
AWS の規範的ガイダンス

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



AWS の規範的ガイドンス: Amazon を使用したロギングとモニタリングの 設計と実装 CloudWatch

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

Amazon の商標およびトレードドレスは、お客様に混乱を招く可能性がある態様、または Amazon の信用を傷つけたり、失わせたりする態様において、Amazon のものではない製品またはサービスに関連して使用してはなりません。Amazon が所有しないその他すべての商標は、それぞれの所有者の所有物であり、これらの所有者は、必ずしも Amazon と提携している、関連がある、または Amazon の後援を受けているとはかぎりません。

Table of Contents

イントロダクション	1
ターゲットを絞ったビジネス成果	4
オペレーショナルレディネスの迅速化	4
オペレーショナルエクセルエンスの向上	4
オペレーショナルレジリエンスの向上	4
オペレーションの拡張とオーバーヘッドコストの削減	4
の計画 CloudWatch 配備	6
を使用する CloudWatch 集中型、または分散型アカウント	6
標準の作成と保存をします。 CloudWatch エージェント設定ファイル	9
保存 CloudWatch S3 バケットの設定ファイル	9
設定: CloudWatch EC2 インスタンスとオンプレミスサーバー用のエージェント	11
設定: CloudWatch エージェント	11
EC2 インスタンスのログキャプチャの設定	12
EC2 インスタンスのメトリクスキャプチャの設定	14
システムレベル CloudWatch 設定	15
システムレベルのログの設定	15
システムレベルのメトリクスを設定する	17
アプリケーションレベル CloudWatch 設定	17
アプリケーションレベルのログの設定	17
アプリケーションレベルのメトリクスを設定する	18
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	20
インストール: CloudWatch 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す るエージェント	20
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 CloudWatch エージェント のデプロイと設定	21
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。	22
クイックセットアップの代わりに AWS CloudFormation を使用する	23
AWS CloudFormation スタックを使用して、単一のアカウントとリージョンでカスタマイズした クイックセットアップ	24
AWS CloudFormation スタックセットを使用して複数のリージョンおよび複数のアカウントでカ スタマイズされたクイックセットアップ	25
オンプレミスサーバーを構成する際の考慮事項	26
エフェメラル EC2 インスタンスに関する考慮事項	27
をデプロイするための自動化ソリューションの使用 CloudWatch エージェント	27
をデプロイする CloudWatch 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 中のエージェント	27
を含む CloudWatch AMI のエージェント	28
Amazon ECS でのログ記録とモニタリング	29
の設定 CloudWatch EC2 起動タイプの場合	29
EC2 および Fargate 起動タイプの Amazon ECS コンテナログ	30
でカスタムログルーティングを使用する FireLens Amazon ECS 用	31
Amazon ECS のメトリクス	31
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	32
Amazon EKS でのログ記録とモニタリング	34
Amazon EKS のログ記録	34
Amazon EKS コントロールプレーンのログ記録	34
Amazon EKS ノードとアプリケーションのログ記録	35
Fargate での Amazon EKS のログ記録	36
Amazon EKS および Kubernetes のメトリクス	36
Kubernetes コントロールプレーンのメトリクス	37
Kubernetes のノードとシステムメトリック	37
アプリケーションメトリクス	38
Fargate での Amazon EKS のメトリクス	38

Amazon EKS における Prometheus モニタリング	39
AWS Lambda のログ記録とメトリクス。	40
Lambda 関数のログを記録する	40
CloudWatch から他の宛先にログを送信する	41
Lambda 関数メトリクス	41
システムレベルのメトリクス	41
アプリケーション・メトリクス	42
CloudWatch でのログの検索および分析	43
でアプリケーションをまとめてモニタリングおよび分析する CloudWatch アプリケーションインサイ ト	43
でログ分析を実行する CloudWatch ログインサイト	45
Amazon でログ分析を実行する OpenSearch サービス	46
CloudWatch によるアラームのオプション	48
を使用する CloudWatch をモニタリングし、アラームを発する	48
を使用する CloudWatch 異常検出とアラームの実行	48
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	49
EC2 インスタンスタグを使用したアラーム作成の自動化	49
アプリケーションとサービスの可用性のモニタリング	50
AWS X-Ray でアプリケーションをトレースする	51
X-Ray デモンをデプロイし、Amazon EC2 でのアプリケーションとサービスをトレースする	51
X-Ray デモンをデプロイし、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	52
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	52
X-Ray 向けにアプリケーションをインストールする	52
X-Ray のサンプリングルールを設定する	53
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	54
クロスサービスダッシュボードを作成する	54
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	54
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	55
Metric Math を使用してオペレータビリティとアラームを微調整する	55
で、Amazon ECS、Amazon EKS、および Lambda 自動ダッシュボードを使用する	56
CloudWatchContainer インサイトと CloudWatch Lambda Insights	56
AWS サービスと CloudWatch の統合。	57
ダッシュボードと可視化のための Amazon Managed Grafana (プレビュー)	58
よくある質問	60
どこに保存すればよいですか CloudWatch 設定ファイル?	60
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?	60
どのように使用しますか CloudWatch コンテナにログファイルをキャプチャするには?	60
AWS サービスのヘルス問題をモニタリングするにはどうしたらいいですか?	61
カスタムを作成するにはどうすればよいですか CloudWatch エージェントサポートが存在しない場合 のメトリック	61
AWS の既存のログおよびモニタリングツールを統合するにはどうしたらよいですか?	61
リソース	62
イントロダクション	62
ターゲットを絞ったビジネス成果	62
の計画 CloudWatch 配備	62
設定: CloudWatch EC2 インスタンスとオンプレミスサーバー用のエージェント	62
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	63
Amazon ECS でのログ記録とモニタリング	63
Amazon EKS でのログ記録とモニタリング	63
AWS Lambda のログ記録とメトリクス。	63
CloudWatch でのログの検索および分析	64
CloudWatch によるアラームのオプション	64
アプリケーションとサービスの可用性のモニタリング	65
AWS X-Ray でアプリケーションをトレース	65

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	65
CloudWatch との統合 AWS サービス	65
ダッシュボードと可視化のための Amazon マネージド Grafana	65
AWS 規範的ガイダンスの用語集	67
ドキュメント履歴	75
.....	lxxvi

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

AWSプロフェッショナルサービス部門 コンサルタント Khurram Nizami 氏

2021年4月最終更新 (p. 75): 2022年1月)

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。このガイドはオペレーションチームを対象としており、DevOps のワークロードを管理するエンジニア、およびアプリケーションエンジニアAWSクラウド

ロギングとモニタリングのアプローチは、AWS Well-Architected Framework の 5 つの柱に基づいている必要があります。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 など多岐にわたります。また、AWS Cost Explorer、AWS Budgets、および CloudWatch 請求メトリクス はコスト最適化のためのものです。

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。多数AWSサービスは自動的に提供する CloudWatch メトリク

ス、CloudWatch モニタリングと分析のためのロギング統合 CloudWatch また、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、ロギング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。

- [AWS Systems Manager Distributor](#), [Systems Manager ステートマネージャー](#), および [Systems Manager Automation](#) を自動化、構成、および更新するには CloudWatch EC2 インスタンスとオンプレミスサーバーのエージェント
- [アマゾン OpenSearch サービス \(Amazon Elasticsearch Service の後継サービス\)](#) 高度なログ集計・検索・分析機能
- [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。
- [Amazon Managed Grafana](#) 複数のソース (例えば、CloudWatch、Amazon など) のデータを可視化し、分析することができます。OpenSearch サービス、および [Amazon Timestream](#))

選択する AWS コンピューティングサービスは、ロギングとモニタリングの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 (OS) にまたがるシステム、およびアプリケーションのログとメトリクスです。
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、以下の領域で CloudWatch と関連する AWS サービスに対応するロギングとモニタリングのソリューションを提供します。

- [の計画 CloudWatch 配備 \(p. 6\)](#)— お客様の計画に関する考慮事項 CloudWatch の一元化に関する導入とガイダンス CloudWatch の設定
- [設定: CloudWatch EC2 インスタンスとオンプレミスサーバー用のエージェント \(p. 11\)](#)— CloudWatch 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 \(p. 20\)](#)— インストールのアプローチ CloudWatch 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エージェント

- [Amazon ECS でのログ記録とモニタリング \(p. 29\)](#)-設定用ガイド CloudWatch Amazon ECS のクラスターレベル、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です。
- [Amazon EKS でのログ記録とモニタリング \(p. 34\)](#)-設定用ガイド CloudWatch Amazon EKS のクラスターレベル、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です。
- [Amazon EKS における Prometheus モニタリング \(p. 39\)](#)-Amazon Managed Service for Prometheus を導入し、比較します CloudWatch Prometheus のコンテナインサイトモニタリング
- [AWS Lambda のログ記録とメトリクス。 \(p. 40\)](#)-設定用ガイド CloudWatch Lambda 関数用です。
- [CloudWatch でのログの検索および分析 \(p. 43\)](#)— Amazon を使用してログを分析する方法 CloudWatch アプリケーションインサイトです CloudWatch ログインサイト、およびログ分析を Amazon に拡張する OpenSearch サービス。
- [CloudWatch によるアラームのオプション \(p. 48\)](#)— について説明します CloudWatch アラームと CloudWatch 異常検出とアラームの作成とセットアップのガイドを提供します。
- [アプリケーションとサービスの可用性のモニタリング \(p. 50\)](#)-について説明します。 CloudWatch 可用性の自動モニタリングのための Synthetics と Route 53 ヘルスチェック
- [AWS X-Ray でアプリケーションをトレースする \(p. 51\)](#) - 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 \(p. 54\)](#) - CloudWatch ダッシュボードを紹介し、AWS ワークロード全体の観測性を向上させます。
- [AWS サービスと CloudWatch の統合。 \(p. 57\)](#)— について説明します CloudWatch さまざまなと統合できますAWSのサービス。
- [ダッシュボードと可視化のための Amazon Managed Grafana \(プレビュー\) \(p. 58\)](#)— Amazon マネージド Grafana を紹介し、比較します。 CloudWatch ダッシュボードと可視化に使用します。

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、[AWS サンプル GitHub 倉庫](#)。

ターゲットを絞ったビジネス成果

AWS クラウド用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の作成は、[クラウドコンピューティングの 6 つの利点](#)の実現に不可欠です。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 Cloud のロギング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、Infrastructure as Code (IaC) アプローチで使用できる[詳細な例とオープンソース、再利用可能なテンプレート](#)も提供し、AWS サービスを使って、Well-Architected ロギングおよびモニタリングソリューションを実装できます。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。

オペレーショナルレジリエンスの向上

お客様のビジネスプロセスとアプリケーションは、さまざまな IT リソースによってサポートされ、オンプレミスまたは AWS Cloud で、さまざまなコンピューティングタイプにホストされる可能性があります。オペレーショナルレジリエンスは、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レジリエンスを改善し、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。

の計画 CloudWatch 配備

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。
- オンプレミスの拠点と AWS インフラの両方があるかどうかです。
- 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。確立されたログとモニタリングアプローチでは、異なるドメイン（たとえば、セキュリティ、パフォーマンス、ネットワークング、分析など）に対して、複数の AWS または独立系ソフトウェア・ベンダー（ISV）のサービスや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。

を使用することをお勧めします。CloudWatch 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できます。多数AWSのサービスで使用 CloudWatch 追加の構成を必要とせず、ログとメトリクスの記録、監視、公開を行います。CloudWatch を提供する。ソフトウェアエージェント異なる 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、のデプロイ、インストール、および設定する方法について説明します。CloudWatch 複数のアカウント、リージョン、および構成のエージェント:

トピック

- [を使用する CloudWatch 集中型、または分散型アカウント \(p. 6\)](#)
- [標準の作成と保存をします。CloudWatch エージェント設定ファイル \(p. 9\)](#)

