます。クイックスタートでは、CloudWatch エージェントによるメトリクスのモニタリングと Fluent Bit による口ギングを設定するため、口ギングとモニタリングのために一度だけデプロイする必要があります。

Amazon EKS ノードは EC2 インスタンスであるため、Amazon EC2 に対して定義した標準を使用して、コンテナインサイトによってキャプチャされたメトリクスに加えて、システムレベルのメトリックスをキャプチャする必要があります。 Amazon EKS クラスターの CloudWatch エージェントをインストールして設定するには、CloudWatch エージェントのデプロイと設定のステートマネージャーとディストリビューターをセットアップする このガイドのセクションと同じ方法で使用できます。Amazon EKS 固有の CloudWatch 設定ファイルを更新して、メトリクスと Amazon EKS 固有の口グ設定を含めることができます。

Prometheus をサポートするエージェントは、Prometheus メトリクスを自動的に検出して サポートされているコンテナ化されたワークロードとシステム をスクレイプできます。CloudWatch Logs Insights を使用して分析するために、組み込みメトリックス形式の CloudWatch Logs としてそれらを取り込み、CloudWatch メトリクスを自動的に作成します。

> **Important**
>
> Prometheus メトリクスを収集するための CloudWatch エージェントの 特殊バージョンをデプロイ を行う必要があります。これは、コンテナインサイト用にデプロイされた CloudWatch エージェントとは別のエージェントです。Prometheus メトリクスの検出を実証するための CloudWatch エージェントと Amazon EKS ポッドデプロイのデプロイと設定ファイルが含まれている prometheus_jmx サンプル の Java アプリケーションを使用できます。詳細については、CloudWatch のドキュメント内の「Amazon EKS および Kubernetes で Java/JMX サンプルワークロードをセットアップ」を参照してください。また、Amazon EKS クラスターで実行中の他の Prometheus ターゲットからメトリクスをキャプチャするように CloudWatch エージェントを設定することもできます。

アプリケーションメトリクス

CloudWatch の埋め込みメトリクス形式 を使用して、独自のアプリケーションメトリクスを作成、取得することも可能です。埋め込みメトリック形式ステートメントを取り込むには、埋め込みメトリック形式のエントリを埋め込みメトリック形式のエンドポイントに送信する必要があります。CloudWatch エージェントは、Amazon EKS ポッドのサイドカーコンテナ を設定できます。CloudWatch エージェント設定は Kubernetes ConfigMap として保存され、CloudWatch エージェントサイドカーコンテナによって読み込まれ、埋め込みメトリックス形式のエンドポイントを開始します。

また、アプリケーションを Prometheus ターゲットとして設定し、Prometheus サポートを使用して CloudWatch エージェントを設定して、メトリクスを検出、スクレープ、CloudWatch に取り込むこともできます。たとえば、オープンソースの JMX エクスポーター と一緒に Java アプリケーションを使用して、CloudWatch エージェントによる Prometheus 消費用の JMX Bean を公開します。

埋め込みメトリクス形式を使用しない場合は、AWS API または AWS SDK を使用して CloudWatch メトリクスを作成および更新することもできます。ただし、モニタリングとアプリケーションロジックが混在するため、このアプローチはお勧めしません。

Fargate での Amazon EKS のメトリクス

Fargate は Kubernetes ポッドを実行するために Amazon EKS ノードを自動的にプロビジョニングするため、ノードレベルのメトリクスを監視および収集する必要はありません。ただし、Fargate の Amazon EKS ノードで実行されているポッドのメトリクスを監視する必要があります。コンテナイ

ンサイトは、現在サポートされていない次の機能が必要なため、Fargate 上の Amazon EKS では現在利用できません。

- DaemonSets は現在サポートされていません。コンテナインサイトは、CloudWatch エージェントを各クラスターノードで DaemonSet として実行することによってデプロイされます。
- HostPath パーシステントボリュームはサポートされていません。CloudWatch エージェントコンテナは、コンテナメトリックスデータを収集するための前提条件として HostPath パーシステントボリュームを使用します。
- Fargate は、特権コンテナーとホスト情報へのアクセスを防止します。

Fargate 用の内蔵ログルータ を使用して、埋め込みメトリクスフォーマットステートメントを CloudWatch に送信します。ログルーターは Fluent Bit を使用します。Fluent Bit には CloudWatch プラグインがあり、組み込みメトリックス形式のステートメントをサポートするように設定できます。

Fargate ノードのポッドレベルのメトリクスを取得してキャプチャするには、Prometheus サーバーを Amazon EKS クラスターにデプロイして Fargate ノードからメトリクスを収集します。Prometheus は永続ストレージを必要とするため、永続ストレージに Amazon Elastic File System (Amazon EFS) を使用する場合は、Prometheus を Fargate にデプロイできます。また、Amazon EC2 バッキングされたノードに Prometheus をデプロイすることもできます。詳細については、 ブログの「Monitoring Amazon EKS on AWS Fargate using Prometheus and Grafana」を参照してください。 AWS

Amazon EKS における Prometheus モニタリング

Amazon Managed Service for Prometheus は、オープンソース Prometheus 用のスケーラブルで安全な AWS マネージドサービスを提供します。Prometheus クエリ言語 (PromQL) を使用して、オペレーションメトリクスの取り込み、格納、およびクエリのための基盤インフラストラクチャを管理することなく、コンテナ化されたワークロードのパフォーマンスを監視できます。Amazon EKS と Amazon ECS から Prometheus メトリクスを収集するには、コレクションエージェントとして AWS Distro for OpenTelemetry (ADOT) または Prometheus サーバーを使います。

CloudWatch コンテナインサイトの Prometheus モニタリングでは、CloudWatch エージェントの設定と使用、Amazon ECS、Amazon EKS、および Kubernetes ワークロードから Prometheus メトリクスの検出、そして CloudWatch メトリクスとして取り込むことができます。このソリューションは、CloudWatch が主要なオブザーバビリティおよびモニタリングソリューションである場合に適切です。ただし、次のリストでは、Prometheus 向けの Amazon Managed Services が Prometheus メトリクスの取り込み、保存、クエリをより柔軟に提供するユースケースの概要を示します。

- Prometheus 向け Amazon Managed Services では、Amazon EKS または自己管理型 Kubernetes にデプロイされた既存の Prometheus サーバーを使用し、ローカルで設定されたデータストアの代わりに Prometheus 向けの Amazon Managed Services に書き込むように設定できます。これにより、Prometheus サーバーとそのインフラストラクチャの可用性の高いデータストアを管理するという未分化の重労働がなくなります。Amazon Managed Service for Prometheus は、 AWS クラウドで活用したい成熟した Prometheus デプロイがある場合に適しています。
- Grafana は可視化のためのデータソースとして Prometheus を直接サポートしています。コンテナのモニタリングに CloudWatch ダッシュボードの代わりに Prometheus で Grafana を使用する場合は、Prometheus 向け Amazon Managed Services でお客様の要件を満たすことができます。Prometheus 向け Amazon Managed Services は Amazon Managed Grafana と統合し、マネージドオープンソースのモニタリングおよび可視化ソリューションを提供します。
- Prometheus を使用すると、PromQL クエリを使用してオペレーションメトリクスの分析を実行できます。対照的に、CloudWatch エージェントは Prometheus メトリクスを埋め込みメトリクスフォーマットで CloudWatch Logs に取り込み、その結果 CloudWatch メトリクスが生成されます。埋め込みメトリクスフォーマットのログは、CloudWatch Logs インサイト を使用して、クエリできます。
- モニタリングとメトリクスのキャプチャに CloudWatch を使用する予定がない場合は、Prometheus サーバーと Grafana などの可視化ソリューションで Prometheus 向け Amazon Managed Services を使用する必要があります。Prometheus サーバーを構成して、Prometheus ターゲットからメトリクスをスクレイピングし、サーバーを Prometheus 向け Amazon Managed

Services ワークスペースにリモート書き込みできるように 設定する必要があります。Amazon Managed Grafana を使えば、付属のプラグインを使用して、Amazon Managed Grafana を Prometheus 向け Amazon Managed Services データソースと直接統合できます。メトリクスデータは Prometheus 向け Amazon Managed Services に保存されるため、CloudWatch エージェントをデプロイする依存性や CloudWatch にデータを取り込む必要はありません。CloudWatch エージェントは、Prometheus のコンテナインサイトモニタリングに必要です。

ADOT コレクターを使用して Prometheus で計測したアプリケーションからスクレイピングし、Prometheus 向けの Amazon Managed Services に、メトリクスを送信することもできます。ADOT コレクターの詳細については、AWS Distro for OpenTelemetry のドキュメントを参照してください。

のログ記録とメトリクス AWS Lambda

Lambda は、ワークロードのためのサーバーを管理および監視する必要性を排除し、アプリケーションのコードにさらなる設定やインスツルメンテーションを行うことなく、自動的に CloudWatch メトリクス、および CloudWatch Logs と連動します。このセクションでは、Lambda で使用されるシステムのパフォーマンス特性と、構成の選択がパフォーマンスにどのように影響するかを理解することができます。また、パフォーマンスの最適化やアプリケーションレベルの問題の診断のために Lambda 関数のログに記録して監視するのに役立ちます。

Lambda 関数のログを記録する

Lambda は、Lambda 関数からの標準出力と標準エラーメッセージを CloudWatch Logs に自動的にストリーミングし、ロギングドライバを必要としません。Lambda はまた、Lambda 関数を実行するコンテナを自動的にプロビジョニングし、個別のログストリームにログメッセージを出力するように設定します。

それ以降の Lambda 関数の呼び出しでは、同じコンテナを再利用し、同じログストリーミングに出力することができます。Lambda は新しいコンテナをプロビジョニングし、新しいログストリーミングに起動を出力することもできます。

Lambda 関数が最初に呼び出されると、Lambda は自動的にロググループを作成します。Lambda 関数には複数のバージョンがあり、実行するバージョンを選択できます。Lambda 関数の呼び出しのすべてのログは、同じロググループに保存されます。名前は変更できず、`/aws/lambda/<YourLambdaFunctionName>` の形式で設定します。Lambda 関数のインスタンスごとに、ロググループ内に個別のログストリーミングが作成されます。Lambda には、ログストリームの標準的な命名規則があり、`YYYY/MM/DD/[<FunctionVersion>]<InstanceId>` の形式で設定します。`InstanceId` は、Lambda 関数インスタンスを識別 AWS するために によって生成されます。

CloudWatch Logs Insights でより簡単にクエリできるため、ログメッセージを JSON 形式でフォーマットすることをお勧めします。また、より簡単にフィルタリングやエクスポートを行うことができます。ロギングライブラリを使用すると、このプロセスを簡素化したり、独自のログ処理関数を記述したりできます。ログメッセージの書式と分類に役立つログライブラリを使用することをお勧めします。例えば、Lambda 関数が Python で書かれている場合は、Python ログモジュールをクリックして、メッセージをログに記録し、出力形式を制御することができます。Lambda は Python で記述された Lambda 関数の Python ログ記録ライブラリをネイティブに使用しており、Lambda 関数内のロガーを取得してカスタマイズできます。 AWS Labs は AWS Lambda Powertools for Python 開発者

ツールキットを作成しているため、コールドスタートなどのキーデータを使用してログメッセージを簡単に強化できます。ツールキットは、Python、Java、Typescript、および .NET で使用できます。

また、ログ出力レベルを変数で設定し、環境や要件に応じて調整することもベストプラクティスの一つです。Lambda 関数のコードは、使用されるライブラリに加えて、ログ出力レベルに応じて大量のログデータを出力する可能性があります。これは、ログのコストやパフォーマンスに影響を与える可能性があります。

Lambda では、コードを更新することなく、Lambda 関数のランタイム環境の環境変数を設定することができます。例えば、コードから取得できるログ出力レベルを定義する LAMBDA_LOG_LEVEL 環境変数を作成することができます。次の例では LAMBDA_LOG_LEVEL 環境変数を取得し、その値を使用してログ出力を定義しようとしています。環境変数が設定されていない場合、デフォルトで INFO レベルになります。

```
import logging
from os import getenv

logger = logging.getLogger()
log_level = getenv("LAMBDA_LOG_LEVEL", "INFO")
level = logging.getLevelName(log_level)
logger.setLevel(level)
```

CloudWatch から他の宛先にログを送信する

サブスクリプションフィルターを使用して、他の送信先 (Amazon OpenSearch Service や Lambda 関数など) にログを送信できます。Amazon OpenSearch Service を使用しない場合は、Lambda 関数を使用してログを処理し、 AWS SDKs を使用して任意の AWS サービスに送信できます。

また、Lambda 関数の AWS クラウド外のログ送信先に SDKs を使用して、選択した送信先にログステートメントを直接送信することもできます。このオプションを選択する場合、レイテンシー、追加の処理時間、エラーおよびリトライ処理、運用ロジックの Lambda 関数への結合などの影響を考慮することをお勧めします。

Lambda 関数のメトリクス

Lambda を使用すると、サーバーを管理したりスケーリングすることなくコードを実行できるため、システムレベルの監査と診断の負担がほぼなくなります。ただし、Lambda 関数のシステムレベルで

パフォーマンスと呼び出しメトリクスを理解することは重要です。これは、リソース構成を最適化し、コードのパフォーマンスを向上させるのに役立ちます。パフォーマンスを効果的に監視および測定することで、ユーザーエクスペリエンスを向上させ、Lambda 関数を適切にサイジングすることでコストを下げることができます。一般的に、Lambda 関数として実行されるワークロードには、取得、および分析が必要なアプリケーションレベルのメトリクスもあります。Lambda は、アプリケーションレベルの CloudWatch メトリクスの取得を容易にするために、埋め込みメトリクス形式を直接サポートします。

システムレベルのメトリクス

Lambda は CloudWatch メトリクスと自動的に統合され Lambda 関数の標準的なメトリクス のセットを提供します。また Lambda は、これらのメトリクスを使用して Lambda 関数ごとに個別のモニタリングダッシュボードも提供します。モニタリングする必要がある重要なメトリクスは、エラーと呼び出しエラーの 2 つです。呼び出しエラーと他のエラータイプの違いを理解することは、Lambda デプロイの診断とサポートに役立ちます。

呼び出しエラー は Lambda 関数の実行を防ぎます。これらのエラーは、コードが実行される前に発生するため、コード内でエラー処理を実装して識別することができません。代わりに、これらのエラーを検出し、オペレーションとワークロードの所有者に通知する Lambda 関数のアラームを設定する必要があります。これらのエラーは、多くの場合、設定や権限のエラーに関連しており、設定や権限の変更が原因で発生することがあります。呼び出しエラーは、関数の複数回の呼び出しを引き起こすリトライを開始する可能性があります。