を使用する CloudWatch 集中型、または分散型アカウント

でもね CloudWatch 監視するように設計されているAWS1つのアカウントとリージョン内のサービスまたはリソースでは、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のログとメトリクスを取得することができます。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。

AWS の規範的ガイダンス Amazon を使用したロ
ギングとモニタリングの設計と実装 CloudWatch
を使用する CloudWatch 集中型、または分散型アカウント

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	<p>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。</p>
アクセスの要件	<p>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。</p> <p>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。中央ログ記録アカウントは、Amazon など、別の検索および集計オプションを持つこともできます。OpenSearch サービスクラスター。アマゾン OpenSearch サービス きめ細かなアクセスコントロールを提供 ログのフィールドレベルまで下げます。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。</p>
オペレーション	<p>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「あなたが構築し、あなたが実行する」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。</p>
環境	<p>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。</p>

CloudWatchが提供する [複数のオプション](#) を使用してログをリアルタイムで処理するには CloudWatch サブスクリプションフィルタ。サブスクリプションフィルタを使用して、カスタム処理、分析、および他のシステムへのロードのために、AWS サービスにリアルタイムでログをストリーミングすることができます。

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 サービスの例です。

- **Amazon Kinesis Data Firehose** - Kinesis Data Firehose は、生成されるデータ量に基づいて自動的にスケールアップとリサイズを行うストリーミングソリューションを提供します。Amazon Kinesis データストリームのシャード数を管理する必要はなく、Amazon Simple Storage Service (Amazon S3)、Amazon S3 (Amazon S3) に直接接続することができます。OpenSearch サービス、または追加のコーディングなしで Amazon Redshift。Kinesis Data Firehose は、それらの AWS サービスにログを一元管理したい場合に有効なソリューションです。
- **Amazon Kinesis Data Streams** - Kinesis Data 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 が適切なソリューションとなります。Amazon を作成できる CloudWatch アカウントとリージョンで Kinesis Data Streams を指定し、中央アカウントで Kinesis Data Streams を指定し、AWS Identity and Access Management ストリームにシールドを配置するアクセス権限を付与する (IAM) ロール。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。

- **アマゾン OpenSearch サービス (Amazon Elasticsearch Service の後継サービス)** - CloudWatch ログは、ロググループ内のログを OpenSearch 個別アカウントまたは集中型アカウントでのクラスタ化。データをストリームするようにロググループを設定すると OpenSearch クラスタでは、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 関数は、OpenSearch クラスタ。Amazon への取り込みをカスタマイズするだけでなく、Lambda 関数をカスタマイズして前処理を追加で行うことも可能です。OpenSearch サービス。Amazon を使用した集中ロギング機能 OpenSearch サービスにより、クラウドアーキテクチャ内の複数のコンポーネントにおける問題の分析、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。

に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。CloudWatch ファイル形式でログインします。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 エージェント設定は、credentials フィールド内の **agent セクション**。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。

また、**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作成し、アカウントからログやメトリクスを読み取り、データを集中管理アカウントなどに送信して、さらなる処理とモニタリングを行うことも可能です。

標準の作成と保存をします。 CloudWatch エージェント設定ファイル

標準を作成することをお勧めします。 CloudWatch EC2 インスタンスとオンプレミスサーバーすべてで取得したいシステムログとメトリクスを含むエージェント設定ファイル。 CloudWatch エージェント設定ファイルウィザードを使用して、設定ファイルを作成するのに役立ちます。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 Parameter Store](#)。しかし、この方法は、パラメータ値を各アカウントとリージョンで作成し、同期する必要があるため、複数のアカウントとリージョンにまたがる拡張が難しくなります。

-amazon-cloudwatch-agent-ctlに付属するスクリプト CloudWatch agent 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一切含まれません。デフォルト設定は、を使用しても適用されます。 [Systems Manager 高速セットアップ](#)向けの CloudWatch エージェント。

デフォルトの設定はロギングを含まず要件に合わせてカスタマイズされていないため、設定ファイルを作成しロードすることをお勧めします。 CloudWatch 設定ファイルディレクトリ。 Linux の場合、CloudWatch 設定ディレクトリは次の場所にあります。 /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d。 Windows の場合、設定ディレクトリは C:\ProgramData\Amazon\AmazonCloudWatchAgent\Configs にあります。

スタート時 CloudWatch エージェントの場合、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 複合構成ファイル。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。また、コピーする必要があります。 CloudWatch 設定ファイルディレクトリ。この方法の詳細については、このガイドの [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 CloudWatch エージェントのデプロイと設定](#) (p. 21) セクションを参照してください。

保存 CloudWatch S3 バケットの設定ファイル

を使用することをお勧めします。 CloudWatch 以下の理由により、Parameter Store パラメータの代わりに、設定ファイルを指定します。

- 複数のリージョンを使用する場合は、同期する必要があります。 CloudWatch 各リージョンのパラメータストアで設定が更新されます。 Parameter Store はリージョンのサービスであり、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。 CloudWatch エージェント。
- 複数ある場合 CloudWatch 設定の場合は、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。それぞれを個別に取得する必要があります。 CloudWatch パラメータストアから設定を行い、新しい設定を追加するたびに取得メソッドを更新します。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。

保存することができます CloudWatch 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 バケットの設定。その後、S3 バケットから、これらの設定をコピーできます。 CloudWatch Systems Manager Automation ランプブックと Systems Manager、ステートマネージャーを使用して、設定ディレクトリ。 [cloudwatch-config-s3-bucket.yaml](#) AWS CloudFormation テンプレートを使用すると、組織内の複数のアカウントからアクセス可能な S3 バケットを AWS Organizations に作成することができます。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。

S3 バケットを作成した後、キーまたはフォルダーのプレフィックス構造を作成し、CloudWatch 設定ファイル。次の表は、サンプルのフォルダ構造の概要を示しています。

/config/standard/windows/ec2	Windows 標準の Windows 固有の保存 CloudWatch Amazon EC2 の設定ファイル。また、このフォルダの下にあるさまざまな Windows バージョン、EC2 インスタンスタイプ、および環境の標準 OS 構成をさらに分類することもできます。
/config/standard/windows/onpremises	Windows 標準の Windows 固有の保存 CloudWatch オンプレミスサーバー用の設定ファイル。また、このフォルダの下に、Windowsのバージョン、サーバーの種類、環境ごとに標準的な OS の設定をさらに分類することができます。
/config/standard/linux/ec2	標準的な Linux 固有のものを保存する CloudWatch Amazon EC2 の設定ファイル。また、このフォルダーでは、Linuxディストリビューション、EC2 インスタンスタイプ、環境ごとに標準 OS の構成をさらに分類することができます。
/config/standard/linux/onpremises	標準的な Linux 固有のものを保存する CloudWatch オンプレミスサーバー用の設定ファイル。また、Linux ディストリビューション、サーバータイプ、環境ごとに標準的な OS の設定をこのフォルダーにさらに分類することができます。
/config/ecs	保存 CloudWatch Amazon ECS コンテナインスタンスを使用する場合は、Amazon ECS に固有の設定ファイル。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保存をします。CloudWatch 設定ファイル。また、環境およびバージョン用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することもできます。

このガイドの [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 CloudWatch エージェントのデプロイと設定 \(p. 21\)](#) セクションで提供される拡張されたサンプル Systems Manager Runbook は、[cloudwatch-config-s3-bucket.yaml](#) AWS 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ます。

代わりに、バージョン管理システムを使用することもできます (たとえば、GitHub または [AWS CodeCommit](#)) を使用して設定ファイルを保存します。バージョン管理システムに保存された設定ファイルを自動的に取得したい場合は、認証情報のストレージを管理または一元化し、アカウントとリージョン間で認証情報を取得するために使用する Systems Manager オートメーションランブックを更新する必要があります。

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

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。

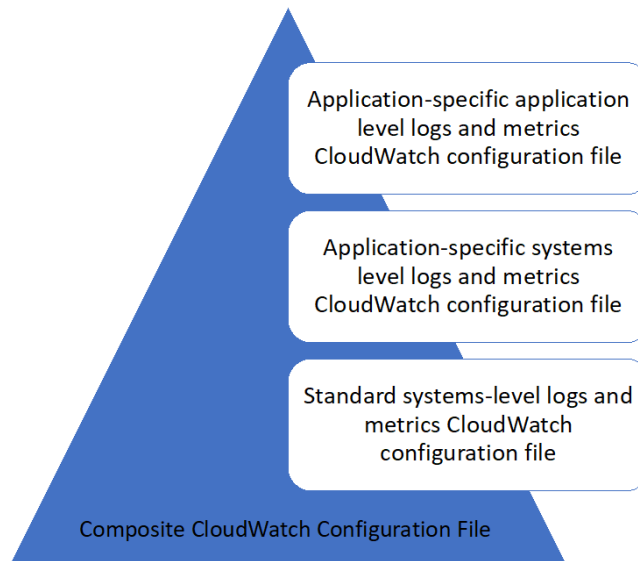
IT 投資を移行または拡張する Organizations AWS クラウドが活用できる CloudWatch 統合ロギングおよびモニタリングソリューションを実現します。CloudWatch 料金は、取得するメトリクスとログに対して段階的に料金を支払うことを意味します。同様のを使用して、オンプレミスサーバーのログとメトリクスをキャプチャすることもできます。CloudWatch Amazon EC2 のエージェントインストールプロセス。

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。

設定: CloudWatch エージェント

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。インストールを開始する前に、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。CloudWatch アカウント内の大規模なエージェント。

複数を組み合わせることができます。CloudWatch コンポジットを形成するためのエージェント構成 CloudWatch エージェント設定。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、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、ログファイルがキャプチャされ、に取り込まれます。CloudWatch によるログ CloudWatch EC2 インスタンスにインストールされるエージェントソフトウェアAWSAPI、またはAWS Command Line Interface(AWS CLI) を使用することをお勧めします。CloudWatch ログファイルを取り込むエージェント CloudWatch Amazon EC2 およびオンプレミスサーバーのログ。

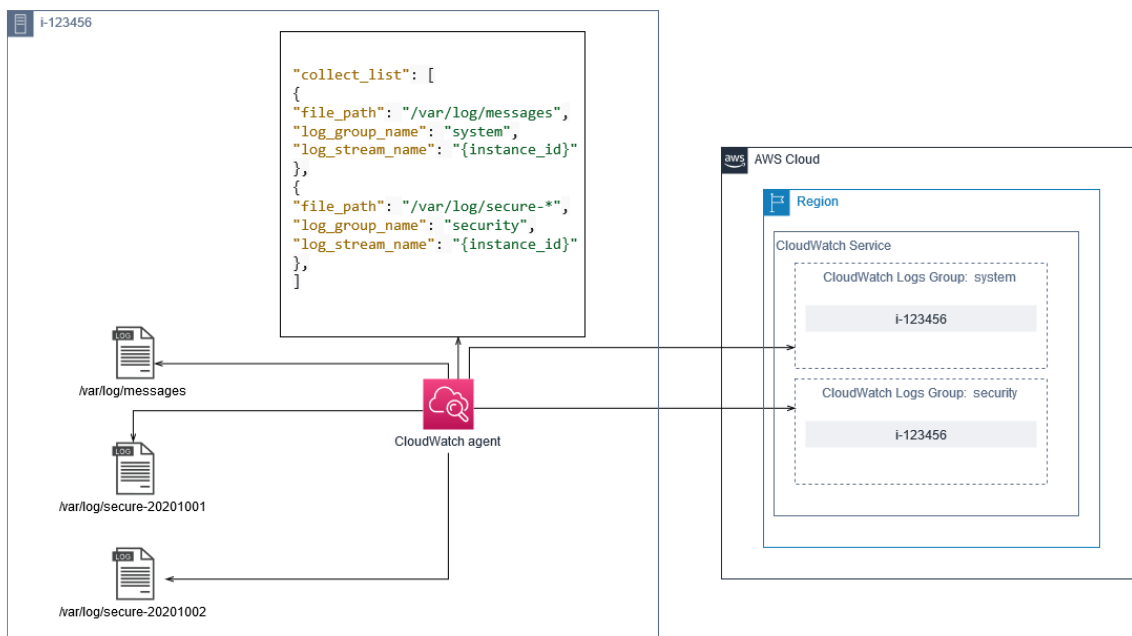
CloudWatch のログファイルからのパターンパッチ適用に基づいて、ログを検索してフィルタリングしたり、メトリクスを抽出したり、自動化を実行したりできます。CloudWatch は、プレーンテキスト、スペース区切り、および JSON 形式のフィルタおよびパターン構文オプションをサポートし、JSON 形式のログが最も柔軟になります。フィルタリングと分析オプションを増やすには、プレーンテキストではなくフォーマットされたログ出力を使用する必要があります。

- 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 次に、各ログファイルを**ログストリーム**これらのログストリームを**log group**。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。ログストリームの名前は、CloudWatch ロググループ。次を使用できます。instance_id,hostname,local_hostname,またはip_addressログストリームとロググループ名の動的置換の場合。つまり、同じものを使用できます。CloudWatch 複数の EC2 インスタンスにわたるエージェント設定ファイル。

次の図は、CloudWatch ログをキャプチャするためのエージェント設定。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。

AWS の規範的ガイドライン Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch EC2 インスタンスのログキャプチャの設定



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、**標準メトリックセット**(CPU、ネットワーク、ストレージ関連のメトリクスなど)は、自動的に送信されます。CloudWatch 5分ごとに。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。**バーストパフォーマンスインスタンス**CPU クレジットのメトリックがあります。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、のデフォルトのメトリクスは含まれません。CloudWatch を使用する必要があります。CloudWatch エージェント、AWS CLI、またはAWSメトリクスをキャプチャする SDK。つまり、キャプチャするメトリクス (CPU 使用率など) を CloudWatch 設定ファイル。一意を作成できます。CloudWatch 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設定ファイル。標準に加えて適用します。CloudWatch の設定

のメトリクスに CloudWatch 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、AWS (例えば、AWS/EC2) で始まる名前空間があり、非 AWS メトリクスは、カスタムメトリクスと見なされます。で設定してキャプチャするメトリクス CloudWatch エージェントはすべてカスタムメトリクスと見なされます。作成された指標の数は、CloudWatch コスト、各メトリクスがすべての EC2 インスタンスまたは一部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、が公開するメトリクスのデフォルトの名前空間空間です。CloudWatch エージェント。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。

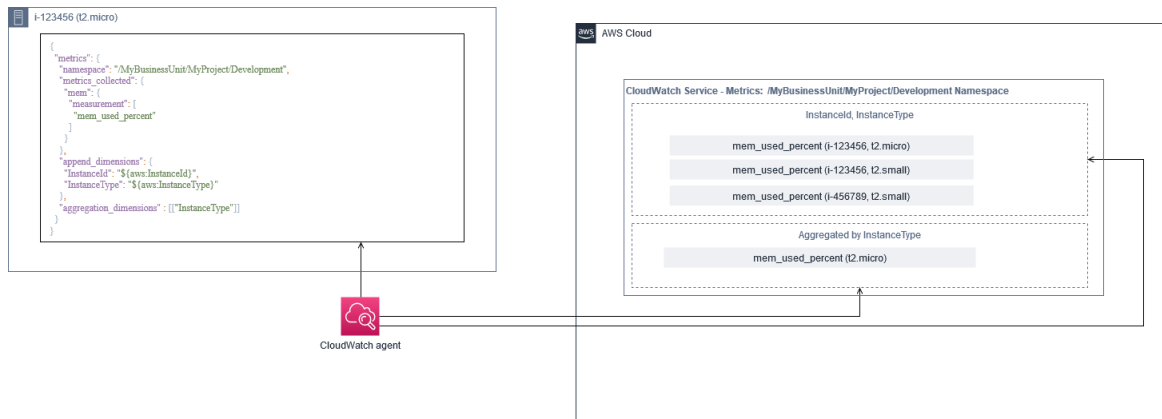
指標はディメンションによって識別され、一連の条件に対して分析するのに役立ち、観測値が記録されるプロパティです。Amazon EC2 には EC2 インスタンス用の **個別のメトリクス** が InstanceId およびAutoScalingGroupName ディメンションで含まれます。また、詳細モニタリングを有効にする場合、ImageId および InstanceType ディメンションでメトリクスを受け取ります。例えば、Amazon EC2 は、InstanceId ディメンション用の別の CPU 使用率メトリクスに加えて、InstanceType ディメンション CPU 使用率に関す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

`append_dimensions`プロパティを使用して、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 CloudWatch の設定 また、動的に追加することもできます。ImageId,InstanceId,InstanceType, およびAutoScalingGroupName内のすべてのメトリクス CloudWatch の設定 または、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。append_dimensionsそのメトリックのプロパティ。CloudWatch は、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。aggregation_dimensionsプロパティ。

例えば、InstanceType デイ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。

次の図表は、サンプルを示しています。CloudWatch カスタム名前空間、追加されたデイメンション、および集計を使用するメトリクス設定 InstanceType。



システムレベル CloudWatch 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 用の特定の構成オプションがあります。

を使用することをお勧めします。CloudWatch 設定ファイルウィザードまたは、設定ファイルスキーマ CloudWatch サポートする予定の各 OS のエージェント設定ファイル。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に定義できます。CloudWatch 設定ファイルを指定し、標準コンフィギュレーションに追加します。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「保存 CloudWatch S3 バケットの設定ファイル (p. 9)」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 Cloud 上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

/var/log/boot.log* — ブートアップログ

AWS の規範的ガイドライン Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch システムレベルのログの設定

(Amazon Linux、CentOS、Red Hat Enterprise Linux の場合)	/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。たとえば、Systems Manager と CloudWatch エージェントは設定にログインします。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、ログファイルが定義されている場合、ログファイルを無視します。CloudWatch エージェントの設定ですが、見つかりません。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、をインストールして設定する必要があります。CloudWatch EC2 インスタンスのエージェント。- CloudWatch エージェント設定ウィザードでは CloudWatch での設定 **定義済みメトリクス** 必要に応じて、メトリクスを追加または削除できます。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。これらのメトリック定義は、別のファイルに保存、バージョン管理、および保守する必要があります。CloudWatch エージェント設定ファイル。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 CloudWatch オンプレミスインスタンスで使用されるエージェント設定ファイル。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 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。標準のログとメトリックを作成できます。CloudWatch IIS の構成は、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。- 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に自動的に結合します。CloudWatch の設定 最終結果は、CloudWatch 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を含む構成 CloudWatch 構成。

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かどうかによって異なります。off-the-shelf (COTS) またはカスタム開発されたアプリケーション。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。

カスタム開発アプリケーションをと統合できます。CloudWatch のアプリケーションにアプリケーションのログファイルを含めることでログを記録します。CloudWatch の設定 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。

に書き込むこともできます。CloudWatch でのログストリーム CloudWatch ログ `PutLogEvents` API 呼び出し、または AWSSDK。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 ログファイルを読み取り、CloudWatch にストリーミングするエージェント。

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、このデータを測定、グラフ化、アラームできます。CloudWatch ロググループ。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

カスタム指標とは、によって直接提供されない指標です。AWSへのサービス CloudWatch また、カスタムネームスペースでパブリッシュされます。CloudWatch メトリクス。すべてのアプリケーションメトリクスは、カスタムと見なされます。CloudWatch メトリクス。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 コスト。

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
- 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 を使用して、ログを記録します。CloudWatch 埋め込みメトリクス形式。
- アプリケーションは、メトリクスをに送信します。CloudWatch での API または AWSSDK。
- アプリケーションは、メトリクスをに送信します。収集したまたは `statsD` 設定済みのデーモン CloudWatch エージェント。

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。複数を使用することもできます。CloudWatch メトリクスごとに異なるディメンションがある場合、アプリケーションのエージェント設定ファイル。

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ー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、アプリケーションの開発者と協力して、CloudWatch 埋込みメトリクス形式AWSSDK、またはAWSアピ。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 CloudWatch 設定](#)をクリックして、組み込みメトリクスフォーマットエージェントを含めます。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

を自動化する CloudWatch エージェントのインストールプロセスを使用すると、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。

- [インストール: CloudWatch エージェントは、Systems Manager デистриビューターと Systems Manager State Manager を使用して \(p. 20\)](#)— EC2 インスタンスとオンプレミスサーバーで Systems Manager エージェント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 エージェントは最新の状態に保たれ、サーバーがないサーバーについてレポートして修正できます。CloudWatch エージェント。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。
- [をデプロイする CloudWatch EC2 インスタンスのプロビジョニング中のユーザーデータスクリプトの一部としてのエージェント \(p. 27\)](#)— 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、AWS CloudFormation スクリプトおよび AWS Service Catalog 製品を含みます。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 \(p. 28\)](#) - 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。

インストール: CloudWatch Systems Manager デистриビューターとステートマネージャーを使用するエージェント

Systems Manager State Manager を Systems Manager のデистриビューターとともに使用して、CloudWatch サーバーおよび EC2 インスタンス上のエージェント。デистриビューターには AmazonCloudWatchAgent AWS 最新の CloudWatch エージェントバージョンをインストールする管理パッケージが含まれます。

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。
- IAM ロールまたは認証情報必須 [CloudWatch Systems Manager の権限](#) 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ている必要があります。例え

AWS の規範的ガイドライン Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch 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 CloudWatch エージェントのデプロイと設定

ば、以下を含む IAM ロールを作成できます。AWS マネージドポリシー: Systems Manager 用の AmazonSSMManagedInstanceCore と CloudWatch 用の CloudWatchAgentServerPolicy。 [ssm-cloudwatch-instance-role.yaml](#) AWS CloudFormation テンプレートを使用して、これらのポリシーを含む IAM ロールとインスタンスプロファイルをデプロイします。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、CloudWatch を使用するためのエージェント [Systems Manager サービスロール](#) オンプレミスサーバー用に構成されています。詳細については、「」を参照してください。 [Systems Manager Agent と統合サーバーを使用するオンプレミスサーバーを構成するにはどうすればよいですか](#) CloudWatch エージェントは一時的な資格情報のみを使用しますか? のAWSナレッジセンター。

次のリストに、Systems Manager ディストリビューターおよびステートマネージャーのアプローチを使用して、CloudWatch エージェント:

- 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。

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既にインストールまたは設定されている場合 CloudWatch エージェントの場合、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ーは、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設定ファイルを使用する場合 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。一元化または同期する必要があります CloudWatch 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構成します。

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 CloudWatch エージェントのデプロイと設定

次を使用できます。 [Systems Manager 高速セットアップ](#) Systems Manager の自動インストールと更新など、Systems Manager 機能をすばやく設定するには CloudWatch EC2 インスタンスのエージェント。クイックセットアップでは、選択に基づいて AWS CloudFormation Systems Manager リソースをデプロイして構成するスタックをデプロイします。

次のリストに、クイックセットアップによって自動化のために実行される 2 つの重要なアクションを示します。CloudWatch エージェントのインストールと更新:

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ーで使用するために、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。

AWS の規範的ガイドライン Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch 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。

- `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`— をインストールします。CloudWatch エージェントとディストリビューターを使用し、各 EC2 インスタンスをデフォルトで 1 回設定する CloudWatch を使用したエージェントの設定 `AWS-ConfigureAWSPackage` Systems Manager のドキュメント。
 - `UpdateCloudWatchDocument`— を更新しています CloudWatch エージェントは、最新の CloudWatch エージェントをインストールして、`AWS-ConfigureAWSPackage` Systems Manager のドキュメント。エージェントを更新またはアンインストールしても、既存のエージェントは削除されません。CloudWatch EC2 インスタンスからの設定ファイル。
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 アクセス権限を含めて、カスタム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。CloudWatch 構成。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。別の方法として、独自のものを使用できます。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 リソースを設定およびデプロイし、クイックセットアップを使用しないこと。