Lambda 関数を正常に起動した場合、関数から例外がスローされた場合でも HTTP 200 レスポンスが返されます。Lambda 関数は、エラーハンドリングを実装し、例外を発生させ、`Errors` メトリクスが Lambda 関数の失敗した実行を捕捉し識別するようにする必要があります。Lambda 関数の呼び出しから、実行が完全に失敗したか、部分的に失敗したか、成功したかを判断するための情報を含むフォーマットされたレスポンスを返す必要があります。

CloudWatch では、個々の Lambda 関数に対して有効化できる CloudWatch Lambda インサイト を提供しています。Lambda インサイトは、システムレベルのメトリクス (例えば、CPU時間、メモリ、ディスク、ネットワーク使用量) を収集、集約、要約します。また、Lambda インサイトは、診断情報 (コールドスタートや Lambda ワーカーのシャットダウンなど) を収集、集約、要約し、問題の切り分けや迅速な解決に役立てます。

Lambda インサイトは、埋め込みメトリック形式を使用して、Lambda 関数の名前に基づいたログストリーミング名のプレフィックスを持つ `/aws/lambda-insights/` ロググループにパフォーマンス情報を自動的に出力します。これらのパフォーマンスログイベントは、自動 CloudWatch ダッ

シュボードの基礎となる CloudWatch メトリクスを作成します。パフォーマンステストや本番環境では、Lambda インサイトを有効にすることをお勧めします。Lambda Insightsによって作成される追加のメトリクスには、Lambda 関数を正しくサイジングして、必要ない容量に対する支払いを回避するのに役立つ `memory_utilization` が含まれます。

アプリケーション・メトリクス

CloudWatch では、埋め込みメトリクス形式を使用して、独自のアプリケーションメトリクスを作成、取得することも可能です。AWS 埋め込みメトリクス形式に提供されているライブラリを活用して、埋め込みメトリクス形式ステートメントを作成して CloudWatch に出力できます。統合された Lambda CloudWatch ロギング機能は、適切にフォーマットされた埋め込みメトリクスフォーマットのステートメントを処理し、抽出するように構成されています。

CloudWatch でのログの検索および分析

ログとメトリクスを一貫したフォーマットと場所にキャプチャした後、それらを検索および分析して、問題の特定とトラブルシューティングに加えて、オペレーション効率を向上させることができます。ログを検索および分析しやすくするために、ログを適切な形式 (JSON など) でキャプチャすることをお勧めします。ほとんどのワークロードでは、ネットワーク、コンピューティング、ストレージ、データベースなどの AWS リソースのコレクションを使用します。可能であれば、すべての AWS ワークロードを効果的にモニタリングおよび管理するために、これらのリソースのメトリクスとログをまとめて分析し、関連付ける必要があります。

CloudWatch には、さまざまな AWS リソースにわたるアプリケーションのメトリクスとログをまとめて定義およびモニタリングする CloudWatch Application Insights、メトリクスの異常を明らかにする CloudWatch Anomaly Detection、CloudWatch Logs でログデータをインタラクティブに検索および分析する CloudWatch Log Insights など、ログとメトリクスの分析に役立ついくつかの機能が用意されています。 CloudWatch

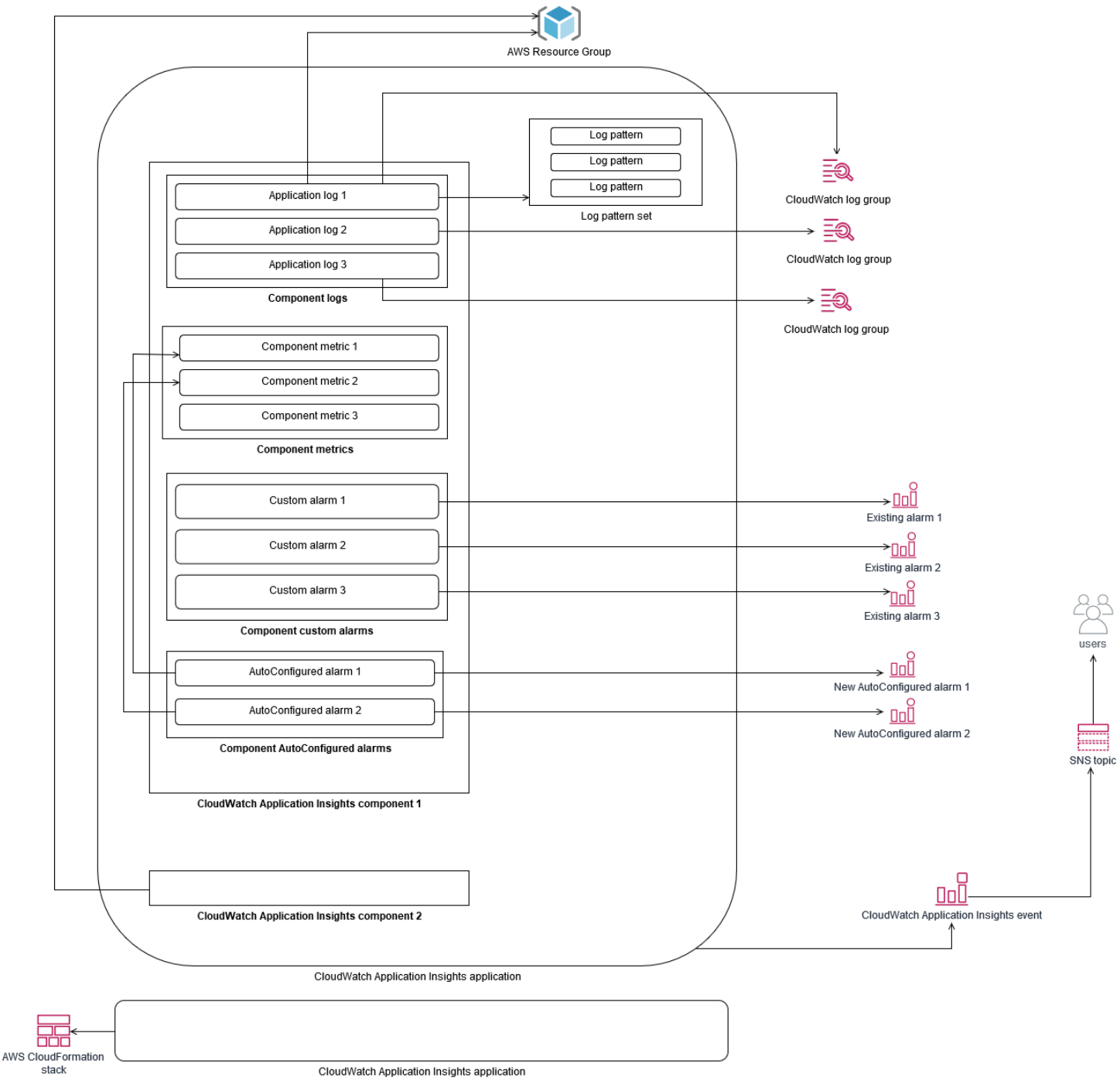
CloudWatch アプリケーションインサイトを使用したアプリケーションのモニタリングと分析

アプリケーションの所有者は Amazon CloudWatch アプリケーションインサイトを使用して、ワークロードの自動モニタリングと分析を設定できます。これは、アカウント内のすべてのワークロードに対して設定されたデフォルトのシステムレベルのモニタリングに追加して、設定できます。CloudWatch アプリケーションインサイトによるモニタリングの設定は、アプリケーションチームが事前にオペレーションと連携し、平均復旧時間（MTTR）を削減するのに役立ちます。CloudWatch アプリケーションインサイトは、アプリケーションレベルのロギングとモニタリングを確立するために必要な労力を削減するのに役立ちます。また、チームによるロギングとモニタリングの責任分担を支援するコンポーネントベースのフレームワークも提供します。

CloudWatch アプリケーションインサイトは、Resource Groups を使用して、アプリケーションとしてまとめてモニタリングする必要があるリソースを特定します。Resource Groups 内のサポートされているリソースは、CloudWatch アプリケーションインサイトのアプリケーションの個別に定義されたコンポーネントになります。CloudWatch アプリケーションインサイトのアプリケーションの各コンポーネントには、独自のログ、メトリクス、アラームがあります。

ログについては、コンポーネントと CloudWatch アプリケーションインサイトのアプリケーション内で使用するログパターンセットを定義します。ログパターンセットは、正規表現に基づいて検索するログパターンの集まりで、パターンが検出されたときの重大度が低、中、または高です。メト

リクスについては、サービス固有およびサポートされているメトリクスのリストから、各コンポーネントについてモニタリングするメトリクスを選択します。アラームの場合、CloudWatch アプリケーションインサイトは、モニタリング対象のメトリクスのスタンダードまたは異常検出アラームを自動的に作成し、設定します。CloudWatch アプリケーションインサイトは、「CloudWatch のドキュメント」のCloudWatch アプリケーションインサイトで説明されているテクノロジーのメトリクスとログキャプチャの自動設定があります。次の図は、CloudWatch アプリケーションインサイトのコンポーネントとそのロギングおよびモニタリングの設定との関係を示します。各コンポーネントは、CloudWatch Logsとメトリクスを使用してモニタリングする独自のログとメトリクスを定義しています。



CloudWatch アプリケーションインサイトによってモニタリングされる EC2 インスタンスには、Systems Manager と CloudWatch エージェントとアクセス許可が必要です。詳細については、「CloudWatch のドキュメント」にある「CloudWatch アプリケーションインサイトを使用したアプリケーションの設定の前提条件」を参照してください。CloudWatch アプリケーションインサイトでは、CloudWatch エージェントをインストールおよび更新するために、Systems Manager を使用します。CloudWatch アプリケーションインサイトで設定されたメトリクスとログは、CloudWatch エージェント設定ファイルを作成し、このファイルは、各 CloudWatch アプリケーションインサイト

のコンポーネントの AmazonCloudWatch-ApplicationInsights-SSMParameter プレフィックスを付けて、Systems Managerのパラメータに格納されます。これにより、別の CloudWatch エージェント設定ファイルが EC2 インスタンスの CloudWatch エージェント設定ディレクトリに追加されます。Systems Manager コマンドを実行して、EC2 インスタンスのアクティブな設定にこの設定を追加します。CloudWatch アプリケーションインサイトを使用しても、既存の CloudWatch エージェントの設定には影響しません。独自のシステムおよびアプリケーションレベルの CloudWatch エージェント設定に加えて、CloudWatch アプリケーションインサイトを使用できます。ただし、設定が重複しないようにする必要があります。

CloudWatch Logs インサイトを使用したログ分析の実行

CloudWatch Logs インサイトを使用すると、単純なクエリ言語を使用して複数のロググループを簡単に検索できます。アプリケーションログが JSON 形式で構造化されている場合、CloudWatch Logs インサイトは、複数のロググループでログストリーミング全体の JSON フィールドを自動的に検出します。CloudWatch Logs インサイトを使用して、アプリケーションおよびシステムログを分析できます。これにより、将来の使用に備えてクエリが保存されます。CloudWatch Logs インサイトのクエリ構文は、アプリケーションやパフォーマンス分析のトラブルシューティングに役立つ sum ()、avg ()、count ()、min ()、max () などの関数による集計などの機能をサポートしています。

組み込みメトリクス形式を使用して CloudWatch メトリクスを作成する場合、サポートされている集計関数を使用して、埋め込みメトリクス形式のログをクエリして 1 回限りのメトリクスを生成できます。これにより、カスタムメトリクスとしてアクティブにキャプチャするのではなく、必要に応じて特定のメトリクスを生成するために必要なデータポイントをキャプチャすることで、CloudWatch モニタリングのコストを削減できます。これは、基数が高いディメンションで多数のメトリクスが発生する場合に特に効果的です。CloudWatch コンテナインサイトもこのアプローチをとり、詳細なパフォーマンスデータをキャプチャしますが、このデータのサブセットの CloudWatch メトリクスのみを生成します。

たとえば、次の埋め込みメトリクスエントリは、組み込みメトリクス形式ステートメントでキャプチャされたメトリクスデータから CloudWatch メトリクスの限定されたセットのみを生成します。

```
{
 "AutoScalingGroupName": "eks-e0bab7f4-fa6c-64ba-dbd9-094aee6cf9ba",
 "CloudWatchMetrics": [
 {
 "Metrics": [
 {
 "Unit": "Count",
 "Name": "pod_number_of_container_restarts"
```

```
}
],
"Dimensions": [
[
"PodName",
"Namespace",
"ClusterName"
]
],
"Namespace": "ContainerInsights"
}
],
"ClusterName": "eksdemo",
"InstanceId": "i-03b21a16b854aa4ca",
"InstanceType": "t3.medium",
"Namespace": "amazon-cloudwatch",
"NodeName": "ip-172-31-10-211.ec2.internal",
"PodName": "cloudwatch-agent",
"Sources": [
"cadvisor",
"pod",
"calculated"
],
"Timestamp": "1605111338968",
"Type": "Pod",
"Version": "0",
"pod_cpu_limit": 200,
"pod_cpu_request": 200,
"pod_cpu_reserved_capacity": 10,
"pod_cpu_usage_system": 3.268605094109382,
"pod_cpu_usage_total": 8.899539221131045,
"pod_cpu_usage_user": 4.160042847048305,
"pod_cpu_utilization": 0.44497696105655227,
"pod_cpu_utilization_over_pod_limit": 4.4497696105655224,
"pod_memory_cache": 4096,
"pod_memory_failcnt": 0,
"pod_memory_hierarchical_pgfault": 0,
"pod_memory_hierarchical_pgmajfault": 0,
"pod_memory_limit": 209715200,
"pod_memory_mapped_file": 0,
"pod_memory_max_usage": 43024384,
"pod_memory_pgfault": 0,
"pod_memory_pgmajfault": 0,
"pod_memory_request": 209715200,
```

```
 "pod_memory_reserved_capacity": 5.148439982463127,
 "pod_memory_rss": 38481920,
 "pod_memory_swap": 0,
 "pod_memory_usage": 42803200,
 "pod_memory_utilization": 0.6172094650851303,
 "pod_memory_utilization_over_pod_limit": 11.98828125,
 "pod_memory_working_set": 25141248,
 "pod_network_rx_bytes": 3566.4174629544723,
 "pod_network_rx_dropped": 0,
 "pod_network_rx_errors": 0,
 "pod_network_rx_packets": 3.3495665260575094,
 "pod_network_total_bytes": 4283.442421354973,
 "pod_network_tx_bytes": 717.0249584005006,
 "pod_network_tx_dropped": 0,
 "pod_network_tx_errors": 0,
 "pod_network_tx_packets": 2.6964010534762948,
 "pod_number_of_container_restarts": 0,
 "pod_number_of_containers": 1,
 "pod_number_of_running_containers": 1,
 "pod_status": "Running"
}
```

ただし、キャプチャしたメトリクスをクエリして、さらなるインサイトを得ることができます。例えば、次のクエリを実行して、メモリページ障害のある最新の 20 ポッドを表示できます。

```
fields @timestamp, @message
| filter (pod_memory_pgfault > 0)
| sort @timestamp desc
| limit 20
```

Amazon OpenSearch Service を使用したログ分析の実行

CloudWatch は Amazon OpenSearch Service と統合され、サブスクリプションフィルターを使用してCloudWatch ロググループから任意の Amazon OpenSearch Service クラスターにログデータをストリーミングできます。CloudWatch は、プライマリログとメトリクスのキャプチャと分析に使用し、次のユースケースでは Amazon OpenSearch Service で拡張できます。

- きめ細かなデータアクセスコントロール – Amazon OpenSearch Service を使用すると、データへのアクセスをフィールドレベルに制限し、ユーザーのアクセス許可に基づいてフィールド内のデータを匿名化できます。これは、機密データを公開せずにトラブルシューティングをサポートしたい場合に便利です。

- 複数のアカウント、リージョン、インフラストラクチャ間でログを集約して検索する – 複数のアカウントとリージョンから共通の Amazon OpenSearch Service クラスターにログをストリーミングできます。集中化されたオペレーションチームは、トレンドや問題を分析し、アカウントとリージョン間で分析を実行できます。CloudWatch ログを Amazon OpenSearch Service にストリーミングすると、マルチリージョンアプリケーションを一元的に検索および分析することもできます。
- ElasticSearch エージェントを使用して Amazon OpenSearch Service に直接ログを出荷して強化する – アプリケーションおよびテクノロジースタックコンポーネントは、CloudWatch エージェントでサポートされていない OSs を使用できます。また、ログデータがログソリューションに出荷される前に、ログデータを強化して変換することもできます。Amazon OpenSearch Service は、ログデータを Amazon OpenSearch Service に送信する前にログのエンリッチメントと変換をサポートする Elastic Beats ファミリーデータシッパーや Logstash などの標準 OpenSearchクライアントをサポートしています。
- 既存の運用管理ソリューションでは、ログ記録とモニタリングに ElasticSearch、Logstash、Kibana (ELK) スタックを使用しています。多くのワークロードが既に設定されている Amazon OpenSearch Service またはオープンソースの Elasticsearch にすでに多額の投資を行っている可能性があります。また、継続して使いたいKibanaで作成されているオペレーションダッシュボードがある場合もあります。

CloudWatch ログを使用する予定がない場合は、Amazon OpenSearch Service がサポートするエージェント、ログドライバー、ライブラリ (Fluent Bit、Fluentd、logstash、Open Distro for ElasticSearch API など) を使用して、ログを Amazon OpenSearch Service に直接送信し、CloudWatch をバイパスできます。ただし、 AWS サービスによって生成されたログをキャプチャするソリューションも実装する必要があります。CloudWatch Logs は、多くの AWS サービスの主要なログキャプチャソリューションであり、複数のサービスが CloudWatch に新しいロググループを自動的に作成します。たとえば、Lambda は各 Lambda 関数に対して新しいロググループを作成します。ロググループのサブスクリプションフィルターを設定して、ログを Amazon OpenSearch Service にストリーミングできます。Amazon OpenSearch Service にストリーミングする個々のロググループごとにサブスクリプションフィルターを手動で設定できます。または、新しいロググループを ElasticSearch クラスターに自動的にサブスクライブするソリューションをデプロイすることもできます。ログは、同じアカウントまたは集中型アカウントの ElasticSearch クラスターにストリーミングできます。同じアカウント内の ElasticSearch クラスターにログをストリーミングすることで、ワークロードの所有者はワークロードの分析とサポートを強化できます。

アカウント、リージョン、およびアプリケーションのログを集約するために、集中型のアカウントまたは共有アカウントで ElasticSearch クラスターを設定することを検討する必要があります。たとえば、 は集中ログ記録に使用されるログアーカイブアカウント AWS Control Tower を設定します。

で新しいアカウントが作成されると AWS Control Tower、その AWS CloudTrail および AWS Config ログはこの一元化されたアカウントの S3 バケットに配信されます。によって計測されるログ記録 AWS Control Tower は、設定、変更、監査ログ記録用です。

Amazon OpenSearch Service を使用して一元化されたアプリケーションログ分析ソリューションを確立するには、1 つ以上の一元化された Amazon OpenSearch Service クラスターを一元化されたログ記録アカウントにデプロイし、他のアカウントのロググループを設定して、ログを一元化された Amazon OpenSearch Service クラスターにストリーミングできます。

個別の Amazon OpenSearch Service クラスターを作成して、アカウント全体に分散される可能性のあるクラウドアーキテクチャのさまざまなアプリケーションまたはレイヤーを処理できます。個別の Amazon OpenSearch Service クラスターを使用すると、セキュリティと可用性のリスクが軽減され、共通の Amazon OpenSearch Service クラスターを使用することで、同じクラスター内のデータの検索と関連付けが容易になります。

CloudWatch によるアラームのオプション

重要なメトリクスを一度だけ自動分析することで、ワークロードに影響を与える前に問題を検出して解決できます。CloudWatch では、特定の期間に複数の統計情報を使用して、複数のメトリクスを簡単にグラフ化および比較できます。CloudWatch を使用して、必要なディメンション値を持つすべてのメトリクスを検索し、分析に必要なメトリクスを見つけることができます。

ワークロードをモニタリングするためのベースラインとして使用する、メトリクスとディメンションの初期セットを含めることで、メトリクス取得のアプローチを開始することをお勧めします。時間の経過と共に、ワークロードは成熟し、さらに分析とサポートに役立つメトリクスとディメンションを追加できます。アプリケーションまたはワークロードは複数の AWS リソースを使用し、独自のカスタムメトリクスを持つ場合があります。これらのリソースを名前空間にグループ化して識別しやすくする必要があります。

また、特定の問題を診断するための関連するログやモニタリングデータをすばやく特定できるように、ログとモニタリングデータの相関関係を考慮する必要があります。AWS X-Ray トレースマップを使用して、問題を診断するためのトレース、メトリクス、ログ、アラームを関連付けることができます。また、システムやサービス全体の問題をすばやく検索して特定できるように、メトリクスのディメンションやワークロードのログに識別子を追加することも検討する必要があります。

CloudWatch アラームを使用したモニタリングとアラームの実行

CloudWatch アラーム を使用することで、ワークロードまたはアプリケーションでの手動でのモニタリングを減らすことができます。まず、各ワークロードコンポーネントについて取得しているメトリクスを確認し、各メトリクスの適切なしきい値を決定する必要があります。しきい値に違反した時に通知する必要のあるチームメンバーを特定してください。個々のチームメンバーではなく、ディストリビューショングループを確立してターゲットにする必要があります。

CloudWatch アラームでは、サービス管理ソリューションと統合することで、新しいチケットを自動的に作成し、オペレーションワークフローを実行できます。たとえば、 は AWS Service Management Connector for ServiceNow と AWS を提供しAWS Service Management Connector、統合をすばやくセットアップできるようにします。このアプローチは、発生したアラームが認識され、これらの製品ですでに定義されている既存のオペレーションワークフローに整合させるために非常に重要です。

また、同じメトリクスに対して、異なるしきい値と評価期間を持つ複数のアラームを作成することもできます。これにより、エスカレーションプロセスの確立に役立ちます。例えば、顧客の注文を追

跡する OrderQueueDepth メトリクスがある場合、平均 1 分間の短時間で低いしきい値を定義し、アプリケーションチームメンバーにメールや Slack で通知することができます。また、同じしきい値で 15 分間以上の同じメトリクスに別のアラームを定義し、それをアプリケーションチームとアプリケーションチームのリードにページ、メール、および通知することもできます。最後に、30 分間以上のハードアベレッジのしきい値に 3 番目のアラームを定義して、上層部に通知し、以前通知されたすべてのチームメンバーに通知することができます。複数のアラームを作成することで、条件ごとに異なるアクションを実行できます。簡単な通知プロセスから始めて、必要に応じて調整および改善することができます。

CloudWatch 異常検出を使用したモニタリングとアラームの実行

特定のメトリクスに適用するしきい値が不明な場合、またはアラームで観測された履歴値に基づいたしきい値を自動的に調整する場合、CloudWatch 異常検出 を使用できます。CloudWatch 異常検出は、例えば、当日配送の注文が締切時間前に増加するなど、アクティビティに定期的で予測可能な変化がある可能性のあるメトリクスに特に有効です。異常検出により、自動調整されるしきい値が有効になり、誤警報を減らすことができます。各メトリクスと統計値に対して異常検出を有効にし、異常値に基づいてアラームを発するように CloudWatch を設定できます。

例えば、CPUUtilization メトリクスと EC2 インスタンスでのAVG 統計の異常検出を可能にできます。異常検出では、最大 14 日間の履歴データを使用して、machine learning (ML) モデルを作成します。異なるしきい値を持つ複数のスタンダードアラームを作成するのと同様に、異なる異常検出帯域を持つ複数のアラームを作成し、アラームエスカレーションプロセスを確立することができます。

このセクションの詳細については、CloudWatch ドキュメント内の 異常検出に基づく CloudWatch アラームの作成 を参照してください。

複数のリージョンとアカウントにまたがるアラーム設定

アプリケーションおよびワークロードの所有者は、複数のリージョンにまたがるワークロードに対してアプリケーションレベルのアラームを作成する必要があります。ワークロードがデプロイされている各アカウントとリージョン内に個別のアラームを作成することをお勧めします。アカウントとリージョンに依存しない CloudFormation StackSets とテンプレートを使用して、必要なアラームでアプリケーションリソースをデプロイすることで、このプロセスを簡素化および自動化できます。templateYou一般的な Amazon Simple Notification Service (Amazon SNS) トピックをターゲットにするようにアラームアクションを設定できます。つまり、アカウントやリージョンに関係なく、同じ通知または修復アクションが使用されます。

マルチアカウントおよびマルチリージョン環境では、アカウントとリージョンの問題を監視するために、 CloudFormation StackSets やすべての EC2 インスタンス間の平均 `CPUUtilization` などの集計メトリクスを使用して、アカウントとリージョンの集約アラームを作成することをお勧めします。

また、キャプチャするスタンダード CloudWatch メトリクスとログ用に設定されたワークロードごとにスタンダードアラームを作成することも検討する必要があります。例えば、各 EC2 インスタンスに対して個別のアラームを作成すると、CPU 使用率メトリクスをモニタリングし、平均 CPU 使用率が日常的に 80% を超えると中央のオペレーションチームに通知することができます。また、日常的に 10% 未満の平均 CPU 使用率をモニタリングするスタンダードアラームを作成することもできます。これらのアラームで、中央オペレーションチームが特定のワークロードオーナーと協力して、必要に応じて EC2 インスタンスのサイズを変更することができます。

EC2 インスタンスタグを使用したアラーム作成の自動化

EC2 インスタンスのアラームのスタンダード設定を作成すると、時間がかかり、一貫性がなく、エラーが発生しやすくなります。amazon-cloudwatch-auto-alarms ソリューションを使用して、アラーム作成プロセスを高速化すると、EC2 インスタンスの CloudWatch アラームのスタンダードセットを自動的に作成し、EC2 インスタンスタグに基づいてカスタムアラームを作成することができます。このソリューションにより、標準アラームを手動で作成する必要がなくなり、CloudEndure などのツールを使用する EC2 インスタンスの大規模な移行時に役立ちます。このソリューションを CloudFormation StackSets でデプロイして、複数のリージョンとアカウントをサポートすることもできます。詳細については、 AWS ブログ上の タグを使用して Amazon EC2 インスタンスの Amazon CloudWatch アラームを作成および維持する を参照してください。

アプリケーションとサービスの可用性のモニタリング

CloudWatch は、アプリケーションとワークロードのパフォーマンスとランタイムの側面をモニタリングおよび分析するのに役立ちます。また、アプリケーションとワークロードの可用性と到達可能性についてもモニタリングする必要があります。これは、Amazon Route 53 ヘルスチェック そして CloudWatch Synthetics でアクティブモニタリングアプローチを使用することができます。

Route 53 ヘルスチェックは、HTTP または HTTPS 経由でウェブページへの接続を監視する場合や、パブリックドメインネームシステム (DNS) 名または IP アドレスへの TCP 経由のネットワーク接続を監視する場合に使用できます。Route 53 ヘルスチェックは、10 秒または 30 秒間隔で指定したリージョンからの接続を開始します。ヘルスチェックを実行する複数のリージョンを選択でき、各ヘルスチェックは個別に実行され、少なくとも 3 つのリージョンを選択する必要があります。ヘルスチェック評価のために返された最初の 5,120 バイトのデータに HTTP または HTTPS リクエストのレスポンス本文を検索して、特定の部分文字列を検索できます。HTTP または HTTPS リクエストは、2xx または 3xx 応答を返せば、正常と見なされます。Route 53 ヘルスチェックを使用して、他のヘルスチェックの健全性をチェックして、複合ヘルスチェックを作成できます。これは、複数のサービスエンドポイントがあり、そのうちの 1 つが異常になったときに同じ通知を実行したい場合に実行できます。DNS に Route 53 を使用する場合は、Route 53 を 別の DNS エントリにフェールオーバーし、ヘルスチェックが異常になった場合に設定できます。重要なワークロードごとに、通常の操作で重要な外部エンドポイントに対して Route 53 ヘルスチェックを設定することを検討する必要があります。Route 53 ヘルスチェックは、アプリケーションにフェールオーバーロジックを書き込むのを防ぐのに役立ちます。

CloudWatch シンセティックスを使用すると、ワークロードの健全性と可用性を評価するスクリプトとして canary を定義できます。Canaries は、Node.js または Python で記述され、HTTP または HTTPS プロトコルで動作するスクリプトです。Node.js または Python をフレームワークとして使用する Lambda 関数をアカウント内に作成します。定義する各 canary は、異なるエンドポイントに対して複数の HTTP または HTTPS 呼び出しを実行できます。つまり、ユースケースやダウンストリームの依存関係を持つエンドポイントなど、一連のステップの正常性をモニタリングできます。Canaries は、実行された各ステップを含む CloudWatch メトリクスを作成し、異なるステップを個別にアラームおよび測定できるようにします。カナリアは、Route 53 ヘルスチェックよりも多くの計画と労力を必要とするが、高度にカスタマイズ可能なモニタリングと評価アプローチを提供します。Canaries は、仮想プライベートクラウド (VPC) 内で実行されるプライベートリソースもサポートしているため、エンドポイントのパブリック IP アドレスがない場合の可用性の監視に最適です。また、VPC 内からエンドポイントへの接続がある限り、Canaries を使用してオンプレミスのワークロードを監視することもできます。これは、オンプレミスに存在するエンドポイントを含むワークロードがある場合に特に重要です。