Important

高速セットアップにより AWS CloudFormation スタックして、選択内容に基づいて Systems Manager リソースをデプロイして構成します。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。

次のセクションでは、クイックセットアップで作成された Systems Manager リソースを手動で更新する方法と、独自の AWS CloudFormation 更新したクイックセットアップを実行するテンプレートの使用方法について説明します。AWS CloudFormation クイックセットアップと AWS CloudFormation で作成されたリソースの手動更新を回避するための独自のテンプレートを使用することをお勧めします。

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、必要なリソースを含める必要があります。CloudWatch エージェントの権限と複数のサポート CloudWatch 設定ファイル。このセクションでは、IAM ロールおよび Systems Manager ドキュメントを更新して、以下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 複数のアカウントからアクセスできる構成。を保存するための S3 バケットの作成 CloudWatch 設定ファイルについては、『』を参照してください。保存 [CloudWatch S3 バケットの設定ファイル](#) (p. 9) このガイドの「」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 インスタンスが既存のインスタンスプロファイルを使用して割り当てられる AWS アクセス許可を失うのを防ぐことができます。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 ポリシーを添付するには、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 エージェントを選択し、デフォルトで設定します。CloudWatch エージェントの設定。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、ステップを追加して、CloudWatch のコンフィギュレーションファイル CloudWatch S3 バケットの設定。

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。

クイックセットアップの代わりに AWS CloudFormation を使用する

Quick Setup を使用する代わりに AWS CloudFormation をクリックして、Systems Manager を構成します。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。CloudWatch 構成。

クイックセットアップ機能では、AWS CloudFormation を使用し、選択に基づいて Systems Manager リソースをデプロイして構成するための AWS CloudFormation スタックセットを作成します。使用する前に AWS CloudFormation スタックセットを使用するには、使用する IAM ロールを作成する必要があります。AWS CloudFormation StackSets 複数のアカウントまたはリージョンにまたがるデプロイメントをサポートします。クイックセットアップは、AWS CloudFormation スタックセットでマルチリージョンまたはマルチアカウントの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。の前提条件を満たしている必要があります。AWS CloudFormation StackSets 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する場合。詳細については、AWS CloudFormation ドキュメント [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 Quick Setup の [AWS-quickSetup-ssmhostMgmt.yaml](#) AWS CloudFormation テンプレートを確認します。

AWS CloudFormation テンプレートのリソースと機能を確認し、要件に応じて調整します。必要な結果を確認するために、変更を使用して段階的にテストする AWS CloudFormation テンプレートのバージョン管理を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。

AWS CloudFormation 単一のテストアカウントとリージョンでのスタックをデプロイし、必要なテストケースを実行して、望ましい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。

AWS CloudFormation スタックを使用して、単一のアカウントとリージョンでカスタマイズしたクイックセットアップ

アカウントとリージョンを1つだけ使用している場合は、AWS CloudFormation スタックセットの代わりに完全な例を AWS 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。を使用するAWS CloudFormation StackSets これにより、future 追加のアカウントとリージョンへの拡張が容易になります。

1つのアカウントおよび1つのリージョンで AWS CloudFormation スタックとして [AWS-QuickSetup-SSMHostMgmt.yaml](#) AWS CloudFormation テンプレートをデプロイするには、次の手順を使用します。

1. テンプレートをダウンロードし、お好みのバージョン管理システムにチェックインします (例: AWS CodeCommit)。
2. 組織の要件に基づいてデフォルトの AWS 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、以下を含む 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。CloudWatch の設定
5. 以下を含む 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。CloudWatch 構成。詳細については、このガイドの [保存 CloudWatch S3 バケットの設定ファイル \(p. 9\)](#) セクションを参照してください。AWS CloudFormation アカウントに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](#) AWS Organizations テンプレートを使用できます。
6. カスタマイズしたクイックセットアップの AWS 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。

- AWS の規範的ガイド Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch
AWS CloudFormation スタックセットを使用して複数のリージョンおよび複数のアカウント
-
- **を確認します。** CloudWatch カスタマイズされた表示して正常に動作している CloudWatch に基づくログとメトリクス CloudWatch S3 バケットからの設定。

AWS CloudFormation スタックセットを使用して複数のリージョンおよび複数のアカウントでカスタマイズされたクイックセットアップ

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](#) AWS CloudFormation テンプレートをスタックセットとして指定します。スタックセットを使用する前に [AWS 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。AWS Organizations 管理アカウントまたは委任された管理者アカウントからサービスマネージド型のスタックセットを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、管理者権限が委任された自動化に使用される集中型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 [AWS CloudFormation スタックを使用して、単一のアカウントとリージョンでカスタマイズしたクイックセットアップ](#) (p. 24) セクションを参照してステップ 1 ~ 5 を完了します。
2. AWS Management Console にサインインして、AWS CloudFormation コンソールを開いてスタックセット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。AWS CloudFormation 要件に合わせてカスタマイズした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。
 - 向けのCloudWatchConfigBucketARNパラメータで、の ARN を入力します。CloudWatch 設定の S3 バケット。
 - スタックセットオプションを指定し、AWS Organizations または自己管理型のアクセス許可で、サービス管理権限を使用するかどうかを選択します。
 - 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。
 - 使用するリージョンを入力します。CloudWatch Systems Manager です。
 - 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。
 - Systems Manager をテストし、CloudWatch デプロイされたアカウントで、ステップ 7 に従って、正常に動作している [AWS CloudFormation スタック](#) を使用して、[単一のアカウントとリージョ](#)

ンでカスタマイズしたクイックセットアップ (p. 24) このガイドの「」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

- CloudWatch オンプレミスサーバーおよび VM のエージェントは、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、次の表に、をインストールおよび設定する
ときに評価する必要がある考慮事項を示します。 CloudWatch オンプレミスサーバーと VM 上のエーজে
ント。

<p>をポイントする 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。</p>	<p>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。</p> <p>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。君を指し示すことができ る CloudWatch 同じファイルにエージェント設定。 次のプロセスを使用できます。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が、AWS ナレッジ センターで一時的な認証情報のみを使用するよう に構成します。</p> <p>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。</p>
<p>VPN または AWS Direct Connect アクセスおよび 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。</p>	<p>AWS Direct Connect または AWS Virtual Private Network (AWS 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。AWSPrivateLink は、へのプライベート接続を 確立します。CloudWatch インターフェイス VPC エンドポイントでログを記録します。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。</p>
<p>すべてのメトリクスを含める必要があります CloudWatch 設定ファイル。</p>	<p>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 する必要があります。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、 設定を標準に</p>

追加できます。CloudWatch プラットフォームのメトリック設定。

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、非常に多くを引き起こす可能性があります。CloudWatch 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ストリームされます。

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。例えば、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 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。そのため、CloudWatch ブート時にエージェント設定ファイル、および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、メトリクスとログが CloudWatch エフェメラル EC2 インスタンスが終了する前のエージェント。 - CloudWatch エージェントには `flush_interval` ログバッファとメトリクスバッファのフラッシュの時間間隔を定義するように構成できるパラメータ。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します。

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

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、自動的にインストールおよび更新できます。CloudWatch エージェント。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、定期的に更新をチェックする必要があります。CloudWatch エージェントを使用するか、エージェントを定期的にアンインストールして再インストールします。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、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。

をデプロイする CloudWatch 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のエージェント

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 回限りの場合、または特殊な構成が必要な場合に使

用されます。AWS 提供する [直接リンク](#) 向けの CloudWatch スタートアップ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エージェント。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、インストール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。CloudWatch エージェント。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。
- 定期的に更新する必要があります CloudWatch 新しいバージョンのエージェント (利用可能な場合)。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、起動時に設定ファイルを取得し、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。CloudWatch エージェント。

を含む CloudWatch AMI のエージェント

この方法を使用する利点は、次のことを待つ必要がないことです。CloudWatch エージェントをインストールして構成し、すぐにロギングとモニタリングを開始できます。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 に最新のもの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。CloudWatch エージェントのバージョン。- CloudWatch AMI にインストールされているエージェントは、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たびに、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、[インストール: CloudWatch 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するエージェント \(p. 20\)](#) 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。
- おれの 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、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 エージェント。
- - CloudWatch エージェントは、CloudWatch 設定が更新されました。
- AWS 認証情報は AMI に保存されてはいけません。ローカルの 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、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。CloudWatch エージェント。

Amazon ECS でのログ記録とモニタリング

Amazon Elastic Container Service (Amazon ECS) は、[2つの起動タイプ](#) コンテナを実行し、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次のとおり、AWS Fargate と Amazon EC2 です。どちらの起動タイプも CloudWatch だし、構成とサポートは異なります。

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

トピック

- [の設定 CloudWatch EC2 起動タイプの場合 \(p. 29\)](#)
- [EC2 および Fargate 起動タイプの Amazon ECS コンテナログ \(p. 30\)](#)
- [でカスタムログルーティングを使用する FireLens Amazon ECS 用 \(p. 31\)](#)
- [Amazon ECS のメトリクス \(p. 31\)](#)

の設定 CloudWatch EC2 起動タイプの場合

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 最適化された AMI には、[Amazon ECS コンテナエージェント](#)以下の内容を指定します。CloudWatch Amazon ECS クラスターのメトリクス。

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。AWS Management Console を利用し EC2 起動タイプの Amazon ECS クラスターをプロビジョニングすることにより、AWS CloudFormation 起動設定を持つグループをデプロイする Amazon EC2 Auto Scaling スタックを作成します。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、CloudWatch Amazon ECS コンテナインスタンスのエージェント。EC2 インスタンスのインストールアプローチは、[インストール: CloudWatch 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するエージェント \(p. 20\)](#) 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ーの関連付けを適用して、をインストール、設定、および更新できます。CloudWatch エージェント。ステート・マネージャーが実行され、CloudWatch エージェントの設定、標準化されたシステムレベルも適用されます CloudWatch Amazon EC2 の設定。標準化を保存することもできます CloudWatch の S3 バケット内の Amazon ECS の設定 CloudWatch 設定を行い、ステートマネージャーで自動的に適用します。

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](#) AWS CloudFormation Linux ベースの Amazon ECS クラスターをプロビジョニングするためのテンプレートを使用できます。このテンプレートは、Systems Manager をインストールし、カスタムをデプロイするカスタ

AWS の規範的ガイダンス Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch EC2 および Fargate 起動タイプの Amazon ECS コンテナログ

起動設定を持つ Amazon ECS クラスターを作成します。CloudWatch Amazon ECS 固有のログファイルを監視するための設定。

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 AWS 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パラメータストアパラメータを提供します。これは、AWS CloudFormation これらのパラメータを参照するテンプレートは、AMI を更新しなくても、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、コンテナをタスクおよび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、次のログドライバオプションのサブセットを提供します。awslogs, splunk, および awsfirelens。AWS 以下の内容を指定します。awslogs コンテナ出力をキャプチャして送信するログドライバ CloudWatch ログ。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と、自動設定 CloudWatch ログオプションの AWS Management Console です `/ecs/<task_name>`。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	<code>/<Business unit>/<Project or application name>/<Environment>/<Cluster name>/<Task name></code>
---------	--

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「[計画 CloudWatch 配備 \(p. 6\)](#)」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。

でカスタムロギングを使用する FireLens Amazon ECS 用

FireLens for Amazon ECS は、ログをルーティングするのに役立ちます [Fluentd](#) または [Fluent Bit](#) コンテナログをに直接送信できるようにする AWS サービスと AWS パートナーネットワーク (APN) の送信先およびサポートログの配送 CloudWatch ログ。

AWS 以下の内容を指定します。 [流暢ビットのドッカー画像](#) Amazon Kinesis Data Streams、Amazon Kinesis Data Firehose、および CloudWatch ログ。 FireLens ログドライバではなく awslogs ログドライバで送信されるログをさらにカスタマイズして制御できます CloudWatch ログ。