を使用したアプリケーションのトレース AWS X-Ray

アプリケーションを介したリクエストは、Amazon EC2、コンテナ、または Lambda のオンプレミスサーバーで実行されているウェブサービスと、アプリケーションおよびデータベースへの呼び出しで構成されます。アプリケーショントレースを実装することで、分散コンポーネントとサービスを使用するアプリケーションの問題の根本原因をすばやく特定できます。AWS X-Ray を使用して、複数のコンポーネントにわたりお客様のアプリケーションのリクエストをトレースします。X-Ray は、アプリケーションコンポーネントを流れるリクエストをサンプリングして サービスグラフに可視化し、各コンポーネントはセグメントとして表現されます。X-Ray はリクエストが複数のコンポーネントを流れるときに、トレース識別子を生成して相関させることができるため、リクエストをエンドツーエンドで表示できます。注釈とメタデータを含めることにより、リクエストの特性を一意に検索して識別できるようにすることで、この機能をさらに強化できます。

X-Ray を使用して、アプリケーション内の各サーバーまたはエンドポイントを構成し、計測することをお勧めします。X-Ray は、X-Ray サービスを呼び出すことによって、アプリケーションコードに実装されます。X-Ray は AWS SDKs も提供します。X-Ray SDK は、他のサービス (HTTP、MySQL、PostgreSQL、MongoDB など) への呼び出しに使用される共通ライブラリへのパッチを提供します。

X-Ray は Amazon EC2 および Amazon ECS にインストールして実行できる X-Ray デーモンを提供し、X-Ray にデータを中継できます。X-Ray は、リクエストを処理した X-Ray デーモンを実行しているサーバーとコンテナからパフォーマンスデータをキャプチャするアプリケーションのトレースを作成します。X-Ray は、 AWS SDK にパッチを適用することで、Amazon DynamoDB などの AWS サービスへの呼び出しをサブセグメントとして自動的に計測します。X-Ray は Lambda 関数と自動的に統合することもできます。

アプリケーションコンポーネントが X-Ray デーモンを設定およびインストールできない、またはコードをインストルメント化できない外部サービスを呼び出す場合は、外部サービスへの呼び出しをラップするサブセグメントを作成できます X-Ray は、 を使用している場合、CloudWatch ログとメトリクスをアプリケーショントレースと関連付けます。つまり AWS X-Ray SDK for Java、関連するメトリクスとログのリクエストをすばやく分析できます。

X-Ray デーモンをデプロイし、Amazon EC2 でのアプリケーションとサービスをトレースする

X-Ray デーモンは、アプリケーションコンポーネントまたはマイクロサービスが実行される EC2 インスタンスにインストールして実行する必要があります。ユーザーデータスクリプトを使用し

て、EC2 インスタンスがプロビジョニングされるときに X-Ray デーモンをデプロイするか、独自の AMI を作成する場合は AMI ビルドプロセスに含めることができます。これは、EC2 インスタンスがエフェメラルである場合に特に便利です。

ステートマネージャーを使用して、X-Ray デーモンが EC2 インスタンスに一貫してインストールされていることを確認する必要があります。Amazon EC2 Windows インスタンスの場合、Systems Manager AWS-RunPowerShellScript ドキュメントを使用して、X-Ray エージェントをダウンロードしてインストールする Windows スクリプトを実行できます。Linux の EC2 インスタンスでは、AWS-RunShellScript ドキュメントを使用して、エージェントをサービスとしてダウンロードしてインストールする Linux スクリプトを実行できます。

Systems Manager である AWS RunRemoteScript ドキュメントを使用して、マルチアカウント環境でスクリプトを実行します。すべてのアカウントからアクセス可能な S3 バケットを作成する必要があります。 AWS Organizationsを使用する場合、組織ベースのバケットポリシーを使用した S3 バケットの作成を推奨します。次に、スクリプトを S3 バケットにアップロードしますが、EC2 インスタンスの IAM ロールにバケットとスクリプトへのアクセス許可があることを確認します。

ステートマネージャーを設定して、X-Ray エージェントがインストールされている EC2 インスタンスにスクリプトを関連付けるようにすることもできます。すべての EC2 インスタンスが X-Ray を必要としない場合や使用しない可能性があるため、インスタンスタグとの関連付けをターゲットにすることができます。例えば、`InstallAWSXRayDaemonWindows` または `InstallAWSXRayDaemonLinux` のタグに基づいてステートマネージャーの関連付けを作成できます。

X-Ray デーモンをデプロイし、Amazon ECS または Amazon EKS でのアプリケーションとサービスをトレースする

Amazon ECS や Amazon EKS などのコンテナベースのワークロードのサイドカーコンテナとして、X-Ray デーモンをデプロイできます。Amazon ECS を使用している場合、アプリケーションコンテナはコンテナリンクを使用してサイドカーコンテナに接続できます。または、awsvpc ネットワークモードを使用する場合は localhost のサイドカーコンテナに直接接続できます。

Amazon EKS の場合、アプリケーションのポッド定義に X-Ray デーモンを定義し、指定したコンテナポートの localhost 経由でアプリケーションがデーモンに接続できます。

X-Ray へのリクエストをトレースするように Lambda を設定する

アプリケーションには Lambda 関数の呼び出しが含まれる場合があります。デーモンプロセスは、Lambda によるフルマネージドプロセスであり、ユーザーによる設定ができないため、Lambda 用 X-Ray デーモンをインストールする必要はありません。Lambda 関数に対して有効にするには、を使用し AWS マネジメントコンソール 、X-Ray コンソールでアクティブトレースオプションを確認します。

さらに計測するには、X-Ray SDK を Lambda 関数にバンドルして送信呼び出しを記録し、注釈またはメタデータを追加できます。

X-Ray 向けにアプリケーションをインストルメントする

アプリケーションのプログラミング言語と一致する X-Ray SDK を評価し、アプリケーションが他のシステムに対して行うすべての呼び出しを分類する必要があります。選択したライブラリから提供されたクライアントを確認し、SDK がアプリケーションのリクエストまたはレスポンスのトレースを自動的に計測できるかどうかを確認します。SDK によって提供されるクライアントを他のダウンストリームシステムに使用できるかどうかを確認します。アプリケーションで呼び出される外部システムや、X-Ray では計測できない外部システムの場合は、カスタムサブセグメントを作成して、トレース情報でキャプチャして識別する必要があります。

アプリケーションをインストルメントするときは、リクエストの特定と検索に役立つ注釈を作成してください。例えば、アプリケーションでは、`customer id` など、顧客に識別子を使用したり、アプリケーションでの役割に基づいて異なるユーザーをセグメント化する可能性があります。

各トレースに対して最大 50 個の注釈を作成できますが、セグメントドキュメントが 64 キロバイトを超えない限り、1 つ以上のフィールドを含むメタデータオブジェクトを作成できます。注釈を選択して情報を検索し、メタデータオブジェクトを使用して、検索後のリクエストのトラブルシューティングに役立つコンテキストを増やす必要があります。

X-Ray のサンプリングルールを設定する

サンプリングルールをカスタマイズすることで、コードを変更または再デプロイすることなく、記録するレコードの量を制御したり、サンプリング動作を変更したりできます。サンプリングルールにより、X-Ray SDK に一連の基準に対して記録するリクエスト数を指示します。デフォルトで、X-Ray SDK は 毎秒、 最初のリクエストを記録し、追加リクエストの 5% を記録します。1 秒あたり 1 つのリクエストがリザーバです。これにより、サービスがリクエストを処理している限り、毎秒少なくと

も 1 つのトレースが記録されます。5% は、リザーバサイズを超えて追加リクエストがサンプリングされるレートです。

デフォルトの構成を確認して更新して、アカウントに適切な値を決定する必要があります。開発環境、テスト、パフォーマンステスト、本番稼働環境では、要件が異なる場合があります。受信するトラフィックの量または重要度のレベルに基づいて、独自のサンプリングルールを必要とするアプリケーションがある場合があります。ベースラインから始めて、ベースラインが要件を満たしているかどうかを定期的に再評価する必要があります。

CloudWatch を使用したダッシュボードとビジュアライゼーション

ダッシュボードを使用すると、アプリケーションとワークロードの懸念事項にすばやく焦点を当てるのに役立ちます。CloudWatch は自動ダッシュボードを提供し、CloudWatch メトリクスを使用するダッシュボードを簡単に作成することもできます。CloudWatch ダッシュボードは、複数のメトリクスを関連づけ、傾向を特定するのに役立つため、メトリクスを単独で表示するよりも多くのインサイトを提供します。たとえば、受信した注文、メモリ、CPU 使用率、データベース接続を含むダッシュボードは、注文数が増減している間に、複数の AWS リソース間でワークロードメトリクスの変化を関連付けるのに役立ちます。

ワークロードとアプリケーションをモニタリングするには、アカウントおよびアプリケーションレベルでダッシュボードを作成する必要があります。サービス固有のメトリクスで事前構成された AWS サービスレベルダッシュボードである、CloudWatch 自動ダッシュボードを使用して開始できます。自動サービスダッシュボードには、サービスのデフォルトの CloudWatch メトリクスがすべて表示されます。自動ダッシュボードは、各サービスメトリクスに使用されているすべてのリソースをグラフ化し、アカウント全体の異常値のリソースをすばやく特定するのに役立ちます。これにより、使用率の高いリソースと使用率の低いリソースを特定し、コストの最適化に役立ちます。

クロスサービスダッシュボードを作成する

クロスサービスダッシュボードを作成するには、 AWS サービスの自動サービスレベルダッシュボードを表示し、アクションメニューからダッシュボードに追加オプションを使用します。その後、他の自動ダッシュボードのメトリクスを新しいダッシュボードに追加し、メトリクスを削除してダッシュボードの焦点を絞り込むことができます。また、独自のカスタムメトリクスを追加して、主要な観測データを追跡する必要があります (例えば、受取注文や秒あたりの取引など)。独自のカスタムクロスサービスダッシュボードを作成すると、ワークロードに最も関連性の高いメトリクスに集中できます。キーメトリクスをカバーし、アカウント内のすべてのワークロードを表示するアカウントレベルのクロスサービスダッシュボードを作成することを推奨します。

クラウドオペレーションチーム用のセントラルオフィススペースまたは共用エリアがある場合、CloudWatch ダッシュボードを大型テレビモニターにフルスクリーンモードで表示し、自動更新できます。

アプリケーションまたはワークロード固有のダッシュボードを作成する

本番環境のすべての重要なアプリケーションまたはワークロードに関するキーメトリクスとリソースに焦点を当てた、アプリケーションおよびワークロード固有のダッシュボードを作成することをお勧めします。アプリケーションおよびワークロード固有のダッシュボードは、カスタムアプリケーションまたはワークロードメトリクスと、パフォーマンスに影響を与える重要な AWS リソースメトリクスに焦点を当てています。

インシデント発生後にキーメトリクスを追跡するために、CloudWatch アプリケーションまたはワークロードダッシュボードを定期的に評価してカスタマイズする必要があります。また、機能が導入または使用停止されたときに、アプリケーションまたはワークロード固有のダッシュボードを更新する必要があります。ワークロードおよびアプリケーション固有のダッシュボードの更新は、ロギングとモニタリングに加えて、品質を継続的に改善するために必要なアクティビティでなければなりません。

クロスアカウントまたはクロスリージョンダッシュボードを作成する

AWS リソースは主にリージョン別であり、メトリクス、アラーム、ダッシュボードはリソースがデプロイされているリージョンに固有です。このため、リージョンを変更して、クロスリージョンワークロードおよびアプリケーションのメトリクス、ダッシュボード、アラームを表示する必要があります。アプリケーションとワークロードを複数のアカウントに分割する場合は、再認証と各アカウントへのサインインが必要になる場合があります。ただし、CloudWatch では、単一のアカウントからのクロスアカウントおよびクロスリージョンデータの表示がサポートされています。つまり、単一のアカウントとリージョンでメトリクス、アラーム、ダッシュボード、ログウィジェットを表示できます。これは、ロギングおよびモニタリングアカウントを一元管理している場合に非常に便利です。

アカウント所有者とアプリケーションチームの所有者は、アカウント固有のクロスリージョンアプリケーション用のダッシュボードを作成して、キーメトリクスを一元的にモニタリングする必要があります。CloudWatch ダッシュボードは、クロスリージョンウィジェットを自動的にサポートします。つまり、追加の設定を行わずに、複数のリージョンのメトリクスを含むダッシュボードを作成できます。

重要な例外は CloudWatch Logs インサイトウィジェットです。ログデータは、現在ログインしているアカウントとリージョンにのみ表示できるためです。メトリクスフィルターを使用してログからリージョン固有のメトリクスを作成でき、これらのメトリクスはクロスリージョンダッシュボードに

表示できます。その後、それらのログをさらに分析する必要がある場合は、特定のリージョンに切り替えることができます。

オペレーションチームは、重要なクロスアカウントおよびクロスリージョンメトリクスをモニタリングする一元化されたダッシュボードを作成する必要があります。例えば、各アカウントとリージョンの CPU 使用率の合計 CPU 使用率を含むクロスアカウントダッシュボードを作成できます。また、Metric Math を使用し、複数のアカウントおよびリージョンにわたるデータを集約およびダッシュボードに表示できます。

Metric Math を使用してオブザーバビリティとアラームを微調整する

Metric Math を使用すると、ワークロードに関連する形式と数式でメトリクスを計算できます。計算されたメトリクスは、追跡目的でダッシュボードに保存および表示できます。例えば、デフォルトの Amazon EBS ボリュームメトリクスでは、特定の期間にわたって実行される読み取りオペレーション (`VolumeReadOps`) と書き込みオペレーション (`VolumeWriteOps`) の回数を示します。

ただし、 は IOPS での Amazon EBS ボリュームのパフォーマンスに関するガイドライン AWS を提供します。Amazon EBS ボリュームの IOPS を Metric Math でグラフ化して計算するには、`VolumeReadOps` と `VolumeWriteOps` を足して、次にそのメトリクスに選択した期間で割ります。

この例では、期間の IOPS を合計し、期間の長さで割って IOPS を取得します。次に、このメトリクスの数式に対してアラームを設定して、ボリュームの IOPS がボリュームタイプの最大容量に近づいたときに警告することができます。メトリクス計算を使用して CloudWatch メトリクスで Amazon Elastic File System (Amazon EFS) ファイルシステムをモニタリングする方法の詳細と例については、 AWS ブログの「Amazon CloudWatch メトリクス計算は Amazon EFS ファイルシステムのほぼリアルタイムのモニタリングを簡素化する」を参照してください。

CloudWatchContainer インサイトと CloudWatch Lambda インサイトで、Amazon ECS、Amazon EKS、および Lambda 自動ダッシュボードを使用する

CloudWatch コンテナインサイトは、Amazon ECS および Amazon EKS で実行されるコンテナワークロードの動的な自動ダッシュボードを作成します。コンテナインサイトを有効にして、CPU、メモリ、ディスク、ネットワーク、およびコンテナの再起動失敗などの診断情報をモニタリングできる

ようにする必要があります。コンテナインサイトは、クラスター、コンテナインスタンスまたはノード、サービス、タスク、ポッド、および個々のコンテナレベルですばやくフィルタリングできる動的なダッシュボードを生成します。コンテナインサイトは、 AWS サービスに応じて、クラスター、ノード、またはコンテナインスタンスレベルで構成されます。

コンテナインサイトと同様に、CloudWatch Lambda インサイトは、Lambda 関数用の動的な自動ダッシュボードを作成します。このソリューションでは、CPU 時間、メモリ、ディスク、ネットワークなどのシステムレベルのメトリクスが収集、集約、要約されます。また、コールドスタートや Lambda ワーカーシャットダウンなどの診断情報が収集、集約、要約されるため、Lambda 関数に関する問題を特定し、迅速に解決できます。Lambda は関数レベルで有効であり、エージェントを必要としません。

コンテナインサイトと Lambda インサイトは、アプリケーションまたはパフォーマンスログ、X-Ray トレース、およびサービスマップにすばやく切り替えて、コンテナのワークロードを可視化するのに役立ちます。両方とも CloudWatch の組み込みメトリクス形式を使用して CloudWatch メトリクスとパフォーマンスログをキャプチャします。

コンテナインサイトと Lambda インサイトによってキャプチャされたメトリクスを使用するワークロードの共有 CloudWatch ダッシュボードを作成できます。これを行うには、CloudWatch コンテナインサイトを使用して自動ダッシュボードをフィルタリングして表示し、ダッシュボードに追加 オプションを選択して、デフォルトの CloudWatch ダッシュボードに表示されるメトリクスを追加することができます。その後、メトリクスを削除またはカスタマイズし、ワークロードを正しく表すために他のメトリクスを追加できます。

CloudWatch と AWS サービスとの統合

AWS は、ログ記録とメトリクスの追加設定オプションを含む多くのサービスを提供します。多くの場合、これらのサービスでは、ログ出力に CloudWatch Logs を設定し、メトリクス出力の CloudWatch メトリクスを設定できます。これらのサービスの提供に使用される基盤となるインフラストラクチャは によって AWS 管理され、アクセスできませんが、プロビジョニングされたサービスのログ記録とメトリクスオプションを使用して、さらなるインサイトを得て、問題をトラブルシューティングできます。例えば、CloudWatch への VPC フローログ を発行できたり、または Amazon Relational Database Service (Amazon RDS) インスタンスを設定して、CloudWatch にログを発行することもできます。

ほとんどの AWS サービスは、 との統合 AWS CloudTrailにより API コールを記録します。CloudTrail は CloudWatch Logs との統合もサポートしているため、 AWS サービスのアクティビティを検索および分析できます。または Amazon EventBridge を使用して、 AWS サービスで実行される特定のアクションのイベントルールを使用してオートメーションと通知を作成および設定することもできます。特定のサービスは EventBridge と直接統合されます。また、CloudTrail を通じて配信されるイベントを作成することもできます。

ダッシュボードと可視化のための Amazon マネージド Grafana

Amazon Managed Grafana を使用して、 AWS ワークロードを監視および視覚化できます。Amazon Managed Grafana は、お客様のオペレーションデータを大規模に可視化して分析するのに役立ちます。Grafana は、メトリクスの保存場所を問わず、クエリ、可視化、アラート表示、把握に役立つオープンソースの分析プラットフォームです。Amazon Managed Grafana は、お客様の組織が既存のワークロードを可視化するために既に Grafana を使用しており、カバレッジを AWS ワークロードに拡大したい場合に特に役に立ちます。Amazon Managed Grafana は データソースとして追加することで CloudWatch で使用することができます。つまり、CloudWatch メトリクスを使用して可視化を実現できます。Amazon Managed Grafana は をサポートし AWS Organizations 、複数のアカウントとリージョンの CloudWatch メトリクスを使用してダッシュボードを一元化できます。

次の表に、CloudWatch ではなく Amazon Managed Grafana をダッシュボードに使用する場合の利点と注意点を示します。お客様のエンドユーザー、ワークロード、アプリケーションのさまざまな要件に基づき、ハイブリッドアプローチが適している場合があります。

Amazon Managed Grafana とオープンソース Grafana でサポートされているデータソースと統合した可視化の実現とダッシュボードの作成	Amazon Managed Grafana は、CloudWatch メトリクスなど、さまざまなデータソースから可視化を実現し、ダッシュボードを作成するのに役立ちます。Amazon Managed Grafana には、 AWS サービス、オープンソースソフトウェア、COTS ソフトウェアにまたがる多数の組み込みデータソースが含まれています。詳細については、Amazon Managed Grafana のドキュメントのビルトインデータソースを参照してください。お客様のワークスペースを Grafana エンタープライズにアップグレードして、より多くのデータソースに対するサポートを追加することもできます。Grafana は、お客様がさまざまな外部システムと通信ができるよう、データソースプラグインもサポートしています。データが CloudWatch ダッシュボードに表示されるよう、CloudWatch ダッシュボードには、CloudWatch メトリクスまた

	は CloudWatch Logs インサイトクエリが必要です。
ダッシュボードソリューションへのアクセスを AWS アカウントアクセスとは別に管理する	Amazon Managed Grafana では、認証と認可 AWS Organizations に AWS IAM Identity Center (IAM Identity Center) と を使用する必要があります。これにより、IAM Identity Center または で既に使用している ID フェデレーションを使用して、Grafana に対してユーザーを認証できます AWS Organizations。ただし、IAM Identity Center または を使用していない場合は AWS Organizations、Amazon Managed Grafana のセットアッププロセスの一部として設定されます。これは、組織が IAM Identity Center または の使用を制限している場合に問題になる可能性があります AWS Organizations。
AWS Organizations 統合により複数のアカウントとリージョンにまたがるデータの取り込みとアクセス	Amazon Managed Grafana は と統合 AWS Organizations され、すべてのアカウントで CloudWatch や Amazon OpenSearch Service などの AWS ソースからデータを読み取ることができます。これにより、アカウント全体のデータを使用して可視化されたダッシュボードを作成できます。間でデータアクセスを自動的に有効にするには AWS Organizations、AWS Organizations 管理アカウントで Amazon Managed Grafana ワークスペースを設定する必要があります。これは、管理アカウント用の AWS Organizations ベストプラクティスでは推奨されていません。対照的に、CloudWatch では、CloudWatch メトリクス用のクロスアカウント、クロスリージョンのダッシュボードがサポートされています。

オープンソースコミュニティで利用可能な、高度な可視化ウィジェットと Grafana の定義を使用する	Grafana は、お客様のダッシュボードの作成時に使用できる大規模なビジュアルのコレクションを提供します。また、コミュニティ貢献の大規模なダッシュボードのライブラリもあり、必要に応じて編集して再利用することができます。
新規および既存の Grafana デプロイでダッシュボードを使用する	Grafana を既に使用している場合は、Grafana デプロイからダッシュボードをインポートおよびエクスポートし、Amazon Managed Grafana で使用できるようにカスタマイズできます。Amazon Managed Grafana では、お客様のダッシュボードソリューションとして Grafana を標準化できます。
ワークスペース、許可、およびデータソースの高度な設定と構成	Amazon Managed Grafana では、独自の設定済みデータソース、ユーザー、ポリシーのセットを持つ複数の Grafana ワークスペースを作成できます。これにより、より高度なユースケース要件やセキュリティ構成に対応するのに役立ちます。お客様のチームが高度な機能を利用するために必要なスキルをまだ保持していない場合は、Grafana で経験を積む必要があります。

CloudWatch のよくある質問を使用したロギングとモニタリングの設計と実装

このセクションでは、CloudWatch のロギングおよびモニタリングソリューションの設計と実装について、よくある質問にお答えしています。

CloudWatch 設定ファイルはどこに保存すればよいですか?

Amazon EC2 の CloudWatch エージェントは、CloudWatch 設定ディレクトリに保存されている複数の設定ファイルを適用できます。CloudWatch の設定は、複数のアカウントや環境でバージョン管理し、再度使用することができるため、ファイルセットとして保存することが理想的です。詳細については、このガイドの CloudWatch 設定の管理 セクションを参照してください。または、設定ファイルを GitHub のリポジトリに保存しておき、新規 EC2 インスタンスがプロビジョニングされる時に設定ファイルを自動取得することもできます。

アラームが発生した時に、サービス管理ソリューションでチケットを作成するにはどうすればよいですか？

サービス管理システムを Amazon Simple Notification Service (Amazon SNS) トピックに統合し、アラームが発生したときに SNS トピックを通知するように CloudWatch アラームを設定します。統合システムは SNS メッセージを受信し、サービス管理システムの API または SDK を使用してチケットを作成できます。

CloudWatch を使用してコンテナ内のログファイルをキャプチャするにはどうすればよいですか？

Amazon ECS タスクと Amazon EKS ポッドは、STDOUT および STDERR 出力を CloudWatch に自動的に送信するように設定できます。コンテナ化されたアプリケーションをログに記録するためには、コンテナの出力をSTDOUT および STDERR に送信する方法が推奨されています。これは、Twelve-Factor アプリケーションマニフェスト もカバーしています。

ただし、特定のログファイルを CloudWatch に送信する場合は、アプリケーションがログファイルを書き込む場所に Amazon EKS ポッドまたは Amazon ECS タスク定義にボリュームをマウントし、Fluentd または Fluent Bit のサイドカーコンテナを使用して CloudWatch にログを送信できま

す。コンテナ内の特定のログファイルを /dev/stdout と /dev/stderr にシンボリックリンクすることを考慮する必要があります。詳細については、Docker ドキュメントの コンテナまたはサービスのログを表示する を参照してください。

AWS サービスのヘルス問題をモニタリングするにはどうすればよいですか?

を使用して AWS ヘルスイベントAWS Health Dashboardをモニタリングできます。また、 AWS ヘルスイベントに関連するサンプルオートメーションソリューション用の、aws-health-tool GitHub リポジトリも参照できます。

エージェントサポートが存在しない場合、カスタム CloudWatch メトリクスを作成するにはどうすればよいですか ?

埋め込みメトリクスフォーマットを使用して、メトリクスをCloudWatch に取り込むことができます。 AWS SDK (put_metric_data など)、 AWS CLI (put-metric-data など)、または AWS API (PutMetricData など) を使用してカスタムメトリクスを作成することもできます。カスタムロジックが長期的に維持される方法を考慮する必要があります。1 つのアプローチとしては、組み込みメトリクスフォーマットサポートを統合した Lambda を使用して、メトリックの期間を設定するために CloudWatch Events イベント スケジュールルール と共にメトリクスを作成することです。

既存の口グ記録およびモニタリングツールを と統合するにはどうすればよいですか AWS?

との統合については、ソフトウェアまたはサービスベンダーが提供するガイダンスを参照してください AWS。エージェントソフトウェア、SDK、または提供された API を使用して、ログとメトリクスをソリューションに送信できる場合があります。また、ベンダーの仕様に合わせて構成された Fluentd や Fluent Bit などのオープンソースソリューションを使用することもできます。また、Lambda および Kinesis Data Streams の AWS SDK および CloudWatch Logs サブスクリプションフィルターを使用して、カスタムログプロセッサとシッパーを作成することもできます。最後に、複数のアカウントとリージョンを使用している場合は、ソフトウェアをどのように統合するかについても考慮する必要があります。

リソース

序章

- AWS Well-Architected

ターゲットを絞ったビジネス成果

- ログ監視-apg-guide-例
- クラウドコンピューティングの 6 つの利点

CloudWatch デプロイを計画する

- AWS Organizations の用語と概念
- AWS Systems Manager 高速セットアップ
- CloudWatch エージェントを使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集
- cloudwatch-config-s3-bucket.yaml
- ウィザードを使用して CloudWatch エージェント設定ファイルを作成する
- エンタープライズ DevOps: ビルドしたものを実行すべき理由
- Amazon S3 へのログデータのエクスポート
- 「Amazon OpenSearch Service のきめ細かなアクセスコントロール」
- Lambda クォータ
- CloudWatch エージェント設定ファイルを手動で作成または編集する
- サブスクリプションによるログデータのリアルタイム処理
- で構築するツール AWS

EC2 インスタンスとオンプレミスサーバー用の CloudWatch エージェントの設定

- Amazon EC2 メトリクスディメンション

- バーストパフォーマンスインスタンス
- CloudWatch エージェント事前定義されたメトリクスセット
- procstat プラグインを使用してプロセスメトリクスを収集する
- procstat 用の CloudWatch エージェントの設定
- EC2 インスタンスの詳細なモニタリングを管理する
- 高カーディナリティログの取り込みと CloudWatch 埋め込みメトリクス形式によるメトリクスの生成
- ロググループとログストリームの使用
- インスタンスで使用可能な CloudWatch メトリクスを一覧表示する
- PutLogEvents
- collectd を使用してカスタムメトリクスを取得する
- StatsD を使用してカスタムメトリクスを取得する

Amazon EC2 およびオンプレミスサーバーに対する CloudWatch エージェントのインストールアプローチ

- ハイブリッドおよびマルチクラウド環境で Systems Manager に必要な IAM サービスロールを作成する
- ハイブリッド環境のマネージドインスタンスアクティベーションを作成する
- CloudWatch エージェントで使用する IAM ロールとユーザーを作成する
- コマンドラインを使用して CloudWatch エージェントをダウンロードして設定する
- Systems Manager エージェントと統合 CloudWatch エージェントを使用するオンプレミスサーバーを、一時的な認証情報のみを使用するように構成するにはどうすればよいですか。
- スタックセットオペレーションの前提条件
- スポットインスタンスの使用

Amazon ECS での口グ記録とモニタリング

- Amazon-cloudwatch-logs-for-fluent-bit
- Amazon ECS CloudWatch メトリクス

- 「Amazon ECS Container Insights メトリクス」
- Amazon ECS コンテナエージェント
- Amazon ECS 起動タイプ
- CloudWatch エージェントをデプロイして Amazon ECS で EC2 インスタンスレベルのメトリクスを収集する
- ecs_cluster_with_cloudwatch_linux.yaml
- ecs_cw_emf_example
- ecs_firelense_emf_example
- ecs-task-nginx-firelense.json
- Amazon ECS に最適化された AMI メタデータを取得する
- awslogs ログドライバーの使用
- クライアントライブラリを使用して埋め込みメトリクス形式のログを生成する

Amazon EKS でのログ記録とモニタリング

- Amazon EKS コントロールプレーンのログ記録
- amazon_eks_managed_node_group_launch_config.yaml
- Amazon EKS ノード
- amazon-eks-nodegroup.yaml
- Amazon EKS サービスレベルアグリーメント
- コンテナインサイトの Prometheus メトリクスのモニタリング
- Prometheus を使用したコントロールプレーンメトリクス
- Fargate ログ記録
- Fargate で Amazon EKS の流暢なビット
- Fargate で Amazon EKS を使用するときにアプリケーションログをキャプチャする方法
- CloudWatch エージェントをインストールして Prometheus メトリクスを収集する
- Kubernetes メトリクスサーバーのインストール
- Kubernetes /ダッシュボード
- Kubernetes 水平ポッドオートスケーラ
- Kubernetes コントロールプレーンのコンポーネント