例えば、FireLens ログ形式の出力を制御するログドライバ。つまり、Amazon ECS コンテナの CloudWatch ログは自動的に JSON オブジェクトとしてフォーマットされ、次の JSON 形式のプロパティが含まれます。 ecs_cluster, ecs_task_arn, ecs_task_definition, container_id, container_name, および ec2_instance_id。 Fluent ホストは、 FLUENT_HOST そして FLUENT_PORT 環境変数は、 awsfirelens ドライバ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、コードからロギング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。

使用することを選択した場合 FireLens Amazon ECS では、と同じ設定を構成できます。 awslogs ログドライバ [他の設定も使って](#)。例えば、 [ecs-task-nginx-firelense.json](#) 使用するように構成された NGINX サーバーを起動する Amazon ECS タスク定義 FireLens CloudWatch へのログ記録用 それはまた起動する FireLens ログ記録用のサイドカーとしての Fluent Bit コンテナ。

Amazon ECS のメトリクス

[Amazon ECS は標準を提供します CloudWatch 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 EC2 および Fargate 起動タイプの (CPU およびメモリ使用率など)。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することもできます。 CloudWatch Container Insights、または埋め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できます。

コンテナインサイトは、CloudWatch クラスター、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クスを提供する機能。 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せん。

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

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、次の内容を指定します。で起動する CloudWatch Amazon ECS クラスター上のデーモンコンテナとしてのエージェント。♪[cagent-ecs-instance-metric-cfn.yaml](#) AWS CloudFormationをデプロイするテンプレート CloudWatch Amazon ECS サービスとしてのエージェント 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 エージェントのデーモンコンテナとしてデプロイされた CloudWatch Container Insights には、次のような追加のディスク、メモリ、CPU メトリクスが含まれます。instance_cpu_reserved_capacityそして instance_memory_reserved_capacityと ClusterName, ContainerInstanceName, InstanceId, InstanceId デイメンション。コンテナインスタンスレベルのメトリクスは、Container Insights によって、CloudWatch 埋め込みメトリクスフォーマット。Amazon ECS コンテナインスタンスにからのガイドの [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 CloudWatch エージェントのデプロイと設定 \(p. 21\)](#) 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。ただし、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。awslogs ログドライバが解釈しない CloudWatch 埋め込みメトリクスフォーマットステートメント。CloudWatch エージェントのサイドカーコンテナを組み込みメトリクス形式のエンドポイントとして設定した場合、CloudWatch Amazon ECS タスク内のサイドカーコンテナとしてエージェントを使用して、組み込みメトリクス形式ステートメントを取り込みます。