- Kubernetes ポッド仕様
- 起動テンプレートのサポート
- マネージドノードグループ
- マネージド型ノードの更新動作
- メトリクスサーバー
- Prometheus とGrafanaを使ってFargateでAmazon EKSをモニタリングする
- prometheus_jmx
- Prometheus /JMX_Exporter
- 追加の Prometheus ソースのスクレイピングとそれらのメトリクスのインポート
- セルフマネージド型ノード
- CloudWatch Logs にログを送信する
- CloudWatch Logs へログを送信する DaemonSet として Fluent Bit を設定する
- Amazon EKS と Kubernetes で Java/JMX サンプルワークロードを設定する
- 新しい Prometheus スクレイプターゲットを追加するためのチュートリアル: Prometheus API Server メトリクス
- 垂直ポッドオートスケーラー

のログ記録とメトリクス AWS Lambda

- Lambda 呼び出しエラー
- ロギング — Python のロギング機能
- クライアントライブラリを使用して埋め込みメトリクス形式のログを生成する
- Lambda 関数のメトリクスの使用

CloudWatch でのログの検索および分析

- Beats ファミリー
- Elastic Logstash
- 弾性スタック
- CloudWatch Logs データを Amazon OpenSearch Service にストリーミングする

CloudWatch によるアラームのオプション

- amazon-cloudwatch-auto-alarms
- AWS Jira Service Management Cloud 用の Service Management Connector
- AWS Jira サービス管理データセンター用のサービス管理コネクタ
- AWS Service Management Connector for ServiceNow

アプリケーションとサービスの可用性のモニタリング

- DNS フェイルオーバーの設定

を使用したアプリケーションのトレース AWS X-Ray

- Amazon ECS タスクネットワーキング
- X-Ray コンソールでのサンプリングルールの設定
- Windows PowerShell コマンドまたはスクリプトを実行する
- Amazon EC2 での X-Ray デーモンの実行
- トレースデータを X-Ray に送信する
- X-Ray でのサービスグラフ

CloudWatch を使用したダッシュボードとビジュアライゼーション

- Amazon CloudWatch Metric Math により、Amazon EFS ファイルシステムのほぼリアルタイムモニタリングが簡素化されます
- CloudWatch コンテナインサイトの設定
- メトリクス数学の使用

CloudWatch と AWS サービスとの統合

- AWS CloudTrail がサポートするサービスと統合
- Amazon EventBridge AWS のサービス の からのイベント
- 経由で配信される AWS サービスイベント AWS CloudTrail

- Amazon CloudWatch Logs による CloudTrail ログファイルをモニタリングする
- CloudWatch Logs へのデータベースログの発行
- CloudWatch Logs へのフローログの発行

ダッシュボードと可視化のための Amazon マネージド Grafana

- の管理アカウントのベストプラクティス AWS Organizations
- Amazon マネージド Grafana 用の組み込みのデータソース
- CloudWatch でのクロスアカウントダッシュボードとクロスリージョンダッシュボード
- Grafana プラグイン

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィード をサブスクライブできます。

変更	説明	日付
ログ情報の更新	のログ記録に関するセクションを更新しました AWS Lambda。	2023 年 4 月 17 日
更新された設定情報	CloudWatch 設定の作成と保存に関するセクションを更新し、名前を変更しました。	2023 年 2 月 9 日
メトリクス情報の更新	Amazon ECS のメトリクスセクションのカスタムアプリケーションメトリクス情報を更新しました。	2023 年 1 月 31 日
プレビュー通知を削除しました	Amazon Managed Grafana は一般利用可能です。	2022 年 5 月 25 日
削除済みのセクション	CloudWatch SDK メトリクスのサポートは終了しました。	2022 年 1 月 7 日
初版発行	—	2021 年 4 月 30 日

AWS 規範ガイダンスの用語集

以下は、 AWS 規範ガイダンスによって提供される戦略、ガイド、パターンで一般的に使用される用語です。エントリを提案するには、用語集の最後のフィードバックの提供リンクを使用します。

数字

7 Rs

アプリケーションをクラウドに移行するための 7 つの一般的な移行戦略。これらの戦略は、ガートナーが 2011 年に特定した 5 Rs に基づいて構築され、以下で構成されています。

- リファクタリング/アーキテクチャの再設計 — クラウドネイティブ特徴を最大限に活用して、俊敏性、パフォーマンス、スケーラビリティを向上させ、アプリケーションを移動させ、アーキテクチャを変更します。これには、通常、オペレーティングシステムとデータベースの移植が含まれます。例: オンプレミスの Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行します。
- リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: オンプレミスの Oracle データベースを の Oracle 用 Amazon Relational Database Service (Amazon RDS) に移行します AWS クラウド。
- 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: カスタマーリレーションシップ管理 (CRM) システムを Salesforce.com に移行します。
- リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: オンプレミスの Oracle データベースを の EC2 インスタンス上の Oracle に移行します AWS クラウド。
- 再配置 (ハイパーバイザーレベルのリフトアンドシフト) – 新しいハードウェアを購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例: Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。

- 使用停止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

ABAC

「属性ベースのアクセスコントロール」を参照してください。

抽象化されたサービス

「 マネージドサービス」を参照してください。

ACID

アトミック性、一貫性、分離性、耐久性を参照してください。

アクティブ - アクティブ移行

(双方向レプリケーションツールまたは二重書き込み操作を使用して) ソースデータベースとターゲットデータベースを同期させ、移行中に両方のデータベースが接続アプリケーションからのトランザクションを処理するデータベース移行方法。この方法では、1 回限りのカットオーバーの必要がなく、管理された小規模なバッチで移行できます。より柔軟ですが、アクティブ/パッシブ移行よりも多くの作業が必要です。

アクティブ - パッシブ移行

ソースデータベースとターゲットデータベースが同期されるデータベース移行方法。ただし、データがターゲットデータベースにレプリケートされている間、ソースデータベースのみが接続アプリケーションからのトランザクションを処理します。移行中、ターゲットデータベースはトランザクションを受け付けません。

集計関数

行のグループで動作し、グループの単一の戻り値を計算する SQL 関数。集計関数の例としては、 `SUM`や などがありま`MAX`す。

AI

「人工知能」を参照してください。

AIOps

「人工知能オペレーション」を参照してください。

匿名化

データセット内の個人情報を完全に削除するプロセス。匿名化は個人のプライバシー保護に役立ちます。匿名化されたデータは、もはや個人データとは見なされません。

アンチパターン

繰り返し起こる問題に対して頻繁に用いられる解決策で、その解決策が逆効果であったり、効果がなかったり、代替案よりも効果が低かったりするもの。

アプリケーションコントロール

マルウェアからシステムを保護するために、承認されたアプリケーションのみを使用できるようにするセキュリティアプローチ。

アプリケーションポートフォリオ

アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の需要要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。

人工知能 (AI)

コンピューティングテクノロジーを使用し、学習、問題の解決、パターンの認識など、通常は人間に関連づけられる認知機能の実行に特化したコンピュータサイエンスの分野。詳細については、「人工知能 (AI) とは何ですか?」を参照してください。

AI オペレーション (AIOps)

機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。 AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。

非対称暗号化

暗号化用のパブリックキーと復号用のプライベートキーから成る 1 組のキーを使用した、暗号化のアルゴリズム。パブリックキーは復号には使用されないため共有しても問題ありませんが、プライベートキーの利用は厳しく制限する必要があります。

原子性、一貫性、分離性、耐久性 (ACID)

エラー、停電、その他の問題が発生した場合でも、データベースのデータ有効性と運用上の信頼性を保証する一連のソフトウェアプロパティ。

属性ベースのアクセス制御 (ABAC)

部署、役職、チーム名など、ユーザーの属性に基づいてアクセス許可をきめ細かく設定する方法。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「 の ABAC AWS」を参照してください。

信頼できるデータソース

最も信頼性のある情報源とされるデータのプライマリーバージョンを保存する場所。匿名化、編集、仮名化など、データを処理または変更する目的で、信頼できるデータソースから他の場所にデータをコピーすることができます。

アベイラビリティーゾーン

他のアベイラビリティーゾーンの障害から AWS リージョン 隔離され、同じリージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供する 内の別の場所。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 のガイドラインとベストプラクティスのフレームワークです。 AWS CAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用という 6 つの重点分野にガイダンスを整理しています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションのためのガイダンスを提供します。詳細については、AWS CAF ウェブサイト と AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

データベース移行ワークロードを評価し、移行戦略を推奨し、作業見積もりを提供するツール。AWS WQF は AWS Schema Conversion Tool (AWS SCT) に含まれています。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。

B

不正なボット

個人や組織を混乱させたり、損害を与えたりすることを意図したボット。

BCP

「事業継続計画」を参照してください。

動作グラフ

リソースの動作とインタラクションを経時的に示した、一元的なインタラクティブビュー。Amazon Detective の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントのData in a behavior graphを参照してください。

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。エンディアン性も参照してください。

二項分類

バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。

ブルームフィルター

要素がセットのメンバーであるかどうかをテストするために使用される、確率的でメモリ効率の高いデータ構造。

ブルー/グリーンデプロイ

2 つの異なる同一の環境を作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (青) で実行し、新しいアプリケーションバージョンを別の環境 (緑) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティややり取りをシミュレートするソフトウェアアプリケーション。インターネット上の情報のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボットと呼ばれる他のボットの中には、個人や組織を混乱させたり、損害を与えたりすることを意図したものもあります。

ボットネット

マルウェアに感染し、ボットハーダーまたはボットオペレーターとして知られる単一の当事者によって制御されているボットのネットワーク。ボットは、ボットとその影響をスケールするための最もよく知られているメカニズムです。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発したり、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

例外的な状況では、承認されたプロセスを通じて、ユーザーが AWS アカウント 通常アクセス許可を持たない にすばやくアクセスできるようにします。詳細については、 Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータ AWS を参照してください。

ブラウンフィールド戦略

環境の既存インフラストラクチャ。システムアーキテクチャにブラウンフィールド戦略を導入する場合、現在のシステムとインフラストラクチャの制約に基づいてアーキテクチャを設計します。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど)。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、ホワイトペーパー AWSでのコンテナ化されたマイクロサービスの実行 の ビジネス機能を中心に組織化 セクションを参照してください。

ビジネス継続性計画 (BCP)

大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。

C

CAF

AWS 「クラウド導入フレームワーク」を参照してください。

Canary デプロイ

エンドユーザーへのバージョンのスローリリースと増分リリース。確信できたら、新しいバージョンをデプロイし、現在のバージョン全体を置き換えます。

CCoE

「Cloud Center of Excellence」を参照してください。

CDC

「データキャプチャの変更」を参照してください。

変更データキャプチャ (CDC)

データソース (データベーステーブルなど) の変更を追跡し、その変更に関するメタデータを記録するプロセス。CDC は、ターゲットシステムでの変更を監査またはレプリケートして同期を維持するなど、さまざまな目的に使用できます。

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストします。AWS Fault Injection Service (AWS FIS) を使用して、 AWS ワークロードにストレスを与え、その応答を評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。

クライアント側の暗号化

ターゲットがデータ AWS のサービス を受信する前のローカルでのデータの暗号化。

Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、 AWS クラウド エンタープライズ戦略ブログの CCoE 投稿を参照してください。

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的にエッジコンピューティングテクノロジーに接続されています。

クラウド運用モデル

IT 組織において、1 つ以上のクラウド環境を構築、成熟、最適化するために使用される運用モデル。詳細については、「クラウド運用モデルの構築」 を参照してください。

導入のクラウドステージ

組織が に移行するときに通常実行する 4 つのフェーズ AWS クラウド:

- プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。AWS 移行戦略とどのように関連しているかについては、「移行準備ガイド」を参照してください。

CMDB

「設定管理データベース」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、 GitHubまたは が含まれますBitbucket Cloud。コードの各バージョンはブランチと呼ばれます。マイクロサービス

の構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

空である、または、かなり空きがある、もしくは、古いデータや無関係なデータが含まれているバッファキャッシュ。データベースインスタンスはメインメモリまたはディスクから読み取る必要があり、バッファキャッシュから読み取るよりも時間がかかるため、パフォーマンスに影響します。

コールドデータ

めったにアクセスされず、通常は過去のデータです。この種類のデータをクエリする場合、通常は低速なクエリでも問題ありません。このデータを低パフォーマンスで安価なストレージ階層またはクラスに移動すると、コストを削減することができます。

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオなどのビジュアル形式から情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI は CV 用の画像処理アルゴリズムを提供します。

設定ドリフト

ワークロードの場合、設定が想定状態から変化します。これにより、ワークロードが非準拠になる可能性があり、通常は段階的かつ意図的ではありません。

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンの単一のエンティティとしてデプロイすることも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性

の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

ネットワーク内のデータを重要度と機密性に基づいて識別、分類するプロセス。データに適した保護および保持のコントロールを判断する際に役立つため、あらゆるサイバーセキュリティのリスク管理戦略において重要な要素です。データ分類は、 AWS Well-Architected フレームワークのセキュリティの柱のコンポーネントです。詳細については、データ分類を参照してください。

データドリフト

実稼働データと ML モデルのトレーニングに使用されたデータとの間に有意な差異が生じたり、入力データが時間の経過と共に有意に変化したりすることです。データドリフトは、ML モデル予測の全体的な品質、精度、公平性を低下させる可能性があります。

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

一元管理とガバナンスを備えた分散型の分散型データ所有権を提供するアーキテクチャフレームワーク。

データ最小化

厳密に必要なデータのみを収集し、処理するという原則。でデータ最小化を実践 AWS クラウドすることで、プライバシーリスク、コスト、分析のカーボンフットプリントを削減できます。

データ境界

AWS 環境内の一連の予防ガードレール。信頼された ID のみが、期待されるネットワークから信頼されたリソースにアクセスできるようにします。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

データの前処理

raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。

データ出所

データの生成、送信、保存の方法など、データのライフサイクル全体を通じてデータの出所と履歴を追跡するプロセス。

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには通常、大量の履歴データが含まれており、通常はクエリや分析に使用されます。

データベース定義言語 (DDL)

データベース内のテーブルやオブジェクトの構造を作成または変更するためのステートメントまたはコマンド。

データベース操作言語 (DML)

データベース内の情報を変更 (挿入、更新、削除) するためのステートメントまたはコマンド。

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせる。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

ディープラーニング

人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を採用するときは AWS、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加して、リソースの安全性を確保します。たとえば、多層防御アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS Organizations ドキュメントのAWS Organizationsで使用できるサービスを参照してください。

デプロイ

アプリケーション、新機能、コードの修正をターゲットの環境で利用できるようにするプロセス。デプロイでは、コードベースに変更を施した後、アプリケーションの環境でそのコードベースを構築して実行します。

開発環境

「環境」を参照してください。

検出管理

イベントが発生したときに、検出、ログ記録、警告を行うように設計されたセキュリティコントロール。これらのコントロールは副次的な防衛手段であり、実行中の予防的コントロールをすり抜けたセキュリティイベントをユーザーに警告します。詳細については、Implementing security controls on AWSのDetective controlsを参照してください。

開発バリューストリームマッピング (DVSM)

ソフトウェア開発ライフサイクルのスピードと品質に悪影響を及ぼす制約を特定し、優先順位を付けるために使用されるプロセス。DVSM は、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

建物、工場、産業機器、生産ラインなど、現実世界のシステムを仮想的に表現したものです。デジタルツインは、予知保全、リモートモニタリング、生産最適化をサポートします。

ディメンションテーブル

スタースキーマでは、ファクトテーブル内の量的データに関するデータ属性を含む小さなテーブル。ディメンションテーブル属性は通常、テキストフィールドまたはテキストのように動作する離散数値です。これらの属性は、クエリの制約、フィルタリング、結果セットのラベル付けに一般的に使用されます。

ディザスタ

ワークロードまたはシステムが、導入されている主要な場所でのビジネス目標の達成を妨げるイベント。これらのイベントは、自然災害、技術的障害、または意図しない設定ミスやマルウェア攻撃などの人間の行動の結果である場合があります。

ディザスタリカバリ (DR)

災害によるダウンタイムとデータ損失を最小限に抑えるために使用する戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用する方法の詳細については、コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件への準拠に影響するランディングゾーンの変更を検出したりできます。

DVSM

「開発値ストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

IoT ネットワークのエッジにあるスマートデバイスの計算能力を高めるテクノロジー。クラウドコンピューティングと比較すると、エッジコンピューティングは通信レイテンシーを短縮し、応答時間を短縮できます。

電子データ交換 (EDI)

組織間のビジネスドキュメントの自動交換。詳細については、「電子データ交換とは」を参照してください。

暗号化

人間が読み取り可能なプレーンテキストデータを暗号文に変換するコンピューティングプロセス。

暗号化キー

暗号化アルゴリズムが生成した、ランダム化されたビットからなる暗号文字列。キーの長さは決まっておらず、各キーは予測できないように、一意になるように設計されています。

エンディアン

コンピュータメモリにバイトが格納される順序。ビッグエンディアンシステムでは、最上位バイトが最初に格納されます。リトルエンディアンシステムでは、最下位バイトが最初に格納されます。

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS

Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (会計、MES、プロジェクト管理など) を自動化および管理するシステム。

エンベロープ暗号化

暗号化キーを、別の暗号化キーを使用して暗号化するプロセス。詳細については、 AWS Key Management Service (AWS KMS) ドキュメントの「エンベロープ暗号化」を参照してください。

環境

実行中のアプリケーションのインスタンス。クラウドコンピューティングにおける一般的な環境の種類は以下のとおりです。

- 開発環境 — アプリケーションのメンテナンスを担当するコアチームのみが使用できる、実行中のアプリケーションのインスタンス。開発環境は、上位の環境に昇格させる変更をテストするときに使用します。このタイプの環境は、テスト環境と呼ばれることもあります。
- 下位環境 — 初期ビルドやテストに使用される環境など、アプリケーションのすべての開発環境。
- 本番環境 — エンドユーザーがアクセスできる、実行中のアプリケーションのインスタンス。CI/CD パイプラインでは、本番環境が最後のデプロイ環境になります。
- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

エピック

アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。例えば、 AWS CAF セキュリティエピックには、ID とアクセスの管理、検出コントロール、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。 AWS 移行戦略のエピックの詳細については、プログラム実装ガイド を参照してください。

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。

F

ファクトテーブル

星スキーマの中央テーブル。事業運営に関する量的データを保存します。通常、ファクトテーブルには、メジャーを含む列とディメンションテーブルへの外部キーを含む列の 2 つのタイプの列が含まれます。

フェイルファスト

開発ライフサイクルを短縮するために頻繁で段階的なテストを使用する哲学。これはアジャイルアプローチの重要な部分です。

障害分離の境界

では AWS クラウド、障害の影響を制限し、ワークロードの耐障害性を高めるのに役立つアベイラビリティーゾーン AWS リージョン、コントロールプレーン、データプレーンなどの境界。詳細については、AWS 「障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

特徴量

お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。

特徴量重要度

モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

機能変換

追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械

学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。

数ショットプロンプト

同様のタスクの実行を求める前に、タスクと必要な出力を示す少数の例を LLM に提供します。この手法は、プロンプトに埋め込まれた例 (ショット) からモデルが学習するコンテキスト内学習のアプリケーションです。少数ショットプロンプトは、特定のフォーマット、推論、またはドメインの知識を必要とするタスクに効果的です。「ゼロショットプロンプト」も参照してください。

FGAC

「きめ細かなアクセスコントロール」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

段階的なアプローチを使用する代わりに、変更データキャプチャによる継続的なデータレプリケーションを使用して、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

一般化およびラベル付けされていないデータの大規模なデータセットでトレーニングされている大規模な深層学習ニューラルネットワーク。FMs は、言語の理解、テキストと画像の生成、自然言語の会話など、さまざまな一般的なタスクを実行できます。詳細については、「基盤モデルとは」を参照してください。

G

生成 AI

大量のデータでトレーニングされ、シンプルなテキストプロンプトを使用してイメージ、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できる AI モデルのサブセット。詳細については、「生成 AI とは」を参照してください。

ジオブロッキング

地理的制限を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リストを使って指定します。詳細については、CloudFront ドキュメントのコンテンツの地理的ディストリビューションの制限を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローはレガシーと見なされ、トランクベースのワークフローはモダンで推奨されるアプローチです。

ゴールデンイメージ

そのシステムまたはソフトウェアの新しいインスタンスをデプロイするためのテンプレートとして使用されるシステムまたはソフトウェアのスナップショット。例えば、製造では、ゴールデンイメージを使用して複数のデバイスにソフトウェアをプロビジョニングし、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

新しい環境に既存のインフラストラクチャが存在しないこと。システムアーキテクチャにグリーンフィールド戦略を導入する場合、既存のインフラストラクチャ (別名ブラウンフィールド) との互換性の制約を受けることなく、あらゆる新しいテクノロジーを選択できます。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

ガードレール

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは AWS Config、、Amazon GuardDuty AWS Security Hub CSPM、、AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

H

HA

「高可用性」を参照してください。

異種混在データベースの移行

別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCTを提供します。

ハイアベイラビリティ (HA)

課題や災害が発生した場合に、介入なしにワークロードを継続的に運用できること。HA システムは、自動的にフェイルオーバーし、一貫して高品質のパフォーマンスを提供し、パフォーマンスへの影響を最小限に抑えながらさまざまな負荷や障害を処理するように設計されています。

ヒストリアンのモダナイゼーション

製造業のニーズによりよく応えるために、オペレーションテクノロジー (OT) システムをモダナイズし、アップグレードするためのアプローチ。ヒストリアンは、工場内のさまざまなソースからデータを収集して保存するために使用されるデータベースの一種です。

ホールドアウトデータ

機械学習モデルのトレーニングに使用されるデータセットから保留される、ラベル付きの履歴データの一部。モデル予測をホールドアウトデータと比較することで、ホールドアウトデータを使用してモデルのパフォーマンスを評価できます。

同種データベースの移行

お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。

ホットデータ

リアルタイムデータや最近の翻訳データなど、頻繁にアクセスされるデータ。通常、このデータには高速なクエリ応答を提供する高性能なストレージ階層またはクラスが必要です。

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性が高いため、通常の DevOps のリリースワークフローからは外れた形で実施されます。

ハイパーケア期間

カットオーバー直後、移行したアプリケーションを移行チームがクラウドで管理、監視して問題に対処する期間。通常、この期間は 1～4 日です。ハイパーケア期間が終了すると、アプリケーションに対する責任は一般的に移行チームからクラウドオペレーションチームに移ります。

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

AWS クラウド 環境内のアクセス許可を定義する 1 つ以上の IAM プリンシパルにアタッチされたポリシー。

アイドル状態のアプリケーション

90 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。

IIoT

「産業用モノのインターネット」を参照してください。

イミュータブルインフラストラクチャ

既存のインフラストラクチャを更新、パッチ適用、または変更する代わりに、本番環境のワークロード用に新しいインフラストラクチャをデプロイするモデル。イミュータブルインフラストラクチャは、本質的にミュータブルインフラストラクチャよりも一貫性、信頼性、予測性が高くなります。詳細については、 AWS 「 Well-Architected フレームワーク」の「イミュータブルインフラストラクチャを使用したデプロイ」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリ

ケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

アプリケーションを 1 回ですべてカットオーバーするのではなく、小さい要素に分けて移行するカットオーバー戦略。例えば、最初は少数のマイクロサービスまたはユーザーのみを新しいシステムに移行する場合があります。すべてが正常に機能することを確認できたら、残りのマイクロサービスやユーザーを段階的に移行し、レガシーシステムを廃止できるようにします。この戦略により、大規模な移行に伴うリスクが軽減されます。

インダストリー 4.0

2016 年に Klaus Schwab によって導入された用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩によるビジネスプロセスのモダナイゼーションを指します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

アプリケーションのインフラストラクチャを一連の設定ファイルを使用してプロビジョニングし、管理するプロセス。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。

産業分野における IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「Building an industrial Internet of Things (IIoT) digital transformation strategy」を参照してください。

インスペクション VPC

AWS マルチアカウントアーキテクチャでは、VPC (同一または異なる 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

インターネットまたはローカル通信ネットワークを介して他のデバイスやシステムと通信する、センサーまたはプロセッサが組み込まれた接続済み物理オブジェクトのネットワーク。詳細については、「IoT とは」を参照してください。

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「モノのインターネット」を参照してください。

IT 情報ライブラリ (ITIL)

IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。

IT サービス管理 (ITSM)

組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、オペレーション統合ガイド を参照してください。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

強制アクセス制御 (MAC) の実装で、ユーザーとデータ自体にそれぞれセキュリティラベル値が明示的に割り当てられます。ユーザーセキュリティラベルとデータセキュリティラベルが交差する部分によって、ユーザーに表示される行と列が決まります。

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロー

ドとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、安全でスケーラブルなマルチアカウント AWS 環境のセットアップ を参照してください。

大規模言語モデル (LLM)

大量のデータに対して事前トレーニングされた深層学習 AI モデル。LLM は、質問への回答、ドキュメントの要約、テキストの他の言語への翻訳、文の完了など、複数のタスクを実行できます。詳細については、LLMs」を参照してください。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースのアクセスコントロール」を参照してください。

最小特権

タスクの実行には必要最低限の権限を付与するという、セキュリティのベストプラクティス。詳細については、IAM ドキュメントの最小特権アクセス許可を適用するを参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。エンディアン性も参照してください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

パターン認識と学習にアルゴリズムと手法を使用する人工知能の一種。ML は、モノのインターネット (IoT) データなどの記録されたデータを分析して学習し、パターンに基づく統計モデルを生成します。詳細については、「機械学習」を参照してください。

メインブランチ

「ブランチ」を参照してください。

マルウェア

コンピュータのセキュリティまたはプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスにつながる可能性があります。マルウェアの例としては、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス はインフラストラクチャレイヤー、オペレーティングシステム、プラットフォーム AWS を運用し、エンドポイントにアクセスしてデータを保存および取得します。Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB は、マネージドサービスの例です。これらは抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するためのソフトウェアシステムで、原材料を工場の完成製品に変換します。

MAP

「移行促進プログラム」を参照してください。

メカニズム

ツールを作成し、ツールの導入を推進し、調整を行うために結果を検査する完全なプロセス。メカニズムは、動作中にそれ自体を強化して改善するサイクルです。詳細については、 AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

組織の一部である管理アカウント AWS アカウント 以外のすべて AWS Organizations。 アカウントが 組織のメンバーになることができるのは、一度に 1 つのみです。

MES

「製造実行システム」を参照してください。

メッセージキューイングテレメトリトランスポート (MQTT)

リソースに制約のある IoT デバイス用の、パブリッシュ/サブスクライブパターンに基づく軽量 machine-to-machine (M2M) 通信プロトコル。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

組織がクラウドに移行するための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー、およびスプリントで作業する DevOps プロフェッショナルが含まれます。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

移行メタデータ

移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例としては、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

移行パターン

移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

に移行するためのビジネスケースを検証するための情報を提供するオンラインツール AWS クラウド。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイド を参照してください。MRA は、AWS 移行戦略の第一段階です。

移行戦略

ワークロードを に移行するために使用するアプローチ AWS クラウド。詳細については、この用語集の「7 Rs エントリ」と「組織を動員して大規模な移行を加速する」を参照してください。

ML

???「機械学習」を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、「」の「アプリケーションをモダナイズするための戦略 AWS クラウド」を参照してください。

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定されたギャップに対処するためのアクションプランが得られます。詳細については、『』の「アプリケーションのモダナイゼーション準備状況の評価 AWS クラウド」を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、モノリスをマイクロサービスに分解する を参照してください。

MPA

「移行ポートフォリオ評価」を参照してください。

MQTT

「Message Queuing Telemetry Transport」を参照してください。

多クラス分類

複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」 または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。

ミュータブルインフラストラクチャ

本番ワークロードの既存のインフラストラクチャを更新および変更するモデル。Well-Architected AWS フレームワークでは、一貫性、信頼性、予測可能性を向上させるために、イミュータブルインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセスコントロール」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

移行プロセス中にソースワークロードを停止させる移行方法。この方法はダウンタイムが長くなるため、通常は重要ではない小規模なワークロードに使用されます。

OI

「 オペレーションの統合」を参照してください。

OLA

「運用レベルの契約」を参照してください。

オンライン移行

ソースワークロードをオフラインにせずにターゲットシステムにコピーする移行方法。ワークロードに接続されているアプリケーションは、移行中も動作し続けることができます。この方法はダウンタイムがゼロから最小限で済むため、通常は重要な本番稼働環境のワークロードに使用されます。

OPC-UA

「Open Process Communications - Unified Architecture」を参照してください。

オープンプロセス通信 - 統合アーキテクチャ (OPC-UA)

産業用オートメーション用のmachine-to-machine (M2M) 通信プロトコル。OPC-UA は、データの暗号化、認証、認可スキームを備えた相互運用性標準を提供します。

オペレーショナルレベルアグリーメント (OLA)

サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。

運用準備状況レビュー (ORR)

インシデントや潜在的な障害の理解、評価、防止、または範囲の縮小に役立つ質問とそれに関連するベストプラクティスのチェックリスト。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携するハードウェアおよびソフトウェアシステム。製造では、OT と情報技術 (IT) システムの統合が、Industry 4.0 変換の主な焦点です。