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。

```
{
  "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 あなたの保管に秘密 CloudWatch シークレットポリシーを構成し、組織と共有するように構成します。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。

♪AWS提供された [オープンソースの組み込みメトリック形式ライブラリアプリケーション](#) で、AWS_EMF_AGENT_ENDPOINTに接続する環境変数 CloudWatch 組み込みメトリック形式のエンドポイントとして機能するエージェントサイドカーコンテナです。例えば、[ecs_cw_emf_example](#)埋め込みメトリクス形式のメトリクスをに送信する Python アプリケーションのサンプル CloudWatch 組み込みメトリック形式のエンドポイントとして構成されたエージェントサイドカーコンテナ。

-Fluent Bit [プラグイン](#)にとって CloudWatch もサポートしています json/emf形式オプションとして、ログを通じてメトリクスを作成できることを意味します FireLens コンテナログドライバ。このオプションを使用する場合は、log_keyオプションを選択し、ログそのように CloudWatch は、埋め込みメトリック形式のメッセージを正しく認識します。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。

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

埋め込みメトリクスフォーマットを使用しない場合は、作成および更新できません CloudWatch によるメ
トリクス [AWS API](#) または [AWS SDK](#)。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、CloudWatch Kubernetes コントロールプレーンのログを記録します。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 Amazon EKS ノードとコンテナログをキャプチャするために、エージェントをデプロイすることもできます。[Fluent Bit and Fluentd](#)は、へのコンテナログの送信にもサポートされています。CloudWatch ログ。

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、[をセットアップする CloudWatch Prometheus メトリクスをスクレイプするエージェント](#)そして作成する CloudWatch メトリクス、他の Prometheus エンドポイントの消費に加えて、[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。

をインストールして設定できます。CloudWatch Amazon EKS ノード上のエージェント。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 クラスターを管理し、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、CloudWatch とコンテナインサイトを使用して、各 Amazon EKS ノードでこれらのログをキャプチャします。

コンテナは、Kubernetes クラスター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。を設定できます。CloudWatch とコンテナインサイトを使用して、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、AWS によって管理されるアカウントで実行されます。Amazon EKS クラスターコントロールプレーンノードは、CloudWatch 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。AWS コントロールプレーンノードの正常性を管理し、[Kubernetes エンドポイントのサービスレベルアグリーメント \(SLA\)](#) が提供されます。

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使用することをお勧めします。Container Insights は、クラスター、ノード、ポッドレベルのメトリクスを実装します。CloudWatch エージェント、CloudWatch へのログキャプチャ用のフルエージェントまたは Fluentd を使用します。Container Insights は、キャプチャされたレイヤー化されたビューを含む自動ダッシュボードも提供します。CloudWatch メトリクス。コンテナインサイトは CloudWatch としてデプロイされます。DaemonSet Fluent Bit DaemonSet これはすべての Amazon EKS ノードで実行されます。Fargate ノードはノードが AWS によって管理されており、DaemonSets をサポートしていないため Container Insights ではサポートされていません。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、CloudWatch でキャプチャされたロググループとログデフォルトの [Fluentd](#) または [Fluent Bit](#) [ログキャプチャ設定](#) Amazon EKS 用。

<code>/aws/containerinsights/Cluster_Name/application</code>	<code>/var/log/containers</code> のすべてのログファイル このディレクトリは、 <code>/var/log/pods</code> 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 <code>stdout</code> または <code>stderr</code> 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 <code>aws-vpc-cni-init</code> , <code>kube-proxy</code> , および <code>coreDNS</code> のログも含まれます。
<code>/aws/containerinsights/Cluster_Name/host</code>	<code>/var/log/dmesg</code> , <code>/var/log/secure</code> , および <code>/var/log/messages</code> からのログ
<code>/aws/containerinsights/Cluster_Name/dataplane</code>	<code>/var/log/journal</code> , <code>kubelet.service</code> , および <code>kubeproxy.service</code> に対する <code>docker.service</code> 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、CloudWatch Amazon EKS ノードにインストールされているエージェント。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。をインストールすると CloudWatch デイストリビューターとステートマネージャーを使用するエージェントの場合、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。

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。

<code>/var/log/containers</code>	このディレクトリは、 <code>/var/log/pods</code> 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 <code>stdout</code> または <code>stderr</code> 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 <code>aws-vpc-cni-init</code> , <code>kube-proxy</code> , および <code>coreDNS</code> のログも含まれます。重要: Container Insights を使用している場合、これは必須ではありません。
----------------------------------	---

<code>var/log/aws-routed-eni/ipamd.log</code>	L-IPAM デーモンのログはここにありますが
<code>/var/log/aws-routed-eni/plugin.log</code>	

Amazon EKS ノードが CloudWatch 適切なシステムレベルのログとメトリクスを送信するためのエージェント。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。を使用して [起動テンプレート](#) では、Systems Manager エージェントのインストールとデフォルト設定を自動化できます。CloudWatch Amazon EKS 固有の重要なログをキャプチャし、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設定します。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、[テンプレートを起動する](#) これには、Systems Manager エージェントのインストールを自動化するためのユーザーデータセクションと CloudWatch の設定 カスタマイズして使うことができる [amazon_eks_managed_node_group_launch_config.yaml](#) AWS CloudFormation Systems Manager エージェントをインストールする起動テンプレートを作成するためのテンプレート CloudWatch エージェント。また、Amazon EKS 固有のロギング設定を CloudWatch 設定ディレクトリ。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 infrastructure-as-code (IaC) アプローチ。更新するたびに AWS CloudFormation 起動テンプレートは新しいバージョンの起動テンプレートをプロビジョニングします。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た IAM ロールとインスタンスプロファイルに、CloudWatchAgentServerPolicy として AmazonSSMManagedInstanceCore AWS マネージドポリシーを確認します。

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。お客様は Amazon EKS クラスターに自己管理ノードをデプロイするのに AWS CloudFormation、つまり、Amazon EKS クラスターに IaC およびマネージド変更アプローチを使用できます。AWS は、[amazon-eks-nodegroup.yaml](#) AWS CloudFormation そのまままたは、カスタマイズできるテンプレートを提供できます。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](#) AWS CloudFormation テンプレートは、必要な Systems Manager エージェントをインストールする更新バージョンです。CloudWatch エージェント。また、Amazon EKS 固有のロギング設定を CloudWatch 設定ディレクトリ。

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、ログを次の場所に自動的にルーティングできます。CloudWatch 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する必要はありません。詳細については、「[ブログ上の Fargate ログ記録](#) Amazon EKS のドキュメントおよび [Amazon EKS のための流暢なビットAWS](#)」を参照してください。このソリューションでは、STDOUT として STDERR コンテナからの入出力 (I/O) ストリームをコンテナから送信し、CloudWatch Fargate 上の Amazon EKS クラスター用に確立された流暢ビット設定に基づいて、Fluent Bit を使用します。

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 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ます。Prometheus 形式でそして CloudWatch では、これらのメトリックをキャプチャして取り込むことができます。CloudWatch Container Insights 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設定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みます。

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 クラスター上で実行します。これらのメトリックは、[Horizontal Pod Autoscaler](#)そして[Vertical Pod Autoscaler](#)。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。このソリューションは、Amazon EKS のドキュメント内の [Kubernetes ダッシュボードのデプロイ方法](#) でのステップに従って個々のクラスターにデプロイできます。

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、point-in-time 分析と履歴分析ではありません。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。

コンテナインサイトでは、コンテナ化されたバージョンの CloudWatch Kubernetes で実行されるエージェント DaemonSet を使用して、クラスター内のすべての実行中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。AWS クイックスタートまたはコンテナインサイトを個別に設定 からクイックスタートを使用できます。クイックスタートは、CloudWatch Fluent Bit でエージェントとロギングを行うため、ロ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クスをキャプチャする必要があります。と同じアプローチを使用できます。[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 CloudWatch エージェントのデプロイと設定 \(p. 21\)](#)をインストールして設定するには、このガイドのセクション CloudWatch Amazon EKS クラスター用のエージェント。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 固有のログ設定を含めることができます。

- CloudWatch 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スクレイプできます。[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](#)。それはそれらを次のように取り込みます。CloudWatch で分析するために、埋め込みメトリック形式でログインします。CloudWatch インサイトを記録し、CloudWatch メトリクスを自動的に作成します。

Important

する必要があります [特殊バージョンをデプロイする](#) の CloudWatch Prometheus メトリクスを収集するエージェント。これは別のエージェントです。CloudWatch コンテナインサイト用にエージェントがデプロイされました。[prometheus_jmx](#) サンプル Java アプリケーション。このアプリケーションには、のデプロイメントファイルと構成ファイルが含まれています。CloudWatch エージェントと Amazon EKS ポッドのデプロイで、Prometheus メトリクスの検出を実証し

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、CloudWatch 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エージェント。

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](#)。- CloudWatch エージェント構成は Kubernetes として保存されます。ConfigMap そして、あなたによって読んでください CloudWatch エージェントサイドカーコンテナを使用して、組み込みメトリ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ep、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 エクスポート](#) Java アプリケーションを使用して、Prometheus 消費用の JMX Bean を公開するには CloudWatch エージェント。

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](#)で CloudWatch メトリクスを作成および更新できます。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 としてのエージェント DaemonSet 各クラスタノードで。
- 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。ログルータは Fluent Bit を使用します。Fluent Bit には CloudWatch 埋め込みメトリクス形式のステートメントをサポートするように設定可能なプラグイン。

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 バックアップされたノードに Prometheus をデプロイすることもできます。詳細については、「[AWS Fargate ブログ上の AWS Prometheus と Grafana を使用した Amazon EKS のモニタリング](#)」を参照してください

Amazon EKS における Prometheus モニタリング

[Prometheus 向けの Amazon Managed Services](#) は、スケーラブルで安全な、オープンソースの Prometheus のための AWS Managed Services 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 メトリクスの収集は、[AWS Distro for F OpenTelemetry \(ADOT\)](#) または Prometheus サーバーをコレクションエージェントとして使います。

[CloudWatch コンテナインサイトの Prometheus モニタリング](#) では、を設定して使用できます。CloudWatch エージェント。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。この解決策は、次の場合に適切です。CloudWatch 主要なオプザバビリティおよびモニタリングソリューションです。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。Prometheus 向け Amazon Managed Services は、AWS クラウド内で、成熟した Prometheus デプロイを活用したい場合に適した選択です。
- 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。代わりに Prometheus で Grafana を使いたい場合は CloudWatch コンテナのモニタリングにダッシュボードを使用すると、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、[その CloudWatch エージェントは、埋め込みメトリクスフォーマットで Prometheus メトリクスを取り込みます。](#)に CloudWatch 結果となるログ CloudWatch メトリクス。埋め込みメトリクスフォーマットログは、次の方法でクエリできます。CloudWatch ログインサイト。
- 使用する予定がない場合は CloudWatch モニタリングとメトリクスのキャプチャには、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、デプロイする依存性はありません。CloudWatch CloudWatch にデータを取り込むためのエージェントまたは要件。- 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。

AWS Lambda のログ記録とメトリクス。

Lambda 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、自動的に CloudWatch メトリクス、および CloudWatch アプリケーションのコードの設定やインストールなしでログを記録します。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自動的にストリーミングします。CloudWatch ログ、ドライバをログに記録する必要はありません。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 ファンクションインスタンスを識別するために AWS によって生成されます。

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。また、を使用してより簡単にクエリを実行できるため、JSON 形式でメッセージを記録することをお勧めします。CloudWatch インサイトを記録します。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。

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

ログを他の送信先 (例えば Amazon など) に送信することができます。OpenSearch サブスクリプションフィルタを使用することで、サービスまたは Lambda 関数)。Amazonを使わないと OpenSearch サービスでは、Lambda 関数でログを処理し、ログをAWSを使用して、任意のサービスAWSSDK。

また、Lambda 関数で AWS クラウド外のログ送信先用の SDK を使用し、ログステートメントを直接任意の送信先に送ることも可能です。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ルアップ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の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することができます。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キャプチャを行うために、埋め込みメトリクス形式を直接サポートします。CloudWatch メトリクスは簡単です。

システムレベルのメトリクス

Lambda は自動的に統合されます。CloudWatch メトリクスを提供し、一連の情報を提供します。[Lambda 関数の標準メトリクス](#)。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ちます。

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が作成される CloudWatch 自動の基礎となるメトリクス CloudWatch ダッシュボード。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

では、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。CloudWatch 埋め込みメトリクス形式を使用する。[AWS が提供する埋め込みメトリクス形式のライブラリ](#) を活用して、埋め込みメトリクス形式のステートメントを作成し、CloudWatch に発行することができます。統合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 リソースをまとめたものを使用します。可能な場合は、これらのリソースからのメトリクスとログをまとめて分析し、それらを関連づけ、AWS ワークロードの全てを効果的にモニタリングおよび管理する必要があります。

CloudWatch には、ログとメトリクスの分析に役立つ複数の機能が搭載されています。[CloudWatch Application Insights](#)異な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してモニタリングするには AWS リソース、[CloudWatch 異常検出](#)メトリクスの異常を明らかにするため、[CloudWatch ログインサイト](#)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には CloudWatch ログ。

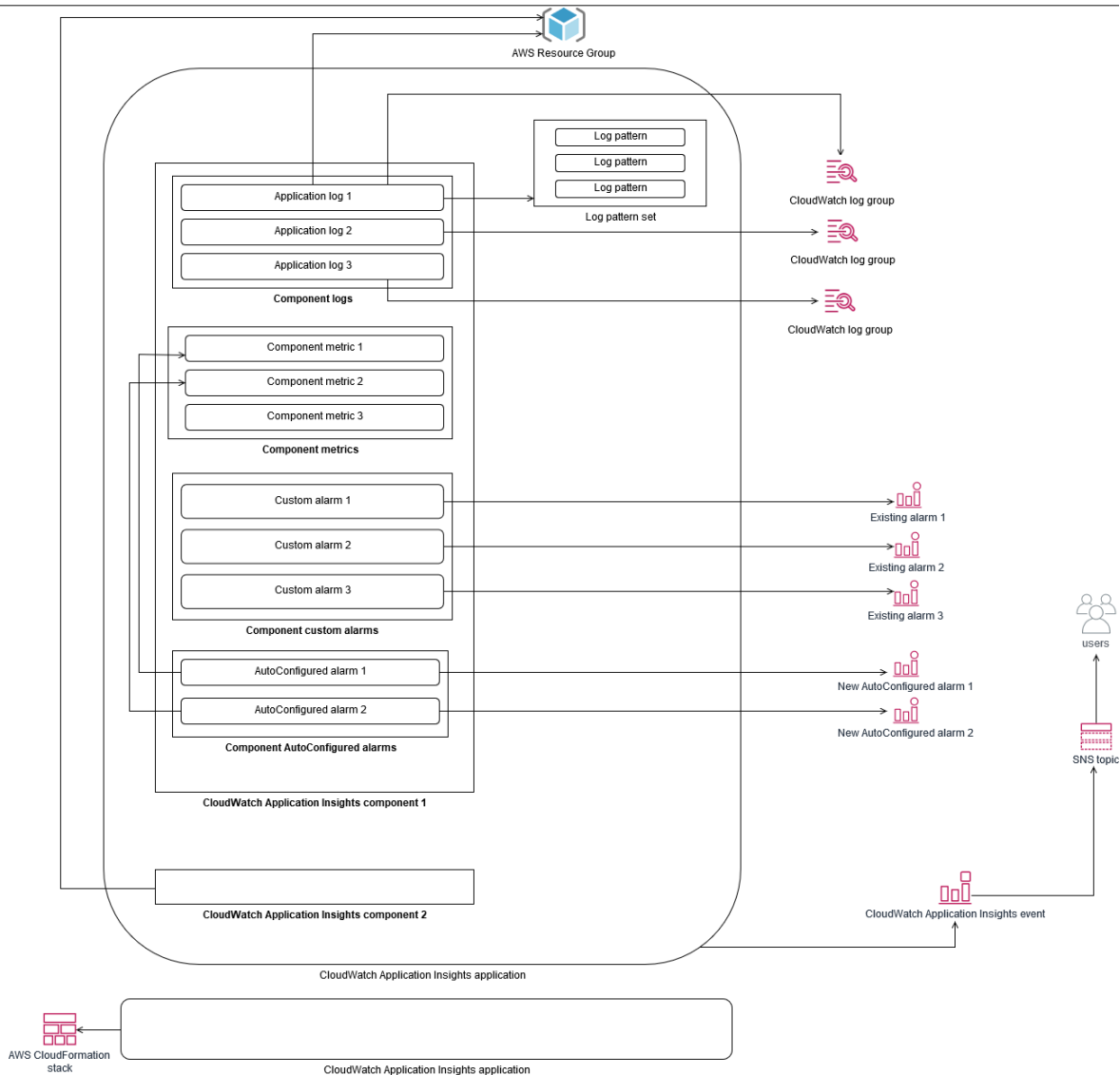
でアプリケーションをまとめてモニタリングおよび分析する CloudWatch アプリケーションインサイト

アプリケーションオーナーは Amazon を使用できます CloudWatch Application Insights では、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。を使用してモニタリングをセットアップする CloudWatch アプリケーションインサイトは、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。リソースグループ内のサポートされているリソースは、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アプリケーション。のそれぞれのコンポーネント CloudWatch Application Insights アプリケーションには、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アプリケーション。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。アラームの場合は、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標準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には、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。[でサポートされているログとメトリクス CloudWatch アプリケーションインサイトの](#) CloudWatch ドキュメント内) を参照してください。次の図は、の関連性を示しています。CloudWatch Application Insights のコンポーネントとそのロギングとモニタリングの設定 各コンポーネントは、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 ログとメトリクス

AWS の規範的ガイドライン Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch でアプリケーションをまとめてモニタリングおよび分析する CloudWatch アプリケーションインサイト



によって監視される EC2 インスタンス CloudWatch アプリケーションインサイトには、Systems Manager と CloudWatch エージェントと権限。詳細については、「」を参照してください。を使用してアプリケーションを構成するための前提条件 CloudWatch アプリケーションインサイトの CloudWatch ドキュメント内)を参照してください。CloudWatch Application Insights は、Systems Manager を使用して、CloudWatch エージェント。CloudWatch アプリケーションインサイトで設定されたメトリックスとログは、CloudWatch Systems Manager パラメータに格納されるエージェント設定ファイル AmazonCloudWatch-ApplicationInsights-SSMParameter 各 CloudWatch アプリケーションインサイトコンポーネントのプレフィックス。これにより、別の結果になります。CloudWatch に追加されるエージェント設定ファイル CloudWatch EC2 インスタンス上のエージェント設定ディレクトリ。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。を使用する CloudWatch アプリケーションインサイトは既存に影響を与えません CloudWatch エージェントの設定です。次を使用できます。CloudWatch 独自のシステムおよびアプリケーションレベルに加えて、Application Insights CloudWatch エージェントの設定。ただし、設定が重複しないようにする必要があります。

でログ分析を実行する CloudWatch ログインサイト

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内のログストリーム全体の JSON フィールドを自動的に検出します。次を使用できます。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。のクエリ構文 CloudWatch Logs インサイトは、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。

埋め込みメトリクスフォーマットを使用して作成する場合 CloudWatch メトリクスの場合は、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、CloudWatch 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、コストを監視できます。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、生成するだけです。CloudWatch このデータのサブセットのメトリクス。

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 でログ分析を実行する OpenSearch サービス

CloudWatch との統合 [アマゾン OpenSearch サービス](#) からログデータをストリーミングできるようにする CloudWatch グループを Amazon にログに記録する OpenSearch お好みのサービスクラスター [サブスクリプションフィルタ](#)。次を使用できます。 CloudWatch プライマリログとメトリクスのキャプチャと分析を行い、Amazon で強化する OpenSearch 次のユースケースに対応するサービス:

- きめ細かなデータアクセス制御— アマゾン OpenSearch サービスを使用すると、データへのアクセスをフィールドレベルまで制限し、ユーザー権限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の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のログを共通の Amazon にストリーミングできます。 OpenSearch サービスクラスター。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。ストリーミング CloudWatch Amazon にログ OpenSearch サービスは、複数のリージョンのアプリケーションを一元的に検索および分析する際にも役立ちます。

- Amazonに直接ログを出荷して豊かにする OpenSearch ElasticSearch エージェントを使用したサービス—アプリケーションおよびテクノロジースタックのコンポーネントは、サポートされていない OS を使用できます。CloudWatch エージェント。また、ログデータをログソリューションに出荷する前に、ログデータをエンリッチして変換することもできます。アマゾン OpenSearch サービスは、次のような標準的な Elasticsearch クライアントをサポートしています。[Elastic Beats ファミリーデータの荷送人](#)そして[Logstash](#)これは、ログデータを Amazon に送信する前に、ログの強化と変換をサポートします。OpenSearch サービス。
- 既存の運用管理ソリューションでは[ElasticSearch](#)、[Logstash](#)、[Kibana](#)(ELK) ログとモニタリング用のスタック—Amazonにすでに多額の投資がある可能性があります OpenSearch 多くのワークロードがすでに設定されたサービスまたはオープンソース Elasticsearch。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。

使用する予定がない場合 CloudWatch ログ、Amazonが使える OpenSearch サービスがサポートするエージェント、ログドライバ、ライブラリ (Fluent Bit、Fluentd、など) [Logstash](#)であり、[用のデストリビューションを開く ElasticSearch API](#)) ログをAmazonに直接発送する OpenSearch CloudWatch をサービスとバイパスします。ただし、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。AWSのサービス。CloudWatch ログは、多くの人にとって主要なログキャプチャソリューションです。AWSサービスおよび複数のサービスが CloudWatch で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。Amazon にログをストリーミングするロググループのサブスクリプションフィルターを設定できます。OpenSearch サービス。Amazon にストリーミングしたい個々のロググループに対して、サブスクリプションフィルターを手動で設定できます。OpenSearch サービス。または、新しいロググループを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。ElasticSearch クラスターを試すことができます。ログをストリーミングするには ElasticSearch 同じアカウントまたは集中型アカウント内のクラスター化にログをストリーミングする ElasticSearch 同じアカウント内のクラスターは、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するのに役立ちます。

のセットアップを検討してください。ElasticSearch アカウント、リージョン、およびアプリケーション全体のログを集約するために、集中型アカウントまたは共有アカウントでクラスター化します。例えば、ログの集中化に使用されるログアーカイブアカウントを AWS Control Tower 設定します。AWS Control Tower で新しいアカウントが作成する時、AWS CloudTrail および AWS Config ログは、この集中型アカウントの S3 バケットに配信されます。AWS Control Tower によって計測されるロギングは、設定、変更、および監査ロギング用です。

Amazon で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 OpenSearch サービス、一元化された Amazon を 1 つ以上デプロイできます OpenSearch 集中型ロギングアカウントにクラスターをサービスし、他のアカウントのロググループを設定して、集中型 Amazon にログをストリーミングすることができます。OpenSearch サービスクラスター。

別個の Amazon を作成できる OpenSearch アカウントに分散しているアプリケーションや、クラウドアーキテクチャのレイヤーを処理するために、サービスクラスター。別々の Amazon を使用する OpenSearch サービスクラスターは、セキュリティと可用性のリスクを軽減し、共通の Amazon を持つことに役立ちます。OpenSearch サービスクラスターを使用すると、同じクラスター内のデータの検索や関連付けを簡単にできるようになります。

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。次を使用できます。CloudWatch では、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が、これらのリソースを名前空間の下にグループ化して、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。[CloudWatch ServiceLens](#) を使用することで、トレース、メトリクス、ログ、アラームを関連付けて、問題を診断できます。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。

を使用する CloudWatch をモニタリングし、アラームを発する

[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。例えば、AWS は [ServiceNow](#) と [JIRA Service Desk](#) の AWS サービス管理コネクタを提供することで、統合をすばやくセットアップできます。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ージのしきい値に 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。

を使用する CloudWatch 異常検出とアラームの実行

次を使用できます。[CloudWatch 異常検出](#) 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、CloudWatch 外れ値に基づいてアラームを発します。

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、「」 [異常検出に基づく CloudWatch アラームの作成](#) の CloudWatch ドキュメント内) 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、このプロセスを簡素化および自動化できます。AWS CloudFormation StackSets とテンプレートをを使用して、必要なアラームを備えたアプリケーションリソースをデプロイできます。templateYでは、共通の 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。これは、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ることを意味します。

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集約アラームを作成して、アカウントとリージョンの問題を監視することをお勧めします。AWS CloudFormation StackSets 平均などの指標を集計するCPUUtilizationすべての EC2 インスタンス間

また、スタンダード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。CloudWatch キャプチャしたメトリックとログ。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。では、AWS CloudFormation StackSets 複数のリージョンとアカウントをサポートします。詳細については、次を参照してください。[タグを使用して Amazon を作成および保守する CloudWatch Amazon EC2 インスタンス用のアラーム](#)でAWSブログ。

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 CloudWatch 実行された各ステップを含む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。

AWS X-Ray でアプリケーションをトレースする

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関連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、X-Ray に自動的にデータを送信するインストルメント化クライアントなど、複数の言語に対応した AWS SDK を提供します。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 のパッチによって、サブセグメントとして Amazon DynamoDB などの AWS サービスへの通話を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 相関関係 CloudWatch アプリケーショントレースを使用したログとメトリクスAWS X-Ray SDK for Javaつまり、リクエストに関連するメトリクスとログ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際に 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 インスタンスでは、Systems Manager である [AWS RunPowerShellScript ドキュメント](#) を使用し、X-Ray エージェントをダウンロード、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 ドキュメントを実行して、[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 スクリプトを実行します。

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 の規範的ガイダンス Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch X-Ray デーモンをデプロイし、Amazon ECS または Amazon EKS でのアプリを使用する場合と、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せん。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 関数で有効にするには、AWS Management Console を使用して、X-Ray コンソールの 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ールメントする

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ー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ブです。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、リザーブ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、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 メトリクス。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。例えば、受注やメモリ、CPU 使用率や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、複数の AWS リソースにわたりワークロードメトリクスの変更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。を使用すると、開始できます。CloudWatch 自動ダッシュボード。AWSサービス固有のメトリクスで事前構成されたサービスレベルダッシュボード。自動サービスダッシュボードにすべての標準が表示されます CloudWatch サービスのメトリクス。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確認し、アクション メニューの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 フルスクリーンモードの大型TVモニターのダッシュボード。自動リフレッシュ機能付き。

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のメトリクス、それからパフォーマンスに影響する重要な AWS リソースメトリクスに重点を置きます。

定期的に評価してカスタマイズするべき CloudWatch インシデント発生後にキーメトリクスを追跡するためのアプリケーションまたはワークロードダッシュボード。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であり、メトリクス、アラーム、およびダッシュボードは、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 ログインサイトウィジェット。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、AWS では、IOPS での Amazon EBS ボリュームパフォーマンスに関するガイドラインを示します。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。で Amazon Elastic File System (Amazon EFS) ファイルシステムのモニタリングにメトリクス数学を使用する方法の詳細と例については、[CloudWatch 指標](#)、「」を参照してください。[アマゾン CloudWatch メトリクス計算により、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](#)で AWS ブログ。

AWS の規範的ガイドライン Amazon を使用したロギングとモニタリングの設計と実装 CloudWatch で、Amazon ECS、Amazon EKS、および Lambda 自動ダッシュボードを使用する CloudWatch Container インサイトと CloudWatch Lambda Insights

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

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に似て、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。両者とも使用している CloudWatch キャプチャする埋込みメトリクスフォーマット CloudWatch メトリクスおよびパフォーマンスログ。

共有を作成できます。CloudWatch 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ダッシュボード。これを行うには、自動ダッシュボードをフィルタリングして表示し、CloudWatch コンテナインサイトを選択し、ダッシュボードに追加標準の CloudWatch ダッシュボードに表示されるメトリクスを追加できるオプション。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。

AWS サービスと CloudWatch の統合。

AWS は、ロギングとメトリクスの追加構成オプションを含む多くのサービスを提供します。これらのサービスでは、多くの場合、構成できます。CloudWatch ログ出力用のログと CloudWatch メトリック出力のメトリクス。これらのサービスを提供するために使用される基盤のインフラストラクチャは、AWS によって管理され、はアクセス不可のとなります。ただし、ロギングとメトリクスオプションを使用して、お客様のプロビジョニングされたサービスが詳細なインサイトを取得し、問題のトラブルシューティングを行うことができます。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。

MustAWSによる API コールのログ記録との統合AWS CloudTrail。CloudTrail またでとの統合がサポートされます。[CloudWatch ログ](#)これは、でアクティビティを検索して分析できることを意味しますAWSのサービス。Amazonを使用することもできます CloudWatch Amazon Events EventBridge で自動化と通知を作成および構成するには CloudWatch で実行される特定のアクションに関するイベントイベントルールAWSのサービス。特定のサービス[直接統合](#)と CloudWatch イベントと EventBridge。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 Managed Grafana (プレ ビュー)

Important

Amazon Managed Grafana がプレビュー中です。一般提供を発表する前に、機能が追加または変更されることがあります。現在、プレビューでは、次のリージョンがサポートされています。米国東部 (バージニア北部) およびヨーロッパ (アイルランド)

Amazon Managed Grafana を使用し、お客様の AWS ワークロードを観察し、可視化することができます。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 マネージド Grafana をで使える CloudWatch によって [データソースとして追加する](#)。つまり、CloudWatch メトリクスを使用してビジュアライゼーションを作成できるということです。Amazon Managed Grafana サポート AWS Organizations また、ダッシュボードを一元化することもできます。CloudWatch 複数のアカウントおよびリージョンからのメトリクス

次の表に、の代わりに Amazon Managed Grafana を使用する利点と注意点を示します。CloudWatch ダッシュボード用。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、次のようなさまざまなデータソースからビジュアライゼーションおよびダッシュボードを作成するのに役立ちます。CloudWatch メトリクス。Amazon Managed Grafana には、AWS サービス、オープンソースソフトウェア、および COTS ソフトウェアにまたがる数多くのビルトイン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。CloudWatch ダッシュボードには、CloudWatch のメトリクスまたは CloudWatch Logs Insights のクエリを表示するデータが CloudWatch ダッシュボード。
AWS アカウントアクセスとは別にダッシュボードソリューションへのアクセスを管理する	Amazon Managed Grafana には、AWS Single Sign-On の使用 (AWS SSO) と認証と認可のための AWS Organizations が必要です。これにより、すでに AWS SSO または AWS Organizations で

	<p>使用している可能性のあるアイデンティティフェ デレーションを使用して Grafana にユーザーを 認証できます。ただし、AWS SSO または AWS Organizations を使用していない場合、Amazon Managed Grafana の設定プロセスの一部として 設定されます。お客様の組織が AWS SSO または AWS Organizations の使用を制限している場合、 これは問題となる可能性があります。</p>
<p>AWS Organizations 統合を使用して、複数のアカ ウントとリージョンでデータの取り込みとアクセ スを行う</p>	<p>Amazon Managed Grafana はAWS Organizations データを読み込めるようにするにはAWSなどの 出典 CloudWatch そしてアマゾン OpenSearch す べてのアカウントでサービスを行えます。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。AWS Organizations での自動的なデータアクセスを有効 にするには、AWS Organizations 管理アカウント で、お客様の 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 またのクロスアカウント、クロスリー ジョンダッシュボードをサポート CloudWatch メ トリックス。</p>
<p>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</p>	<p>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。</p>
<p>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</p>	<p>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。</p>
<p>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</p>	<p>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。</p>

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

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。

どこに保存すればよいですか CloudWatch 設定ファイル?

- CloudWatch Amazon EC2 エージェントは、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 設定ディレクトリ。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 [保存 CloudWatch S3 バケットの設定ファイル \(p. 9\)](#) セクションを参照してください。または、設定ファイルをリポジトリに保存することもできます。GitHub および、新しい EC2 インスタンスがプロビジョニングされる時の設定ファイルの取得を自動化します。