オペレーション統合 (OI)

クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイド を参照してください。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail をログに記録する、 によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの組織の証跡の作成を参照してください。

組織変更管理 (OCM)

人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイド を参照してください。

オリジンアクセスコントロール (OAC)

Amazon Simple Storage Service (Amazon S3) コンテンツを保護するための、CloudFront のアクセス制限の強化オプション。OAC は AWS リージョン、すべての S3 バケット、 AWS KMS (SSE-KMS) によるサーバー側の暗号化、S3 バケットへの動的 PUT および DELETEリクエストをサポートします。

オリジンアクセスアイデンティティ (OAI)

CloudFront の、Amazon S3 コンテンツを保護するためのアクセス制限オプション。OAI を使用すると、CloudFront が、Amazon S3 に認証可能なプリンシパルを作成します。認証されたプリンシパルは、S3 バケット内のコンテンツに、特定の CloudFront ディストリビューションを介してのみアクセスできます。OACも併せて参照してください。OAC では、より詳細な、強化されたアクセスコントロールが可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション内から開始されたネットワーク接続を処理する VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

P

アクセス許可の境界

ユーザーまたはロールが使用できるアクセス許可の上限を設定する、IAM プリンシパルにアタッチされる IAM 管理ポリシー。詳細については、IAM ドキュメントのアクセス許可の境界を参照してください。

個人を特定できる情報 (PII)

直接閲覧した場合、または他の関連データと組み合わせた場合に、個人の身元を合理的に推測するために使用できる情報。PII の例には、氏名、住所、連絡先情報などがあります。

PII

個人を特定できる情報を参照してください。

プレイブック

クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。

PLC

「プログラム可能なロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

アクセス許可を定義 (アイデンティティベースのポリシーを参照)、アクセス条件を指定 (リソースベースのポリシーを参照)、または の組織内のすべてのアカウントに対する最大アクセス許可を定義 AWS Organizations (サービスコントロールポリシーを参照) できるオブジェクト。

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。詳細については、マイクロサービスでのデータ永続性の有効化 を参照してください。

ポートフォリオ評価

移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行準備状況ガイド」を参照してください。

述語

`true` または を返すクエリ条件。一般的に`false`は `WHERE`句にあります。

述語プッシュダウン

転送前にクエリ内のデータをフィルタリングするデータベースクエリ最適化手法。これにより、リレーショナルデータベースから取得して処理する必要があるデータの量が減少し、クエリのパフォーマンスが向上します。

予防的コントロール

イベントの発生を防ぐように設計されたセキュリティコントロール。このコントロールは、ネットワークへの不正アクセスや好ましくない変更を防ぐ最前線の防御です。詳細については、Implementing security controls on AWSのPreventative controlsを参照してください。

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM AWS アカウントロール、または ユーザーのルートユーザーです。詳細については、IAM ドキュメントのロールに関する用語と概念内にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通じてプライバシーを考慮するシステムエンジニアリングアプローチ。

プライベートホストゾーン

1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナ。詳細については、Route 53 ドキュメントの「プライベートホストゾーンの使用」を参照してください。

プロアクティブコントロール

非準拠リソースのデプロイを防ぐように設計されたセキュリティコントロール。これらのコントロールは、プロビジョニング前にリソースをスキャンします。リソースがコントロールに準拠していない場合、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「 コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

設計、開発、発売から成長と成熟、拒否と削除まで、ライフサイクル全体にわたる製品のデータとプロセスの管理。

本番環境

「環境」を参照してください。

プログラム可能なロジックコントローラー (PLC)

製造では、マシンをモニタリングし、製造プロセスを自動化する、信頼性の高い適応可能なコンピュータです。

プロンプトの連鎖

1 つの LLM プロンプトの出力を次のプロンプトの入力として使用して、より良いレスポンスを生成します。この手法は、複雑なタスクをサブタスクに分割したり、事前レスポンスを繰り返し改善または拡張したりするために使用されます。これにより、モデルのレスポンスの精度と関連性が向上し、より詳細でパーソナライズされた結果が得られます。

仮名化

データセット内の個人識別子をプレースホルダー値に置き換えるプロセス。仮名化は個人のプライバシー保護に役立ちます。仮名化されたデータは、依然として個人データとみなされます。

パブリッシュ/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。スケーラビリティと応答性を向上させます。たとえば、マイクロサービスベースの MES では、マイクロサービスは他のマイクロサー

ビスがサブスクライブできるチャネルにイベントメッセージを発行できます。システムは、公開サービスを変更せずに新しいマイクロサービスを追加できます。

Q

クエリプラン

SQL リレーショナルデータベースシステムのデータにアクセスするために使用される手順などの一連のステップ。

クエリプランのリグレッション

データベースサービスのオプティマイザーが、データベース環境に特定の変更が加えられる前に選択されたプランよりも最適性の低いプランを選択すること。これは、統計、制限事項、環境設定、クエリパラメータのバインディングの変更、およびデータベースエンジンの更新などが原因である可能性があります。

R

RACI マトリックス

責任、説明責任、相談、情報 (RACI) を参照してください。

RAG

「取得拡張生成」を参照してください。

ランサムウェア

決済が完了するまでコンピュータシステムまたはデータへのアクセスをブロックするように設計された、悪意のあるソフトウェア。

RASCI マトリックス

責任、説明責任、相談、情報 (RACI) を参照してください。

RCAC

「行と列のアクセスコントロール」を参照してください。

リードレプリカ

読み取り専用に使用されるデータベースのコピー。クエリをリードレプリカにルーティングして、プライマリデータベースへの負荷を軽減できます。

再設計

「7 Rs」を参照してください。

目標復旧時点 (RPO)

最後のデータリカバリポイントからの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の とは独立しています。詳細については、「アカウントで使用できる を指定する AWS リージョン」を参照してください。

回帰

数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

プラットフォーム変更

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または回復するアプリケーションの機能。高可用性とディザスタリカバリは、 で回復性を計画する際の一般的な考慮事項です AWS クラウド。詳細については、AWS クラウド「レジリエンス」を参照してください。

リソースベースのポリシー

Amazon S3 バケット、エンドポイント、暗号化キーなどのリソースにアタッチされたポリシー。このタイプのポリシーは、アクセスが許可されているプリンシパル、サポートされているアクション、その他の満たすべき条件を指定します。

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任 (A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートを含めると、そのマトリックスは RASCI マトリックスと呼ばれ、サポートを除外すると RACI マトリックスと呼ばれます。

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、Implementing security controls on AWSのResponsive controlsを参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

取得拡張生成 (RAG)

LLM がレスポンスを生成する前にトレーニングデータソースの外部にある信頼できるデータソースを参照する生成 AI テクノロジー。例えば、RAG モデルは組織のナレッジベースまたはカスタムデータのセマンティック検索を実行する場合があります。詳細については、「RAG とは」を参照してください。

ローテーション

攻撃者が認証情報にアクセスすることをより困難にするために、シークレットを定期的に更新するプロセス。

行と列のアクセス制御 (RCAC)

アクセスルールが定義された、基本的で柔軟な SQL 表現の使用。RCAC は行権限と列マスクで構成されています。

RPO

「目標復旧時点」を参照してください。

RTO

目標復旧時間を参照してください。

ランブック

特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。

S

SAML 2.0

多くの ID プロバイダー (IdP) が使用しているオープンスタンダード。この機能を使用すると、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 にログイン AWS マネジメントコンソール したり AWS 、API オペレーションを呼び出すことができます。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントのSAML 2.0 ベースのフェデレーションについてを参照してください。

SCADA

「監視コントロールとデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存する AWS Secrets Managerパスワードやユーザー認証情報などの機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値は、バイナリ、1 つの文字列、または複数の文字列にすることができます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの内容」を参照してください。

設計によるセキュリティ

開発プロセス全体でセキュリティを考慮するシステムエンジニアリングアプローチ。

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、予防的、検出的、応答的、プロアクティブの 4 つの主なタイプがあります。

セキュリティ強化

アタックサーフェスを狭めて攻撃への耐性を高めるプロセス。このプロセスには、不要になったリソースの削除、最小特権を付与するセキュリティのベストプラクティスの実装、設定ファイル内の不要な機能の無効化、といったアクションが含まれています。

Security Information and Event Management (SIEM) システム

セキュリティ情報管理 (SIM) とセキュリティイベント管理 (SEM) のシステムを組み合わせたツールとサービス。SIEM システムは、サーバー、ネットワーク、デバイス、その他ソースからデータを収集、モニタリング、分析して、脅威やセキュリティ違反を検出し、アラートを発信します。

セキュリティレスポンスの自動化

セキュリティイベントに自動的に応答または修復するように設計された、事前定義されたプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例としては、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータの、それ AWS のサービス を受け取る による暗号化。

サービスコントロールポリシー (SCP)

AWS Organizationsの組織内の、すべてのアカウントのアクセス許可を一元的に管理するポリシー。SCP は、管理者がユーザーまたはロールに委任するアクションに、ガードレールを定義したり、アクションの制限を設定したりします。SCP は、許可リストまたは拒否リストとして、許可または禁止するサービスやアクションを指定する際に使用できます。詳細については、 AWS Organizations ドキュメントの「サービスコントロールポリシー」を参照してください。

サービスエンドポイント

のエントリポイントの URL AWS のサービス。ターゲットサービスにプログラムで接続するには、エンドポイントを使用します。詳細については、AWS 全般のリファレンスの「AWS のサービス エンドポイント」を参照してください。

サービスレベルアグリーメント (SLA)

サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットなど、サービスのパフォーマンス側面の測定。

サービスレベルの目標 (SLO)

サービスレベルのインジケータによって測定される、サービスの状態を表すターゲットメトリクス。

責任共有モデル

クラウドのセキュリティとコンプライアンス AWS について と共有する責任を説明するモデル。AWS はクラウドのセキュリティを担当しますが、お客様はクラウドのセキュリティを担当します。詳細については、責任共有モデルを参照してください。

SIEM

セキュリティ情報とイベント管理システムを参照してください。

単一障害点 (SPOF)

システムを中断する可能性のあるアプリケーションの 1 つの重要なコンポーネントの障害。

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お

客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、『』の「アプリケーションをモダナイズするための段階的アプローチ AWS クラウド」を参照してください。

SPOF

単一障害点を参照してください。

スタースキーマ

1 つの大きなファクトテーブルを使用してトランザクションデータまたは測定データを保存し、1 つ以上の小さなディメンションテーブルを使用してデータ属性を保存するデータベース組織構造。この構造は、データウェアハウスまたはビジネスインテリジェンスの目的で使用するように設計されています。

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズを参照してください。

サブネット

VPC 内の IP アドレスの範囲。サブネットは、1 つのアベイラビリティーゾーンに存在する必要があります。

監視制御とデータ収集 (SCADA)

製造では、ハードウェアとソフトウェアを使用して物理アセットと本番稼働をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして潜在的な問題を検出したり、パフォーマンスをモニタリングしたりする方法でシステムをテストします。Amazon CloudWatch Synthetics を使用して、これらのテストを作成できます。

システムプロンプト

LLM にコンテキスト、指示、またはガイドラインを提供して動作を指示する手法。システムプロンプトは、コンテキストを設定し、ユーザーとのやり取りのルールを確立するのに役立ちます。

T

tags

AWS リソースを整理するためのメタデータとして機能するキーと値のペア。タグは、 リソースの管理、識別、整理、検索、フィルタリングに役立ちます。詳細については、「AWS リソースのタグ付け」を参照してください。

ターゲット変数

監督された機械学習でお客様が予測しようとしている値。これは、結果変数 のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。

タスクリスト

ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。

テスト環境

「環境」を参照してください。

トレーニング

お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。

トランジットゲートウェイ

VPC とオンプレミスネットワークを相互接続するために使用できる、ネットワークの中継ハブ。詳細については、 AWS Transit Gateway ドキュメントの「トランジットゲートウェイとは」を参照してください。

トランクベースのワークフロー

デベロッパーが機能ブランチで機能をローカルにビルドしてテストし、その変更をメインブランチにマージするアプローチ。メインブランチはその後、開発環境、本番前環境、本番環境に合わせて順次構築されます。

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定したサービスにアクセス許可を付与します。信頼されたサービスは、サービスにリンクされたロールを必要なときに各アカウントに作成し、ユーザーに代わって管理タスクを実行します。詳細については、 ドキュメントの「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

チューニング

機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。

ツーピザチーム

2 枚のピザで養うことができるくらい小さな DevOps チーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。

U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。詳細については、深層学習システムにおける不確実性の定量化 ガイドを参照してください。

未分化なタスク

ヘビーリフティングとも呼ばれ、アプリケーションの作成と運用には必要だが、エンドユーザーに直接的な価値をもたらさなかったり、競争上の優位性をもたらしたりしない作業です。未分化なタスクの例としては、調達、メンテナンス、キャパシティプランニングなどがあります。

上位環境

???「環境」を参照してください。

V

バキューミング

ストレージを再利用してパフォーマンスを向上させるために、増分更新後にクリーンアップを行うデータベースのメンテナンス操作。

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

プライベート IP アドレスを使用してトラフィックをルーティングできる、2 つの VPC 間の接続。詳細については、Amazon VPC ドキュメントの「VPC ピア機能とは」を参照してください。

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

頻繁にアクセスされる最新の関連データを含むバッファキャッシュ。データベースインスタンスはバッファキャッシュから、メインメモリまたはディスクからよりも短い時間で読み取りを行うことができます。

ウォームデータ

アクセス頻度の低いデータ。この種類のデータをクエリする場合、通常は適度に遅いクエリでも問題ありません。

ウィンドウ関数

現在のレコードに関連する行のグループに対して計算を実行する SQL 関数。ウィンドウ関数は、移動平均の計算や、現在の行の相対位置に基づく行の値へのアクセスなどのタスクの処理に役立ちます。

ワークロード

ビジネス価値をもたらすリソースとコード (顧客向けアプリケーションやバックエンドプロセスなど) の総称。

ワークストリーム

特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。

WORM

「書き込み 1 回」、「読み取り多数」を参照してください。

WQF

AWS 「ワークロード認定フレームワーク」を参照してください。

Write Once, Read Many (WORM)

データを 1 回書き込み、データの削除や変更を防ぐストレージモデル。承認されたユーザーは、必要な回数だけデータを読み取ることができますが、変更することはできません。このデータストレージインフラストラクチャはイミュータブルと見なされます。

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を利用する攻撃、通常はマルウェア。

ゼロデイ脆弱性

実稼働システムにおける未解決の欠陥または脆弱性。脅威アクターは、このような脆弱性を利用してシステムを攻撃する可能性があります。開発者は、よく攻撃の結果で脆弱性に気付きます。

ゼロショットプロンプト

LLM にタスクを実行する手順を提供しますが、タスクのガイドに役立つ例 (ショット) はありません。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

ゾンビアプリケーション

平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。