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、CloudWatch アラームが発生したときに SNS トピックを通知するアラーム。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。

をどのように使用しますか CloudWatch コンテナにログファイルをキャプチャするには？

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に送信する場合 CloudWatch 次に、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したらいいですか？

♪[AWS Health Dashboard](#) モニターの方法 AWSヘルスイベント。また、[aws-Healths ツール](#) [GitHub](#) 関連するサンプル自動化ソリューションのリポジトリ [AWSヘルスイベント](#)。

カスタムを作成するにはどうすればよいですか CloudWatch エージェントサポートが存在しない場 合のメトリック

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。また、AWS SDK (例えば、[put_metric_data](#))、AWS CLI (例えば、[put-metric-data](#))、または AWS API (例えば、[PutMetricData](#)) を使用して、カスタムメトリクスを作成します。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、メトリクスを作成することです。CloudWatch イベントイベントスケジュールルールを選択して、メトリックの期間を設定します。

AWS の既存のログおよびモニタリングツールを統 合するにはどうしたらよいですか？

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。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、[aws-logs](#) を使用することもできます。AWSSDK および CloudWatch Lambda および Kinesis Data Streams のサブスクリプションフィルターを記録して、カスタムログプロセッサとシッパーを作成します。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

イントロダクション

- [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](#)

の計画 CloudWatch 配備

- [AWS Organizations の用語と概念](#)
- [AWS Systems Manager 高速セットアップ](#)
- [を使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する CloudWatch エージェント](#)
- [cloudwatch-config-s3-bucket.yaml](#)
- [の作成 CloudWatch ウィザードを使用したエージェント設定ファイル](#)
- [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 でのきめ細かなアクセス制御 OpenSearch サービス](#)
- [Lambda クォータ](#)
- [を手動で作成または編集する CloudWatch エージェント設定ファイル](#)
- [サブスクリプションを使用したログデータのリアルタイム処理](#)
- [AWS に構築するツール](#)

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

- [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 エージェントの事前定義されたメトリクスセット](#)
- [procstat プラグインでプロセスメトリクスを収集する](#)
- [設定: CloudWatch procstat 用のエージェント](#)
- [インスタンスの詳細モニタリングを有効または無効にする](#)
- [高カーディナリティログの取り込みとメトリクスの生成 CloudWatch 組み込みメトリクス形式](#)
- [ロググループとログストリームの操作](#)
- [利用可能なものを一覧表示します。 CloudWatch インスタンスのメトリクス](#)

- [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

- ハイブリッド環境用の 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のアクティベーションを作成する
- で使用する IAM ロールとユーザーを作成する CloudWatch エージェント
- をダウンロードして設定します CloudWatch コマンドラインを使用したエージェント
- 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 でのロ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](#)
- のデプロイ CloudWatch Amazon ECS 上で EC2 インスタンスレベルのメトリクスを収集するためのエージェント
- [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](#)を使用する
- クライアントライブラリを使用した組み込みメトリクスフォーマットログの生成

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 を使用したプレーンメトリクスのコントロール
- チュートリアル: Kubernetes ダッシュボード (ウェブ UI) のデプロイ
- Fargate ログ記録
- Fargate で Amazon EKS の流暢なビット
- Fargate で Amazon EKS を使用するときアプリケーションログをキャプチャする方法
- インストール: CloudWatch Prometheus メトリクスの収集エージェント
- 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
- マネージド型ノードグループ
- 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
- へのログの送信 CloudWatch ログ
- fluentD を DaemonSet にログを送信するには CloudWatch ログ
- 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する
- 新しい Prometheus スクレイピングターゲットを追加するためのチュートリアル: Prometheus API サーバーメトリクス
- Vertical Pod Autoscaler

AWS Lambda のログ記録とメトリクス。

- 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た組み込みメトリクスフォーマットログの生成
- 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
- ストリーミング CloudWatch Amazon にデータをログする OpenSearch サービス

CloudWatch によるアラームのオプション

- amazon-cloudwatch-auto-alarms

- [AWS AWS Service Management Connector for Jira Service Management](#)
- [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](#)

AWS X-Ray でアプリケーションをトレース

- [Amazon ECS タスクネットワークング](#)
- [X-Ray コンソールでのサンプリングルールの設定](#)
- [Windows を実行 PowerShell コマンドまたはスクリプト](#)
- [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

- [アマゾン 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](#)
- [セットアップ CloudWatch Container Insights](#)
- [Metric Math を使用する](#)

CloudWatch との統合 AWS サービス

- [AWS CloudTrail でサポートされるサービスと統合](#)
- [サポートされているサービスからの CloudWatch Events イベントの例](#)
- [CloudTrail 経由で配信されるイベント](#)
- [Monitoring CloudTrail とのログファイル CloudWatch ログ](#)
- [へのデータベースエンジンログの発行 CloudWatch ログ](#)
- [へのフローログの発行 CloudWatch ログ](#)

ダッシュボードと可視化のための Amazon マネージド Grafana

- [AWS Organizations の管理アカウントのベストプラクティス](#)
- [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](#)

AWS 規範的ガイダンスの用語集

[AI と ML の用語 \(p. 67\)](#) | [移行の用語 \(p. 68\)](#) | [モダナイゼーションの用語 \(p. 72\)](#)

AI と ML の用語

以下は、AWS 規範的ガイダンスによって提供される人工知能 (AI) および機械学習 (ML) に関連する戦略、ガイド、および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。

二項分類	バイナリ結果 (2 つの可能なクラスの中の 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。
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。
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。
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。
特徴	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、機能は製造ラインから定期的にキャプチャされるイメージである可能性があります。
特徴の重要度	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、次を参照してください。[AWS を使用した機械学習モデルの解釈可能性](#)。

機能変換	追加のソースによるデータのエンリッチ化、値のスケールリング、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なパターンを学習するのに役立ちます。
解釈	機械学習モデルの特性で、モデルの予測が入力にどのように依存するかを人間が理解できる程度を記述します。詳細については、次を参照してください。 AWS を使用した機械学習モデルの解釈可能性 。
複数クラス分類	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。
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。
.training	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。
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。
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。
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。認識論の不確実性が限られた、不完全なデータによって引き起こされる。一方、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。

移行の条件

以下は、AWS 規範的ガイダンスによって提供される移行関連戦略、ガイド、および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。

- 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する。

- 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: お客様のオンプレミスの Oracle データベースをの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行するAWSCloud。
- 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: お客様の顧客関係管理 (CRM) システムを Salesforce.com に移行する。
- 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: の EC2 インスタンス上の Oracle 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 に移行するAWSCloud。
- 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の購入、アプリケーションの書き換え、お客様の既存のオペレーションの変更を行うことなく、インフラストラクチャをクラウドに移行できます。この移行シナリオは AWS の VMware Cloud に固有のもので、お客様のオンプレミス環境と、および AWS の間の仮想マシン (VM) 互換性とワークロードの移植性をサポートします。インフラストラクチャを AWS の VMware Cloud に移行するときに、お客様のオンプレミスのデータセンターから VMware Cloud Foundation テクノロジーを使用できます。例: Oracle データベースをホストしているハイパーバイザーを VMware Cloud 上に再配置する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、およびそれらを行移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。

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。

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。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドに正常に移行するための効率的で効果的な計画を立てるのを支援する AWS から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。AWSCAF 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、運用の観点と呼ばれる 6 つの重点を置く分野にガイダンスを編成しています。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重点を当てています。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、クラウドの導入を成功させるための組織の準備を支援するために、人材開発、トレーニング、コミュニケーションに関する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 Cloud Adoption Framework の概要](#) を参照してください。

AWS ランディングゾーン

ランディングゾーンは、Well-Architected の、スケーラブルで安全なマルチアカウント 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

	<p>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。</p>
AWS ワークロード資格フレームワーク (AWS WQF)	<p>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、作業の見積もりを提供するツール。AWSWQF は AWS Schema Conversion Tool (AWS SCT) と共に含まれます。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。</p>
ビジネス継続性計画 (BCP)	<p>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。</p>
Cloud Center of Excellence (CCoE)	<p>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 の投稿 を参照してください。</p>
導入のクラウドステージ	<p>組織が、AWS クラウドへの移行時に通常実行する 4 つの段階。</p> <ul style="list-style-type: none">• 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 <p>これらのステージは Stephen Orban が AWS クラウドエンタープライズ戦略ブログの クラウドファーストジャーニーと導入ステージ というブログ記事で定義したものです。これらが、AWS 移行戦略とどのような関係があるかについては、移行準備ガイド を参照してください。</p>
構成管理データベース (CMDB)	<p>企業のハードウェアおよびソフトウェア製品、構成、相互依存関係に関する情報を含むデータベース。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。</p>
エピック	<p>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、アイデンティティとアクセスの管理、検出型制御、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。</p>
異種混在データベースの移行	<p>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。</p>
同種データベースの移行	<p>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。</p>
アイドル状態のアプリケーション	<p>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。</p>
IT 情報ライブラリ (ITIL)	<p>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。</p>

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。
大規模な移行	300 台以上のサーバの移行。
Migration Acceleration Program (MAP)	組織がクラウドへの移行の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。
移行ポートフォリオ評価 (MPA)	AWS クラウド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。MPA は、詳細なポートフォリオ評価 (サーバ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、すべての人に無料で利用できる AWS コンサルタントと APN パートナーコンサルタントです。
移行準備状況評価 (MRA)	組織のクラウド対応状況に関するインサイトを獲得し、長所と短所を特定し、AWS CAF を使用して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。
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、およびが含まれます。DevOps スプリントで働くプロフェッショナル。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 移行ファクトリーの議論 と CloudEndure 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に関する情報。移行パターンごとに、異なる移行メタデータのセットが必要です。移行メタデータの例には、ターゲットサブネット、セキュリティグループ、およびAWSアカウント。
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: で Amazon EC2 への移行を再ホストするAWSアプリケーション移行サービス。
移行戦略	ワークロードをAWSCloud。詳細については、「」を参照してください。 7 Rs (p. 68) この用語集に記入し、 組織を動員して大規模な移行を加速する 。
運用レベル契約 (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。
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更のスピードから、このフレームワークは人材の高速化と呼ばれます。詳細については、[OCM ガイド](#) を参照してください。

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の運用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。
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、次を参照してください。 移行準備状況の評価 。
実行責任者、説明責任者、協業先、報告先 (RACI) マトリックス	プロジェクト内のロールと責任を定義して割り当てるマトリックス。例えば、お客様が RACI を作成して、セキュリティコントロールの所有権を定義したり、移行プロジェクト内の特定のタスクの役割と責任を特定したりできます。
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。
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ことを約束するものを明確にする契約。
タスクリスト	Runbook 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、Runbook の概要と完了する一般的なタスクのリストが含まれています。一般的なタスクごとに、推定所要時間、所有者、進捗状況が含まれます。
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、ウェーブプランニング、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、これらの資産を移行ワークストリームに配信し、サーバーとアプリケーションを移行します。
ゾンビアアプリケーション	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。

モダナイゼーションの条件

以下は、AWS 規範的ガイダンスによって提供されるモダナイゼーション関連戦略、ガイド、および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。

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、 AWS でのコンテナ化されたマイクロサービスの実行 ホワイトペーパーの 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。
ドメイン駆動型の設計	各コンポーネントが提供する進化するドメイン、つまり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・エヴァンスの著書で紹介され、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 (ポストン: 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、「」を参照してください。 コンテナと Amazon API Gateway を使用して、 レガシー Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ 。

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てマイクロサービスを統合する 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、 AWS でのマイクロサービスの実装 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、 AWS クラウドのアプリケーションのモダナイズ戦略 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、 AWS クラウドで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評価する 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。
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。
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、 AWS クラウドで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 を参照してください。
絞め師イチジクのパターン	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、確立された木に成長し、最終的にその宿主を克服して置き換えるイチジクのブドウの類推を使用します。パターンは マーティン・ファウラーによる導入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、このパターンの適用方法の例については、「」を参照してください。 コンテナと Amazon API Gateway を使用して、 レガシー Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ 。

2 枚ピザチーム

A Small DevOps 2 枚のピザをフィードすることができるチーム。2 枚ピザチームのサイズにより、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。詳細については、「」を参照してください。[2 枚ピザチーム](#)のセクションについて [DevOps オンAWS](#)ホワイトペーパー。

ドキュメント履歴

このガイドは、このドキュメントの大きな変更点をまとめたものです。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。

update-history-change	update-history-description	update-history-date
セクションを削除しました (p. 75)	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 (p. 75)	—	2021 年 4 月 30 日

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。